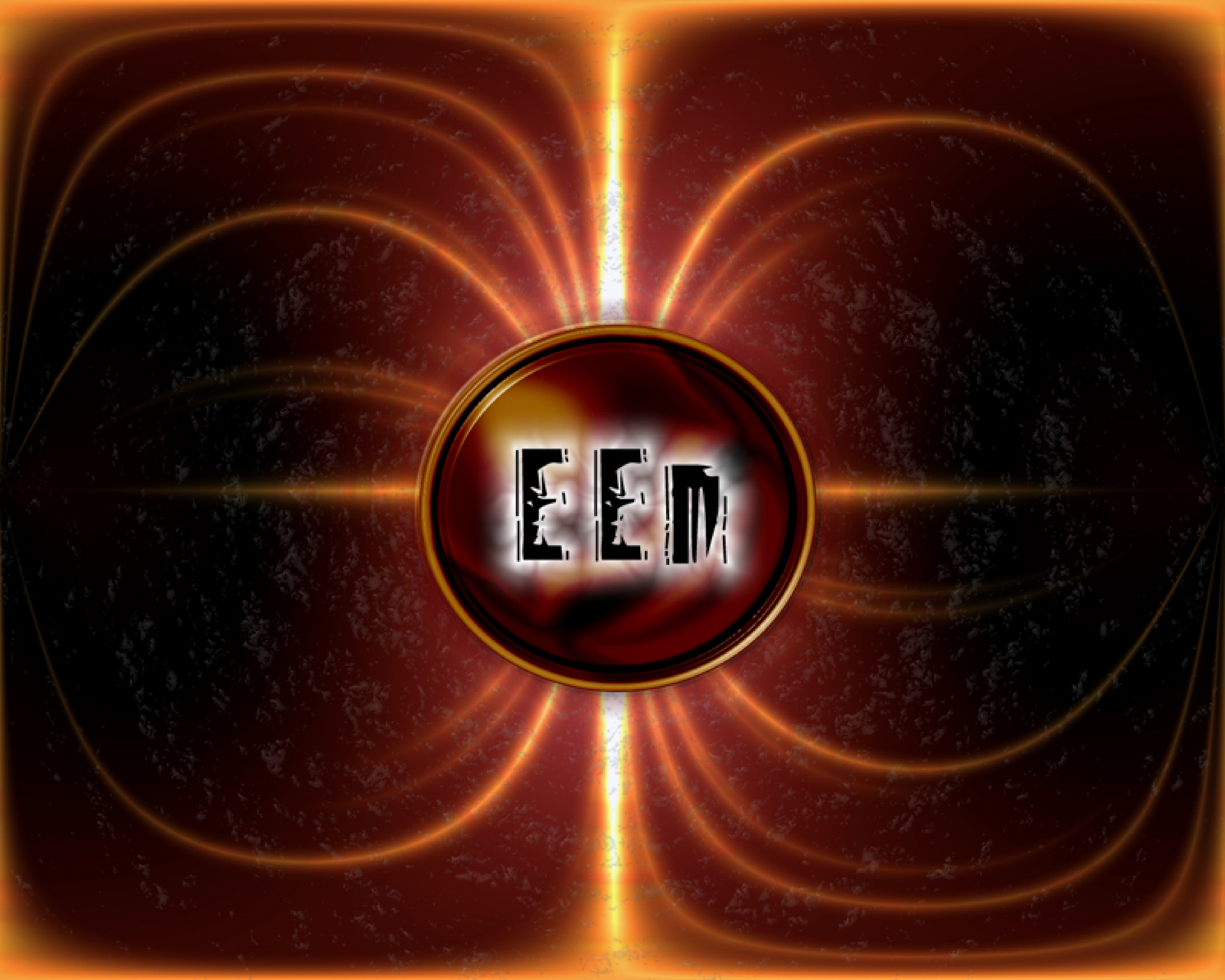
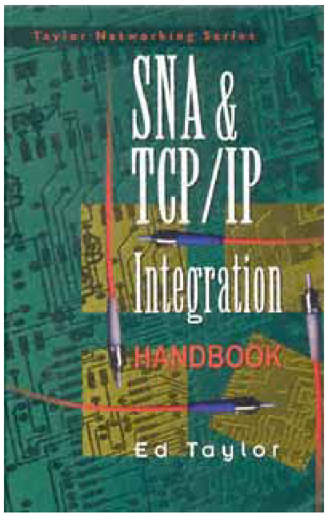

EEM


Taylor Networking Series
SNA & TCP/IP Integration HANDBOOK
Ed Taylor

About the Author

Ed Taylor is founder and chief network architect Information World, Inc, and a former network architect for IBM.

Some of Mr. Taylor's consulting experience includes work for NEC, Orange County, CA, BASF, Chrysler, Hewlett-Packard, Dow Jones, Ore-Ida Foods, Mutual of New York (MONY), and IBM Education.

SNA and TCP/IP Integration Handbook

Ed Taylor

McGraw-Hill
New York San Francisco
Washington, D.C.   Auckland
Bogotá
Caracas Lisbon London Madrid
Mexico City Milan
Montreal New Delhi San Juan
Singapore
Sydney Tokyo Toronto

Library of Congress Cataloging-in-Publication Data

Taylor, Ed, 1958–
SNA and TCP/IP integration handbook / Ed Taylor.
p. cm.
Includes index.
ISBN 0-07-063412-2
1. TCP/IP (Computer network protocol 2. SNA (Computer network
architecture) 3. Internetworking (Telecommunication) I. Title.
TK5105.585.T32 1998
004.6'2—dc21 98-34750
CIP

McGraw-Hill
A Division of The McGraw-Hill Companies

1 2 3 4 5 6 7 8 9 0 DOC/DOC 9 0 3 2 1 0 9 8

ISBN 0-07-063412-2

The sponsoring editor for this book was Steven Elliot, the editing supervisor was Ruth Mannino, and the production supervisor was Clare Stanley. It was set in Century Schoolbook by Priscilla Beer of McGraw-Hill's Desktop Composition unit.

Printed and bound by R. R. Donnelley & Sons Company.

To Jan Hoover
From Ed Taylor

Acknowledgments

I would like to thank the following for contributions to different aspects of this book.

MJH	Sony Corporation
IBM	Information World, Inc.
NetOptics	DHL Airborne
SCO	Emery Airfreight
Creative Labs	Federal Express
3ComUSRobotics	United Parcel Service (UPS)
SysKonnect	Roadway
Hubbell	United States Post Office
Liebert	McGraw-Hill:
Bud Industries	Steve Elliot
Tektronix	Ruth Mannino
Hewlett Packard	Clare Stanley
McAfee	Priscilla Beer
Altec Lansing	Cathy Hertz
Wagner Edstrom	Suzanne Rapcavage
Microsoft	Joe Rivellese

Preface

Purpose of This Book

I wrote this book based upon my experience. I included information that helped me, and I hope it does the same for you.

To work with both of these network protocols is a challenge. No single book can contain all the information one needs to know about them in every environment. The purpose of this book is to get you started in the right direction. Many good books have been published by McGraw-Hill on SNA and TCP/IP, respectively; I recommend you look into acquiring some of these to complement this one.

How to Use This Book

You can read this book from front to back. You can use it as a reference. It can be used to teach SNA and TCP/IP principles. I believe you'll find it most helpful approaching it from a topical perspective.

If you would like additional information, I can be reached through any of the following:

Information World, Inc. (IWI):	Edtaylor@info.com
Internet:	IWIinc@aol.com IWIinc@ibm.net IWIinc@msn.com Edtaylor@aol.com zac0002@ibm.net
AOL:	IWIinc Edtaylor
Compuserve:	72714,1417

1
Perspective on Systems Network Architecture

Systems Network Architecture (SNA) is a complex topic. If your background does not include experience in SNA, you will find this chapter most helpful; if it does, you will find this chapter particularly enlightening. I included background information here for the benefit of all readers. The blueprints included in this chapter are based on the terms, concepts, and architecture presented in the early part of this chapter.

SNA networks are built from hardware and software components. These networks vary in size, and different blueprints presented in this chapter bear this out. Before examining blueprints in this chapter, we will consider some reference material.

1.1 Hardware Architecture

Much development occurred at IBM during the 1940s and 1950s. History has shown that these two decades led to the creation of what has become known as the biggest gamble in the history of the IBM corporation: the System/360 (S/360) hardware architecture. Before exploring the S/360 architecture, we will briefly review hardware offerings preceding the S/360.

During the 1940s, 1950s, and even into the 1960s, IBM offered approximately six popular solid-state mainframe computers. However, a fundamental problem was lack of interchangeability among these systems. This meant that IBM programming, support, sales, and tech nical sales advisors all concentrated on their own areas of expertise and thus that there was seldom any overlap between systems. This was costly and became an increasing problem for IBM. Not only was this scenario a problem for IBM, but its customers had to contend with this if they had more than one type of machine to meet the needs throughout a given corporation. Following are some examples of these machines and their strengths:

• 604. This was an electronic calculating punch-card machine. It was first available in approximately 1948. The strength of this machine and its major selling points were speed (which was especially advantageous for the user), a pluggable circuitry, and its concentrated components in such a small physical location.

• 650. This machine initially became available around 1954 but was not announced until 1953. This was a magnetic drum storage machine whose primary strength was its general computing capability. This machine was extremely successful after its introduction into the marketplace.

• 701. This machine, announced in 1952, had faster input/output and higher processing speed than some of its predecessors and was especially powerful in scientific and related areas of computation. Thus it was not a general-purpose machine such as the 650, for example.

• 702. This large system, which focused on the ease of character handling, was announced in 1953 and made its debut in approximately 1955. Interestingly, the concept of this system originated in the late 1940s.

• 1401. This system was announced in 1959. It soon grasped a large market share after shipping began in 1960. It had increased speed, a fast printer, and other peripherals such as tape- and card-processing capabilities, which enhanced its popularity. Also its multiple components had selective capabilities, yet it was reasonably priced.

These systems, and others IBM offered during the 1940s through the early 1960s, proved IBM’s ability to meet a diversity of needs. However, such system diversity led to complexity in terms of one corporation attempting to maintain this posture in a technical environment. IBM was aware of the positive and negative contributions that these diverse systems offered.

After years of planning, designing, and reengineering, 1964 proved to be a pivotal point in the history of the IBM corporation. This was when the System/360 was introduced. This system was unique for multiple reasons, but at the heart of the S/360 was now one architecture capable of accommodating what previously was achieved by different systems. In addition, the S/360 architecture was available in different models, offering customers a wide selection for their initial purchase to meet their immediate needs with hardware that could be upgraded to accommodate future architectures as they appeared on the market. At this time, IBM is four architectures removed from the S/360, and, according to a personal account, a program originally written for the S/360 has been executed successfully on the S/390 architecture.

Technical highlights of each IBM hardware architecture are presented below.

1.2 System/360

The System/360 (S/360) was successful, to say the least. Some of the characteristics and functions of the components of this hardware architecture are listed below.

S/360 components included (1) central processing unit (CPU); (2) channels; (3) control unit(s); (4) peripheral devices such as terminals, printers, and tape and disk drives, as well as card punch and readers; and (5) main storage.

1. *CPU characteristics and functions.* The S/360 had a single-processor architecture, but models introduced later supported multiple processors. There were five classes of interrupts, with interrupt priority. There were 16 general-purpose 32-bit registers and four optional 64-bit floating-point registers, with 24-bit addressing. Selected models had dynamic address translation. Supervisor facilities included a timer, direct control capabilities, storage protection, and support for multisystem operation. There was ASCII and EBCDIC character-set support and channel-to-channel adapters were used for interconnection of multiple processors.

2. *Channel characteristics and functions.* The S/360 provided a data path to and from control units and devices and used a protocol for data transfer. Selector channels were used with tape and disk devices for high-speed data transfer, and one subchannel was used. Byte multiplexer channels interleaved input/output (I/O) operations. Slower operating devices were used with this type of channel, which could logically support up to 256 subchannels.

3. *Control unit*(*s*). The S/360 served as an interface between devices such as a terminal, card reader, card punch, and printer.

4. *Peripheral devices.* The S/360 terminals, printers, card punches, and card readers served as I/O devices. Terminals were used interactively, whereas punched cards were used in batch processing.

5. *Main storage.* Main storage in the early models emphasized speed and size. Virtual storage became available in later models.

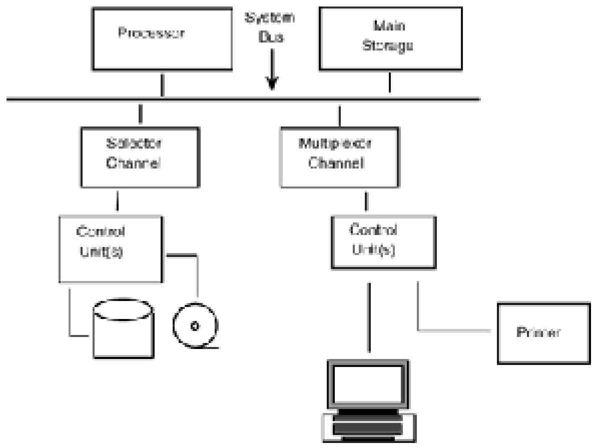


Figure 1-1
Logical view of an S/360.

Consider Fig. 1-1, which is a logical view of the S/360. Interestingly, in a general way, it is basic to current personal computer (PC) architecture. Granted, PC architecture has evolved to become more complex; this illustration does provide the rudiments of the PC's beginning. Figure 1-2 is a physical perspective of the S/360. Ironically, although more components have been inserted into various places, this view is the essence of PC systems today. Think about it. Systems today have a processor, main storage [random-access memory (RAM)], input and output (channels), and peripherals to control other devices. Some of the features and built-in functions of later S/360 models were carried over to the next hardware architectural generation, known as the S/370.

1.3 System/370

System/370 (S/370) architecture, announced in 1970, was the successor to the S/360. During the next 6 years IBM refined more than 14 models based on this architecture. In this section we examine S/370 features and functions on the basis of S/370 architecture as a whole (including its components and their characteristics and functions), not the inception of incremental enhancements.

S/370 components included (1) central processing units (CPUs), (2) channels, (3) control unit(s), (4) peripheral devices, and (5) main and virtual storage.

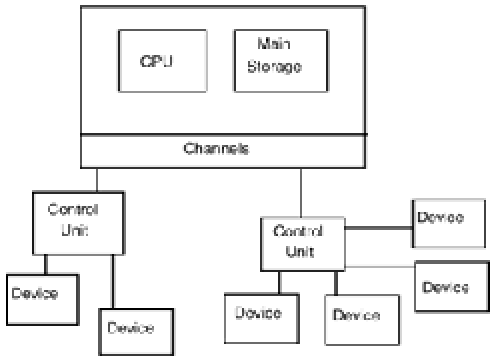


Figure 1-2
Physical view of an S/360.

1. *CPU characteristics and functions.* The S/370 inherited user program upgrade support from the S/360; it also had multiple processor support. It had six classes of interrupts, with interrupt priority. It had 16 general-purpose 32-bit registers and four additional 64-bit floating-point registers, with 24-bit addressing. It had a dynamic address translation (DAT) facility and an extended real addressing facility as an extension to DAT, making 64-MB (megabyte) addressability of real storage possible, plus a translation look-aside buffer to minimize DAT use. It also had a dual-address-space (DAS) facility, supporting semiprivileged programs. Supervisor facilities included a timer, direct control capabilities, storage protection, and support for multisystem operation. An optional vector facility was offered on selected models. The S/370 had Extended Binary-Coded Decimal Interchange Code (EBCDIC) character-set support, with removal of American Standard Code for Information Interchange (ASCII) support as in the S/360. Two page sizes—2 and 4 K (kilobytes)—and two segment sizes—64 K and 1 MB were available. One operating system was used for multiprocessing. Also, the S/370 had approximately 50 more instructions than the S/360.

2. *Channel characteristics and functions.* The S/370 supported three types of channels—selector, byte, and block multiplexer—as well as a 2-byte channel bus-width extension. Each channel had an associated set of subchannels. The S/370 used channel protocol for data transfer.

Data transfer rates of 1.5 and 3 MB could be achieved, depending on the channel bus width. The S/370 could suspend and resume facility for programmed control of channel program execution and remove 16-byte channel prefetching from S/360 channels.

3. *Control unit(s).* The S/370 served as an interface between devices such as a terminal, card reader, card punch, and printer.

4. *Peripheral devices.* The S/370's terminals, printers, card punches, and card readers served as I/O devices. Terminals were used interactively; punched cards were used in batch processing.

5. *Storage.* Main storage had addressability of up to 64 MB; virtual storage, up to 16 MB beyond that of main storage.

Figure 1-3 is a representation of implementation of the S/370 architecture which followed the S/360.

1.4 370/eXtended Architecture

The 370/eXtended architecture (370/XA), announced in 1981, followed the S/370 and what IBM called "S/370 compatibility realized in the 4300 series of systems." From 1981 until 1988 IBM implemented 370/XA hardware architecture in their mainframes. The focus here follows the same scheme as the prior two architectures: the features and functions of 370/XA as a whole.

370/XA components included (1) central processing units (CPUs); (2) a channel subsystem; (3) control unit(s); (4) peripheral devices; and (5) absolute, real, and virtual (addressing types) storage.

1. *CPU characteristics and functions.* The 370/XA had two addressing modes of operation—24- and 31-bit—and 2-gigabyte (GB) addressability with 31-bit addressing mode. This included 13 new instructions; multiple-processor support; six classes of interrupts, with interrupt priority; and 16 general-purpose 32-bit registers and four 64-bit floating-point registers, with 24-bit addressing. The 370/XA also had a dynamic address translation (DAT) facility, with extended real addressing as an extension to DAT, permitting 64-MB real storage addressability, and a translation look-aside buffer to minimize DAT use. A dual-address-space (DAS) facility, supporting semiprivileged programs, was also included. Supervisor facilities included a timer, direct control capabilities, storage protection, and support for multisystem operation. An optional vector facility was offered on selected models. The 370/XA had EBCDIC character-set support and dynamic I/O reconnect. Two page sizes—2 and 4 K—and two segment sizes—64 K and 1 MB—were available. Multiprocessing was performed with one operating system. Approximately 50 new instructions had been added since the launching of the S/370.

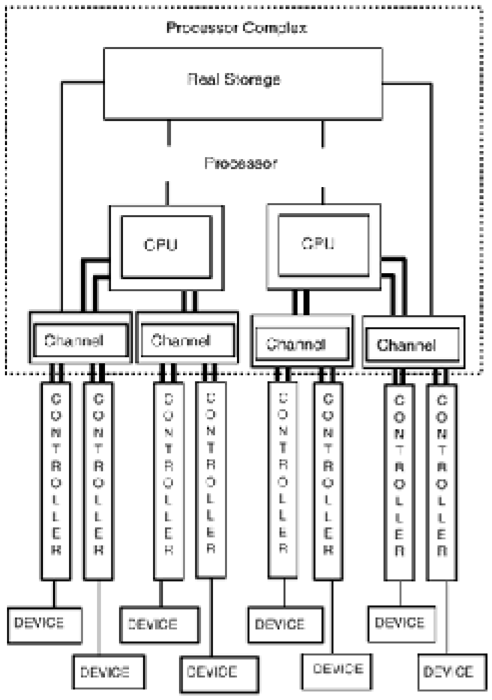


Figure 1-3
A conceptual view of an S/370.

2. *Channel subsystem characteristics and functions.* The 370/XA channel subsystem was simply a processor that served as an interface between I/O devices and processors; it performed preprocessing on data between I/O devices and processors. A *channel path* in 370/XA referred to the physical path between the channel subsystem and a device. Subchannel numbers had a one-to-one relationship with an I/O device, and there was path-independent addressing for I/O devices. The implementation of paths enabled dynamic data routing from I/O device to processor. A *channel path identifier* (CHPID) in 370/XA was associated with devices such as control units. Path management was performed by the channel subsystem. Increased Channel Command Word (CCW) support was included for direct use of 31-bit addressing in channel programs. Thirteen I/O instructions had been added (since the S/370), and two types of channels were supported: byte and block multiplexer. Subchannels were not owned by the channel as in the S/370. The 370/XA used channel protocol for data transfer with data transfer rates of 1.5 and 3 MB, depending on the channel bus width. The 370/XA could suspend and resume facility for programmed control of channel program execution and could remove 16-byte channel prefetching. Figure 1-4 is a flowchart of a channel subsystem.

3. *Control unit(s)*. These served as an interface between devices such as a terminal, card reader, card punch, and printer.

4. *Peripheral devices.* These included terminals, printers, card punches, and card readers, all of which served as I/O devices. Terminals were used interactively; punched cards were used in batch processing.

5. *Storage.* There were three types of storage (differentiated by addressing scheme): absolute, real, and virtual. In *absolute storage,* which was an address in main storage, no transformations were performed on the contents and 2 GB of storage were possible. In *real storage*—which was also an address in main storage, multiple processors accessed the same main storage. A CPU prefix number distinguished processors to maintain storage order. In *virtual storage,* the address reflected an abstract location. Virtual storage does not exist in reality; rather, it is a concept, achieved by main storage, secondary storage, and processor speed as the fundamental components that make virtual storage possible.

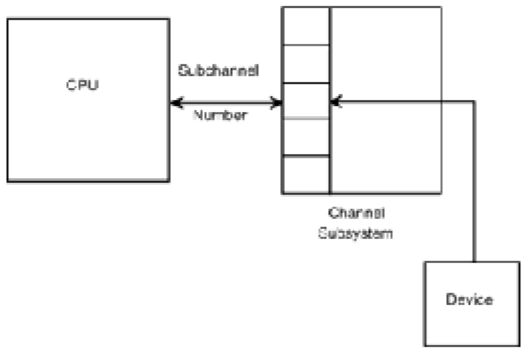


Figure 1-4
370/XA channel subsystem.

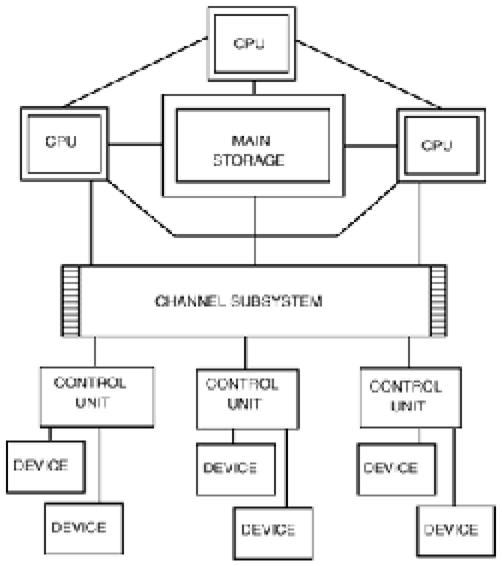


Figure 1-5
370/XA architecture.

The architectural advancement shown in Fig. 1-5 brought about a more segregated and separate function of system component functions.

Other functionality was added to 370/XA until the announcement of Enterprise System Architecture; however the highlights mentioned above constitute the bulk of additions to the XA architecture.

1.5 Enterprise System Architecture/370

Enterprise System Architecture (ESA/370), announced in 1988, succeeded the 370/XA. This architecture built on advances made in 370/XA and S/370. Consequently, different hardware-related functions were implemented. However, from a systems standpoint, considerable advances were made, including a new operating system, as we will see in a later section. Highlights of ESA/370 included addressing, storage, and machine-dependent support.

1. *Addressing.* Key enhancements in ESA/370 included 16 new access registers, which provided the hardware capability for a program to address up to 16 spaces. A *home address space* was a translation mode that permitted the control program to gain control quickly where principal control blocks were maintained. A *private space*, designed to enhance security functions, was also supported in the ESA/370 to prevent use of the translation look-aside buffer for common segments.

2. *Storage.* A major improvement in the ESA/370 was its storage-handling ability. The storage management subsystem was designed to stage data. An I/O boundary existed, and multiple places were available for storing data. With a storage subsystem, the framework was in place for a particular processor complex to capitalize on this feature. (See Fig. 1-6.) This storage hierarchy is quite significant. Today, PCs typically have a similar arrangement. Even the implementation of cache to the processor and RAM is now being phased in to a greater degree with each new model brought to the marketplace. In addition, working with the storage hierarchy in the ESA/370 was the *system control element* (SCE), which routed data through the CPU's main and expanded storage, as well as the channels. The SCE kept track of changes made to data and was the key component for moving data throughout the hierarchy. Notice the centralized connection of the SCE and the other system components in Fig. 1-7.

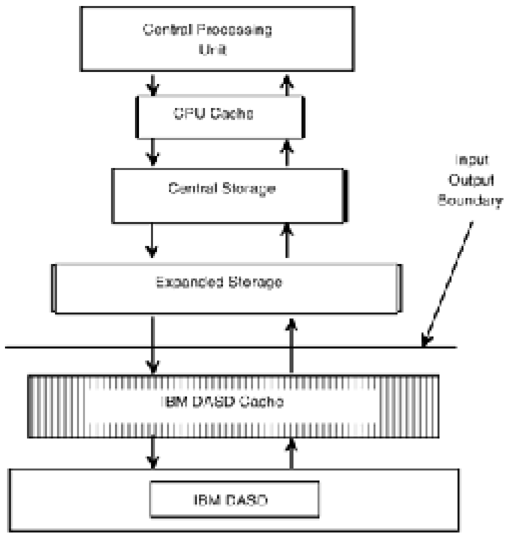


Figure 1-6
IBM's storage hierarchy.

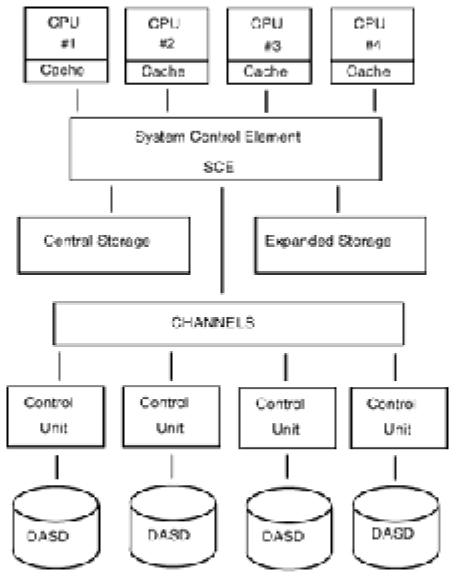


Figure 1-7
SCE logical view.

3. *Machine-dependent support.* A feature known as the *logical partitioned mode* (commonly known by its acronym, LPAR) was a machine-dependent function, tied to hardware and a software component known as the *Processor Resource/Systems Manager* (PR/SM, pronounced *prism*). To function in LPAR mode, the machine had to be supported by that system and selected on power-up time. LPAR permitted a system to run four logical partitions (see Fig. 1-8), each running an operating system simultaneously and all independent of one another. The logical partitioning of a processor included the processor's resources such as storage, channels, and the processor itself. Isolation of the LPARs was enforced via hardware. PR/SM itself was an IBM-supplied option that some machines could take advantage of to offer LPAR. PR/SM was implemented in *microcode,* which is IBM's synonym for *firmware.*

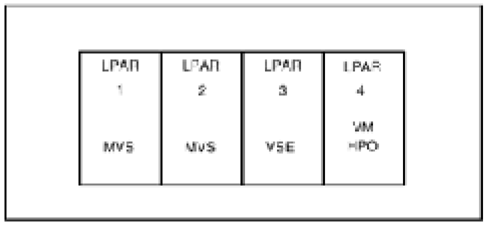


Figure 1-8
A logical view of LPAR.

Although there were few physical differences between the ESA/370 and 370/XA architecture, the former offered storage, addressing, and machine-dependent enhancements; Fig. 1-9 depicts ESA/370 with two processors.

1.6 System/390

System/390 (S/390), also known as *Enterprise System/390,* was announced in September 1990. This announcement was broad in scope, encompassing a new processor line, channel subsystem, many software announcements, and wide-sweeping networking-related support. A brief view of the hardware highlights includes S/390 enhancements, ES/9000 processors, Enterprise Systems Connection (ESCON) Architecture, and networking support.

S/390 enhancements included *storage override protection*, which provided reliability of executing programs by keeping a different applica tion from executing simultaneously within that same address space; *program event recording* (PER) *Facility 2*, which provided a more focused method of event control when compared to PER 1; and *Access List Control*, which permitted different users different functionality within the same address space.

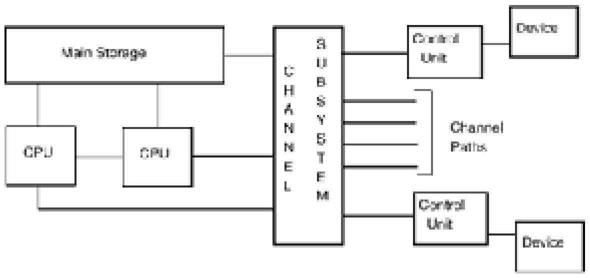


Figure 1-9
ESA/370 architecture.

S/390 built on the framework of ESA/370. Some documents refer to S/390 and ESA/390 simultaneously. IBM documentation uses ESA/390 to refer to those environments which include one or more of the following:

• Enterprise Systems Connection (ESCON) Architecture

• Common Cryptographic Architecture

• An environment providing data spaces for Virtual Memory.

1.7 Summary

Since the 1950s, IBM has brought five hardware architectures to market; they were described in this chapter. Although IBM has additional architectures implemented in different offerings such as the AS/400 and RISC/6000, the architectures presented here were those architectures associated primarily with what were considered *mainframes.*

Any way these architectures are evaluated, it is easy to conclude that the advances made by IBM in the latter half of the twentieth century surpassed the progress made earlier.

2
SNA Hardware Components

SNA is a collection of hardware and software components integrated together to create a functional network. The amount and type of hardware and software installed determine the functionality of an SNA network. This chapter presents the popular hardware used in SNA networks in the past as well as the present.

2.1 Processors

IBM processors are grouped into what IBM calls *series.* IBM has different series that are based on different architectures, such as those mentioned in Chap 1. For example, the ES/9000 series is based on ESA/390 or S/390 architecture. Some processor series IBM has and has offered include ES/9000, 3090, 4300, 303X, 308X, and 9370.

Processor series have models. For example, the announcement of the ES/9000 included 18 models. Some of these models included

120	210	440	720
130	260	480	820
150	320	500	900
170	330	580	
190	340	620	

Each of these models has different processing support levels and other differences such as water-cooled or air-cooled processor capability. Consider Fig. 2-1.

Some of the processor series IBM has offered provide different functionalities. For example, some models in the 9370 series support direct Ethernet network attachment.

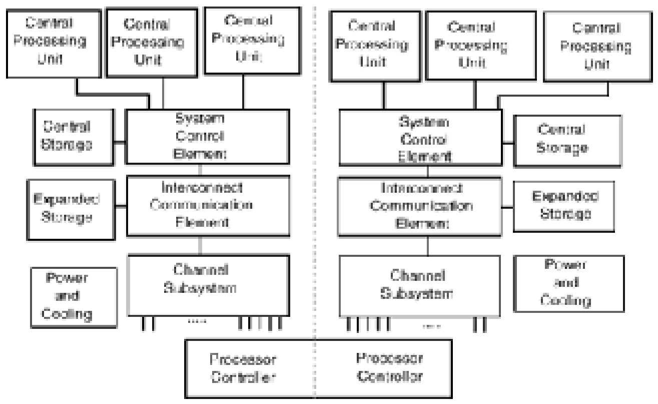


Figure 2-1
A conceptual view of a partitioned ES/9000.

Some 308X series have model numbers that indicate single- or dual-processor capabilities. In general, the model number of a particular series indicates a significant amount of information about the processor. Understanding this numbering scheme helps break the number barrier for those new to IBM equipment and environments.

2.2 Channels

The saying in IBM circles is that all data inbound to a processor must go through a channel to get to the processor. So far, this author has found that to be the case. Channels, a channel subsystem, and channel paths are deeply rooted in the hardware architecture dictating how data are manipulated at the lowest layers in a system. ESCON is IBM's serial fiber channel, and is discussed in greater detail later in this chapter; however, for now, consider Fig. 2-2.

Channels used prior to ESCON used heavy copper-stranded cables called *bus* and *tag* cables. The distance in which parallel channels (those prior to ESCON) could operate was approximately 200 ft if devices were daisy chained. A straight run of 400 ft might be obtained under ideal conditions. The bus cable contains signal lines used to transport data. Tag cables control data traffic on the bus.

Prior to ESCON, IBM offered three types of channels that moved data in parallel from a source to a destination point:

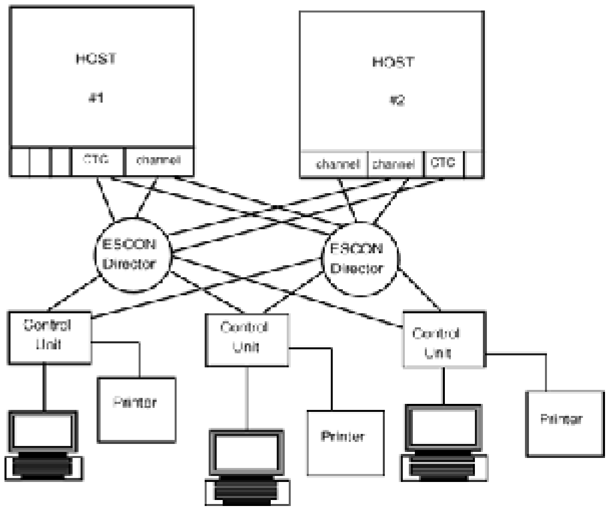


Figure 2-2
ESCON configuration.

1. *Selector channel.* This channel had only one subchannel and could accommodate only one data transfer at a time. However, multiple devices could be connected to this subchannel such as tape and disk devices. The selector channel was intended for high-speed devices only. Once a logical connection was established between a device and the channel, no interruptions occurred for the duration of the data transfer. The selector channel was developed in the early days of IBM channels. They are outdated today and were used primarily in the 1960s and to a reduced degree in the early 1970s, when support for selector channels was discontinued. Figure 2-3 shows a selector channel and devices.

2. *Byte multiplexer channel.* This was a follow-on to the selector channel. Unlike the selector channel, this channel was intended for low-speed devices. One channel could address up to 256 subchannels. The subchannels operated in *burst mode,* meaning that once a logical connection was established between a device and the channel, the data were pushed to the channel. After release, the next data transfer could occur, permitting interleaving of data at a byte level. Since these channels were designed to operate with slow-speed devices, this operation works quite well, exploiting bandwidth and utilizing the available resources. (See Fig. 2-4 for an overall view of the byte multiplexer channel.)

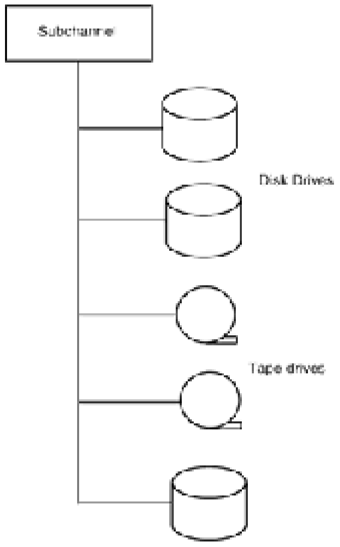


Figure 2-3
Selector channel.

3. *Block multiplexer channel.* This was another follow-on to the selector channel and was introduced with the S/370 hardware architecture. This type of channel had the ability to record the address, byte count, status, and control information for an I/O operation and thus could perform a disconnect from a device if no data were being transferred. This meant that high-speed devices could have overlapping operations. In this sense a greater utilization of resources is realized. Figure 2-5 depicts an overall view of the block multiplexer channel system and shows how data are interleaved as they are passed to the channel.

Another type of channel supported by IBM now is the ESCON channel. Simply stated, it is a fiber-optic data path. Its channel protocols are different from those of the three channels described above. ESCON is referred to as a *serial channel,* whereas the other three channels have been renamed *parallel channels.*

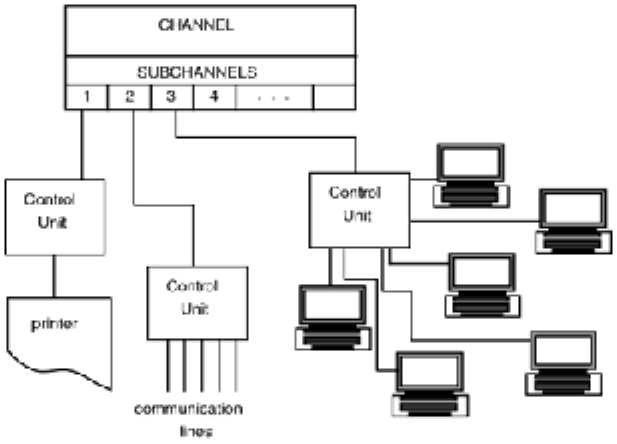


Figure 2-4
A byte multiplexer channel, with its subchannels and devices.

ESCON channels support greater physical cable length because photons, not electrons, are moved; hence no voltage drop is realized. ESCON channels also support dynamic connectivity with the use of ESCON directors. Additionally, ESCON extenders can be used to move the distance in which these cables can operate to approximately 43 km. Figure 2-6 depicts two processors, an ESCON director, ESCON and parallel channels, and multiple control units.

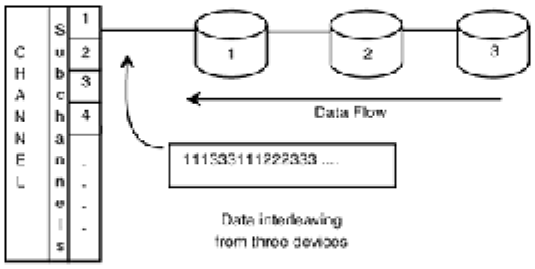


Figure 2-5
Data flow through a block multiplexer channel.

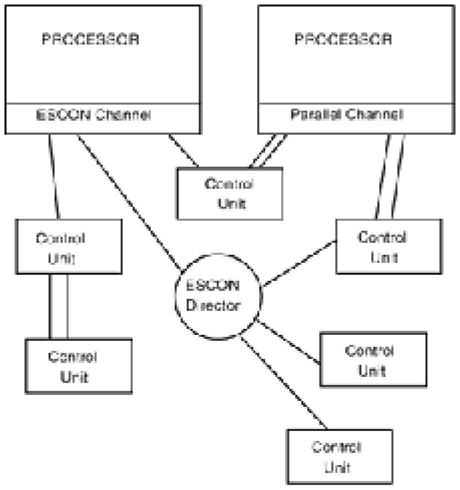


Figure 2-6
Serial and parallel channels.

2.3 Communication Controller

This device, also known as a *front-end processor* (FEP), houses the *network control program* (NCP), a component that we will discuss shortly. The communication controller is available in different sizes and models that dictate the abilities and limitations of the controllers. Communication controllers perform multiple tasks, including routing, flow control, and the point at which communication lines connect and where certain specialized programs operate, allowing non-SNA equipment to access an SNA network. The communication controller system is shown in Fig. 2-7.

This is only one example of how a communication controller is implemented. Later in this chapter the implementation of this device is shown in different scenarios.

2.4 Cluster Controller

The cluster controller was the forerunner of the establishment controller, explained next. In SNA environments, cluster controllers were used to attach terminals and printers. The different cluster controller models dictated how many devices could be attached to the controller. In SNA drawings where a cluster controller is used, it is represented typically as shown in Fig. 2-8.

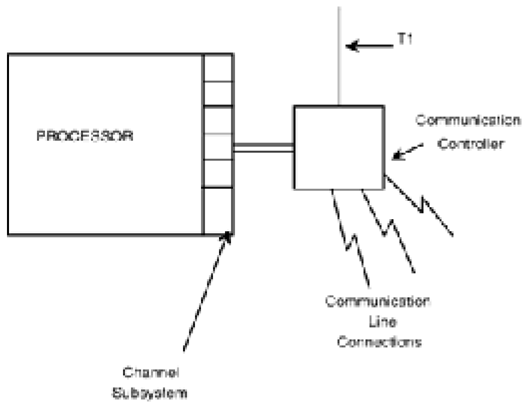


Figure 2-7
Processors, channels, and communication controller.

The cluster controller is known in SNA environments as a *3274 control unit.* The 3274 control unit family has different models, including the 1A, 21A, 31, and 41.

Depending on the model, two possible modes of operation are possible: local and remote.

There are three methods of identifying the operational method of a particular control unit:

1. An *A* indicator means that the control unit is functioning as a channel-attached local device.

2. The *B* and *D* designators indicate that the control units are channel-attached using the processor channel program.

3. The *C* designator indicates that the control unit is operating as a remote unit with SDLC or BSC data-link layer protocols.

Figure 2-8
Cluster controller.

The 3274 is still in use today even though it has been replaced by the 3174. Both work well together, and, depending on the need, justification for a 3274 may be as credible as that for a 3174.

2.5 Establishment Controller

This device succeeded the cluster controller. It provides services offered by the cluster controller and more such as those used in networking. The establishment controller appears as illustrated in Fig. 2-9.

The establishment controller has numerous models in its family. Certain models are capable of performing functions that others cannot. A brief list of those controllers that have been brought to market includes:

1L	11R	21R	52R
1R	12L	22L	61R
2R	12R	22R	62R
3R	13R	23R	63R

11L   21L   51R

Some characteristics and functions that the establishment controller offers are Token Ring support and Token Ring gateway support. Multicast support via the concurrent communication adapter and single-link configuration includes ISDN support, PU2.0 support, PU2.1 support, APPN support, control unit terminal (CUT) support, distributed function terminal (DFT) support, synchronous data-link control (SDLC) support, X.21 and X.25 support, parallel-channel support, ESCON support, binary synchronous communication (BSC) support, 3270 data-stream support, printer support, response-time monitor (RTM) support, Common Management Information Protocol (CMIP) support, Generic Alert support, and T2.1 Channel Command support.

The Advanced Peer-to-Peer Networking (APPN) support offered by the establishment controller is a considerable enhancement over the 3274 cluster controllers, which did not have this support. This device covers a broader support than its predecessor, and is positioned to fit into either SNA or APPN networks or both.

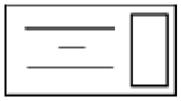


Figure 2-9
Establishment controller.

2.6 Interconnect Controller

This device, which succeeded the cluster controller, provides the services of the cluster controller and more, such as those used in networking. The interconnect controller conceptually appears as shown in Fig. 2-10.

The interconnect controller is known numerically as the 3172. Three models have thus far been introduced to the market: the 001, 002, and 003. The forte of the 003 model is its versatility. The model 003 supports the TCP/IP offload function offered by IBM, which enables a customer to purchase TCP/IP to run as a VTAM application but select the applications and functions desired and offload TCP, UDP, and IP to the 3172 model 003. In turn, only the desired portion of TCP/IP resides as an active application under VTAM. The result is conservation of resources from a processor standpoint.

The offload function of TCP/IP to the model 003 means that protocol conversion of TCP and/or UDP is performed on the 3172—not the processor. Additionally, this means that IP performs routing functions on the 3172. Again the benefit is no need for work on the processor for this function.

The 3172 model 003 communicates effectively with a processor through what IBM calls *Common Link Access to Workstation* (CLAW) protocol. The 3172 003 can achieve this because it implements a CLAW driver to do so. The benefit to this is how the subchannel is utilized with regard to data transfer rate.

Other benefits of the 3172 model 003 is its support for data-link layer protocols. This model supports FDDI, Ethernet version 2, 802.3, Token Ring, and channel protocol (to the processor).

Another powerful support aspect of the 3172 is its ability to support not only NetView but also Simple Network Management Protocol (SNMP). It also provides a system log facility via the Interconnect Control Program (ICP) which can be used for debugging if necessary.

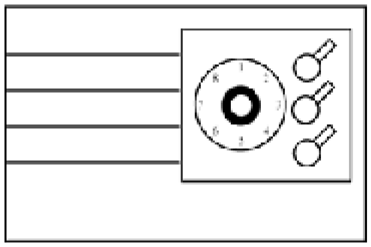


Figure 2-10
Interconnect controller.

Since its announcement, the 3172 has undergone multiple changes. Some functions available now did not exist at the time of its inception. Other functions have been discontinued because of market conditions.

2.7 Direct-Access Storage Device

Direct-access storage device (DASD) is IBM’s term for a disk drive. Significant here are the five deliverable DASD offerings. Many IBM DASD units are still in use in companies and corporations around the world. For the sake of information, the focus here is on these different offerings and their fundamental significance.

The first DASD devices could be characterized as removable media, as their “platters”—those layers of the drive in which data are stored—could be removed. The second DASD devices to appear had more “intelligence” than their predecessors did. They had removable platters also, but a great improvement was the increased storage capacity of some models in these offerings. Third came the DASD offering which brought significant improvement of storage in terms of density and also better diagnostic capabilities. Significant performance was also achieved. The fourth category of DASD offerings from IBM dominated the 1980s. A resounding word recurring about this category of DASD is reliability. Performance, speed, and flexibility in regard to implementation also characterized this category.

The fifth category of IBM DASD was developed in the 1990s. Many in the technical community, including IBM, consider it the DASD architectural foundation for this decade and into the twenty-first century. Strengths of this category include a robust number of models from which to select. Software support used in processors supporting the IBM storage hierarchy has also caught up significantly. Consequently, leveraging advanced hardware along with supporting software brings synergy to the storage subsystem offered in this current category.

Figure 2-11 depicts how reference to DASD is generally made in SNA. Most references are not to a specific DASD model, unless an in-depth discussion of the topic at hand is required.

Figure 2-11
Direct-access storage device.

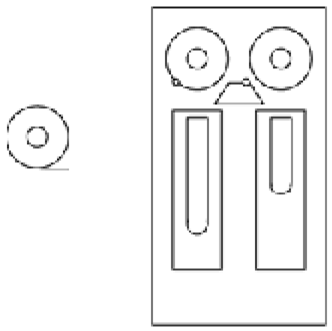


Figure 2-12
Type drive.

2.8 Tape Devices

IBM has two basic groups of tape devices: reel-based and cartridge-based. In many IBM shops tape is a method of backup for data that may be archived. Figure 2-12 shows the general symbols used when reference is made to a disk drive.

2.9 Printers

IBM has different types of printers, which can be categorized by speed and type of technology. Figure 2-13 shows a generic example of a printer. Instances of reference to a printer in this chapter include this representation. Most of IBM's documentation uses this representation for printers. In most cases this symbol suffices to convey the information being discussed.

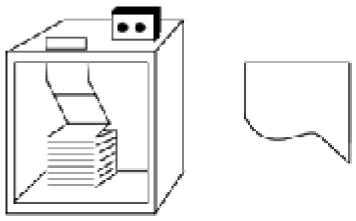


Figure 2-13
Printer.

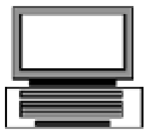


Figure 2-14
Terminal.

2.10 Terminals

Different types of terminals are used in SNA, but most have one commonality: use of the 3270 data stream. This will be discussed in further detail later, but it is important to note here the general nature of IBM terminals used in SNA. There are two categories of terminals: those which do and those which do not support graphics.

Terminals can also be classified according to how many columns and rows they support. For example, in the 3278 terminal family, the basic difference between the four terminal types—models 2 through 5—is how many columns and rows are supported.

Other terminal types are the 3179G, which supports graphic applications, and the 3279, which is a later version of the 3278 terminal.

Terminals will be discussed later in sections on SNA issues such as the type data stream they support. References in this book to a 3270 terminal appear as shown in Fig. 2-14.

2.11 Summary

Categorization of IBM hardware is fairly logical. This chapter presents most of the hardware associated with SNA. Actual implementation of this hardware and IBM's software makes up SNA as it is known in the marketplace. IBM hardware has a unique design—it is designed for functionality. Each piece of equipment performs one or more identifiable function; however, IBM hardware is also designed as an integral component on which the SNA is based.

3
IBM Operating Systems

IBM has many different operating systems. However, this chapter focuses on those three operating systems which work with the larger-scale systems and predominate in terms of market share and presence in the SNA networking environment: Multiple Virtual Storage, Virtual Machine, and Virtual Storage Extended. These three systems are at different versions and releases and reflect the underlying hardware architecture on which they operate. The basic characteristics and functions of these operating systems, as well as the software subsystems (which have considerable market share) that operate under their control, and other pertinent information that seems to be agreed on in the industry are discussed here. In this chapter, as we did in Chaps. 1 and 2, we will attempt to understand the basics of some of the SNA-centered terminology and concepts.

3.1 Multiple Virtual Storage

Multiple Virtual Storage (MVS) originated in the first versions of Operating System/360 (OS/360) designed in the early 1960s. It is generally considered a production-oriented operating system. From a historical viewpoint, this operating system has evolved through many versions and releases, but for our purposes, suffice it to say that MVS's immediate predecessor was Multiprogramming with a Variable (number of) Tasks (OS/MVT).

Many books have been written about MVS by IBM and others not affiliated with the IBM corporation. The intent of this section is to focus on the highlights of MVS. Not all of its features and functions are present ed, just those which this author believes have made significant impact through the versions and releases that MVS has realized thus far.

MVS operated with S/370 hardware architecture and followed two earlier versions of operating systems: OS/VS1 and OS/VS2. MVS could address up to 16 MB of virtual storage. It utilized program areas in the form of names that referred to an *address space*, which is an identifiable amount of memory, or address-storing capacity, that a program can use. Tied to this was the concept of an *address-space identifier*, which identified a particular address space. *Data areas*—known more precisely as *common data areas*—provided capabilities for user program messaging, communication with the operating supervisor, and operating system-program interaction. *I/O buffers* were also utilized to pass data from a telecommunication subsystems to an address space where that data would be moved for operations to be performed.

A major change for MVS in 1981 was related to addressability. The significance of *Multiple Virtual Storage/eXtended Architecture* (MVS/XA) was that it broke the 16-MB boundary and virtual storage addressability was supported up to 2 GB; numerically, this is 2,174,484,684 bytes!

MVS/XA also provided each user with the perception of having a unique address space and could distinguish between programs and user data in each address space. Cross-memory services brought the ability for a user to access other address spaces as necessary.

The concept of task management in MVS/XA meant piecework operation, in which jobs were divided into smaller pieces and each piece "task" was processed as efficiently as possible. This process was controlled by the supervisor.

MVS/XA supported both 24- and 31-bit addressing. Along with MVS/XA came changes in the I/O facilities. The largest change with MVS/XA was the I/O subsystem, which handles I/O operations independently of the processor. This was followed by other changes.

First, the channel, not the operating system, handled channel path selection. Additionally, dynamic reconnection was added to the I/O portion to support dynamic path selection. Another enhancement to I/O facilities that fit into MVS/XA was the increase in the number of supported devices to 4,096; however, a contingent factor was dependence on the I/O configuration program. Support for up to 256 channel paths with 8 paths per device was also supported under MVS/XA.

MVS/XA introduced three types of tracing: address space, branch, and explicit software. The generalized trace facility (GTF) was changed to support 31-bit addressing. Parallel to this was the improved DUMP facility.

MVS/XA controlled work or managed a resource by identifying a control block. Three types of control blocks were associated with MVS/XA: *resource,* which represented a DASD or processor, for example; *system*, which contained systemwide information; and *task*, which represented one unit of work.

The *System Resource Manager* (SRM) is used in MVS/XA to make decisions concerning where an address space should remain, that is, in real storage or DASD; these address spaces represent “resources” in a MVS/XA environment.

In 1988, *Multiple Virtual Storage/Enterprise Systems Architecture* (MVS/ESA) was introduced. Its virtual storage addressability was (and remains) 16 terabytes (TB). New features and functions made this software and architecture the most vigorous to date.

With MVS/ESA *data spaces*—which contain data only, no programs—became available, further extending the abilities of virtual storage. Since data spaces were available, they had to be managed. A data-space manager keeps track of those in use in the system and manages them accordingly. A data space may be assigned to one program only or shared.

Related to data space is the concept of *cyberspace,* which is data in an address space that has a special classification. It is data that is processed in an address space, but typically uses expanded storage or migrates to auxiliary storage.

An *access list,* which is responsible for determining which data spaces a program is authorized to use, is also part of MVS/ESA. This facility is also used by hardware to exploit it for use with the segment table descriptor.

Also included in MVS is the *linkage stack facility,* which is used to retain information about the state a program is in during execution when a call is sent to another program.

The *virtual look-aside facility* (VLF) was designed to reduce the time for *partitioned data set* (PDS) searches and reads. It can also perform multiple reads against the same object. VLF operates in its own address space and assigns names to objects it manages. The naming convention uses three levels: class, major, and minor. This object naming system is used by the VLF to increase retrieval.

Advanced Program-to-Program Communication (APPC) support under MVS enables peer connectivity outside the MVS environment; examples are AIX for the RISC/6000, S/38, OS/400, VM/ESA, VSE, and OS/2.

APPC under MVS by APPC/MVS applications, Time Sharing Option/ Extension (TSO/E) operation, the IBM Information Management System (IMS), and even APPC batch jobs are shown in Fig. 3-1.

Other examination functions in MVS/ESA include sysplex, the cross-system coupling facility, the automatic reconfiguration facility, and hardware configuration definition.

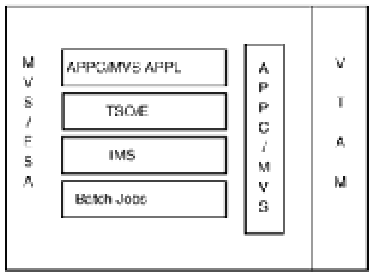


Figure 3-1
Conceptual view of APPC/MVS.

In *sysplex* one or more MVS systems are linked via hardware and software services. In this scenario they are treated as a single complex and initialized as such. To achieve this sysplex concept, an external clock, called the *sysplex timer*, must be used to synchronize the systems brought together. Its purpose is to synchronize the time across the entire central processing complex.

The *cross-system coupling facility* (XCF) is software providing control of members and groups, intercommunication among members, and monitoring of members. Explained another way, it provides the services necessary for programs in a multisystem environment to communicate successfully.

The *automatic reconfiguration facility* (ARF) can be used in a single- or multiprocessor configuration environment. In either case it serves to redistribute a workload in the event of failure without operator intervention.

The *hardware configuration definition* (HCD) facility permits communication between the channel subsystem and I/O definitions at the same time. Prior to MVS version 4 this process was a two-step function: (1) defining the I/O subsystem via the I/O Control Program (IOCP) and (2) defining software by the MVS Control Program (MVSCP). Now, with HCD the two are combined.

Other MVS functions are too numerous to list here. For further information about MVS, contact your local IBM representative.

3.2 Virtual Machine

Virtual Machine/Enterprise Systems Architecture (VM/ESA) combines the VM/System Product (VM/SP), VM/SP High-Performance Option (VM/SP HPO), and VM/SP XA. VM/ESA facilities include APPC/VM VTAM support (AVS), VM/ESA facility that converts APPC/VM into APPC/VTAM protocol for communication throughout an SNA network; the *group control system* (GCS), a facility that manages subsystems that permit VM/ESA to interact within an SNA environment, the *interactive problem control system* (IPCS), a VM/ESA component that provides an operator with online capability for problem diagnosis and management; and the *transparent services access facility* (TSAF), a VM/ESA component that permits communication between programs by name specification instead of user or node ID.

The forte of VM/ESA is its ability to run multiple operating systems on the same processor. This concept is known as *multiple preferred guests* (MPG) (Fig. 3-2).

The concept of multiple preferred guests is that totally different operating systems can be executing simultaneously and one MPG fails but does not cause any other MPG to fail. This type of environment can be advantageous if a production operating system is needed such as MVS/ESA and a development-oriented operating system such as VM/ESA is needed. Both of these can execute under the control of a VM/ESA operating system.

As did MVS, VM originated in the 1970s. Technically, VM dates back to 1965 with what is known as *Control Program 40* (CP-40). By the late 1960s evolution of Control Program 40 was shaping up to what would become known as VM/370. VM/370 was made available in 1972 according to the *IBM System Journal* Volume 18, Number 1, 1979.

A closer view of VM/ESA reveals four components, which are the focus of the remainder of this section: the control program (CP), the conversational monitoring system (CMS), the group control system (GCS), and the interuser communication vehicle (IUCV).

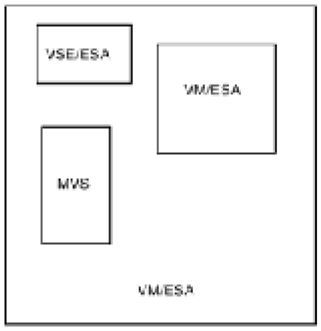


Figure 3-2
Conceptual view of multiple preferred guests.

The CP is used to manage the actual hardware. It communicates with resources such as the real processor, I/O subsystem, and storage and is responsible for management of these resources. It is the CP that utilizes the physical resources to create the logical (virtual machines). CP exploits the hardware architecture mentioned previously. In the case of S/390 architecture, many of the resources available with the MVS operating system are available through the platform where CP is in control. CP has a set of commands that permit the manipulation of resources, both physical and logical. Some of the commands can be used immediately after logon to perform system management functions.

Some CP aspects should be noted here. First, the CP is software. It is transferred to real storage when the system is booted. One way CP works is by defining internal objects by what is called a *control block.* An example of an important control block is the *VM definition block* (VMDBK), which represents a logged-on virtual machine; it is created by the CP when one logs onto VM. A trace table exists in VM, and a CP trace table maintains any events related to system crashes and other problems. The CP knows the real I/O configuration and correlates real I/O device capabilities to virtual devices. CP with VM/ESA architecture also supports ESCON I/O architecture.

The *conversational monitoring system* (CMS), another VM/ESA component, operates with the CP but is used as a two-way communication processor between users. IBM has a large manual of valid CMS commands that can be used to perform a number of functions. CMS is an operating system for VM users.

CMS communication can occur between users and CMS. Users can enter commands to CMS, and CMS can issue messages to users. CMS permits a number of user functions; for example, it permits communication with other users; permits users to create, test, and debug programs; controls workflow in a VM/ESA system; and permits file sharing.

The concept of file sharing under CMS is achieved via the *shared file system* (SFS), which is an extension of CMS and provides file management capabilities. According to IBM's *CMS Shared File System Primer*, document number GG24-3709, SFS features include the following:

• A hierarchical file system.

• Files can be stored in pools.

• User space can be assigned to a spool file.

• A file can be located in more than one directory.

• Files and directories can be shared with users on other systems

• File and directory locks ensure data integrity when multiple users are involved with the same file and/or directory.

• Users can have concurrent access to files and directories.

• Files in a file space are stored in a directory.

CMS also has a facility known as *XEDIT*, a full-screen editor that permits users the ability to create, edit, and manipulate files. It runs under the control of CMS. Some functions capable under XEDIT include CMS file creation, file editing, joining existing files, performing searches in files for specific data, creating XEDIT macros, performing a sort function on data in a file, and providing help for users.

The *group control system* (GCS), a VM component shipped by IBM with each VM/ESA system, manages subsystems that permit interoperation with IBM's SNA. Actually, GCS is a supervisor. IBM's Virtual Telecommunication Access Method (VTAM) runs under a GCS group which, in turn, has a supervisor. VTAM is used for communication between programs in a VM/ESA machine or complex and devices outside it in an SNA network.

The *interuser communication vehicle* (IUCV) facilitates communication between programs running in two different virtual machines. It is also aided by CMS. The IUCV serves the function of facilitating the communication between a virtual machine and a CP service. Some examples of the latter include logging errors, communication with the system console, performing message system service functions, and other services such as a SPOOL system service. Figure 3-3 provides a conceptual view of these IUCV communication functions.

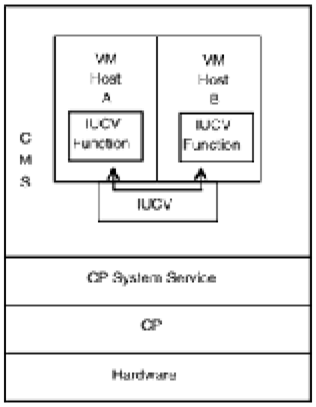


Figure 3-3
IUCV communications.

There are many more aspects of the VM environment, but for our purposes here this information should suffice. IBM has exhaustive documentation about VM and its components, and this author recommends contacting IBM for further information if needed.

3.3 Virtual Storage Extended

Virtual Storage Extended/Enterprise System Architecture (VSE/ESA) is another IBM operating system that operates on some S/370 architectures and the ESA/390 architecture. Its strength is in batch processing capabilities. Also, according to IBM documentation, it can support high transaction volumes. VSE/ESA is *regenerated,* meaning that a generation is not required; IBM ships object code. This is in contrast with MVS/ESA and VM/ESA, which must be generated at installation time for site-specific customization. Because VSE/ESA is supplied this way, much planning can be eliminated.

Before examining the functionality of VSE/ESA, consider its lineage according to IBM's *VSE/ESA Version 1.3. An Introduction Presentation Foil Master,* document number GG24-4008.

Date	**Offering**
1965	DOS
1972	DOS/Virtual Storage (DOS/VS)
1979	DOS/Virtual Storage Extended (DOS/VSE)
1985	VSE/System Product (VSE/SP) version 2
1989	VSE/SP version 4
1990	VSE/ESA version 1.1
1991	VSE/ESA version 1.2
1993	VSE/ESA version 1.3

VSE/ESA can operate in multiple environments. Some of those environments are

• It can serve as the sole operating system on a processor. As a result, both local and remote operations can be realized.

• It can run under an LPAR.

• It can run as a standalone system, without human intervention, in some cases functioning as a node within a network.

• VSE/ESA can also operate under VM as a preferred guest.

The following list includes the major enhancements and inherent features and functions of VSE/ESA, as well as some of its optional facilities. The purpose here is to understand the breadth of change that VSE has undergone with the advent of its support in ESA. The list is not exhaustive, but the core aspects of this release are presented.

• 31-bit virtual addressing

• Support for data spaces

• Support for more I/O devices

• ESCON support

• ESCON director support

• VTAM version 3 release 4 features for VSE/ESA

• National language support including Spanish, German, and Japanese

• Addressability of ≤2 GB of real storage and ≤2 GB of virtual storage

• VSE/ESA POWER operation in a private address space

• Support for the 3172 via VTAM 3.4

• ES/9000 processor support

• ESCON C-T-C adapter support

• Support for virtual disks in storage

• Support for extended functions of the 3990 DASD

• NetView for VES/ESA

• Support for SQL/DS making a VSE/ESA capable of functioning with Distributed Relational Database Architecture (DRDA) as a server

• 3174 ESCON connectivity

So many enhancements were made that the VSE/ESA operating system has become more popular in the past few years (according to reports in credible trade magazines). Beyond the aforementioned enhancements to VSE, some of its system components—Librarian, ICCF, POWER, and VSAM—need to be presented and briefly discussed for those new to the VSE/ESA environment.

The *Librarian* is a utility program that is used to manipulate libraries. It aids in the creation, maintenance, and use of libraries. With VSE/ESA the following were included in the Librarian: COPY and COMPARE commands; a MOVE command; LISTD command, which is a list including the date; a SEARCH command; LOCK and UNLOCK commands to provide data integrity to single library members that may be in the process of being updated; and a BACKUP command, which now supports the backup of an individual member.

Libraries have one or more sublibraries which, in turn, consists of members. These members are where data, programs, source code, and other “data” exist in a VSE/ESA system.

Two types of libraries exist in VSE/ESA: system and ICCF libraries. System library sublibraries can include the following members: *source*—source code to be processed; *object*—a module generated by the output of a language translator and used for input by a linkage editor; *DUMP*—if an abnormal termination results, the contents will be sent to this type of member in a sublibrary; *procedure*—a set of procedures, which may be a set of job control statements; and *phase*—a member that has a program or sections of a program stored in it that is (are) ready to run.

Other system-oriented libraries exist, and if further information is needed, look for the appropriate IBM VSE/ESA manual that explains this information needed or obtain a list of VSE/ESA documentation through your local IBM representative.

The VSE/Interactive Computing and Control Facility (VSE/ICCF) uses libraries. There are many such libraries, but relevant here and most frequently used include the *public library,* which consists of data that may need to be accessed by a number of users systemwide; the *common library,* which contains data in which all users are interested; the *main library,* which is attached to an individual once that person logs on; and the *private library,* which contains data limited to one or a few users.

The *interactive computing and control facility* (ICCF) serves as the interface between a user and a VSE/ESA system. Technically, it is the subsystem for program development and system administration occurs. Libraries, sublibraries, and members can be created through ICCF. *Priority Output Writers and Execution (Processors and Input) Readers* (POWER) is a subsystem that provides networking support, batch job processing, and spooling functions. The *Virtual Storage Access Method* (VSAM) is the access method for storing data, programs, and the like; thus it provides a form of data management. Its supported data organization includes entry-sequenced files (similar to those in a sequential file), key-sequenced data sets (similar to those in an indexed file), relative record data sets (similar to those in direct access files), and variable-length relative record data sets.

VSE/ESA has been placed on equal ground with MVS/ESA and VM/ESA. There are many other aspects of VSE/ESA, but the point of addressing the operating systems mentioned here is to convey some of their strengths and their prevalence in the marketplace. The enhancements necessary to place VSE/ESA on the same level as MVS and VM took time. But now IBM has three refined and tested operating systems to offer a wide variety of customer needs.

3.4 Summary

This chapter presented IBM’s three primary operating systems which work with their large processors. All three operating systems explained here are prevalent in the marketplace today.

These operating systems had different original design intents and the flavor they bring to the marketplace is still discernible today. MVS is basically considered a production-oriented operating system today. Technically, VM could be considered more research and development (R&D) oriented, but since considerable software development is available in the marketplace, even VM can be considered a production-oriented operating system. VSE is primarily production-oriented, but it can be found in many installations that could be considered R&D.

4
IBM Software Offerings

IBM has many software subsystems that operate with the operating systems previously discussed. This chapter focuses on software and methods designed especially for large networking systems.

4.1 Virtual Telecommunication Access Method

The *Virtual Telecommunication Access Method* (VTAM) is a software subsystem that operates under the operating systems described in Chap. 3. It is a critical component in traditional SNA and functions in new ways with IBM's APPN. Practically every application that runs in an MVS host runs as a VTAM application. Applications must be defined to VTAM in such a way that they can be used. The same holds true for hardware and components outside the processor. Figure 4-1 provides a conceptual view of VTAM with some peripherals attached.

One of VTAM's major roles in traditional SNA is aiding in session establishment. When a terminal user asks to use a software subsystem, the request is first interpreted by VTAM and then passed to the appropriate subsystem.

VTAM is also the centralized point for network component activation and deactivation. A VTAM component, the *system services control point* (SSCP), plays a vital role in the area of network management. This role should not be confused with the software that enables network management to be realized; rather, they work together. The SSCP network management role has to do with communicating with hardware and other components throughout the network.

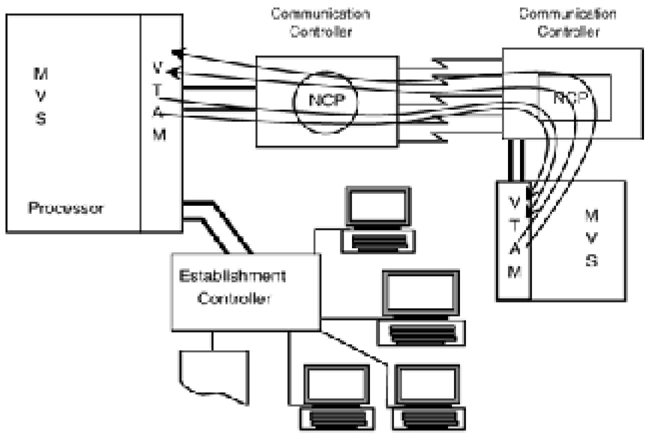


Figure 4-1
Conceptual view of VTAM.

VTAM must know (have defined) software and hardware that operate within an SNA network. Some VTAM components that must be defined for software and hardware operation are application(s), device(s), session operating parameters, logon menu (if utilized), and communication controller and attached devices.

Applications must be defined to VTAM for the application to work. For example, if application *Z* is loaded into a system, then certain parameters about the application must be defined. Different applications have requirements for definition. To determine whether this applies, you should consult the *IBM VTAM Resource Definition* manuals. Table 4-1 shows a basic example of an application definition.

Table 4-1 Application Definition Example

Name of the application VBUILD statement Name of the application Definition statement Operands	MYAPP TYPE = APPL MYAPP APPL ACBNAME AUTH DLGMODE EAS MODETAB SSCPFM USSTAB

Table 4-2 Establishment Controller Definitions

MY3174	VBUILD	TYPE = LOCAL
MY3174	PU	CUADDR =
	DLOGMODE =	
	DISCNT =	
	ISTATUS =	
	MAXBFRV =	
	PUTYPE =	
	USSTAB =	
YOUR327401	LU	LOCADDR =
.	.	
.	.	
.	.	
YOUR317406	LU	LOCADDR =

Table 4-1 is an example of how each application that operates under VTAM would need to be defined. Not all the parameters will be the same. The application and VTAM requirements will dictate how the application should be defined, as will the site requirements.

Devices must be defined to VTAM as well. A number of factors dictate how a device is defined, including how the device is physically attached and the data-link protocol used and its role in the SNA environment (according to SNA definitions). Table 4-2 presents an example of a 3174 establishment controller defined to VTAM.

In Table 4-2 the definition of the device can be divided into three parts. VTAM requires a VBUILD statement, a physical unit (PU) statement, and logical unit (LU) statements for those LUs to be used. Physical and logical units will be discussed further later in this book.

The statements and parameters in this example are typical of how a device is defined to VTAM. Three factors determine how a device should be defined to VTAM: the architectural capability of the device, how VTAM dictates that it must be defined, and how the device is used in a given site.

Session operating parameters must be customized. If a terminal user desires to work with an application, the terminal parameters must be programmed. The location for these session parameters is known as the *logon mode table* (LOGMODE table).

The LOGMODE table consists of numerous entries, each of which defines the session parameters for a particular type of session. Sessions will be discussed later in this chapter, but Fig. 4-2 shows a LOGMODE table.

The logon menu, if utilized, is what users see when viewing a terminal under the control of VTAM. This logon menu is formally called the *unformatted system services* (USS) table. Figure 4-3 is an example of a USS table.

```
*************************************************************
*       THIS IS THE LOGMODE TABLE FOR MY3174        *
****************************************************
*       LOGMODE TABLE ENTRY FOR 3278-M2 EMULATION          *
****************************************************

EDMODE      MODETAB
EDMODE2            MODEENT            LOGMODE = EDM2,
     FMPROF = X'03',            TSPROF = X'03',
               PRIPROT = X'B1',
     SECPROT = X'90',
     COMPROT = X'3080',                        RUSIZES = X'F8F8',
               PSERVIC = X'028000000000000000000200'

****************************************************
*       LOGMODE TABLE ENTRY FOR 3278-M3 EMULATION          *
****************************************************

EDMODE3      MODEENT
     LOGMODE = EDM3,                    FMPROF = X'03',
               TSPROF = X'03',
     PRIPROT = X'B1',             SECPROT = X'90',
          COMPROT = X'3080',
     RUSIZES = X'F8F8',
     PSERVIC = X'028000000000185020507F00'

****************************************************
        LOGMODE TABLE ENTRY FOR 3278-M4 EMULATION          *
****************************************************

EDMODE4            MODEENT            LOGMODE = EDM4,
     FMPROF = X'03',            TSPROF = X'03',
               PRIPROT = X'B1',
     SECPROT = X'90',
     COMPROT = X'3080',                  RUSIZES = X'F8F8',
               PSERVIC = X'02800000000018502B507F00'
```

```
*************************************************
     LOGMODE TABLE ENTRY FOR 3278-M5 EMULATION        *
*************************************************
```

Figure 4-2
A LOGMODE table.

```
EDMODE5              MODEENT              LOGMODE = EDM5,
      FMPROF = X'03',               TSPROF = X'03',
                        PRIPROT = X'B1',
      SECPROT = X'90',
      COMPROT = X'3080',                    RUSIZES = X'F8F8',
                  PSERVIC = X'028000000000018501B847F00'
*************************************************
EDMODE MODEEND      *
*************************************************
```

Figure 4-2
(*Continued*) A LOGMODE table.

```
******************TOP OF DATA*********************  **********
USSTAB TITLE 'ETUSS TABLE' *********************
ETUSS        USSTAB                        LOGON              USSCMD
CMD = LOGON,FORMAT = PL1                          USSPARM
PARM = APPLID                     USSPARM PARM = LOGMODE
            USSPARM PARM = DATA                            TSO
      USSCMD CMD = TSO,REP = LOGON,FORMAT = PL1
      USSPARM PARM = APPLID,DEFAULT = A01TSO
      USSPARM PARM = LOGMODE                      USSPARM
PARM = DATA
CICS              USSCMD CMD = CICS,REP = LOGON,FORMAT = PL1
      USSPARM PARM = APPLID,DEFAULT = DETTCCICS
            USSPARM PARM = LOGMODE
      USSPARM PARM = DATA
JES2              USSCMD CMD = JES2,REP = LOGON,FORMAT = PL1
      USSPARM PARM = APPLID,DEFAULT = JES2        USSPARM
PARM = LOGMODE                          USSPARM PARM = DATA
USSMSGS
USSMSG MSG = 0,TEXT = 'USSMSG0: @@LUNAME LOGON/LOGOFF IN
  PROGRESS'
USSMSG MSG = 1,TEXT = 'USSMSG1: @@LUNAME INVALID
COMMAND
   SYNTAX'
      USSMSG MSG = 2,TEXT = 'USSMSG2: @@LUNAME % COMMAND
UNRECOGNIZED'         USSMSG MSG = 3,TEXT = 'USSMSG3:
@@LUNAME %
PARAMETER UNRECOGNIZED'         USSMSG MSG = 4,TEXT =
'USSMSG4:
```

```
@@LUNAME % PARAMETER INVALID'
USSMSG MSG = 5,TEXT = 'USSMSG5: @@LUNAME UNSUPPORTED
FUNCTION'
```

Figure 4-3
Unformatted system services table.

```
USSMSG MSG = 6,TEXT = 'USSMSG6: @@LUNAME
SEQUENCE ERROR'
USSMSG MSG = 7,TEXT = 'USSMSG7: @@LUNAME SESSION
NOT BOUND'
USSMSG MSG = 8,TEXT = 'USSMSG8: @@LUNAME
INSUFFICIENT
STORAGE'
USSMSG MSG = 9,TEXT = 'USSMSG9: @@LUNAME
MAGNETIC CARD DATA
ERROR'         USSMSG MSG = 10,BUFFER =
MSG10                         USSMSG
MSG = 11,TEXT = 'USSMSG11: @@LUNAME SESSION ENDED'
USSMSG MSG = 12,TEXT = 'USSMSG12: @@LUNAME REQ
PARAMETER
OMITTED'
        USSMSG MSG = 13,TEXT = 'USSMSG13: @@LUNAME
IBMECHO %'
USSMSG MSG = 14,TEXT = 'USSMSG14: @@LUNAME USS
MESSAGE %
NOT DEFINED' MSGBUFF
MSG10       DC        (MSG10E-MSG10-2)         DC         C'
',X'15'         DC         DC         C'              '          DC
DC              C'          USING THE CORRECT      ',X'15'      DC
DC         C'
'                DC
DC              C'         VTAM SYNTAX              ',X'15'        DC
DC              C'         '             DC                      DC          C'
THE
MENU OF              'X'15'      DC                      DC           C'
'              DC
DC C'         CHOICE CAN     ',X'15'                  DC
DC              C'         '               DC
DC              C'         BE DISPLAYED                ',X'15'
DC
DC              C'         '          DC
DC              C'         '          DC
DC              C' ---------------------------------------- 'X'15'
DC
DC              C' ---------------------------------------- ',X'15'
DC
```

```
DC          C'      YOU CAN CREATE YOUR OWN
',X'15'       DC
DC          C'      '            DC
DC          C'      MENU      ',X'15'      DC
DC          C'      '            DC
DC          C'      SO USERS LOGON            ',X'15'      DC
DC          C'      '       DC
DC          C'      BY APPLICATION NAME       ',X'15'      DC
DC          C'      '       DC
DC          C'      SUCH AS            ',X'15'            DC
DC          C'------------------------------------ '       DC
DC          C'      TSO        ',X'15'      DC
DC          C'      '            DC
DC          C'      CICS      ',X'15'      DC
DC        C'
DC
DC          C'      JES2      ',X'15'      DC
DC        C'
            DC
DC          C'      '            DC
DC          C'-------------------------------------',X'15'          DC
DC          C'-------------------------------------',X'15'          DC
DC          C'      '            DC
DC          C'      ',X'15'      DC
END          USSEND
***************** BOTTOM OF DATA
**************************
```

Figure 4-3
(*Continued*) Unformatted system services table.

The USS table shown in Fig. 4-3 can be divided into three parts. The arrangement of this table is based on IBM's *VTAM Resource Definition* manual. Some flexibility exists, but generally the following parts are present: (1) the name of the application a user wants to access is listed, followed by the parameters required to pass the request to that application; (2) VTAM has 15 messages which may be generated if certain conditions exist; and (3) the contents of what is displayed on the menu must be coded. As long as the table is created and does not violate any VTAM regulations, much flexibility exists.

A communication controller [also called a front-end processor (FEP)] has special definitions to VTAM because it has a software program known as the Network Control Program (NCP) operating within it. Because this is so, knowing what is attached directly and indirectly to the FEP is required.

Creating an NCP is site dependent like VTAM. However, restrictions do apply regarding how the NCP can be generated. The term *generate* is often used in a deprecated form known as (GEN). Figure 4-4 shows the location of an NCP with reference to other components in the network.

VTAM is a critical component in an SNA network. Understanding it is no trivial task. As I have said before, and believe to be true, “VTAM is the heart and soul of SNA.”

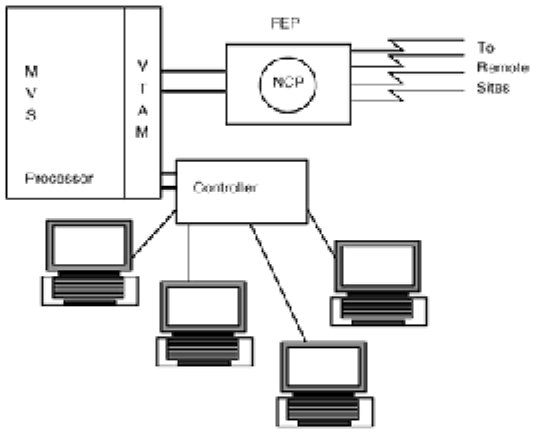


Figure 4-4
NCP location.

4.2 Job Entry Subsystem 2

Job Entry Subsystem 2 (JES2) is a spooling subsystem. According to IBM it is a primary subsystem. An interesting concept about software that operates under the control of VTAM (with the exception of third-party software) is that this software is collectively referred to as *subsystems*. One reason for this is the size of some of them. As we shall see, some subsystems are practically operating systems themselves with sub-subsystems!

JES2 is a spooling subsystem that receives jobs, schedules them, and controls their output. In essence, it serves as an interface to the operating system for job processing. An example of how JES2 is used could be a job that is to be printed. Once the command is issued to print, the job goes to JES2 and then is subsequently printed.

Figure 4-5 provides a conceptual view of JES2.

JES2 functions so that data to be processed (whatever that data may be) may be literally spooled to a DASD for buffer storage. By doing so delays can be minimized because processing continues and jobs are queued to be processed.

The term *remote job entry* (RJE) refers to a site not in the same physical facility as the processor complex that has terminal capabilities and a link to the processor so that jobs may be submitted to JES2. *Network job entry* (NJE) is another term used to refer to a complex of processors where they are dispersed but connected together in a network. NJE functions to provide the ability for multiple JES2 subsystems to communicate as peers in a network environment. This is in contrast to RJE, which permits JES2 interaction with a remote workstation.

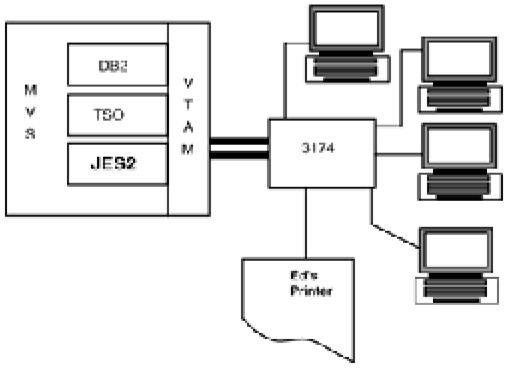


Figure 4-5
The JES2.

JES2 is a descendant of the Houston Automatic Spooling Priority (HASP). It originated by IBM employees in Houston to expedite job processing in a typical university environment—many jobs, short execution times. Early OS/360 spent more time scheduling a job than for its execution. The benefit of HASP was achieving overlap in scheduling and execution through spooling.

Another job management subsystem is JES3. Originating from IBM's Attached Support Processor (ASP) system, JES3 broadens the scope of job management. It manages jobs in the job queue, during the execution phase, and after execution for proper outputting; this management is accomplished by specially tailored algorithms, furnished by the installation. A fundamental difference between JES2 and JES3 is that JES3 operates in an environment in which one copy of JES3 is the "master" (called the *GLOBAL*) and other copies (called *LOCALS*) operate throughout a processor complex in communication with this master copy.

4.3 Network Control Program

Network Control Program (NCP) operates on IBM's communication controllers. It, as does VTAM, plays a critical role in IBM's SNA. The NCP serves two primary functions: routing and flow control. Although it serves other functions as well, these seem to be the primary ones. An NCP must have devices, paths, communication lines, and connections defined to it.

The NCP performs two major functions: controlling data flow (1) through a network with other NCPs and VTAMs and (2) between itself and VTAM. The NCP also routes data throughout a network. Within an NCP generation a list of routes (explicit and virtual) is defined. Also, a *class of service* (COS) can be used to aid in routing data. Depending on how the COS has been implemented, particular data from one source may have a higher priority than data from another source.

The NCP operates in conjunction with other programs on a communications controller. For example, the *Network Packet Switched Interface* (NPSI) option is a program used to permit connectivity with an X.25 network. Another example is the *Network Terminal Option* (NTO), which works in conjunction with an NCP to permit non-SNA devices to connect to the SNA network. NTO's fundamental purpose is to convert this non-SNA protocol into SNA protocol before the data are routed to the processor.

4.4 NetView

NetView, IBM's tool for managing SNA, originated in the 1970s. NetView was announced in 1986. Prior to that time components were used to selectively manage a network. For example, the *Network Communications Control Facility* (NCCF), the command-line interface for NetVeiw, was announced in 1978. At the same time the Network Problem Determination Application program was announced. Both were released in 1979.

NetView has grown to consist of the Network Communications Control Facility (NCCF), the Network Problem Determination Application (NPDA), the Network Logical Data Manager (NLDM), the Browse facility, the Status Monitor, the Graphic Monitor, and the Resource Object Data Manager (RODM). Figure 4-6 is a conceptual view of VTAM and the SNA network.

NetView, VTAM, and MVS system commands can be entered at the command line from NCCF, which is the base of NetView. This capability makes it possible to control an SNA network from a remote location.

Typically, SNA management is accomplished via a console (generally inside a data center). However, with the capability NCCF provides remote operation is possible. As long as a connection can be made to the system running NetView and an individual is authorized to execute system-oriented commands, then NetView, VTAM, and MVS commands can be issued against the NCCF command prompt.

NPDA is that part of NetVeiw used to manage hardware. It can collect and maintain data about devices throughout the network. NPDA has the capability to request data about a particular piece of hardware or accept data sent to it from a given hardware device.

NLDM is the session monitor. A *session* is a logical connection between two endpoints. With NLDM the following information can be obtained about a session:

• Availability

• Configuration

• Error

• Event

• Explicit route

• Response time

• Session partner

• Trace

• Virtual route

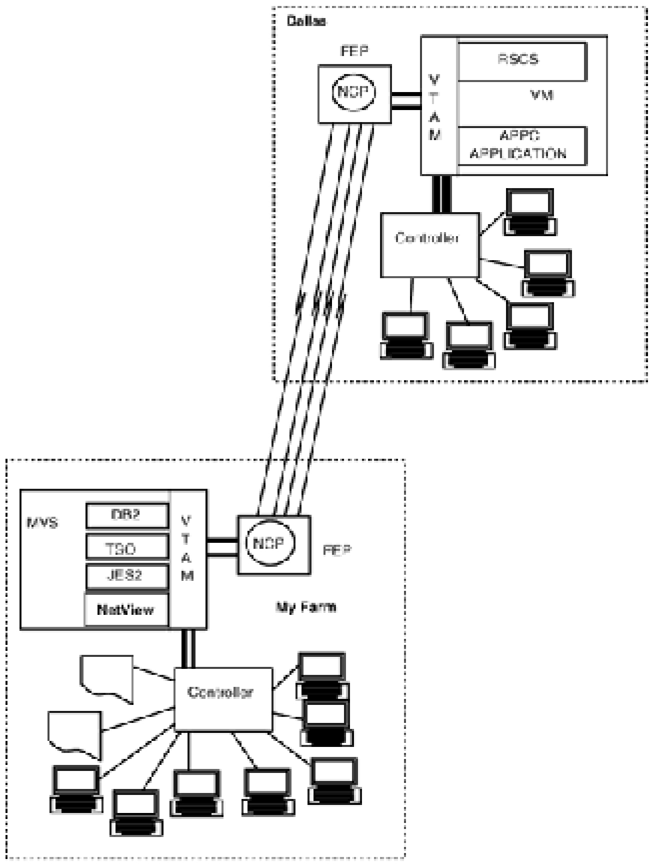


Figure 4-6
Conceptual view of NetView.

The Browse facility enables an operator to view NetView log data, VTAM definitions of devices within the network, command lists, and systemwide definitions.

The Status Monitor collects information about parts of an SNA network. It displays the data in columnar form, or it can be routed to the Graphic Monitor, where the data can be display graphically. The Graphic Monitor is a menu-driven method for monitoring network operations. It has pull-down menus and displays selected parts of a network in color. Certain colors are also used to highlight problem areas.

The Resource Object Data Manager operates in its own address space in memory. It serves as a central repository for storing and retrieving information about resources throughout the network. It can obtain execution information, configuration information, and/or status information. It can make this data available to those applications which need it because of its object-oriented structure.

NetView can operate under MVS, VM, and VSE. Two commonly used functions of NetView are alerts and response time. *Alerts* are messages regarding the status of a given device. These alerts use IBM's Network Management Vector Transport (NMVT) protocol. Information such as the day, date, and time of a failure can be included in an alert. Other status information is also included and can be site-specific.

The *response time monitor* (RTM) is a tool used to measure the time it takes for data to leave a terminal after an *attention identifier* (AID) key is pressed, reach the host application, and return.

NetView has been expanded by IBM to support other platforms now such as the RISC/6000. Other capabilities are possible with NetView, even participating in network management with TCP/IP-based networks.

4.5 Time Sharing Option

The *Time Sharing Option* (TSO) is IBM's interactive facility that operates under MVS. TSO has three modes of operation: the Interactive System Productivity Facility/Program Development Facility (ISPF/PDF), the Information Center Facility, and the line mode.

The ISPF/PDF main menu is typically what users see when they log onto TSO. The ISPF/PDF main menu offers many choices, which lead, in turn, to submenus. Some of the main menu choices and their basic functions are

• *ISPF PARMS*. These are specific parameters for use with ISPF.

• *Browse*. This function permits viewing only of authorized data sets.

• *EDIT*. This command invokes the editor and enables a user to create a memo, program, or basically anything one would use an editor for.

• *Utilities*. This leads to another menu which has a number of selections that permit disk, data-set, and other maintenance and utility functions.

• *Foreground*. This causes the language processor to move to the foreground.

• *Batch*. This enables a user to submit jobs for batch processing.

• *Command*. If chosen, this takes the user to TSO line mode of operation. Here valid TSO commands can be entered.

• *Dialog test*. This function provides a user the ability to perform dialog testing.

• *LM utility*. This provides an individual with capabilities to perform maintenance utility functions.

• *EXIT*. If selected, this causes ISPF/PDF to terminate.

Other functions are available from the main menu. Some of the ones listed above have submenus that provide additional capabilities.

The Information Center Facility (ICF) provides users with a main menu similar in appearance to the ISPF/PDF main menu. Some of the functions available to users via ICF are *News,* allowing users to obtain news from the system; *Names,* which provides a list of names and phone numbers; *Chart,* which permits a user to create a chart or graph; *PDF,* which (if selected) enables a user to use program development services; and *EXIT,* which causes ICF to terminate.

The TSO line mode can be used for multiple reasons. First, valid TSO commands such as LISTCAT can be entered. For example, this command is used to list data sets that are cataloged and accessible by a user.

Another function of TSO line mode is execution of custom programs. If a customized program needs a certain protocol for operation, such as terminal interaction, then a program can be created to execute using line mode.

Figure 4-7 depicts a conceptual view of TSO and other subsystems mentioned above.

4.6 Customer Information Control System

The *Customer Information Control System* (CICS) is an online transaction-processing system. It is supported under MVS, VM, and VSE operating systems. It is oriented generally toward business implementations rather than scientific or engineering computations.

An example of where CICS would typically be implemented is in a banking environment. For example, transaction programming using CICS could be used for customers who want to access their bank accounts via an automatic teller machine (ATM). In this case a program in the ATM communicates with a program running under CICS control in the bank’s computer. The program at the teller machine is communicating with the CICS program in real time. The program running under CICS, in turn, accesses a database that maintains the requesting ATM account balance. After the program under CICS verifies the requesting party’s account, it communicates with the ATM, sending it the appropriate response. Assuming that the money is available, the ATM dispenses the cash.

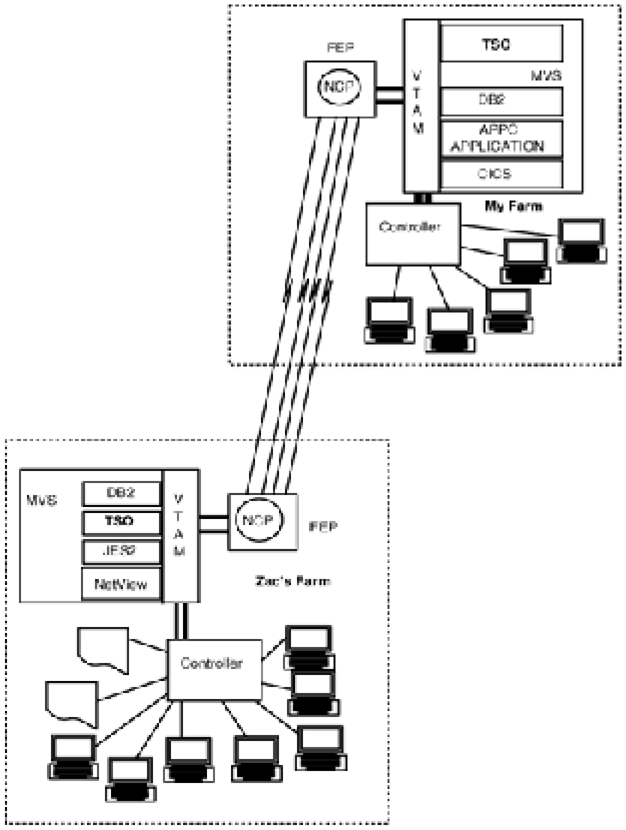


Figure 4-7
Conceptual view of TSO.

Customized programming is possible under CICS, making it attractive to users who need to create online processing programs. CICS also supports communication between transaction programs within one CICS subsystem.

4.7 DATABASE 2

DATABASE 2 (DB2) is IBM's relational database application that provides users with flexibility and power via the functions it supports. Some of those functions are

• Utilization of a single VTAM conversation to manipulate multiple request and responses with other DB2 applications throughout a complex.

• Support for Distributed Relational Database Architecture (DRDA).

• Support for Structured Query Language (SQL) request from remote locations.

• Site independence and capability for interaction with other DB2 sites.

• Multiuser support for concurrent access, including making updates, deletions, and insertions.

• DB2 fits into SAA via CPI-SQL.

• An audit trail can be selectively chosen.

• DB2 can be used in an XRF environment.

• Support for 10,000 open concurrent data sets per address space is provided.

• The maximum number of columns in a DB2 table is 750.

• Multiple simultaneous index recovery can be performed on the same table space.

4.8 Remote Spooling Communication Subsystem

The *Remote Spooling Communication Subsystem* (RSCS) application operates under VM to provide data-transfer capabilities. According to IBM, it has the following support: file transfer, message, commands, and mail between VM, MVS, VSE, NJE, and OS/400; ASCII support to printers and plotters; support for IPDS and SCS data streams; provision of a gateway programming interface for protocols such as TCP/IP; support for 3270-type printers with form control buffering; and capability to share printers.

RSCS provides VM users the ability to send mail, specific messages, and jobs to other users within an SNA network. VM users use RSCS for printing purposes. The basic functionality of RSCS is that the origin node starts communication with a destination node. And, multiple devices may be along the path between the two. Because of the wide protocol support, RSCS can function over multiple types of links.

4.9 Local Area Network Resource Extension and Services

The *Local Area Network Resource Extension and Services* (LANRES) application operates in MVS and VM environments. According to IBM sources, it brings the power behind the S/390 architecture to a NetWare environment. LANRES achieves this by making DASD available to NetWare servers and S/390-based printers available to NetWare clients.

LANRES also permits authorized MVS users to move data to and from a NetWare server. Additionally, NetWare server files and directories can be listed, created, and/or deleted.

LANRES also makes LAN printers available to MVS users. In effect, it brings together NetWare environments with S/390 seamlessly to take advantage of right-positioning of workloads. LANRES also offers centralization of LAN management to the MVS host, if desired. It also permits MVS users to send a PostScript file to a PostScript printer on a LAN. Figure 4-8 shows a hypothetical LANRES environment.

LANRES is versatile because of the connectivity solutions it supports. The following connectivity solutions are supported by LANRES: ESCON, parallel channel, APPC connection, host TCP/IP, and VM Programmable Workstation Services (VM PWSCs).

The method of connection dictates how LANRES is configured on the host. Because of the breadth in support for connectivity solutions, requirements, installation, and definitions are site dependent and directly related to how the product is used. For example, if the product is used with TCP/IP under MVS, LANRES uses sockets and TCP for connectivity. However, if APPC is used, then it connects to APPC MVS via CPI-C, conforming to SAA standards.

IBM has many other software products, too many to list here. If your needs have not been covered in this section, contact your local IBM representative for additional information.

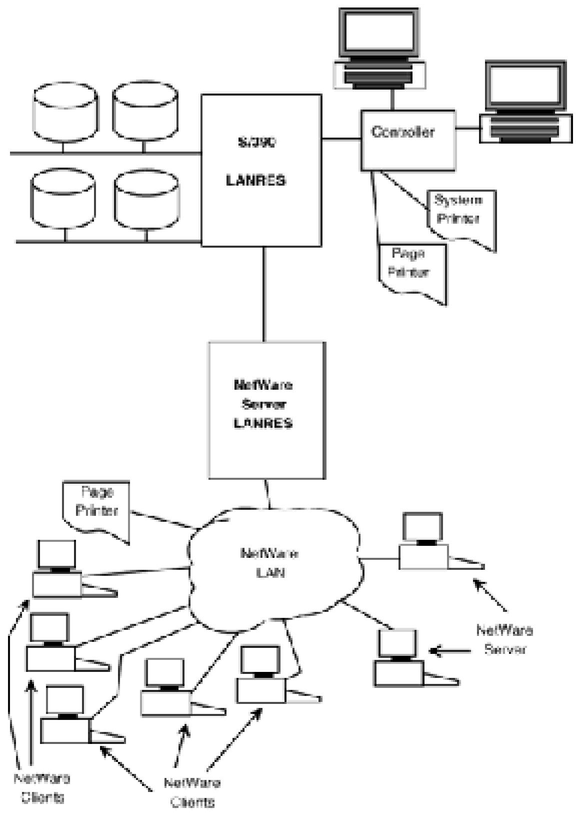


Figure 4-8
Conceptual view of LANRES and NetWare LANs.

4.10 Summary

IBM's software is the other part of the two-part equation to SNA; IBM hardware is the first part. The software presented here is the most popular software in use today, but there are other IBM software offerings. Unless your background or current work includes exposure to large systems, you probably won't come into contact with the software offerings explained in this chapter.

The nature of the software explained here is such that it lends itself to large systems and hence many users. Implementation of this software is the backbone of SNA when considered in conjunction with the hardware offerings. It is through the definition of software that much of SNA is defined and identified.

5
SNA Networking

In 1974 IBM announced SNA. It began as a layered architecture, and traditional SNA remains as such. Traditional SNA is defined by being more hierarchical than peer-oriented. Succinctly, it is pre-1992 with the Networking Blueprint announcement. This does not mean that the Networking Blueprint replaces the functionality of traditional SNA; rather, it provides a different approach to networking in general. A few examples are presented later, at the end of this chapter.

5.1 SNA Layers

The seven traditional SNA networking layers are shown in Fig. 5-1.

The basic functions at these layers are

Layer 1	Effecting linkage between two or more nodes
Layer 2	Moving data across a link
Layer 3	Routing and flow control
Layer 4	Throttling data movement and performing security functions if required
Layer 5	Synchronizing, correlating, and grouping data
Layer 6	Formatting data to protocol
Layer 7	Providing application-required services

In many installations this hierarchy describes the functionality of data flow within the network. This model of networking is considered traditional SNA.

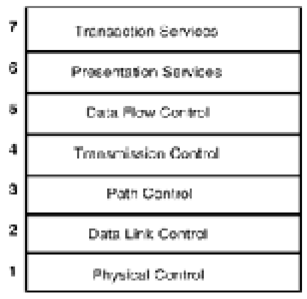


Figure 5-1
SNA tradition layers.

5.2 IBM's Blueprint for Networking

On September 15, 1992, IBM announced their Networking Blueprint, which is a new approach to IBM networking. The blueprint provides a framework for choices to be selected to fit the needs of a situation. Figure 5-2 is a representation of that framework.

The structure of the blueprint, as seen in Fig. 5-2, is entirely different from the layers of traditional SNA. According to IBM, the blueprint structure consists of four major layers, three switching boundaries, and what IBM calls the *systems management plane*. The four layers are addressed first, followed by switching boundaries and the systems management plane.

1. *Subnetworking layer*. This constitutes the lower layer, which IBM divides into four categories: local area network (LAN), wide area network (WAN), channel, and emerging. These categories can be further divided into protocols. For example, protocols that would generally fall into the LAN category include Ethernet, Token Ring, and FDDI; protocols applicable to the WAN category could include FDDI, frame relay, SDLC, and others; protocols applicable to the channel category could include byte multiplexer and block multiplexer channels and ESCON; and protocols applicable to the emerging-technology category include Asynchronous Transfer Mode (ATM) and Fast Ethernet. This list is not exhaustive, but conveys the idea behind the supported protocols at this layer. And, to a considerable degree, these can be selected for what best fits the site requirements.

2. *Transport/networking layer*. This layer is represented by six networking capabilities. The transport/networking capabilities apply differently according to the following supported protocols: SNA, APPN, TCP/IP, OSI, NetBIOS, and NetWare. The functions of this layer are contingent on the protocol selected. This means that TCP/IP works in a particular fashion whereas APPN works according to its structural definitions at this layer. Users have the flexibility to select the protocol of choice.

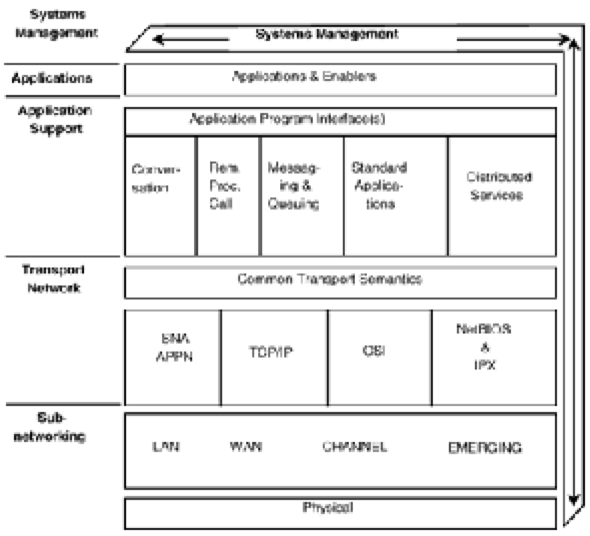


Figure 5-2
IBM’s Networking Blueprint

3. *Application support layer*. This layer provides service support for applications. According to IBM, the prevalent interfaces and services that work at this layer are conversational services (known as CPI-C), dealing with streams of related interactions; remote procedure call (RPC), which is capable of passing parameters to a subroutine; and message queue (known as MQI), which manages the queues that relate messages. The following services may also apply at this layer: distributed system services; different vendor applications such as TELNET for remote login, FTP for file transfer, and SNMP for network management; and other non-transport-layer-dependent applications

4. *Application layer*. This layer comprises applications inherent to the protocols available. These applications include print, mail, file transfer, and remote logon.

5. *Switching boundaries.* The applicable switching boundaries include application program interface (API), common transport semantics (CTS), and subnetwork-access boundary (SAB). The primary purpose of the API switching boundary serves to make the underlying architecture transparent. The CTS switching boundary enables any protocol above it to access any protocol below it. The SAB switching boundary resides between the transport/network part and the physical part of the blueprint. It serves to make link services available to the protocols driving the network.

IBM's stance on its Networking Blueprint indicates a more flexible networking approach than in traditional SNA. IBM has published two documents that are good references on its blueprint: GC31-7057, *Networking Blueprint Executive Overview*; and GC31-7074, *Multiprotocol Transport Networking* (MPTN) *Architecture: Formats.* To obtain these and other IBM documents, contact your local IBM representative or obtain their World Wide Web (WWW) location.

5.3 Traditional SNA Concepts

This section explains terms and concepts that make up the core of SNA. SNA as a network protocol is implemented via hardware and software.

Nodes

The term *node* is used in IBM documentation, and depending on the context, it can take on different meanings. In traditional SNA, which is sometimes referred to as *subarea SNA,* different type nodes exist. These popular nodes in subarea SNA include

• *Host node.* Also known as a *subarea node,* this provides end-user services. It is a type 5 node. (Node types will be explained later.)

• *Communication controller node.* This communication controller (also known as an FEP) is a type 4 node.

• *Peripheral node.* This is a cluster controller or an establishment controller. Depending on the device, it may be a type 2.0 or 2.1 node.

Figure 5-3 shows all three types of nodes.

Subareas

Subareas exist in traditional SNA. No areas are defined. A subarea is defined as either a subarea node and peripheral node(s), a subarea node, or a subarea node and communication controller node. Figure 5-4 depicts these three subareas.

Network-accessible units (NAUs)

IBM defines an NAU categorically as a system services control point (SSCP), physical unit (PU), or a logical unit (LU).

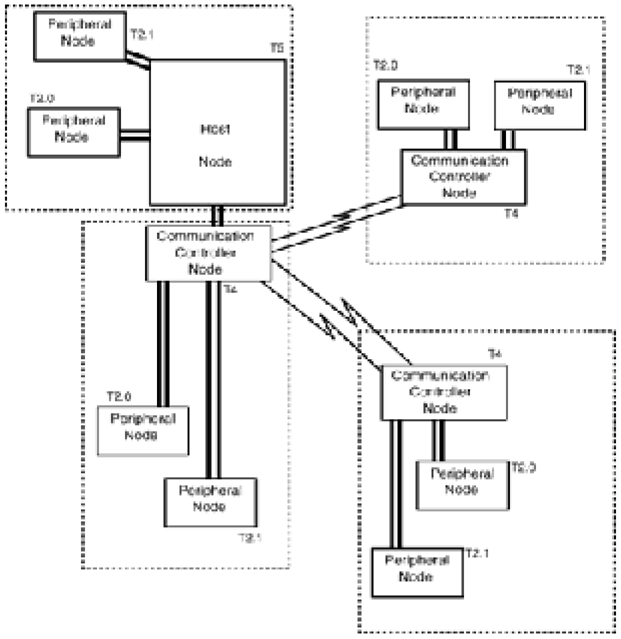


Figure 5-3
Conceptual view of nodes.

The characteristics, functions, and locations of these NAUs are

System services control point (SSCP). This is a controlling point in SNA and is located in the Virtual Telecommunications Access Method (VTAM) (see Fig. 5-5). Some of its characteristics and functions are network control, session management, resource activation and deactivation, focal point for receipt of PU data, passing data to and from NetView, and command execution.

Physical unit (PU). A physical unit pertains more to the functions or capabilities of a node than to describing a particular hardware device. The PU type is architecturally related and part of macrocode and software. Some basic characteristics and functions of PUs are:

• A PU is defined in software or microcode, receives messages from an SSCP, and provides internal network functions, not user-related functions.

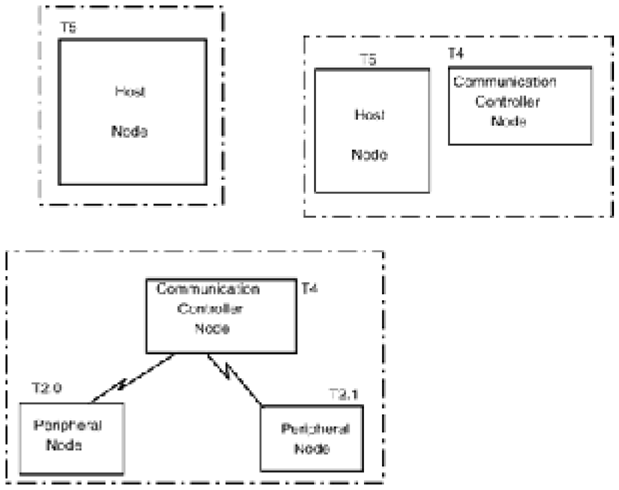


Figure 5-4
SNA subareas.

• Participating entities in an SNA network are known by their node type.

• A PU can manage links and link stations, set up virtual and explicit routes in certain nodes, and communicate with one or more control points.

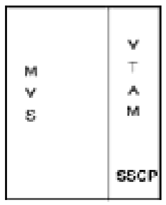


Figure 5-5
Conceptual view of an SSCP.

Four types of PUs—types 5, 4, 2.0, and 2.1—are presented here and are best described according to their functional characteristics:

1. A PU type 5 (T5) node is a host subarea node and functions as a processor, practically speaking, or can provide T5 functions, including management of subarea resources, facilitating session establishment, and monitoring resources.

2. A PU type 4 (T4) node is a communication controller node. It is an FEP or has the capability to emulate PU4 functions; for instance, the control point is called a *physical unit control point* (PUCP), the PU manages its peripheral nodes, and it can communicate with an SSCP.

3. A PU type 2.0 is a peripheral node and is totally dependent on a T5 node for session establishment. The control point for this node is a PUCP, and the PU communicates with a T4 PUCP or an SSCP, monitors its local resources, and sends status related data to an SSCP.

For practical purposes, it is correct to associate a physical device with a PU, but this can be elusive because some devices can act as different PU types depending on how they are GENed. Also, the location where terminals and printers connect on controllers can be considered an LU. Granted IBM documentation does not say this (at least I have not seen any), but it is accurate. The same holds true for PU2.1 devices.

A PU2.1 node is also a peripheral node. Its control point is a PUCP. This node differs from a T2.0 node because it supports peer communications to some degree. It can perform the functions of a T2.0 node, but it can also perform functions native to T2.1 architecture. Additional details on T2.1 nodes are provided in Chap. 6.

Logical units (LU). IBM defines a *logical unit* as an addressable endpoint. This applies to hardware and software. The LU types of concern here and their functions (protocol support) are:

LU0	Create your own program
LU1	SNA Character String printing (SCS)
LU2	A 3270 data stream
LU3	A 3270 data stream for printers
LU6.2	Advanced Program-to-Program Communication (APPC) protocol
LU7	5250 data stream for AS/400 systems

There are two categories of logical units: dependent and independent. The former requires VTAM for session establishment and the latter does not, after initial download of tables.

Figure 5-6 is an example of hardware, software, and concepts presented to this point. It is a good example for obtaining a holistic perspective on SNA.

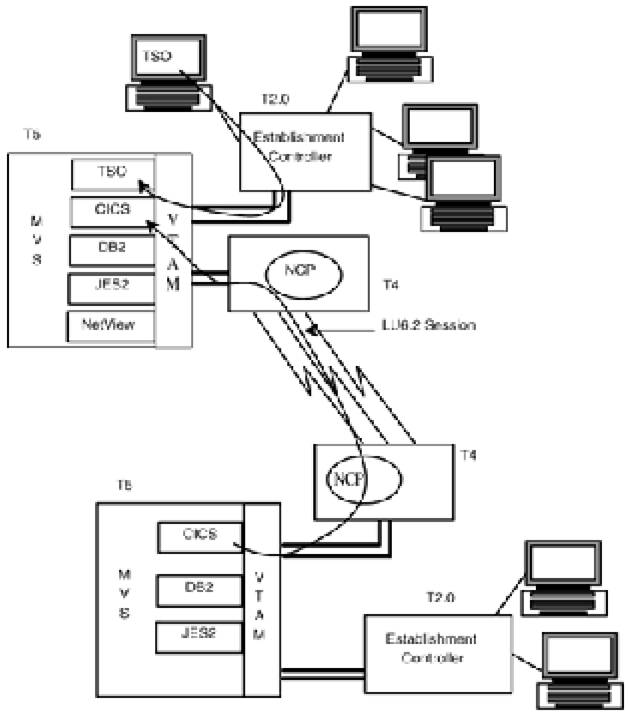


Figure 5-6
Conceptual view of a typical SNA network.

Figure 5-6 shows two processors, with software on each, communication controllers, an establishment controller, a cluster controller, printers, and terminals. It also shows two CICS subsystems communicating with one another via an LU6.2 session. Additionally, it shows a terminal communicating with TSO via LU2 protocol.

Sessions

As mentioned previously, *sessions* are defined as a logical connection between two endpoints. Four types of sessions are considered here: SSCP-SSCP, SSCP-PU, SSCP-LU, and LU-LU.

The *SSCP-SSCP* session is an example of two VTAM subsystems communicating with one another. There are numerous reasons for this; for instance, a session can be set up between a terminal user and a software subsystem that is not in the same processor. An *SSCP-PU* session is used to activate a device. Other functions of this type of session include management-related data flows across the session. An *SSCP-LU* session can be used by VTAM to activate or deactivate an LU. An *LU-LU* session could be a terminal user communicating with a software subsystem. In this case the LU-LU session could be described as *primary logical unit* (PLU) and *secondary logical unit* (SLU), where the former is a software subsystem and the latter is the terminal user. This is considered a *dependent logical unit* (DLU). The LU-LU session can also be described as independent. When this is the case, the LU is referred to as an *independent logical unit* (ILU).

Link stations

A link station (LS) is the intelligence in a device defined as being that point in a device where the data link is managed. IBM's *SNA Technical Overview* document GC30-3073 describes this concept very well, and Fig. 5-7, which is based on their model, depicts this.

Link stations perform functions such as receiving requests from and responding to its control point, controlling link-level data flow, moving data from one link station to another via the medium, and managing error recovery at the link level on the node.

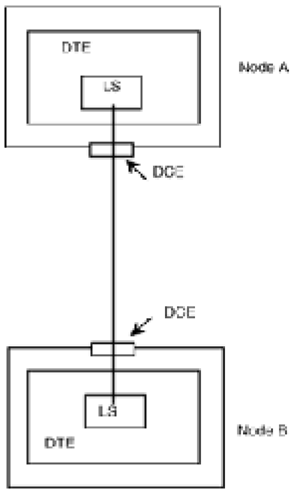


Figure 5-7
Conceptual view of a link station.

Link

In SNA a *link* refers to the data link. SNA links include parallel channel, ESCON, frame relay, Token Ring, SDLC, and Ethernet.

Technically, IBM defines a link as that connection between two link stations. Hence, this includes the medium, data communication equipment (DCEs), and link connection. The link connection is defined as that part consisting of the DCEs and transmission medium.

Domains

Another concept in SNA is the domain, which is that area whose components have a single point for control. In a T5 node the control point is the SSCP; in a T4 node, this is the NCP.

Parallel to the concept of domains is ownership. In SNA all resources are owned; the question is by which device. This concept of ownership is related to the domain. Resources in a given domain are normally owned by a control point in that domain.

Once the concept of a domain is understood, it is necessary to understand the concept of cross-domain resources. The latter is defined as a resource in a domain other than where the requesting party is located.

Figure 5-8 shows two different domains.

In Fig. 5-8 two processors have application subsystems. Each processor owns the applications on that host. If defined appropriately to VTAM, users in domain A can access application subsystems in domain B and vice versa.

5.4 SNA Protocol Structure

SNA protocol structure can be explained by layers. Figure 5-9 is a view of what SNA considers as a message unit.

Three distinct components of the message unit are the basic link unit (BLU), the path information unit (PIU), and the basic information unit (BIU).

1. *Basic link unit.* The BLU is assembled at the data-link layer of the network. It includes data and protocols that have been passed from layers above it. In front of the BLU is a link header (LH). Next is the information field, followed by the link trailer (LT) at the end of the message unit.

2. *Path information unit.* The PIU consists of the transmission header (TH), request or response header (RH), and the request or response unit (RU). The PIU operates at that layer in the network responsible for routing data (message units) through SNA.

3. *Basic information unit.* The BIU consists of a request or response header, then a request or response unit. A request or response unit is located on top of the BIU. Depending on the direction of the mes sage unit flow either a response or some type data end user or SNA data which includes data streams is sent.

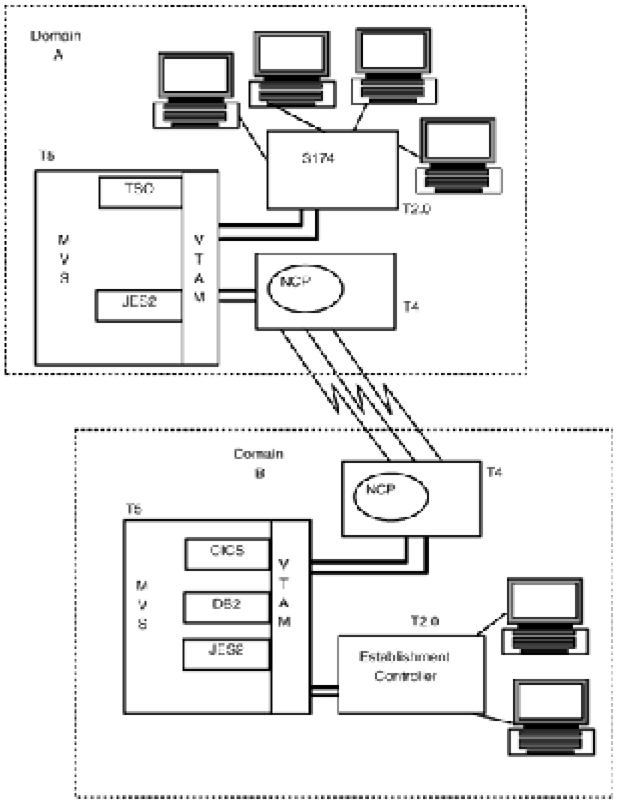


Figure 5-8
Conceptual view of domains.

Considerably more information abounds on this topic of message units. IBM document GA27-3136 is the source for present and future information on discussing data streams, profile concepts, function management header concepts, and request and response concepts. For greater detail on these topics, the IBM document is the best reference.

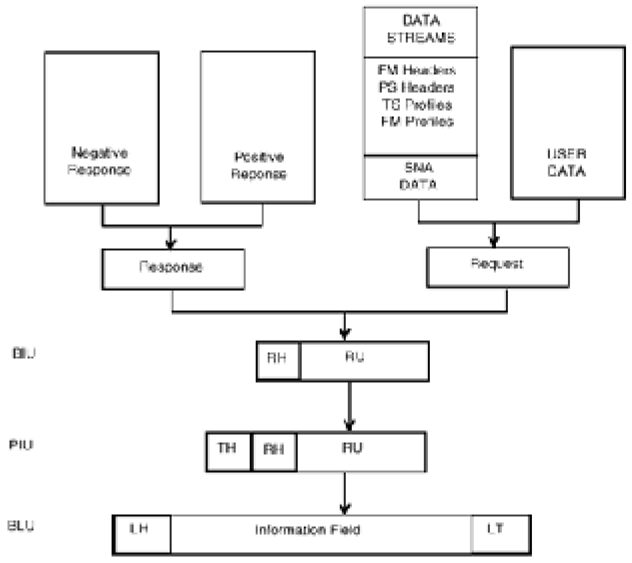


Figure 5-9
Structural view of the message unit.

5.5 SNA Data Streams

IBM has defined data streams which are used in SNA. An example of a dominant data stream is the 3270, which is used by terminals and printers that operate with MVS, VM, and VSE operating systems. Briefly, the data streams used include the SNA Character String (SCS), the 3270, the General Data Stream (GDS) variable, Information Interchange Architecture (IIA), and the Intelligent Printer Data Stream (IPDS).

1. *SNA Character String.* This character string is a protocol used with printers and certain terminals. LU1 and LU6.2 can use this data stream. One unique aspect of this data stream is its lack of data flow control functions.

2. *3270 data stream.* This data stream contains user-defined data. It also includes commands that aid in LU-LU control. LU2 and LU3 use this data stream for terminals and printers, respectively. It can be used by LU6.2 as an option data stream.

3. *General Data Stream.* This data stream is used by transaction programs to interpret data records as they were sent and received. This data stream is used by LU6.2

4. *Information Interchange Architecture.* This data stream is used to define a collection of data streams. It is used by applications exchanging programs. This means that Open Document Architecture (ODA) can be used. Document Content Architecture (DCA) can also be used.

5. *Intelligent Printer Data Stream.* This data stream is used between a host and a printer and can be used with an all-points addressable printer. IPDS can intermix text and graphics-both vector- and raster-based.

5.6 Profile Concepts

Two profiles for examination here include the transmission service and function management profile.

Transmission service profiles are used at the transmission layer in the network. They represent protocols that may be selected at session activation:

TS1	Used with SSCP-PU and SSCP-LU sessions
TS2	Used with LU-LU sessions
TS3	Also used on LU-LU sessions
TS4	Used on LU-LU sessions
TS5	Used on SSCP-PU sessions
TS7	Used on LU-LU sessions
TS17	Used on SSCP-SSCP sessions

These profiles provide a variety of services. They are used at a developmental and debugging level.

The following are *function management profiles:*

FM0	Used on SSCP-PU and SSCP-LU sessions
FM2	Used on LU-LU sessions
FM3	Used on LU-LU sessions
FM4	Used on LU-LU sessions
FM5	Used on SSCP-PU T5 and T4 sessions
FM6	Used on SSCP-LU sessions
FM7	Used on LU-LU sessions
FM17	Used on SSCP-SSCP sessions
FM18	Used on LU-LU sessions
FM19	Used on LU-LU sessions

As do the transmission service profiles, function management profiles provide a variety of service on the basis of session type and need. For in-depth information about these profiles, refer to IBM manual GA27-31-36.

5.7 Function Management Header Concepts

The concept behind a *function management header* (FMH) is that if a session supports these headers, a request header can contain an option indicating that an FMH is present. If present, they indicate specific functionality. Consider the following FMHs:

FMH1	Used to select a destination logical unit
FMH2	Used to handle data management for a task
FMH3	Used for the same purposes as FMH2, but does not have a stack reference
FMH4	Carries logical block commands that are used to define different parameters
FMH5	An LU6.2 ATTACH header; used to carry a request for a conversation; with non-LU6.2 ATTACH originates from the sending half-session program to the destination manager
FMH6	Used during an active transaction program conversation
FMH7	Provides error information for LU6.2 and in a similar fashion for non-LU6.2
FMH8	Used with an IMS application with LU6.1 protocols
FMH10	Prepares a session for sync-point processing
FMH12	Used with LU6.2 for security

5.8 Request/Response Header Concept

These headers are used to perform bit-level operations inside message units. IBM identifies an exhaustive list of RHs to accomplish a variety of tasks. They perform functions such as

• To provide a format indicator

• To indicate sense data

• To indicate beginning of chain

• To indicate the end of a chain

• To indicate the types of a response

• To request a larger window

• To indicate the beginning and end of a bracket

RH formats are dependent on the type of session used. Details provided by these formats are used in the formatting of SNA data.

5.9 SNA Commands

SNA commands differ according to the type of LU and session used. Some commonalities exist in theory, but specific commands can differ.

The theory of command flow can be exemplified as follows. Assume that a terminal user wants to sign onto TSO. What is the theoretical operation and commands that flow between the two? Figure 5-10 depicts this scenario.

The following scenarios describe some of the commands that flow between the terminal user and TSO:

1. A user enters TSO and it is received by VTAM. VTAM sees this as a character-coded logon.

2. A logon exit is scheduled for the primary logical unit.

3. After the PLU receives control of the logon exit, the PLU passes an open session request to the SSCP.

4. As a result of the open session request, the BIND command is sent to the secondary logical unit (SLU).

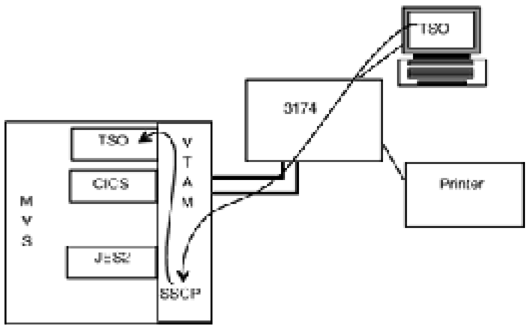


Figure 5-10
Conceptual view of a TSO user.

5. Assuming that the terminal sends back a positive response, the session is bound.

6. If a negative response is returned, a BIND failure command is generated.

IBM has devoted an entire manual (GA27-3136, *SNA Formats*) to describing SNA commands. However, SNA commands fit into the request or response structure explained above. Some examples of SNA commands and their functions are

ACTLU	Activate logical unit
ACTPU	Activate physical unit
DACTLU	Deactivate logical unit
DACTPU	Deactivate physical unit
BIND	Activate a session between LUs
CDINIT	Cross-domain initiate sent between two SSCPs
CINIT	Control initiate; request the PLU to send a BIND
LUSTAT	Used to send status information
NOTIFY	Used to synchronize awareness of an SSCP and PLU
SDT	Used to start data traffic
SESSEND	LU notification to the SSCP that a session has ended
UNBIND	Send to UNBIND two LUs

5.10 Flow Control

SNA has a method for data control. The following three paragraphs partially explain how SNA controls data.

1. *Explicit route (ER).* In SNA, an *explicit route* is a defined set of nodes and transmission groups (TG) of a path. For example, an explicit route could be subarea node X, TG2, subarea node T, TG2, and subarea D. An explicit route is the definition of a path in subarea SNA. It is a physical connection.

2. *Virtual route (VR).* These routes are logical connections between two endpoints. A virtual route is mapped atop explicit routes. Consequently, a virtual route reflects the characteristics of an explicit route. For example, in most scenarios where multiple FEPs are installed, multiple links connect them. These links are physical and defined as *explicit.* The logical route is then mapped to the route that best fits the need of the session.

3. *Class of service (COS).* This includes characteristics such as transmission priority, bandwidth, and security. The following classes can be defined according to COS: a class that provides response times reflect ing high priority, a class reflecting best availability, a class with higher levels of security, and a class for batch processing.

Transmission priority is also used with flow control. The combination of these abilities makes flow control possible in the network.

5.11 Advanced Program-to-Program Communication

Advanced Program-to-Program Communication (APPC) is IBM’s premier peer-oriented protocol. It is based on LU6.2. The flexibility of the protocol enables it to be implemented across a variety of platforms.

Origins and evolution

APPC originated in the early 1980s. It evolved from limited support and is now supported by MVS/ESA and many application subsystems operating under VTAM. Some benefits of using APPC are that one protocol can be used across a variety of architectures and that APPC provides security, offers a distributed approach to transaction processing, and offers multiple ways to create transaction programs. APPC is now widespread among MVS, VM, and VSE operating systems. It is also fundamental to Advanced Peer-to-Peer Networking (APPN). APPC is widely used in the marketplace by third-party vendors.

Conceptual overview

The idea behind APPC is peer communication between programs. This means that customized programs can be written to utilize the power behind APPC. Consider an example of two banks: one in Dallas and the other in Research Triangle Park. Now assume that daily information needs to be exchanged between a bank in Dallas and a bank in Research Triangle Park. Figure 5-11 shows an example of two programs exchanging information between the two banking institutions.

Conversations

Sessions have been explained as being a logical connection between two endpoints. A *conversation* is communication between two or more transaction programs using an LU6.2 session through a defined independent logical unit (ILU). Figure 5-12 explains this concept.

Figure 5-12 illustrates the following components: Node A, multiple transaction programs (TPs), an LU6.2 session, a point at which an ILU is defined, and a conversation between two TPs.

Figure 5-12 also depicts the idea behind a conversation. In this figure any of the transaction programs in node A can communicate with any of the TPs on node B.

There are two types of conversation: basic and mapped. A *basic* conversation provides a low-level interface for those transactions pro grams that need support for privileged functions. A *mapped* conversation provides a protocol boundary in the same way as the basic conversation does but enables arbitrary transmission of message format. System- or user-defined mappers can be used.

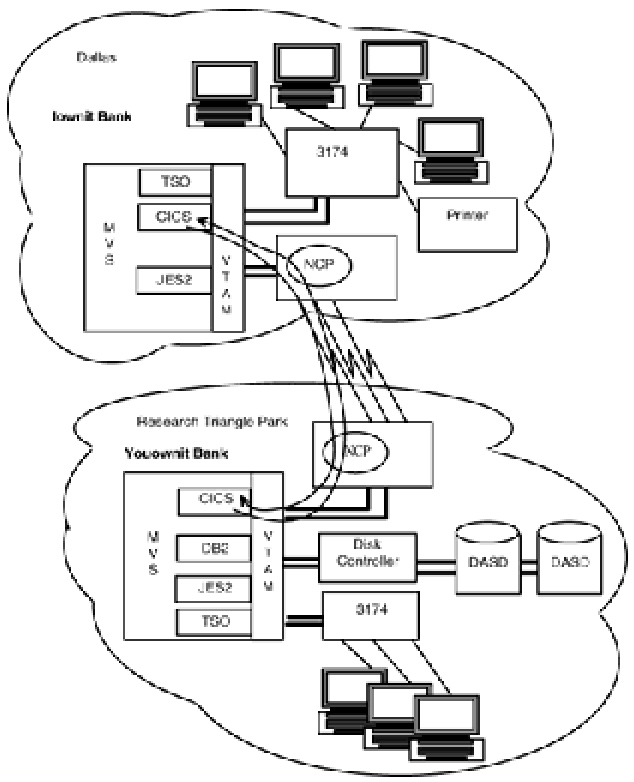


Figure 5-11
Conceptual view of APPC implementation.

Transaction program

IBM defines a *transaction program* as an application that is executed within the LU6.2 protocol. It is a type of application, typically user-written to meet the needs of a specific installation.

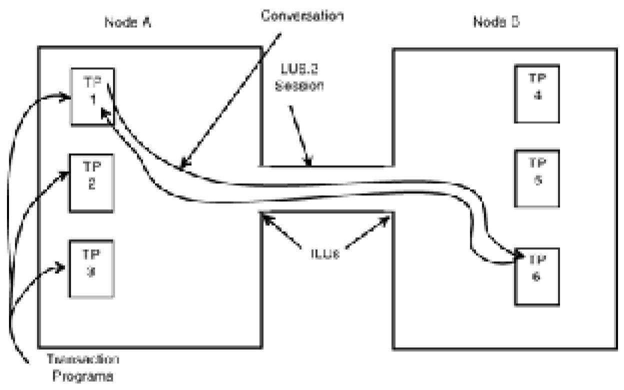


Figure 5-12
Conceptual view of a conversation.

Types of Verbs

APPC is a high-level language and uses “verbs” to achieve communication. There are two verb categories:

1. *Conversation operator verbs.* This category includes mapped, basic, and type-independent verbs. *Mapped verbs* are used by application programs and provide services for programs written in high-level languages. *Basic verbs* are used by LU service programs that provide end-user services or protocol boundaries for application programs. *Type-independent verbs* can be used with basic and mapped conversations. They provide a variety of generic services needed by both conversations.

2. *Control operator verbs.* This category includes several subcategories, such as *change number of sessions* (CNOS), session control, LU definition, and miscellaneous. *CNOS verbs* are used to change the session limit that controls the number of LU-LU sessions per mode name available between two LUs allocated for conversations. *Session control verbs* are used for session control. This includes activation and deactivation sessions and deactivate conversation groups. *LU definition verbs* define or modify local LU operating parameters. *Miscellaneous verbs* are those verbs needed but not defined to another category.

LU6.2 Session Considerations

Some of the concepts and functions of APPC are parallel sessions, single sessions, session pools, session selection, session limits, the concept of contention polarity, and winners and losers.

Parallel sessions are based on the concept of multiple pairs of sessions communicating with the same pair of LUs. Typically, one pair of TPs use a session at a given instance. LU6.2 supports multiple concurrent sessions. Applications must be capable of multiple session support, including the processors and workstations. The concept of multiple session support is called *parallel sessions. Single sessions* are defined as LUs that cannot support more than one session against a given LU in a given instance. *Session pools* are a collection of named LUs that contain active sessions which can be allocated to different conversations if required. *Session selection* indicates how a transaction program controls selection of a session. TPs cannot control session selection directly, but they can map this to a set of characteristics using the mode name parameter. A *session limit* is simply the maximum number of sessions that can be active at a given LU at one time.

The concept of *contention polarity* can be illustrated by two LUs attempting to initiate a session simultaneously. Contention polarity is a method of preventing this by defining multiple LUs for operation to function as a contention winner or contention loser. Typically, multiple winners and losers are defined in each node to prevent a state of contention.

Sync-point Processing

In LU6.2, *sync-point processing* lets transaction programs synchronize their resources at specified time periods called sync points. This is important because multiple transaction programs are exchanging data; thus TPs must be in “sync.”

Many additional LU6.2-based concepts exist. Many books have been written on the topic. A helpful one is IBM’s *SNA Transaction Programmer’s Reference Manual for LU Type 6.2* (GC30-3084).

5.12 A Perspective on Blueprints

The information provided earlier in this chapter is foundational to the blueprints that follow. Most any SNA network can be viewed in light of the following blueprints. Regardless of the network size, SNA networks contain components and abstract infrastructures that can be identified and explained.

The following blueprints are only examples. Further information can be obtained in Taylor, *Network Architectural Design Handbook* (McGraw-Hill, 1998).

Blueprint 1

This blueprint could be explained by listing its major components, which include a processor, a controller, user devices, an operating system, a communication control subsystem, and application programs. (See Fig. 5-13.)

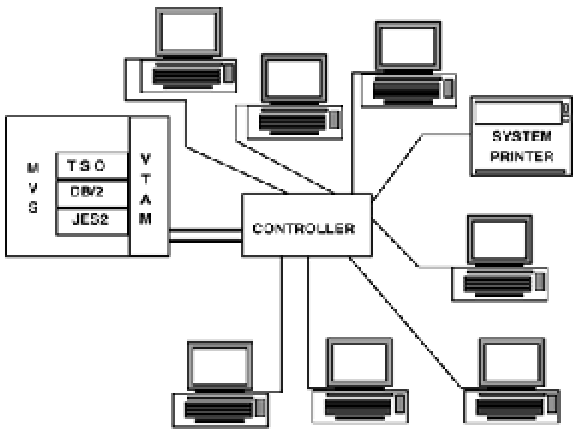


Figure 5-13
SNA blueprint 1.

This figure shows a processor, controller, and user devices such as a printer and terminals. It also shows applications operating under the MVS operating system.

Blueprint 1 shows an SNA implementation. This blueprint is typical for small SNA environments. Although the blueprint is simple, the implementation could accommodate hundreds of users.

This type of blueprint is recommended for use in scenarios where large amounts of data entry are performed and databases are maintained. An ideal example would be a customer service center where a large database is used and numerous operators need access to it. If this scenario were "real," many more controllers would be required to accommodate increased users by way of terminals.

Blueprint 1 is a good place to begin when an SNA network is needed and growth is anticipated. This blueprint is the core of most SNA network blueprints. It can be expanded on and enhanced in every way from the addition of processors, controllers, printers, terminals, and other equipment (as shown later in this chapter) to accommodate remote equipment (hardware and software) and users.

Blueprint 1 can use software different from that shown in Fig. 5-13. For example, a different operating system could be used. Additional software could be used or some of the software referred to in this figure could be replaced with software to meet the needs of a given location.

Blueprint 1 is a reference point for beginning and planning SNA networks. Since SNA networks consist of hardware and software, this blueprint is an example of hardware and software that can be used. Few SNA networks are initially large. Since this is the case, this blueprint can be considered the base blueprint for any network planning for SNA.

Blueprint 2

The next blueprint for SNA network design includes expansion of users and functionality of the network. Some may disagree that diagrams such as Figs. 5.13 and 5.14 do not represent "real" SNA networks. However, they do. Consider Fig. 5-14.

This blueprint illustrates the next logical extension to growth in an SNA network. Here there are two sites located within the same geographic plant. Consider this figure as an example to a facility in which the data center houses the actual processor and controller used to interface the terminals and printer in site A (top). Consider terminal users in site A to be the customer service department, which is in a large room of a company where numerous customer service representatives have their own cubicles.

Now consider site B (bottom) to be the technical support for the same company. In this example, sites A and B are under the same physical “roof” but located in entirely different places in the facility. Consider site B to be located on the second floor of the physical plant. Also, consider site B to be a large area where employees have desks, personal computers in addition to the 3270-based terminals, and a lab area as well in order to test product problems as they are taken by each employee and logged into the company database which is maintained in the SNA network.

Blueprint 2 is an ideal example of a single data center supporting two entirely different divisions within a company. Both customer service and technical support need access to the same database in which customer information is maintained; however, each department in this example has entirely different corporate missions with respect to the customer. This blueprint is a good example of how to lay a strong technical foundation in regard to hardware and software that can be easily enhanced and upgraded as demands grow.

Blueprint 3

SNA blueprint 3 begins a move toward a more complex SNA environment. Figure 5-15 illustrates this migration as well as the complexity involved in SNA network blueprints.

Note that Fig. 5-15 shows the processor (thus core operations) of the SNA network located in Dallas. This site also shows a controller, terminals, and system printer at the Dallas location. In addition, the Dallas location also has a communication controller which houses the NCP.

This blueprint adds an entirely new twist to SNA blueprints. The use of a communications device means a more complex configuration for the processor’s telecommunication access method. Where an NCP is used in conjunction with VTAM, considerably more intelligence is required to make the network function.

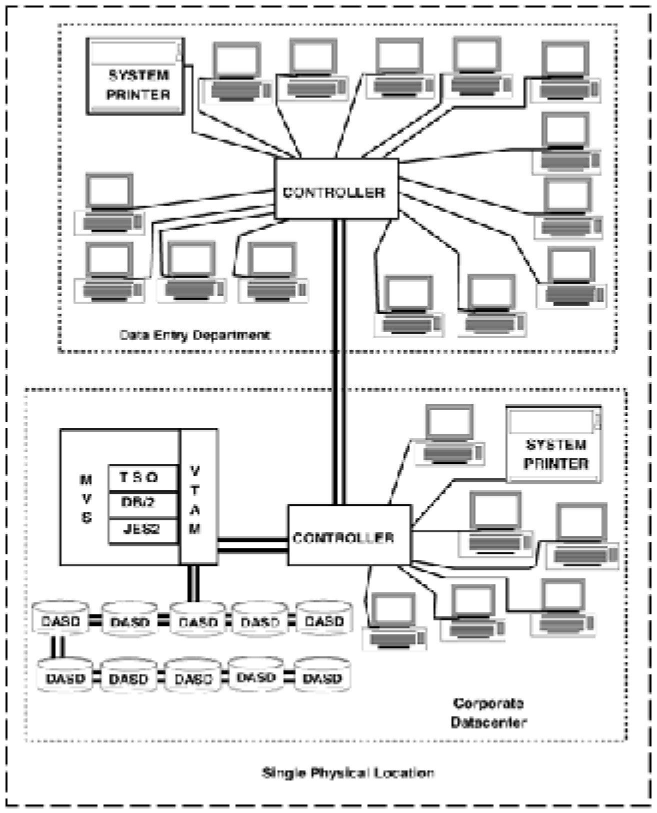


Figure 5-14
SNA blueprint 2.

This blueprint seems to be fairly simple; however, it is not. The remote location in San Jose shows a communication controller with an NCP, controller, terminals, and printer. Technically, the San Jose location has the same functionality as the Dallas location. This blueprint is common in many locations where SNA networks are implemented.

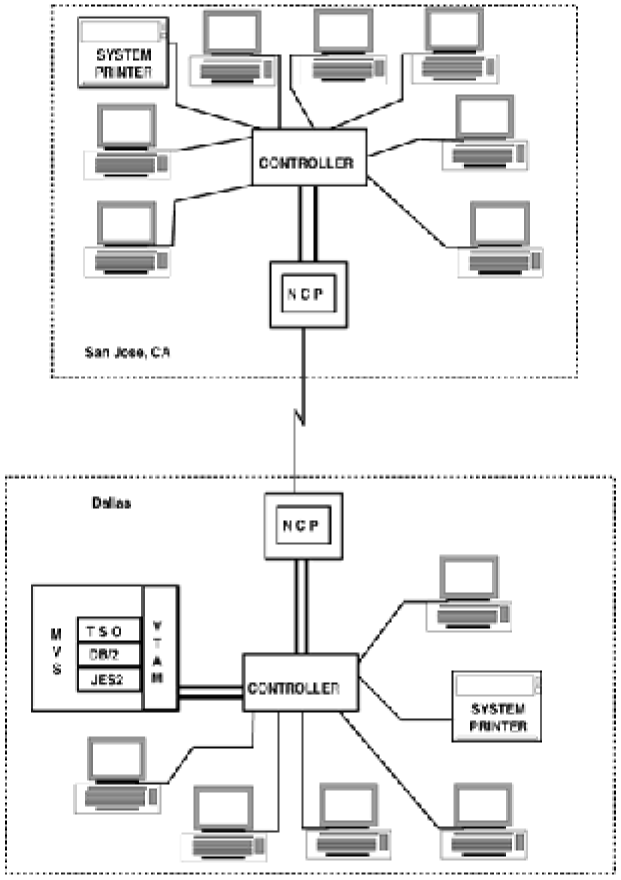


Figure 5-15
SNA blueprint 3.

Blueprint 3 introduces three basic topics for consideration:

1. Operation of such an environment presupposes the existence of a staff to implement and maintain it. It is fair to say that most sites today where this blueprint applies could have staff capable of implementing and maintaining it. However, knowledge of VTAM does not necessarily imply knowledge of NCP. These are two powerful and significant software packages that are complex and require time to master.

2. This blueprint does not explicitly show any network management. In most sites where this blueprint would adequately reflect the hardware and software, network management may in fact not be required. I have worked in a blueprint 3 scenario in which knowledgeable staff could extrapolate functionality of the network without any formal network management software. I do not mean to discourage the reader from using network management products; I simply mean that a sufficiently small SNA network, even though it can be physically dispersed, can be implemented and managed to some degree.

3. This blueprint shows a variable not present in the former blueprints: data-link communication through lines outside the facility in which the processor lies. Regardless of whether the link between Dallas and San Jose in this example is dedicated, the simple fact that distance separates the processor and the remote site requires consideration.

The matter of distance between locations presents some degree of uncertainty regarding manageability. For example, all things being equal, no significant problems should arise. However, line quality, maintenance of the data communication lines themselves, and other factors contribute to some degree of uncertainty.

This blueprint also implies some degree of accessibility to the NCP at the San Jose site if the processor in Dallas is not functional. This blueprint also implies that no redundant links exist between NCPs.

In summary, this blueprint brings link management, remote user operation, and growth in the remote (San Jose) location to mind. This information alone suffices to suggest that functional SNA represent more than simple implementations of hardware and software.

5.13 Summary

SNA networking was presented in this chapter, and these insights make it much easier to understand. SNA is a complex topic. If you really want to learn it well, I suggest that you spend time rereading the first five chapters until the information seems to jell in your mind and you can understand the practical hardware and software offerings in terms of the SNA terminology and concepts presented here.

It takes time to learn any computer architecture. This information is presented to bring the bulk of important topics together and to synthesize it. Many well-written IBM manuals are available; I suggest that you spend the time and money to obtain them. They will be able to take you to the inner depths of SNA.

6
SNA Internetworking with APPN

Advanced Peer-to-Peer Networking (APPN) is a peer-oriented architecture. It predates most of the personal computers (PCs) used in the marketplace today. Most applications in use today are peer-oriented; however, this has not always been the case.

In 1986 IBM announced support for T2.1 node support with System 36 (S/36). That same year IBM introduced its *SNA Type 2.1 Node Reference Manual* (SC30-3422). It defined the beginnings of APPN implementation. Nodes can be understood best by knowing the services they provide or the characteristics that represent them. Both the architectural design and the characteristics supported reflect or define a node type.

It is best to view APPN from this abstract viewpoint because it does not matter (to a certain degree) which host or model number is or is not a device. The core of APPN is that the devices use LU6.2, not that a certain device exists. The question to ask is "Which functions and features are supported by a certain device with respect to APPN?"

6.1 Origins and Evolution

Version 1 can be characterized by low-entry networking end nodes or Advanced Peer-to-Peer Networking end nodes. According to the *IBM Type 2.1 Architecture Manual* (SC30-3422), published in 1991, these two references to an end node are synonymous.

IBM documentation also states that "a T2.1 node is that node which uses protocols that require less system requirements." A T2.1 node provides peer connectivity and session-level connectivity using LU6.2 protocols.

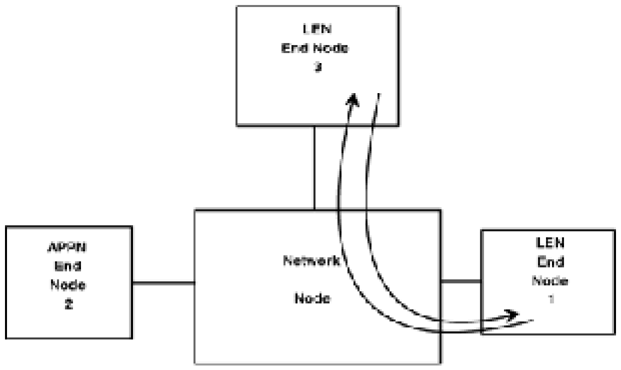


Figure 6-1
Conceptual view of a network node.

APPN version 1 can best be described as an evolution. According to the *Type 2.1 Architecture Manual*, three node types were identifiable with T2.1 architecture: APPN network node (NN), APPN end node (APPN EN), and LEN end node (LEN EN).

APPN NNs are capable of performing intermediate session routing, performing directory searches and route selection, and providing LU-LU service for their local LUs. (See Fig. 6-1.)

APPN ENs can perform limited directory services, register their LUs with an NN, and be attached to multiple NNs. (See Fig. 6-2.)

LEN ENs cannot register their LUs with an NN, must have predefined remote LUs via the system, and use T2.1 protocols *without* APPN enhancements. (See Fig. 6-3.)

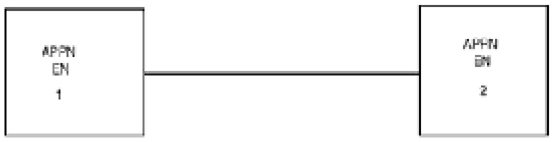


Figure 6-2
Conceptual view of two APPN end nodes.

Figure 6-3
Conceptual view of two LEN end nodes.

Traces of version 1 can be seen through IBM product offerings from 1986 to 1992. An example of one such offering supporting APPN is the AS/400, announced in 1988. Other products were announced during that period, but a significant break with version 1 came in 1992.

APPN version 2 can be identified by IBM's *APPN Architecture Reference* (SC30-3422), published in March 1993. In these 1000-plus pages, IBM clearly defines extensions to earlier APPN and LEN networking.

It seems IBM started to bring APPN together in 1992 with the announcement of VTAM version 4 release 1. In short, VTAM 4.1 (shipped in May 1993) permits VTAM to participate with other nodes in an APPN network and appear as another peer node. Technically this capability was available in VTAM version 3 release 2, and then as support for causal connections in VTAM 3.3, according to the VTAM manuals.

APPN is rooted in T2.1 architecture. It has extended beyond that; nevertheless, its beginnings can be traced back to the mid-1980s with T2.1 node architecture. APPN has evolved into its present offering and it seems IBM will continue to enhance it thus perpetuating its evolution.

APPN simplifies network definition. It also permits dynamic route selection. APPN also provides a distributed directory service. This function determines remote LUs that may be known locally only by name. This means that manual definitions for routes or location of remote LUs is not required.

APPN implementations can select routes according to user-defined criteria. A component in each NN called a *control point* (CP) is used to determine the best route from the initiating LU to the destination LU. APPN also supports *intermediate session routing,* in which data are routed through the NN for sessions that do not originate or terminate with that NN. Transmission priority is based on the class of service (COS) specified by the user.

6.2 APPN Node Types

APPN's approach to networking parallels that of the client/server. This architectural nature lends APPN to a "peer"-oriented network. The pivotal issues concern what node types exist, what functions they perform, and what additional APPN option sets exist. The remainder of this section discusses these points.

6.2.1 APPN Network Node

A major role of the NN is performing the function of *server*. In this context, other nodes participate as *clients*. This client/server concept is somewhat similar to TCP/IP clients/servers. An NN functions as a server to the end nodes attached to it. The NN and its attached end nodes are considered a domain. NN services include directory and routing services, intermediate LU-LU routing, end-node management services, LU-LU session services, support for any APPN or LEN node attachment with the same network ID, functioning as a *server* for clients, and supporting SNA subarea boundary nodes.

6.2.2 APPN End Node

In light of the client/server parallel that APPN purports, an end node functions like a client. End nodes support LU6.2. Without an NN, ENs can communicate only via LU-LU sessions with the partner LU locat ed in an adjacent node. However, with an NN an EN can communicate with remote LUs. These nodes have the ability to inform an NN of their local LUs. ENs can have active links to multiple nodes at any given time, but an EN can have CP-CP sessions with only one NN at a given instance. ENs can have attachments to multiple NNs in case one NN fails. (Examine Fig. 6-4.)

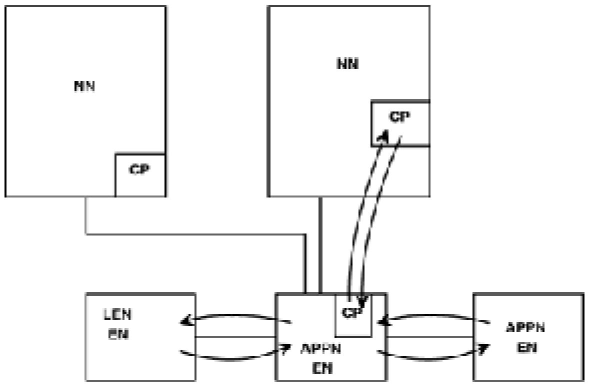


Figure 6-4
Conceptual view of CP-CP session.

This type of node can make a connection to any LEN or APPN node. An EN cannot have CP-CP sessions with another EN.

6.2.3 LEN End Node

This type of node implements basic T2.1 protocols; no APPN enhancements are included. The LEN end node is not capable of having a CP-CP session. Connections with destinations must be predefined. An LEN EN communicates with remote LUs through system definition. At system definition time, the CP name of its adjacent node is defined and indicates how the local LUs can be accessed. The LEN EN accesses some remote LUs by the services provided via the NN server functions. (Consider Fig. 6-5.)

6.2.4 Peripheral Border Node

These nodes do not support intermediate network routing, but they do support directory services, session establishment, route selection, and session establishment of LUs between adjacent subnetworks. Figure 6-6 is an example of this type of node.

6.2.5 Extended-Border Node

This type node supports intermediate network routing, but the subnetworks must be predefined. Additionally, they provide directory ser vices, session establishment, and route selection, and they permit the partitioning of a subnetwork into two or more clusters.

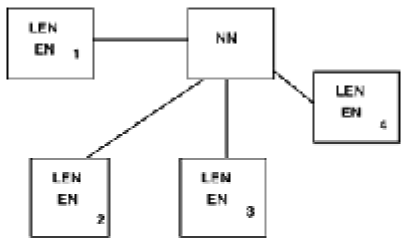


Figure 6-5
LEN ENs defined to the NN.

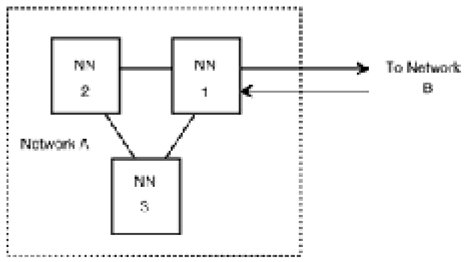


Figure 6-6
Conceptual view of a peripheral border node.

Figure 6-7 depicts four networks with an LU-LU session between peripheral networks via intermediate networks with extended boundary node function support.

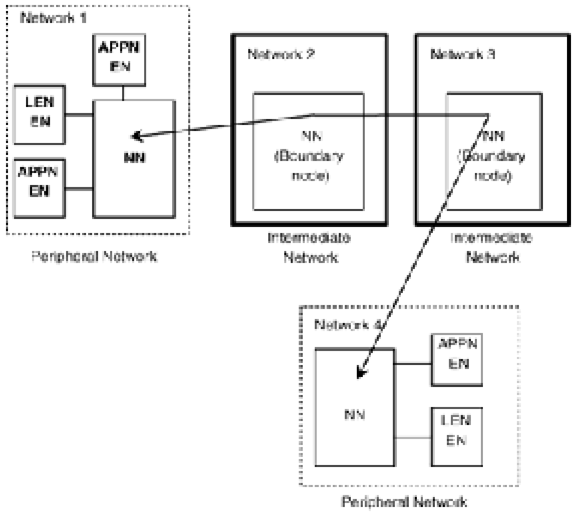


Figure 6-7
Conceptual view of an extended-border node.

.2.6 APPN Subarea Interchange Node

This is a T5 node in SNA feature/function. It permits connectivity between APPN networks and SNA networks. An *interchange node* (IN) achieves this by mapping routing, directory, session setup, and route selection for both network types. Figure 6-8 is an example of this.

6.2.7 Migration Data Host

Beginning with VTAM 4.1, a host node could emulate an APPN end node in an APPN network, support subarea connections, and perform Cross Domain Resource Manager functions. It does not participate in broadcast searches in APPN networks.

The migration data host is primarily a host dedicated to processing applications. It not only functions as an EN in an APPN network but also supports subarea connections in traditional SNA.

6.2.8 High-Performance Routing Node

High-performance routing (HPR) is fundamentally an extension to APPN as defined by IBM. Operationally, HPR is implemented into APPN network nodes or APPN end nodes. Practically, HPR implemented in APPN provides a high-performance method for routing, and the realization of the routing function is enhanced because of the high-speed links that connect. HPR, in APPN, provides a method to control congestion in the network and enables enhanced throughput in the network. Another function of HPR in APPN is its ability to perform routing of sessions away from nodes or links that have become unavailable.

A key component of HPR is the *Rapid Transport Protocol* (RTP). This protocol is connection-oriented and full-duplex. Consequently, these characteristics enable higher speeds within the bandwidth in which the technology is used. A function of RTP is the nondisruptive path switch, which provides a sort of dynamic path allocation ability. Another key function of RTP is the error recovery it employs from end to end. This is performed in all network links. A by-product of this end-to-end error recovery is the congestion control mechanism built into RTP.

Another key component of HPR is the *Automatic Network Routing* (ANR) protocol, which minimizes storage and processing requirements for routing functions. Part of these functions include fast packet switching. Many functions that make ANR possible occur at RTP connection endpoints. Another characteristic of ANR is lack of session awareness, which results in absence of the requirement for maintenance of routing tables related to session connectors. ANR is characteristically a source routing protocol. Hence, required routing information is carried in the network header in each packet. In a sense this could be viewed as “on-the-fly” routing, as this author refers to it.

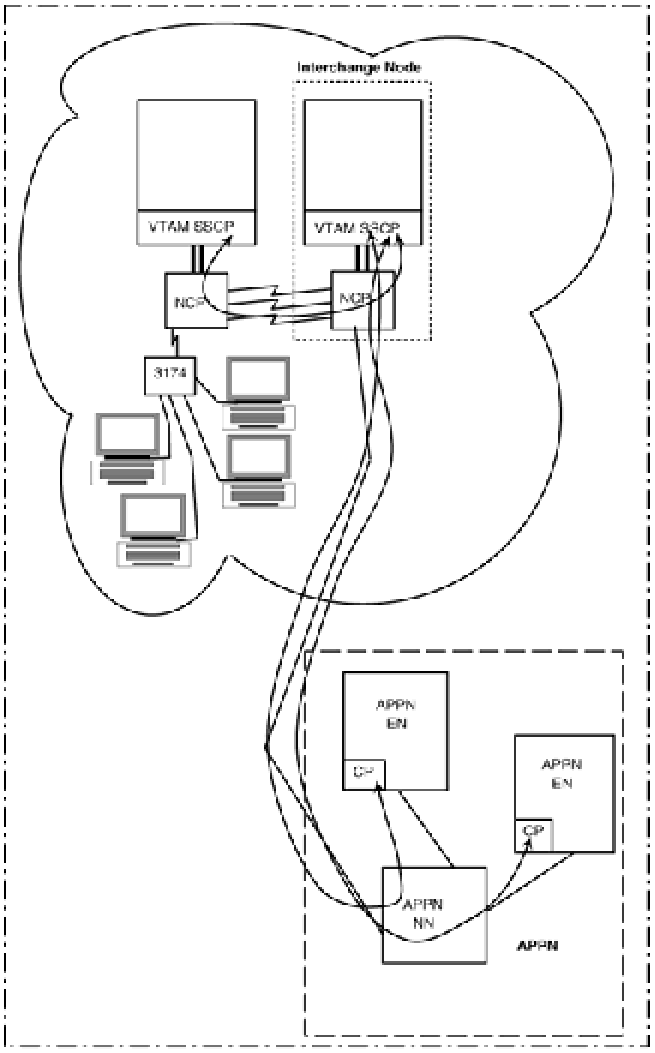


Figure 6-8
APPN and subarea SNA via an interchange node.

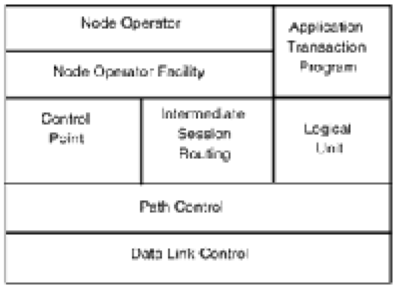


Figure 6-9
Conceptual view of APPN node structure.

6.3 APPN Node Structure

At the core of APPN lie the three nodes mentioned previously: APPN network node, APPN end node, and LEN end node. These nodes are built around the structure displayed in Fig. 6-9.

Components of this node are data-link control, path control, logical unit, intermediate session routing, control point, and node operator facility.

Each component, which has subcomponents and multiple functions, is explained below.

6.3.1 Data-Link Control

The *data-link control* (DLC) is the interface with the link connection. It provides data-link protocols. The functions of DLC are to provide communication establishment between nodes, maintain the synchronization between the two, issue acknowledgments, perform error recovery when required, and sequence data flow. Figure 6-10 shows how the DLC appears conceptually.

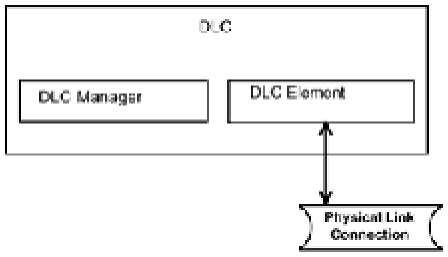


Figure 6-10
Conceptual view of data-link control.

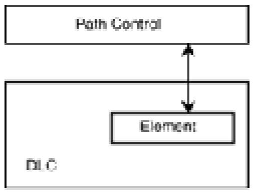


Figure 6-11
Role of the element.

This part of the node is responsible for communication with the physical link. It consists of two components. The DLC element is responsible for the following functions: moving data to the physical medium, performing retransmissions, moving data to and from other DLC elements, managing the DLC and path control (PC) boundary, and receiving traffic from the session.

Figure 6-11 depicts how the element functions within the node.

The DLC manager performs the following functions: activating and deactivating the DLC element, activating and deactivating links, passing parameters to the control point (CP) when a station becomes operative or otherwise, and controlling the boundary between the CP and the DLC.

Figure 6-12 depicts the relationship between the DLC manager and other node components.

Figure 6-13 shows how the DLC communicates with the medium, internally, with the session, and with the control point.

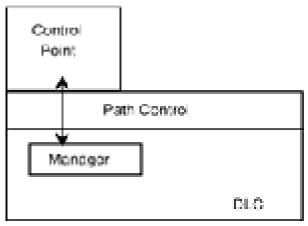


Figure 6-12
Functionality of the manager.

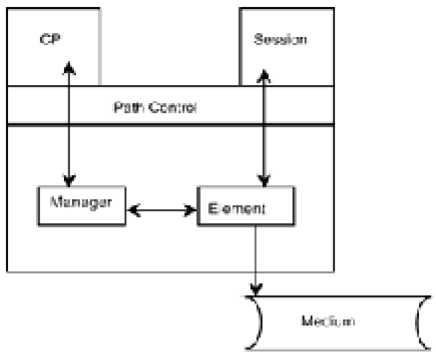


Figure 6-13
Functionality of DLC as a whole.

6.3.2 Path Control

Similar to the DLC, the *path control* (PC) component in the node has an element and a manager. Functions of these PC components include the following:

1. *Element functions.* The element portion of the PC performs functions such as error checking on messages received from the DLC element; generation of segments for outbound messages; converting messages from messages received from the DLC; prioritizing message transmission to the DLC component; and routing messages between the PC manager, the half-session (HS), the session connector (SC), and the DLC component. Figure 6-14 depicts the relationship between the PC components and the DLC component.

2. *Manager functions.* The manager portion of the PC performs functions such as session connection and disconnection, stopping outbound data traffic on notification, and interaction with the control point (CP). Figure 6-15 shows the relationship between the PC, the CP, and the parts that aid in session establishment.

General characteristics of the PC include routing messages between destination nodes and LUs residing in the same node. The PC routes messages from the DLC to the appropriate component, including the CP, LU, or the intermediate session routing (ISR) component.

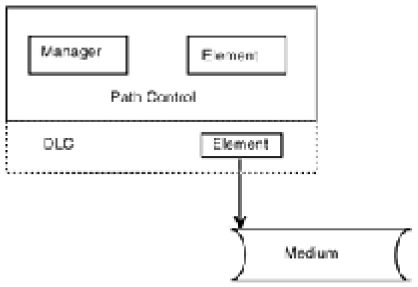


Figure 6-14
Conceptual view of path control.

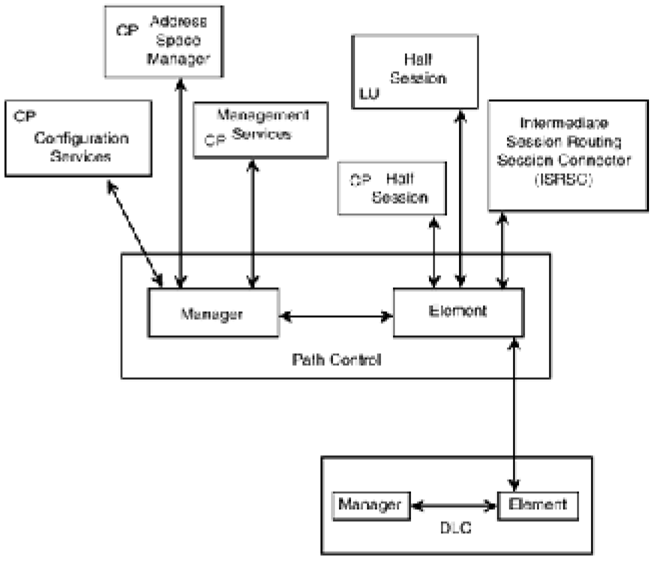


Figure 6-15
Relationship between PC, CP, and session establishment.

6.3.3 Logical Unit

The LU has many ports that enable it to communicate with the PC, LUs in another node, and other components within the same node. Figure 6-16 is a conceptual view of an LU correlated to the APPN node structure as a whole.

The LU serves as a port (addressable point) for application transaction programs. The components that make up an LU include the LU-LU half-session, session manager, presentation service component, resource manager, and service transaction programs.

LU-LU Half-Session

This part of the LU comprises two components: data flow control and transmission control. The half-session component controls local and remote LU communications. The data flow control part of the half-session creates request/response headers (RHs), ensures that proper RH parameters are in place, ensures proper function management (FM) profile for the session, manages bracket protocol, flushes rejected brackets, and is responsible for generating chaining.

The transmission control part of the LU performs session-level pacing, examines received sequence numbers for possible BIU errors, reassembles request/response units (RUs), enforces the exchange of cryptography verification when it is used, enciphers or deciphers session cryptography when used or required, and provides reassembly for RUs that have been segmented.

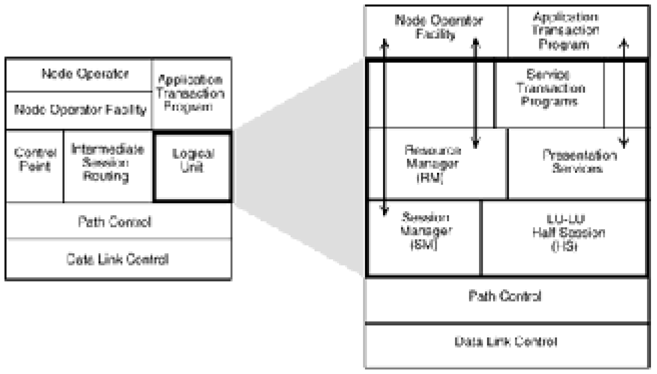


Figure 6-16
The LU with respect to node structure.

Session Manager

The session manager sends and receives the BIND, creates half-session instances, connects half-sessions to the path control, supplies session parameters during the BIND exchange, negotiates parameters during the BIND exchange, and informs the resource manager when session outage occurs.

Presentation services component

The presentation services component calls a transaction program, loads a transaction program, keeps the SEND or RECEIVE state alive with the transaction program, puts data into logical records, maps transaction program data into mapped conversation records, confirms logical record length, generates function management (FM) headers for an ATTACH, and provides error information.

There are other functions in addition to these; however, those listed here cover the major operations of this LU component.

Resource Manager (RM)

The resource manager works in conjunction with presentation services and conversations flowing between transaction programs. Some basic functions of the RM include the creation and destruction of presentation service instances and of conversation resources, connection of conversation resources to the half-session and to presentation services, maintaining data structures, enforcing session-level security, and generating the FM Header 12 (security header).

Service Transaction Programs (STPs)

These programs make up the transaction services layer. They can be used to change the number of sessions (CNOS). They also interact with the node operator facility (NOF).

Figure 6-17 shows the structure of the *intermediate session routing* (ISR) facility.

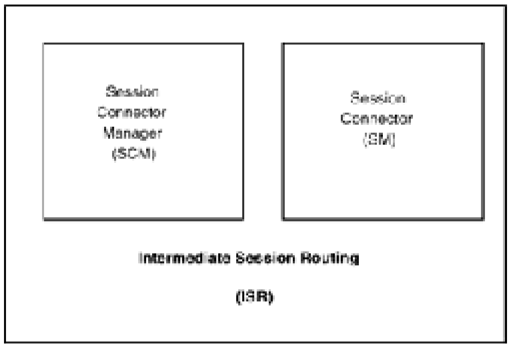


Figure 6-17
Conceptual view of ISR component.

The ISR consists of two components: the session connector (SC) and the session connection manager. An SC is allocated for each session and performs routing, pacing, reassembly of basic information units (BIUs), monitoring of the session for errors, and intermediate reassembly.

This component is responsible for routing session traffic through intermediate nodes. The companion component in the ISR is the *session connection manager* (SCM), which performs the following functions: *intermediate* BIND and UNBIND processing; creating, initializing, and eliminating session connectors, connecting SCs to the PC, and buffer reservation.

Figure 6-18 shows the correlation between ISR components and other components in the node.

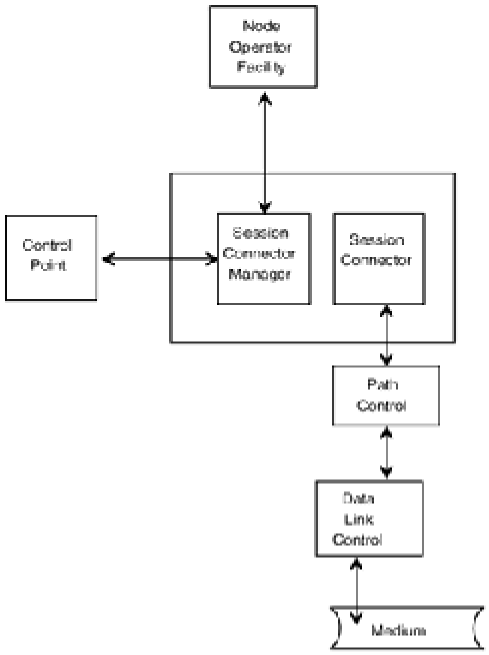


Figure 6-18
ISR communication with other components.

6.3.4 Control Point

The *control point* (CP) manages resources within a node. The CP uses CP-to-CP sessions to exchange management information. According to IBM documentation, the CP may be merged with the LU; this is an implementation issue. If this is done, certain implications do apply. For purposes here, the CP is treated as a separate entity.

Another way of defining the CP is based on the IBM *APPN Architecture* manual (SC30-3422). The definition in the glossary states:

> (1) A component of an APPN or LEN node that manages the resources of that node. In an APPN node, the CP is capable of engaging in CP-CP sessions with other APPN nodes. In an APPN network node, the CP provides services to adjacent end nodes in the APPN network. (2) A component of a node that manages resources of that node and optionally provides services to other nodes in the network. Examples are a system services control point (SSCP) in a type 5 node, a physical unit control point (PUCP) in a type 4 subarea node, a network node control point (NNCP) in an APPN network node, and an end node control point (ENCP) in an APPN or LEN end node. An SSCP and an NNCP can provide services to other nodes.

The focus here is on the components and their functions within the CP. Figure 6-19 is a conceptual example of how the CP appears.

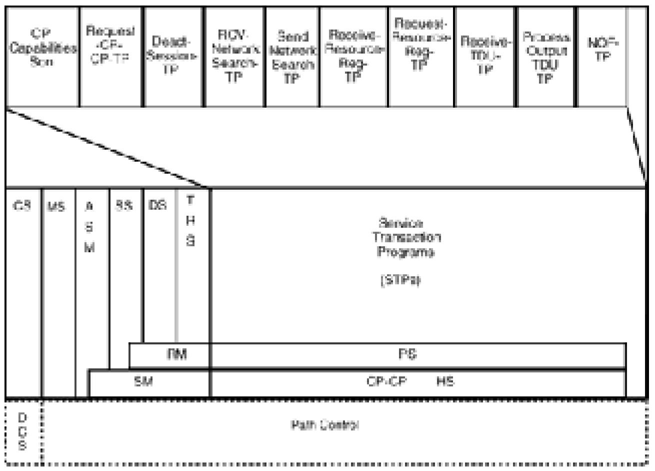


Figure 6-19
Control point components.

The following explains the component function as it appears in Fig. 6-19.

Configuration Services (CS)

This component manages physical-link connections of the node itself. Its functions include

• *Link activation.* The CS exchanges information with the DLC. Exchange Identification parameters (XIDs) are passed to ensure that each node's abilities are understood by the other. Figure 16.19 shows the CS relative to the DLC and the PC.

• *Link deactivation.* When a link is deactivated, the CS performs cleanup functions and then notifies the appropriate components within the node.

• *Link queries.*

• *Exchange of XID3.* This can occur if link or node characteristics change while the link is active.

• *Link definition verification.* The CS saves this definition during link activation and deactivation.

Management Services (MS)

This component handles alert level at a local level. Other components in the node log alert information in the local node. This component communicates with a number of other components within the node such as the ISR, the NOF, and the PC.

Address Space Manager (ASM)

The ASM handles the address spaces related to each local transmission group (TG) within a node. In APPN, a TG is synonymous with a *link.* In each address space a local form session identifier (LFSID) is defined. Additionally, one address space correlates with a TG (link).

The ASM designates session addresses, activates and deactivates address spaces, frees address spaces on request, routes nonrelated session data, and paces BINDs via adaptive pacing. If BINDs are received segmented, the ASM assembles them.

The ASM communicates with the PC, NOF, and other components within the node.

Session Services (SS)

SS focuses primarily on initialization and termination of both CP-CP and LU-LU sessions. Specific functions SS performs include procedure correlation identifiers [also known as *fully qualified procedure correlation identifiers* (FQPCIDs)], initiation of sessions, CP-CP session activation and deactivation, session termination, and monitoring of the active number of sessions.

Directory Services (DS)

DS functions depend on the node type. It functions differently with NNs and ENs. Basically, the DS maintains the node directory within that node, but this is not the case in LEN end nodes. It does, however, provide the ability to search and update directories in other nodes throughout the network. Other functions it performs include

• *End-node searches.* Three types of searches can be identified in end nodes (ENs): a search initiated locally, a search initiated from a remote node, and sending of a search request to an NN. In this latter case the NN actually performs the search; in a sense it functions as a proxy.

• *Network node searches.* DS examines its own directory to determine whether resources are owned by that node or by its associated end nodes.

• *End-node updates.* Local directory updates are performed via system definition.

• *Network node updates.* This is achieved in one of three ways: (1) as a result of systems definition, (2) when dynamic updates occur as a result of communication with the NN server's client nodes, and (3) after network search completion updates may be performed by data cached during the search.

• *End-node registration and deletion.* This function occurs only if the end node is authorized for this function.

• *Network node directory maintenance.* An NN performs updates and deletions on the basis of information received from nodes in which their registration applies.

Topology and Routing Services

This component of the CP performs three primary functions:

• *Class-of-service management* (*COSM*). The COS manager keeps the database updated. It notifies route selection services (RSS) when a class of service changes.

• *Route selection service* (*RSS*). This component functions differently in NNs and end nodes. In an NN the RSS determines preferred routes within the network and determines what transmission priority to use on selected routes. It also updates the topology database to reflect the most current topology and specifies routes computed by the COS requested. The TG from origin to destination is selected by the RSS. In an end node, RSS selects a TG and transmission priority.

• *Topology database management* (*TDM*). This function is based on the node type. In an end node the TD manager maintains the topology database. In an NN it broadcasts to the network once changes are made locally. It performs what is called a *periodic broadcast*—a broadcast throughout the network in intervals of approximately five 24-h days. It also *deletes* resources from the database if no data have been received about a resource in 15 days. It also responds to remote queries.

Service Transaction Programs (STPs)

STPs exist in an APPN node. They exchange information over CP-CP sessions. They do not exist in LEN end nodes. Ten transaction programs (TPs) are described here:

• `CP Capabilities/Sent Outside the Node (SON)`. This is the only SS TP used to attach by a remote node. Its purpose is twofold: to send the PC capabilities to another participating node and to perform processing for session outage.

• `Request_CP_CP_TP`. When the SS wishes to start a CP-CP session with another node, this TP is used to invoke the `Request CP Capabilities TP`.

• `Deact_Session_TP`. This deactivates a CP-CP session to a specific node.

• `Receive_Network_Search_TP`. This program receives a locate search from an adjacent APPN node.

• `Send_Network_Search_TP`. This is an APPN node that sends a locate search request/reply to a remote directory service.

• `Receive_Resource_Registration_TP`. This is where an APPN NN sends registration and deletion variables to an APPN end node pertaining to specified resources.

• `Request_Resource_Registration_TP`. This occurs when an APPN end node sends registration and delete information to the target APPN NN.

• `Receive_TDU_TP`. This occurs when an NN TRS communicates with a `TDP_TP` receiving information from an adjacent node.

• `Process_Output_TDU_TP`. TDM sends a signal to the `Process_Output_TDU_TP` when a local node wants to broadcast topology information.

• `NOF_TP`. This is involved in the start-up of a node operator facility.

6.3.5 Node Operator Facility

This component initializes the CP and ISR on starting the node; functions as the user interface for the CP, ISR, and LUs; and enables the following functions: activating and deactivating links; creating and deleting LUs; ascertaining status information; retrieving database information; and defining directory information, local and remote LUs, node characteristics, session limits, transaction programs, links, TP start-up, and other CP names.

The NOF is discussed in more detail later in this chapter.

6.4 Directory Services

The concept behind the directory services (DS) component is the responsibility for resource searches for the local node and those throughout an APPN network. Directory services is responsible for the registration of resources to network nodes (NNs) where they function as a server for directory services.

The DS component of the control point itself has three major components: (1) directory database function, (2) CP status function maintenance, and (3) network search function. (See Fig. 6-20.)

The directory database function (DDB) performs lookup and maintenance for directory services.

The CP status maintenance function keeps a log of other CPs that wish to communicate with the CP in that node. In NNs it keeps track of end nodes (clients) and also maintains information as to other NNs with which it can establish CP-CP sessions.

The network search function component sends and receives resource search request to and from other nodes in the network.

Understanding the terminology is important for examining the DS component.

6.4.1 Directory Service Terminology

Authorized node When this term is used in conjunction with DS, it means that information sent about itself is accepted. An unauthorized node cannot use certain protocols and is denied registration of its LUs in the distributed directory.

Border node This is an NN connecting APPN networks that maintain different databases reflecting their topology. Peripheral border nodes support directory services, session setup, and route selection between networks of different identifiers. A border node does not support intermediate routing. An extended-border node provides session setup, directory services, and routing through a boundary of different networks where these networks have different topologies.

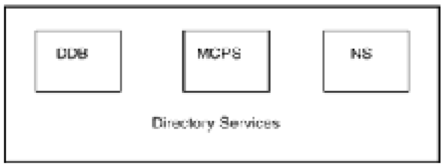


Figure 6-20
Conceptual view of directory service components.

Central directory server (CDS) This function resides in an NN and function differs from that of directory services (DS) because the CDS maintains all resource information within a network. More than one CDS can exist in the same network.

CP send/receive session The DS uses CP-CP sessions between APPN EN and the NN server. This session carries the `Locate_Search` function.

Distributed directory database (DDD) Directories of resources exist throughout the APPN network. The concept of DDD is the collective whole of the databases throughout the network.

DS user A component in a node which uses the DS.

Local Directory Database (LDD) This refers specifically to the local directory database in a given node.

Locate Search This is the method a DS finds resources not in that node. A Locate Search can be either a *broadcast search*—sent throughout the entire network, a *directed search*—sent to a known location for verification, or a *domain search*—an NN communication with its client ENs to verify resources in a given location.

Subnetwork A collection of nodes which have common characteristics such as a common network address or a database in common.

6.4.2 Directory Service Functions

The function of the DS component is present in each T2.1 node; the degree to which it is exploited depends on the node itself. The DS functions in an NN are as follows: (1) a database is maintained of local resources and resources that have been cached because of a locate search function, (2) the DS is able to determine the location of a specified resource, (3) the DS registers resources in a domain via the NOF or an end node (client), (4) the DS deactivates CP-CP sessions if the node is in a state of deadlock with respect to sending a locate search, and (5) the DS provides support for intermediate nodes in locate searches.

Directory service in an APPN EN provides these functions: (1) once notification of a CP-CP session failure has occurred, it cleans up any outstanding searches; (2) it sends and receives locate resource searches with the NN; (3) it registers resources with the NN; (4) it supports CP-CP sessions with the NN; and (5) it maintains a directory database of local resources and adjacent node resources.

6.4.3 Directory Database Function

The directory database (DD) is a distributed database containing lists of resources throughout the APPN network. For example, ENs keep information about their resources in their DD. An NN maintains a *node operator facility* (NOF), defines directory entries of resources in that node and in the nodes it serves. A major function is keeping the database within the storage requirements for that node. The directory database contains different entries.

Types of Entries

Fundamentally, the directory database maintains information about its own resources. Additionally, the following types of entries may be found:

Domain entries including resources in that domain, but located in one of the client nodes (end nodes).

Other domain entries maintaining information about resources in other domains as its name implies.

Other network entries keeping information about resources that can be reached by a different net ID.

Origination of Entries

As mentioned previously, a distributed directory contains individual, local databases viewed as a whole. Information gets into these directories by either NOF definition or APPN end-node registration with an NN server via a CP-CP session. Information may also be entered by the caching function as a result of the locate search function.

The Network Search Function

The *network search function* (NSF) maintains the protocols used while searching the distributed database. This function also maintains control of the transport directory services. It also enforces logic with regard to the sending of directory service messages. However, its primary purpose is locations of network resources and control flow throughout the network of request and replies.

Within the NSF a need may arise to send a request to another node asking for information about the directory in that node; in this case the message that flows is called a *locate search.* Three types of locate searches can be identified: (1) one-hop search, (2) directed search, and (3) broadcast search. The *one-hop search* is a locate search request exchanged between an APPN end node and an NN.

A *directed search* traverses a predefined path from one NN to another NN. In this case the originating NN calculates a CP-CP path to the target node and adds routing information to the search. This works because each NN on the path uses this information to select the next hop. By functioning in this manner, it ensures that the most direct route to the destination node is obtained.

The broadcast search is used by NN to send a locate search request to multiple CPs. Two common types of broadcast searches are domain and network. The *domain broadcast search* sends a locate search for the resource to adjacent APPN end nodes. Because more than one reply may return, the directory service used the first positive reply. The *network broadcast search* is sent to all NN nodes. It is used to ascertain a resource location when it cannot be found otherwise. It is used as the last attempt because this type of search permeates the network with request for the location of the resource.

Locate searches can carry non-DS information. If this is the case, it is used by other CP components normally for the use of transport control data. When this non-DS information is used, the user is theoretically an application. For example, session services could use this to transport information variables. Some examples of some information capable of being transported as non-DS information are fully qualified procedure correlation identifier (FQPCID), destination LU, mode name, class of service (COS), originating LU, and endpoint vectors.

6.4.4 Central Directory Server

The Central Directory Server (CDS) resides in an APPN network. More than one can exist. The CDS accepts registration of resources from other network nodes. Once information is received from the registrations of other nodes a central directory is maintained.

6.4.5 Directory Entry Contents

The contents of a directory entry depends on node type. Briefly the contents found in directory entries are resource name, resource type (NN CP, EN CP, LU information), EN control point (uses a pointer to an adjacent CP status control block), information about the hierarchy (LU entries for NN servers, adjacent entries for an LU entry, the “child” LU entries for adjacent EN control points), classification (home, cached, or registered), and information as to whether the resource has been or can be registered with the central directory server.

6.5 Topology and Routing Services

Topology and routing services (TRS) are present in each NN and in a lesser form (with respect to functionality) in APPN ENs and LEN ENs. In NNs TRS creates and maintains the COS database and is responsible for maintaining a copy of the network topology database.

In end nodes the TRS creates and maintains the local topology database. It is also responsible for the COS table.

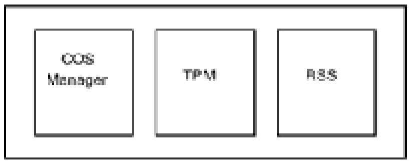


Figure 6-21
Conceptual view of topology and routing services components.

The TRS has three components: (1) class-of-service manager, (2) route selection service, (3) and topology database manager (see Fig. 6-21).

The *COS manager* enables translation of a mode name to a COS name. This is a basic function for NNs; however it is optional for end nodes. *Route selection* service computes routes. It selects the path from origin to destination. Technically, it computes the most efficient route between nodes in an APPN network. The *topology database manager* maintains the COS and topology databases. In NNs the TDM maintains the network topology database, and, on one or the other ends, it maintains local topology information.

6.5.1 Class-of-Service Database

The COS database exists in all NNS and in those end nodes which support them. The COS database contains mode names that include a pointer to a COS name; COS names which have COS definitions representing characteristics of the node, transmission priority, and weight assigned; and weight index structure for computing the actual transmission group weight.

Each COS entry in the COS database contains some basic information, such as COS name, transmission priority, transmission group characteristics, security-level, cost per byte, and propagation delay.

6.5.2 Route Selection Service

A *route* in APPN is a path between two endpoints. This includes the intermediate components that may exist, such as links, NNs, domains, and transmission groups.

The following minimum criteria are used to determine the best route in an APPN network:

• Route characteristics must be known.

• All possible routes are calculated.

• If a resource is not acceptable, it must be excluded; hence, determination of this factor about resources must be performed.

• All resources that will be used during the route must be accounted for and calculated accordingly.

6.5.3 Topology Database

The topology database contains information about the logical structure of the APPN network and about all nodes in the network, transmission groups, intermediate transmission groups, and other pertinent information.

There are two types of topology database: (1) network and (2) local.

Network Topology Database

A *network topology database* is maintained in all NNs. This database includes information on NNs connections to other NNs and connections to virtual routing nodes. Each NN participating in the APPN network is aware of this database because the database is on each NN.

The structure of the network topology database includes two categories: *node table*, which includes information about the node such as CP name, network ID, characteristics, and resource sequence number; and *TG table,* which includes information about transmission groups. Some of that information includes CP-CP session support, status, a pointer to the TG vector, and a pointer to the weight (amount of resource requirement).

Local-Topology Database

Local-topology databases are located in end nodes. This database contains information about each endpoint attached to that node. It is created and maintained by the topology database manager. The local-topology database is used when no CP-CP session exists in an NN, to establish sessions to predefined LUs, and to send information to the NN for the route selection process.

6.6 Configuration Services

This component of the control point (CP), known as *configuration services* (CS) (see Fig. 6-22), is responsible for managing local node resources like links to other nodes.

CS performs a number of functions, including node configuration definition (data-link type, ports, adjacent nodes and links), link activation and deactivation, and nonactivation of an XID exchange. On node initialization, CSs receive the node name, network ID, link station support information, and information concerning TGs.

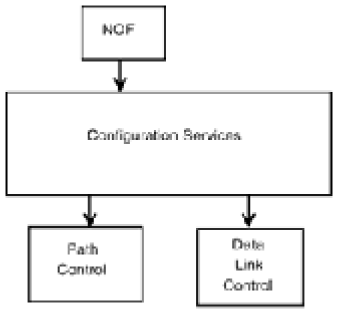


Figure 6-22
Conceptual view of configuration services.

Through CS, the NOF defines the basic node configuration. First is the data-link type. This CP component communicates with the data-link control manager for definition purposes. Ports are also defined by the NOF via CSs. They are considered hardware.

Port type is defined: switched, nonswitched, or as a shared-access facility. Information includes buffer size, limits, timeout values, TG characteristics, and any associated DLC process.

Link stations may be defined at activation time, or their parameters can be negotiated; either way they must match. Nodes require system definition for their local link station. These definitions include role (primary or secondary), address or defined as negotiable, inactivity timer, retry limit on mode setting, and modem delay limits.

6.7 Management Services

The concept of management services (MS) is implemented in each T2.1 node and is known as the *control point management services* (CPMS)—a component of the CP in T2.1 nodes. Its functionality is straightforward.

The NOF sends messages to the CPMS. These messages are converted into local management services, where they are carried out by the receiving component in the T2.1 node.

With respect to the CP, management services interacts with each of the following: (1) session services, (2) configuration services, (3) session manager, (4) resource manager, (5) address-space manager, (6) topology and routing services, (7) directory services, and (8) presentation services.

Some information CPMS can receive from CP components and other components, respectively, is listed below. Some information must be solicited and some need not be.

1. Information about currently active sessions—problems detected in a node by the CP

2. Domain information

3. Information about currently active LU6.2 sessions

4. Information about session conversations

5. Unsolicited information about problems related to this component

6. Information about routing information

7. Locating network resources for the CPMS

8. Providing LU6.2 protocol boundary information

The LU session manager provides information about currently active sessions. The LU resource manager provides information about conversations across an LU6.2 session. The LU management service component provides unsolicited information about the LU. The session connection manager provides information about data passing through the node.

The path control manager reports unsolicited information about any problems detected at this component. The data-link control manager provides a vehicle for testing resources such as links and modems.

Other management information is ascertainable, but the point here is that management services actually interacts with all T2.1 node components, not only the CP.

6.8 Address-Space Manager

The *address-space manager* (ASM) resides in NNs and APPN ENs. Fundamental functions of the ASM include management of session addresses [also known as *local form session identifiers* (LFSIDs); these addresses are used for routing data traffic and local path control] and of flow control of session activation messages (the BIND), informing the appropriate session manager component when a link fails, and routing session activation and deactivation messages.

The ASM is created on initialization of the node by the NOF. Once this is performed, the ASM is notified by the NOF of the CP name, network ID, and nature of BIND assembly supported.

6.8.1 Functions of the ASM

For communication to occur between LUs or CPs and other CPs, a local form session identifier (LFSID) must be allocated by the ASM. By performing this function, the ASM achieves address control in the node. The ASM bases the LFSID on path control instance identifier for that session. The ASM maintains an address-space list and a list of assigned LFSIDs in use.

Address spaces are defined by the ASM in relation to the transmission group attached to the node. The ASM assigns an address space consisting of a number large enough to allocate sufficient LFSIDs for that TG.

Whenever a TG is activated or deactivated, the ASM is informed. At that time the ASM creates or destroys the tables used to control that TG's address space. ASM handles the assigned address space by dividing it into groups. These groups consist of 256 LFSIDs. However, the ASM allocates the LFSIDs only as necessary.

6.8.2 LFSIDs

LFSIDs are 17-bit session identifiers used by the path control to route session traffic and have two components:

1. *A 1-bit assignor indicator.* Every ASM in the nodes connected by a TG selects an LFSID from the TG address space with a different value, so no duplication exists.

2. *A 16-bit session identifier.* This is further broken into an 8-bit identifier (considered high) and an 8-bit identifier (considered low).

An LFSID assigned to a session maintains its active state as long as the session exists. The ASM is the component that terminates the association between the LFSID and the session once the ASM receives notification that an UNBIND or response to UNBIND is delivered from the path control component.

This LFSID is used because on each session hop between two endpoints each node uses distinct session identifiers to identify a session; consequently, the term *local form session identifier* (LFSID) is used.

Considerably more detail accompanies this topic, but this is not a section on programming! For further information on the topic, refer to IBM's *APPN Reference* manual number SC30-3422.

6.9 Session Services

In general, the session services (SS) functional part of the control point (CP) aids in generating unique session identifiers, LU-LU session ini tiation and deactivation, and CP-CP session activation and session deactivation.

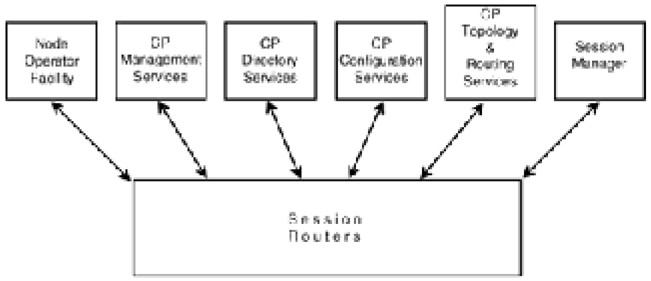


Figure 6-23
Relationships between session services and other components.

Figure 6-23 depicts the relationship between SSs and other components.

6.9.1 Fully Qualified Path Control Identifier

The session identifiers generated by this part of the CP should not be confused with the LFSIDs generated by the address-space manager (ASM) mentioned previously. The session identifiers described here are called *fully qualified procedure correlation identifiers* (FQPCIDs).

Session services assign a *network* unique session identifier, known more commonly as the FQPCID. This FQPCID correlates request and replies between APPN nodes and identifies a session for problem determination, auditing, accounting, performance, cleanup, and other purposes, or to perform recovery actions.

FQPCID is assigned at the originating node and it is of fixed length. It contains an 8-byte session identifier field that includes the network qualified name which generated it.

6.9.2 LU-LU Session Initiation and Deactivation

With APPN NNs, APPN ENs, and LEN ENs, LUs can initiate sessions and respond to the session initiation requisition from another LU, or CP, for that matter. The session activation request (also called a BIND) is sent by a particular LU; that LU is considered the *primary LU* (PLU). The BIND recipient, on the other hand, is called the *secondary LU* (SLU). The LUs go into session once the BIND is sent and is received by the target, and the target LU sends a response (RSP) to the BIND back to the sending LU. This is an “active” session.

Some of the information specified in a BIND request includes network-qualified name of the PLU or SLU, route traversed through the network to the SLU, the FQPCID, and the maximum request/response unit size.

On the other hand, a session is stopped when an UNBIND is sent to the target and the target responds with a response (RSP) back to the originator of the UNBIND. This is session *deactivation,* or an “inactive” session.

Sometimes the terms PLU and SLU are substituted with the terms origin LU (OLU) and destination LU (DLU).

6.9.3 CP-CP Session Activation and Deactivation

CP sessions are always LU6.2, which means that the possibility for contention may exist if there is a CP in two nodes. *Contention* is best explained by both CPs attempting to establish a session with the other at the same time. Since this possibility exists, the question of how to overcome this scenario is in order.

Contention can be overcome by what are called contention “winners” and “losers.” Each CP has contention winner LUs defined and generally the same number of contention losers defined. Because CP-CP sessions are established in parallel, each CP has winner and loser LUs defined. With this configuration, contention can be overcome.

Establishment of CP-CP sessions begins when the session services notifies directory services that a session is pending active. Then the directory service queues network operations that may involve the CP session LU. Session services notifies the resource manager to attempt activation of a winner LU with the destination LU in the target node. The session services is once again invoked to assign FQPCID having a codename of CPSVCMG.

The following information is part of information that flows across CP-CP sessions: topology database updates, session activity, request for data management support, reply to a request for data management support, and a resource search capability.

CP-CP session deactivation may occur for one of two reasons: (1) in the event of *normal deactivation,* which usually means that the node, or its partner, no longer requires the session; or (2) if an *abnormal CP-CP session termination* occurs, which could be the result of protocol violation during the session or a link failure or, in remote cases, both.

6.10 Node Operator Facility

The NOF is the interface between the operator and the T2.1 node. Its purpose is to enable operators to control node operation. A node operator can be either a human, a command list for execution, or a transaction program. Either of these entities can perform node operator functions. A *human operator* can execute a specific dialog between the NOF and the individual and make changes possibly not anticipated or capable of by a program. A *command list* is simply a file with a list of node operator commands to be executed. The NOF interpreter logs the commands and responses from the NOF and maintains this for future reference. *Transaction program control* is used in remote operations. This works when a transaction program actually issues a command against the NOF which is in a remote location. These three forms of communication with the NOF are illustrated in Fig. 6-24.

6.10.1 NOF Functions

Briefly NOF functions include creating other components in the node, issuing commands to initialize the node, converting commands to sig nals capable of being understood of components within the node, starting a log of commands issued and the results of issuing these commands, receiving results from node components, routing signals to the appropriate node components, and managing unsolicited messages from any node components.

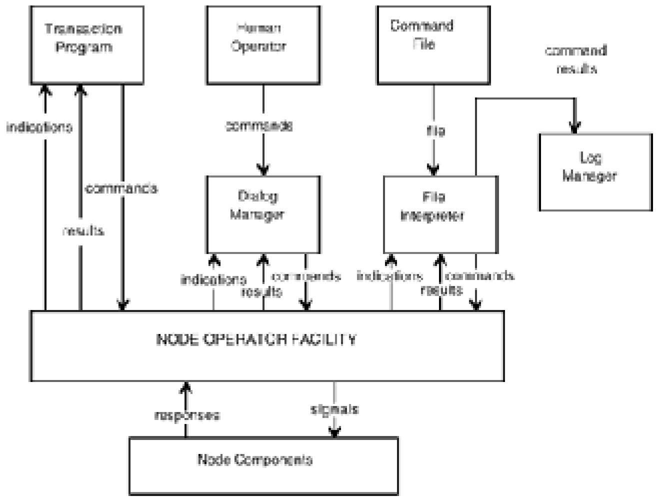


Figure 6-24
A perspective of the node operator facility.

On initialization, the NOF creates the following components in order: (1) address-space manager, (2) session services, (3) directory services, (4) configuration services, (5) management services, (6) topology and routing services, (7) session connector manager, and (8) session manager of the control point (CP).

6.10.2 Commands Listing and Function

When a command is entered and received, the NOF parses the command into a form understood by the NOF. At this time the syntax is verified.

The following are node operator commands which are architecturally defined; brief descriptions of their functions are given in the right column.

`CHANGE_SESSION_LIMIT`	Changes the session limit
`DEFINE_ADJACENT_NODE`	Defines an adjacent node to a local node
`DEFINE_CLASS_OF_SERVICE`	Changes or updates the COS
`DEFINE_CONNECTION_NETWORK`	Defines a connection network to the local APPN node
`DEFINE_DIRECTORY_ENTRY`	Defines or updates directory entries
`DEFINE_DLC`	Defines a data-link control
`DEFINE_ISR_TUNING`	Adds or updates the session connector manager
`DEFINE_LINK_STATION`	Defines a connection to a link station
`DEFINE_LOCAL_LU`	Used to define a new LU
`DEFINE_MODE`	Used to create a new mode definition for a local LU
`DEFINE_NODE_CHARS`	Defines or updates current characteristics in the local node
`DEFINE_PARTNET_LU`	Defines or changes a local LU
`DEFINE_PORT`	Defines a port to a local node
`DEFINE_TP`	Defines or changes a local LU's operation with a local transaction program
`DELETE_ADJACENT_NODE`	Removes a definition of an adjacent node
`DELETE_CLASS_OF_SERVICE`	Removes a COS definition
`DELETE_CONNECTION_NETWORK`	Removes connection network from a local node
`DELETE_DIRECTORY_ENTRY`	Removes an entry from the directory services
`DELETE_DLC`	Removes a data-link control instance
`DELETE_ISR_TUNING`	Removes one or more session connector managers
`DELETE_LINK_STATION`	Removes adjacent link station definition
`DELETE_LOCAL_LU`	Removes a local LU from a node
`DELETE_MODE`	Removes a mode definition from a local LU
`DELETE_PARTNER_LU`	Remove definition that a local LU uses with a remote LU
`DELETE_PORT`	Removes a port definition in the local node
`DELETE_TP`	Removes a local transaction program definition
`INITIALIZE_SESSION_LIMIT`	Initializes the number of sessions allowed
`QUERY_CLASS_OF_SERVICE`	Used to obtain the values defined for a COS
`QUERY_CONNECTION_NETWORK`	Used to obtain the status of a connection network
`QUERY_DLC`	Used to obtain the status of a specific data-link control instance within a node

`QUERY_ISR_TUNING`	Used to ascertain information about the session connector manager
`QUERY_LINK_STATION`	Used to obtain the status within the node of an adjacent link station
`QUERY_PORT`	Used to obtain status of a port within the node
`QUERY_STATISTICS`	Used to obtain detailed information about a link station
`RESET_SESSION_LIMIT`	Resets the number of sessions allowed
`START_DLC`	Starts a specified data link
`START_LINK_STATION`	Establishes communication between a local link station and an adjacent link station
`START_NODE`	Brings up the SNA node
`START_PORT`	Starts a specified port and local link stations
`START_TP`	Requests a local LU to start a TP in a node
`STOP_DLC`	Stops the named data-link control
`STOP_LINK_STATION`	Stops communication with a specified adjacent link station
`STOP_PORT`	Stops a specified port and associated local link stations

6.11 APPN Concepts and Traditional SNA

APPN and SNA are philosophically different. APPN is peer-oriented, using LU6.2 protocols, and is implemented across a variety of equipment. SNA has been hierarchical in nature. This meant that VTAM was involved in practically all session establishment. This began to change with VTAM version 3 release 2 and is more prevalent with VTAM version 4 release 1.

APPN and traditional SNA are becoming less clear-cut. They are evolving into a cooperative way of networking when both are present in one environment. This section explores some differences between the two and presents examples of areas where they are somewhat coming together.

6.11.1 APPN structure

APPN builds on different types of nodes that provide services such as routing, database maintenance, directory services, and end-user services. The growth in different types of APPN nodes has changed and continues to change. No longer is an APPN network considered implemented with midrange and PS/2 systems. Now, APPN can be implemented with SNA via VTAM support.

APPN uses LU6.2, an independent LU capable of initiating a BIND (request for a session with another LU).

6.11.2 SNA Structure

SNA has been built around hardware architectures and VTAM becoming the centerpiece of software for the network. SNA has been hierarchical (some also call it *subarea SNA*) in nature; but it has had support for peer operations. Now, those peer operations are expanding by embracing APPN via VTAM and the NCP.

SNA uses LU6.2, but it primarily utilizes other LU types such as LU1, LU2, and LU3, which are dependent on VTAM for session establishment. As a result, the question becomes "How can LU2s be implemented into an APPN network and access a VTAM host?"

6.11.3 APPN-SNA Mixture

Subarea SNA supports dependent logical units. SNA's roots are in this functionality. This means that an LU requesting a session with a VTAM application must have the services of VTAM (SSCP) or aid via the NCP boundary function. For LUs residing on adjacent nodes to VTAM or the NCP, they traverse the VTAM or NCP boundary function.

If VTAM is configured as an end node, VTAM cannot perform intermediate session routing. However, nodes can attach to VTAM using the boundary function of SNA. Consequently, dependent LUs must access VTAM via this boundary function.

Two terms need clarification and explanation: *dependent logical unit requestor* (DLUR) and *dependent logical unit server* (DLUS). Implementation involving DLUR and DLUS provides the following scenario. When APPN nodes are mixed in networks with nodes such as a PU2.0 device, this PU2.0 must have SSCP-PU sessions and SSCP-LU sessions. Once support for these two sessions is achieved, a dependent LU-LU session can be achieved from the PU2.0 device and a subsystem application.

To realize this in APPN and mixed subarea SNA, these data must be encapsulated within an LU6.2 session and passed to the SSCP and PU, respectively. When this is realized, the need for a T2.0 node to be directly attached, or data-link-switched, to the SSCP, thus providing SSCP access is removed. Hence, integration of PU2.0 and PU2.1 APPN and subarea dependent LU can be achieved.

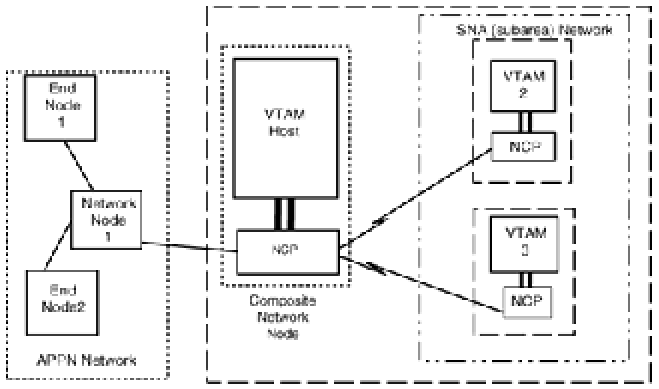


Figure 6-25
A mixed APPN-SNA network.

Figure 6-25 depicts an APPN network and a subarea network with VTAM functioning as a composite network node.

As a result of the implementation shown in Fig. 6-25, session establishment can occur between any LU in the subarea network and any LU in the APPN network. In this case APPN VTAM must be implemented and converts subarea to APPN protocols and vice-versa.

6.11.4 APPN Integration into SNA

Example 1

APPN can be implemented numerous ways. Figure 6-26 shows one illustration of APPN implementation.

In Fig. 6-26, note that two LEN end nodes are communicating via an APPC between Dallas and Houston. This peer-oriented network design shows two hosts in different locations sharing information between two different geographic locations.

Example 2

APPN can be implemented as shown in Fig. 6-27.

In Fig. 6-27, a VTAM and NCP host is acting as an intermediate node. This node is functioning as another LEN end node. This means that the CP information session flow is occurring between end nodes, therefore making the routing function more efficient.

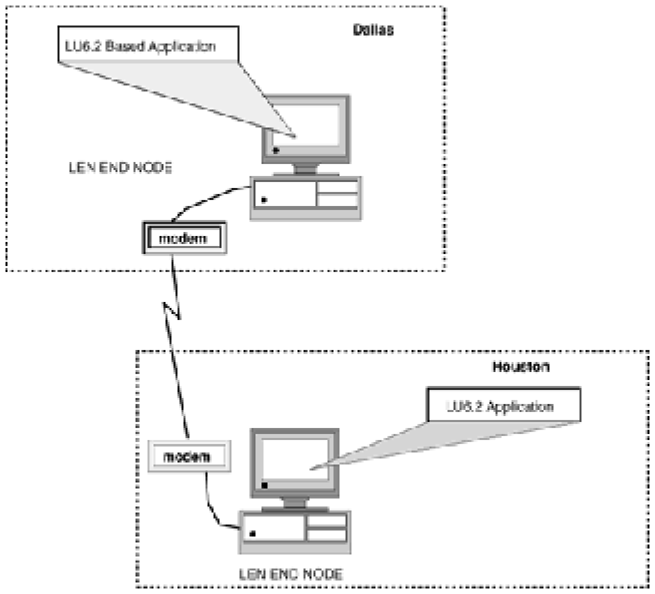


Figure 6-26
APPN-SNAintegration example 1.

6.12 APPN and SNA Summary

This chapter has presented APPN architecture and aspects of APPN-SNA integration. In many respects APPN and SNA are becoming one. The lines are increasingly blurred between pure SNA and pure APPN. The mixture of these once separate ways of networking is making solutions more robust in the marketplace for businesses worldwide.

APPN has its own architecture. It is peer-oriented, whereas SNA is hierarchical in nature. VTAM version 4 release 1 is a tool for integrating APPN and subarea SNA. This integration feature was enhanced in VTAM version 4 release 2.

T2.1 architecture consists of many components. T2.1 node components can be viewed two ways: (1) the components that make up the node itself and (2) those components that constitute the CP.

APPN and subarea SNA are different. Neither is better; they are simply different. Much more information is available about APPN. Your IBM representative can tell you how you can obtain more information on this topic.

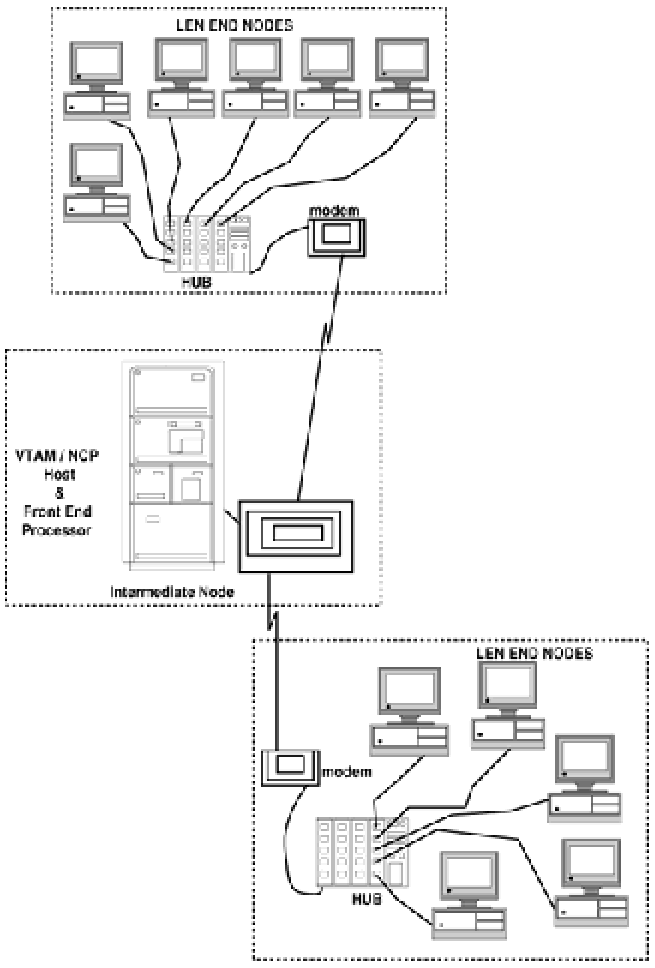


Figure 6-27
APPN-SNA integration example 2.

7
Evolution and Characteristics of TCP/IP

The *Transmission Control Protocol/Internet Protocol* (TCP/IP) is an upper-layer network protocol. It is in widespread use around the world today. This chapter presents the core components and issues related to TCP/IP, beginning with an historical perspective.

7.1 Historical Perspective

A good place to begin is in the late 1960s. An entity in the U.S. government, the Advanced Research Projects Agency (ARPA), was exploring technologies of all sorts. One of those technologies led to a need (desire) to create a network based on packet switching technology to help them experiment with what they built. It was also seen as a means of using the then current telephone lines to connect scientists and personnel in physically different locations to work together in this network.

By late 1969 the necessary components had come together to create the ARPAnet. In short order a few individuals had put together a network that was capable of exchanging data. Time passed, and additions and refinements were made to the ARPAnet.

7.1.1 The 1970s

In 1971 the Defense Advanced Research Projects Agency (DARPA) succeeded ARPA. As a result, the ARPAnet came under the control of DARPA. DARPA's strength was concentration on satellite, radio, and packet switching technology.

During this same time period ARPAnet was using a *Network Control Program* (NCP). Since the NCP was so closely tied to the characteristics of ARPAnet, it had limitations for coping with the areas of research, capabilities, and other requirements. These protocols ARPAnet utilized (viz., the NCP) were characteristically slow and had periods where the network was not stable. Since ARPAnet was now officially under DARPA's umbrella and the realization that a new approach to ARPAnet was needed, a different direction was taken.

Around 1974 DARPA sponsored development for a new set of protocols to replace the ones in use at that time. This endeavor led to the development of protocols that were the basis for TCP/IP. The first TCP/IP began to appear in the 1974–1975 timeframe. While these technical matters were in full force, another phenomenon was occurring.

In 1975 the U.S. Department of Defense (DoD) put the ARPAnet under the control of the Defense Communication Agency (DCA); the DCA was responsible for operational aspects of the network. It was then that the ARPAnet became the foundation for the Defense Data Network (DDN).

Time passed and TCP/IP enhancements continued. Many networks emerged working with and connecting to ARPAnet with TCP/IP protocols. In 1978 TCP/IP was sufficiently stable for a public demonstration from a mobile location connecting to a remote location via a satellite. It was a success.

7.1.2 The 1980s

From 1978 until 1982 TCP/IP gained momentum and was continually refined. In 1982 multiple strides were made. First, the DoD issued a policy statement adopting the TCP/IP protocol and rendering it the overseeing entity for uniting distributed networks. The next year, 1983, the DoD formally adopted TCP/IP as the standard for the protocol to use when connecting to the ARPAnet.

Early 1983, when the DoD formally discontinued support for the Network Control Program (NCP) and adopted TCP/IP protocol, marks the birth of the Internet. The term *Internet* was an outgrowth of the term *internetworking*, a technical term referring to the interconnection of networks. Nevertheless, this term has maintained its association reflecting the multiple networks around the world today.

7.1.3 The 1990s

The Internet today consists of numerous interconnected networks. The National Research and Education Network (NREN) is a dominant part of the Internet today. Other Internet networks include the National Science Foundation (NSF) network, NASA, Department of Education, and many others including educational institutions.

Commercial, educational, and other types of organizations are connected to the Internet. An industry of service providers for the Internet seems to be emerging.

7.2 Forces Contributing to TCP/IP Growth

7.2.1 Technology

The history reviewed sheds some light on the technology surrounding TCP/IP and the Internet, but does not explain certain aspects of the Internet that may aid in understanding the technological impact it had on TCP/IP.

The Internet (uppercase *I*) is based on TCP/IP as the U.S. government made it the standard. The Internet is worldwide, and all sorts of entities are connected to it. Knowing this, we can deduce that those entities connected to it are using TCP/IP. This alone counts for a tremendous amount of TCP/IP in the marketplace. And at the current rate, it is increasing rapidly.

The 1980s can be characterized as a decade of rapid technological growth. Many companies capitalized on the U.S. government endorsement of TCP/IP as the standard for the Internet and began producing products to meet this need.

This influx of TCP/IP products nursed the need for additional products. For example, in the 1980s two technologies dominated: PCs and LANs. With the proliferation of PCs and LANs an entirely new industry began emerging. These technological forces seemed to propel TCP/IP forward because TCP/IP and PCs made for a good match when implementing LANs. TCP/IP implemented on an individual basis is referred to as an *internet* (lowercase *i*).

7.2.2 Market forces

A factor that contributed to the growth of TCP/IP in the market was corporate downsizing. This may seem strange, but during the 1980s I witnessed many cases where TCP/IP-based networks grew while others shrank. Granted this was not the only reason for TCP/IP’s healthy market share, but it did contribute.

For example, I observed a similar scenario. A corporation (which I will not identify by name) had its corporate offices in the northeastern United States. This corporation had many (over 50) satellite offices around the nation. It needed these satellite offices to ensure independence for daily operations and at the same time be connected to the corporate data center. They achieved this by implementing TCP/IP-based LANs in their satellite offices, then connecting them to the data center. This example is one of many I have observed.

7.2.3 Availability

TCP/IP could be purchased off the shelf of many computer stores by the end of the 1980s. This degree of availability says a lot for a product which at the beginning of the decade was not readily available to end users.

Another factor played a role in the availability of TCP/IP. The DoD not only encouraged use of TCP/IP; they funded a company called Bolt, Deranek, and Newman to port TCP/IP to UNIX. In addition, the DoD encouraged the University of California, Berkeley to include TCP/IP in their BSD LICB UNIX operating system. This meant that by acquiring Berkeley UNIX, users got TCP/IP free. Soon after this TCP/IP was added to AT&T's System V UNIX operating system. I suppose this conveys how available it was becoming.

7.2.4 Individual knowledge

By the late 1970s and surely into the 1980s TCP/IP was installed in most colleges and many educational institutions. Since by the mid-1980s it was shipped free with Berkeley UNIX and was available there, it became dominant in learning institutions. The obvious occurred—individuals everywhere began graduating from educational institutions and if their background included computer science, odds were they had been exposed to TCP/IP.

Granted this premise, consider this. These individuals entered the workplace and began penetrating the technical and managerial echelons of corporations. When it came to contributing to a decision about a network protocol, which would be the likely choice in many cases?

In the 1980s the marketplace paid a premium for those who understood TCP/IP. By the mid-1990s the market already had considerable numbers of individuals with varying degrees of TCP/IP knowledge.

All these factors weave together to make TCP/IP as dominant as it is today. Surely other factors have contributed as well and TCP/IP has become a prevalent upper-layer protocol world wide.

7.3 Layer Analysis

In the early days of the Internet the term *gateway* became commonplace. It generally meant a connection from a specific location into the Internet. This was adequate at the time; however, confusion now abounds with the use of this term.

According to the *American Heritage Dictionary*, the term *gateway* is defined as: "1. An opening, as in a wall or a fence, that may be closed by gate. 2. A means of access." I believe the original meaning of this term was "a means of access." This is fine, and you are probably wondering why it is even mentioned. Well, today an entire industry called *internetworking* and *integration* has appeared, and with it specialized devices exist. One such device is a gateway.

A consensus among integrators and those who integrate heterogeneous networks agrees on the definition of the term *gateway*. It is a device that at minimum converts upper-layer protocols from one type to another. It can, however, convert all seven layers of protocols.

The purpose of explaining this here is simple. Throughout the presentation on TCP/IP the term *gateway* may appear. The term has such a foothold in the TCP/IP community that it is still used. Ironically, when the term gateway is used in many instances with TCP/IP and the Internet, technically the term should be *router*.

7.4 Overview and Correlation to the OSI Model

TCP/IP is an upper-layer protocol. TCP/IP is implemented in software; however, some specific implementations have abbreviated TCP/IP protocol stacks implemented in firmware. TCP/IP can, however, operate on different hardware and software platforms, and it supports more than one data-link layer protocol.

The OSI model is a representation of the layers that should exist in a network. Figure 7-1 compares TCP/IP to the Open Systems Interconnection (OSI) model.

Note that TCP/IP has three layers; network, transport, and the upper three layers combined together functioning aggregately as the application layer. TCP/IP is flexible at the lower two layers. It can be implemented in a variety of ways.

TCP/IP can operate with a number of data-link layer protocols. Some are listed in Fig. 7-1. The remainder of this chapter highlights popular components at each layer.

7.5 Network Layer Components and Functions

OSI model layer 3 is the network layer. In TCP/IP it is the lowest layer in the TCP/IP protocol suite. TCP/IP network layer components include the following:

Internet Protocol (IP) IP has an addressing scheme used to identify the host in which it resides and is involved in routing functions.

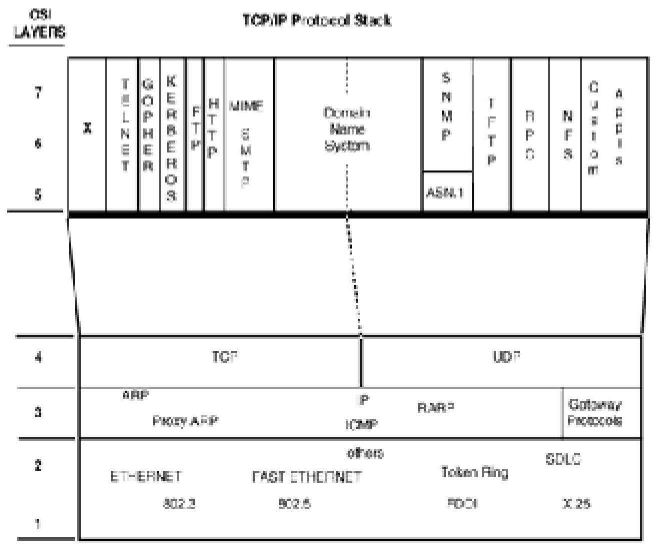


Figure 7-1
TCP/IP and OSI.

Internet Control Message Protocol (ICMP) ICMP is a required component in each TCP/IP implementation. It is responsible for sending messages through the network via the IP header.

Address Resolution Protocol (ARP) ARP dynamically translates IP addresses into physical (hardware interface card) addresses.

Reverse Address Resolution Protocol (RARP) RARP requests its host IP address by broadcasting its hardware address. Typically an RARP server is designated and responds.

Routing Information Protocol (RIP) RIP is a routing protocol used at the network layer. If implemented, it performs routing of packets in the host in which it resides.

Open Shortest Path First (OSPF) This is a routing protocol implemented at the network layer as RIP, but it utilizes knowledge of the internet topology to route messages by the quickest route.

7.6 Transport Layer Components and Functions

Layer 4 of both the OSI model and TCP/IP is the transport layer. Transport layer components include the *Transmission Control Protocol (TCP),* which is considered reliable and performs retransmissions if necessary; and the *User Datagram Protocol (UDP),* which is considered unreliable and does not perform retransmissions; this task remains for the application using its services.

7.7 Popular Application Layer Offerings

Above the transport layer in TCP/IP there are a number of popular applications, including

X A windowing system that can be implemented in a multivendor environment.

TELNET An application that provides remote logon services.

File Transfer Protocol (FTP) An application that provides file transfer capabilities among heterogeneous systems.

Simple Mail Transfer Protocol (SMTP) An application that provides electronic-mail (email) services for TCP/IP-based users.

Domain Name Service (DNS) An application designed to resolve destination addresses in a TCP/IP network. This application is an automated method of providing network addresses without the need to update host tables manually.

Trivial File Transfer Protocol (TFTP) This UDP application is best used in initialization of network devices where software must be downloaded to a device. Since TFTP is a simple file transfer protocol, it meets this need well.

Simple Network Management Protocol (SNMP) This is how most TCP/IP networks are managed. SNMP is based on an agent-and-manager arrangement. The agent collects information about a host, and the manager maintains status information about hosts participating with agents.

Network File Server (NFS) An application that causes remote directories to appear to be part of the directory system to the host which the user is using.

Remote Procedure Call (RPC) An application protocol that enables a routine to be called and executed on a server.

Custom Applications Applications that can be written using UDP as a transport layer protocol. By doing so, peer communications can be achieved between applications.

7.8 TCP/IP Network Requirements

Before exploring details of TCP/IP, basic requirements should be known for a TCP/IP network to function. For example, TCP/IP net works require *all* participating hosts to have TCP/IP operating on them, and they must be connected directly or indirectly to a common link. This may require some gateway functionality for some systems, but Fig. 7-2 is an example of a typical TCP/IP network with different vendor computers.

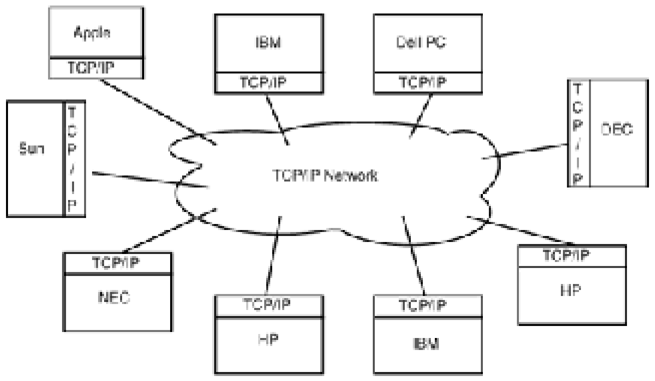


Figure 7-2
TCP/IP networking requirements.

Figure 7-2 includes vendors whose operating systems are different. They also have different hardware platforms. However, if the link to the TCP/IP network is established, the vendor computers shown in Fig. 7-2 can communicate effectively.

With this overview in mind, in the next section we will summarize detailed information about the TCP/IP protocols and applications presented in this chapter.

7.9 Summary

TCP/IP has evolved over the past few decades. Its origins were in development, education, and research entities. Many technologies and market forces converged in the 1980s and caused TCP/IP to gain market momentum.

TCP/IP is technically a four-layer protocol. It can be compared and contrasted to the OSI model. TCP/IP can operate with numerous lower-layer protocols, as we saw in this chapter.

Many of the applications that operate on top of TCP and UDP have become popular today. Their functionalities have become common in many software packages that incorporate them.

One of the many reasons for the popularity of TCP/IP is the unifying factor this protocol offers. It is possible to use TCP/IP with every major operating system in the marketplace today.

8
Common TCP and UDP Applications

This chapter presents popular applications that use TCP and UDP. These are the two transport-layer protocols used by TCP/IP.

8.1 X Window System

X, as it is known in the marketplace, is a distributed windowing system. At MIT (Massachusetts Institute of Technology) in the early 1980s developers were looking for a way to develop applications in a distributed computing environment, which was considered cutting-edge technology at the time. During their work these developers realized that distributed windowing system would meet their needs very well.

After meeting and sharing information with individuals at Stanford University who had performed similar work, these MIT developers managed to give a considerable starting point to begin this endeavor. However, the group at Stanford working with this technology had dubbed it *W*, for windowing. The individuals at MIT renamed it X simply because this was the next letter in the alphabet. The name stuck.

By the late 1980s X commanded a considerable market share specifically in a UNIX-based environment; X provides a dominant user interface in the UNIX environment, and it has spread into MS-DOS (Microsoft Disk Operating System) and VMS environments as well. One reason for its growth was nondependence on hardware and software.

X is asynchronous and is based on a client/server model. It can manipulate two-dimensional graphics on a bitmapped display. Before examining some of the operational aspects of X, consider the X layer and its relationship to the TCP/IP protocol suite shown in Fig. 8-1.

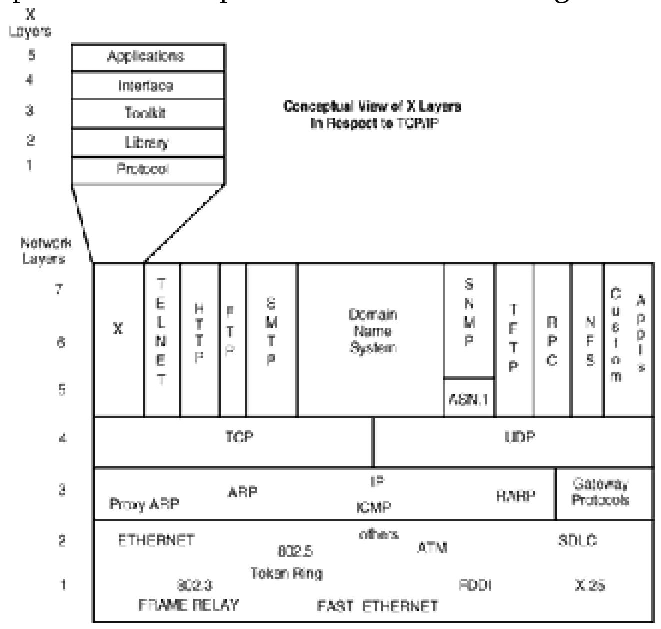


Figure 8-1
X and the TCP/IP protocol suite.

Although Fig. 8-1 shows the TCP/IP protocol suite, to help the reader understand the relationship between X and TCP/IP, the focus here is on X. X is not a transport-layer protocol; however, it uses TCP for a transport protocol.

From a TCP/IP perspective X comprises layers 5 to 7. However, X itself has five layers:

Protocol. This is the lowest layer in X. It hooks into TCP and contains actual X protocol components.

Library. The X library consists of a collection of language routines based on the X protocol. X library routines perform functions such as responding to pressing the left mouse button.

Toolkit. The X toolkit is a higher level of programming tools. Examples of support provided from this layer are the functions they provide in programming related to scrollbar and menu functions.

Interface. The interface is what a user sees. Examples of interface software are Sun, Microsystem's OpenLook, Hewlett-Packard (HP)'s OpenView, Open Software Foundation (OSF)'s Motif, and NeXT Software's interface to name only a few.

Applications. X applications can be defined as client applications that use X and conform to X programming standards that interact with the X server.

8.1.1 The X Theory of Operation

X clients and servers do not function in the same way as other clients and servers in the TCP/IP environment. In "normal" operation, a client *initiates* something and servers *serve* or *answer* the client's request. In X the concept is skewed.

An X display manager exists in the X environment. Its basic function is to start and maintain X server operation. The X display manager (Xdm) itself can be started either manually or automatically and is a client application.

An X display server (Xds) interfaces between hardware components (e.g., a keyboard or a mouse) and X client applications. The Xds operates by catching data entered and directs the data to the appropriate X client application.

The Xdm-Xds relationship can be illustrated by the following scenario. Consider two active windows on a physical display. Each window functions as a client application. From this perspective, the idea of directing data to the appropriate X client application assumes a different meaning. This architectural arrangement is required to maintain order because multiple windows may be on the display (say, four or five).

In summary, the X display manager and the X server control the operations on the display, which is what a user sees. Most entities in an X environment function as X client applications. An example of this is the Xclock, an Xterm which is an emulator, or even a TN3270 emulation software package used to access a 3270 data stream in an SNA environment.

8.2 TELNET

TELNET, a TCP application consisting of a client and a server, provides the ability to perform remote logons to adjacent hosts. The majority of TCP/IP software implementations have TELNET, simply because it is part of the protocol suite. As stated above, *clients* initiate something (in this case a remote logon) and *servers* serve client requests. Figure 8-2 shows the TCP/IP protocol suite with TELNET highlighted.

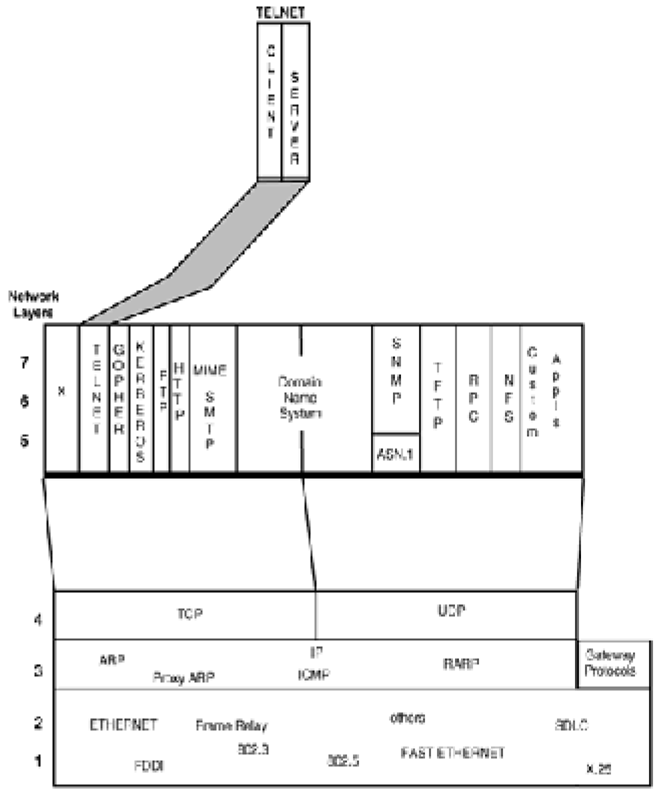


Figure 8-2
TELNET client and server.

The TELNET client/server arrangement shown in Fig. 8-2 represents practically all TCP/IP host implementations in which the protocol suite has been developed according to the request-for-change (RFC) order. Exceptions do apply, however; for example, TCP/IP on a DOS-based PC cannot implement a TELNET server because of the architectural constraints of the PC. In short, the PC cannot truly multitest, and other nuances apply as well. Furthermore, on some network devices this implementation cannot work. However, the point is that on most host implementations such as UNIX, VMS, MVS, VM, VSE, and some other operating systems the TELNET client and server will function.

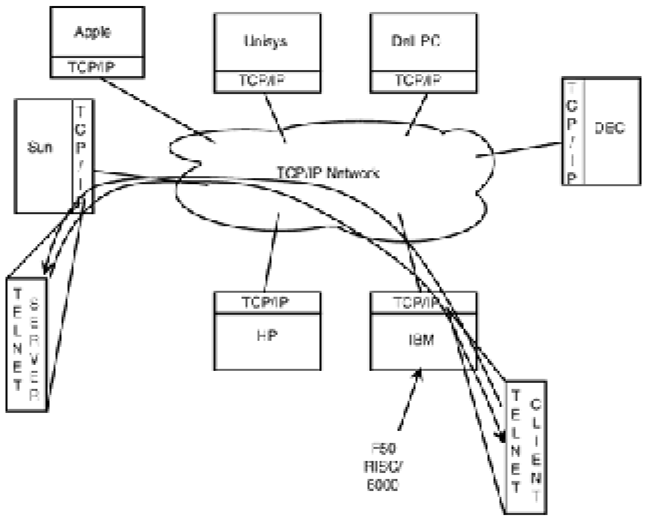


Figure 8-3
TELNET client and server interaction.

Figure 8-3 illustrates TELNET client/server interaction on different hosts.

Figure 8-3 shows a RISC/6000 user invoking a TELNET client, native to that machine because it is in the TCP/IP protocol suite. The RISC/6000 user wants to log on to the Sun host. The Sun host has TELNET in its TCP/IP protocol suite; thus the TELNET server answers the client's request and a *logical* connection is established between the RISC/6000 user and the Sun host; the RISC/6000 user perceives this as a *physical* connection.

This TELNET function works with the majority of major vendors in the marketplace today. The key to understanding the client/server con cept is to remember that clients *initiate* and servers serve client requests.

8.2.1 TN3270 client

A TELNET protocol is presented in this section for readers who wish to design a program based on this protocol. The most common program written using a TELNET (TN) protocol is an emulator application providing data translation services between ASCII and EBCDIC and vice versa. This program (application) is the TN3270 client.

TELNET (within a native TCP/IP protocol stack) has ASCII-based data, which are not compatible with EBCDIC-based SNA. In SNA, the EBCDIC goes a step further and defines *data streams.* The predominant data stream is the 3270 data stream, which is used with terminals interactively.

Because of the ASCII-EBCDIC data incompatibility, it is necessary to convert ASCII into EBCDIC, specifically, into a 3270 or a 5250 data stream. The question is where this process will take place. With the data stream dilemma between TCP/IP networks and SNA networks, this fundamental issue must be resolved.

So, how do users on a TCP/IP-based network have ASCII data converted into the EBCDIC? There are two possible solutions:

1. A raw TELNET client can be used to establish a logical connection between a UNIX or other non-EBCDIC-host and a EBCDIC-based host. If this is the case, then ASCII to EBCDIC translation will occur on the EBCDIC host (except when a gateway is used between the two and translation services are provided).

2. A TN3270 client application can be used like a raw TELNET to gain entry into the SNA environment, but a TN3270 client application performs data translation. This means that it sends an EBCDIC (3270 or 5250) data stream to the destination host. The point here is that the TN3270 client application translates ASCII data into an EBCDIC, which is usually a 3270 or 5250 data stream. In the scenario shown in Fig. 8-4 the Sun user is invoking a TN3270 client and making a connection to the MVS host utilizing the offloaded TCP/IP-activated TELNET server. Note that this figure shows multiple hosts in the network. Because of the TCP/IP configuration on the MVS host many different scenarios can exist, even from remote users.

In Fig. 8-4 the TN3270 client is shown establishing a logical connection with the TELNET server native to the TCP/IP protocol on the MVS machine. The data stream leaving the ASCII-based host is EBCDIC. This works because the data format (either ASCII or EBCDIC) is formatted at layer 6 within a network. By the time the data gets down to the interface card connecting it to the network, the data are represented by voltages or light pulses, whichever the network is based on.

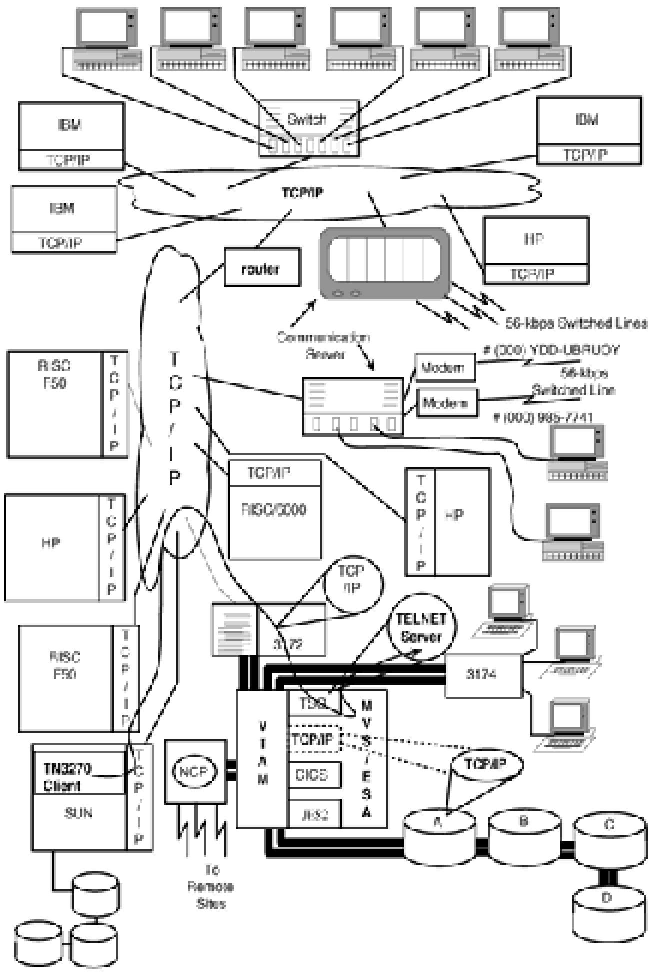


Figure 8-4
TCP/IP-based network.

The net effect of having TN3270 clients is that they do pay for themselves over a period of time. In some instances TN3270 applications are not needed and provide little, if any, benefit to the end user. Both a raw TELNET and a TN3270 client provide the user with remote logon capability, and both are client applications; the difference is merely where data translation is performed.

8.2.2 TELNET Client Usage

As mentioned previously, TELNET consists of a client and a server. A client always initiates a logical connection, and a server always answers the client's request. To use TELNET, a command must be entered to invoke the TELNET client. The command to invoke the TELNET client from the TCP/IP suite is TELNET. Assuming that TCP/IP has been installed properly and normal setup occurred, entering the TELNET command invokes the TELNET client from the TCP/IP protocol stack.

If the TELNET command is entered without a target host name, alias, or Internet address, the following prompt appears: `telnet>`. This command is generated from the TELNET client on that host. When a previous prompt appears, valid TELNET client commands can be entered against it.

8.2.3 Valid TELNET Client Commands

Valid TELNET client commands can be entered at the TELNET client prompt. If a user does not know valid commands to execute against a TELNET client prompt, a question mark (?) can be entered and a list of valid TELNET commands will be displayed. Following is an abbreviated list of valid TELNET client commands, with brief explanations:

? close-	This closes a current connection if one is established.
? display-	This command will display the operating parameters in use for TELNET. Because these parameters can be changed, they are site-dependent.
? mode-	This command indicates whether entry can be made line by line or in one-character-at-a-time mode.
? open-	This command is required prior to the target host name in order for session establishment to occur.
? quit-	This command is entered to exit the telnet> prompt, thus exiting TELNET.
? send-	This command may be entered to accommodate some of the special characters that may need to be transmitted.
? set-	This command is used to set certain parameters to be enforced during a TELNET session.
? status-	This command provides information regarding the connection and any operating parameters in force for the TELNET session.
? toggle-	This command is used to toggle (change) operating parameters.
? z-	This command will suspend the telnet> prompt.
? ?-	This command prints valid TELNET commands that can be entered against the telnet> prompt.

8.2.4 TELNET Use

Using TELNET is straightforward once the newness of the technology wears off. Learning TELNET is easier when one understands basic TELNET operation, TELNET commands, and how to log on to hosts appropriately.

Since TELNET is part of the TCP/IP protocol suite, it does work with other components in the suite. For example, if one attempts to establish a remote logon with a target host, and after a period of time a response such as “host unreachable” is displayed on the terminal, a non-TELNET-related problem may exist. In this example, the “host unreachable” message comes from the *Internet Control Message Protocol* (ICMP) component, which is an integral part of the IP layer, providing messages responding to different conditions. Here, a destination host is not reachable by the TELNET client. The obvious question is Why? There are two possible reasons for this. The host may be (1) unreachable because of a break in the physical cable connecting the hosts together or (2) located on another segment of the network and inaccessible at the moment. There are other possibilities as well.

When messages such as these appear, they are usually generated from the ICMP portion of the TCP/IP suite. It would be helpful to familiarize yourself with common messages and understand their meaning. This can prove to be a valuable troubleshooting tool.

8.3 File Transfer Protocol

The *File Transfer Protocol* (FTP) is a file transfer application that uses TCP for a transport protocol. FTP has a client and a server as does TELNET; operationally they are similar. The difference is that TELNET enables remote logon whereas FTP permits file transfers.

FTP does not actually transfer a file from one host to another—it copies it. Hence, a copy of the original file has been put on a different machine. Figure 8-5 depicts this scenario.

Figure 8-5 shows a user on a Sun host performing two steps: (1) the Sun user executes FTP HP and a logon is established and (2) the Sun user issues the FTP command GET and designates the filename as FILEABC. The line with the arrowheads indicates that the file is copied from the HP disk to the Sun disk.

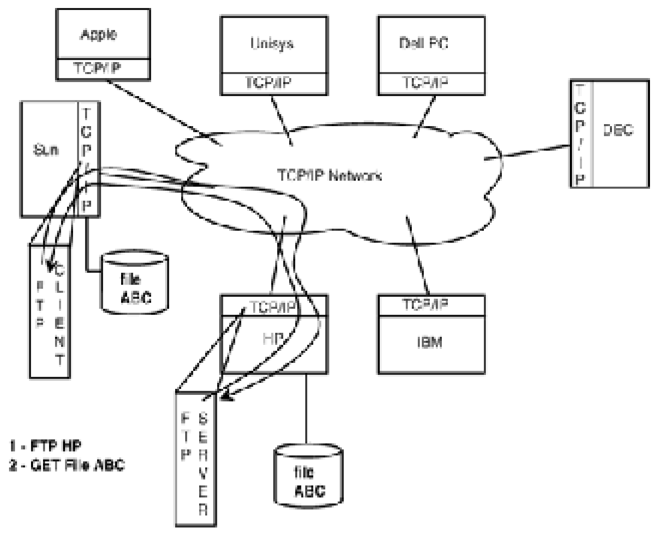


Figure 8-5
File transfer protocol.

This illustration shows multiple hosts. The same operation can be performed by any of these hosts. The Digital Equipment Corp. (DEC) host can perform any of the TCP/IP functions just as the SUN or HP.

8.4 HyperText Transfer Protocol

The *HyperText Transfer Protocol* (HTTP) is an application-level protocol for distributed, collaborative, and hypermedia information systems. It is a generic, stateless, object-oriented protocol which can be used for many tasks, such as name servers and distributed object management systems, through extension of its request methods. A feature of HTTP is the typing and negotiation of data representation, allowing systems to be built independently of the data being transferred. HTTP has been in use by the World Wide Web (WWW) global information initiative since 1990, and it is continuing to evolve.

8.4.1 HTTP Perspective and Purpose

The first version of HTTP, the HTTP/0.9, was a simple protocol for raw data transfer across the Internet. HTTP/1.0, as defined by RFC 1945, improved the protocol by allowing messages to be in the format of Multipurpose Internet Mail Extension (MIME)-like messages, containing metainformation about the data transferred and modifiers on the request/response semantics. However, HTTP/1.0 does not sufficiently take into consideration the effects of hierarchical proxies, caching, the need for persistent connections, and virtual hosts. In addition, the proliferation of incompletely implemented applications calling themselves HTTP/1.0 has necessitated a protocol version change to enable two communicating applications to determine each other's true capabilities.

Practical information systems require more functionality than simple retrieval, including search, front-end update, and annotation. HTTP allows an open-ended set of methods that indicate the purpose of a request. It builds on the discipline of reference provided by the Uniform Resource Identifier (URI), as a location (URL) or a name (URN), for indicating the resource to which a method is to be applied. Messages are passed in a format similar to that used by Internet mail as defined by MIME.

HTTP is also used as a generic protocol for communication between user agents and proxies/gateways to other Internet systems, including those supported by the SMTP, NNTP, FTP, Gopher, and Wide Area Information Server (WAIS) protocols. In this way, HTTP allows basic hypermedia access to resources available from diverse applications.

8.4.2 HTTP Terminology

A number of terms, specific primarily to HTTP, need to be understood to better comprehend the HTTP protocol and its operation. Consider the terms presented here.

age The age of a response is the time since it was sent by, or successfully validated with, the origin server.

cache A program's local store of response messages and the subsystem that controls message storage, retrieval, and deletion. A cache stores cacheable responses in order to reduce the response time and network bandwidth consumption on future, equivalent requests. Any client or server may include a cache, but a cache cannot be used by a server that is acting as a tunnel.

cacheable A response is cacheable if a cache is allowed to store a copy of the response message for use in answering subsequent requests. Even if a resource is cacheable, there may be additional constraints on whether a cache can use the cached copy for a particular request.

client A program that establishes connections for the purpose of sending requests.

connection A transport-layer virtual circuit established between two programs for the purpose of communication.

content negotiation The mechanism for selecting the appropriate representation when servicing a request. The representation of entities in any response can be negotiated (including error responses).

entity The information transferred as the payload of a request or response. An entity consists of metainformation in the form of entity-header fields and content in the form of an entity body.

explicit expiration time The time at which the origin server intends that an entity should no longer be returned by a cache without further validation.

firsthand A response is firsthand if it comes directly and without unnecessary delay from the origin server, perhaps via one or more proxies. A response is also firsthand if its validity has just been checked directly with the origin server.

fresh A response is fresh if its age has not yet exceeded its freshness lifetime.

freshness lifetime The length of time between the generation of a response and its expiration time.

gateway A server which acts as an intermediary for some other server. Unlike a proxy, a gateway receives requests as if it were the origin server for the requested resource; the requesting client may not be aware that it is communicating with a gateway.

heuristic expiration time An expiration time assigned by a cache when no explicit expiration time is available.

message The basic unit of HTTP communication, consisting of a structured sequence of octets matching the syntax and transmitted via the connection.

origin server The server on which a given resource resides or is to be created.

proxy An intermediary program which acts as both a server and a client to make requests on behalf of other clients. Requests are serviced internally or by passing them on, with possible translation, to other servers. A proxy must implement both the client and server requirements of this specification.

representation An entity included with a response that is subject to content negotiation. Multiple representations may be associated with a particular response status.

request An HTTP request message.

resource A network data object or service that can be identified by a URI. Resources may be available in multiple representations (e.g., multiple languages, data formats, size, resolutions) or vary in other ways.

response An HTTP response message.

semantically transparent A cache behaves in a “semantically transparent” manner, with respect to a particular response, when its use affects neither the requesting client nor the origin server, except to improve performance. When a cache is semantically transparent, the client receives exactly the same response (except for hop-by-hop headers) that it would have received had its request been handled directly by the origin server.

server An application program that accepts connections in order to service requests by sending back responses. Any given program may be capable of being both a client and a server; our use of these terms refers only to the role being performed by the program for a particular connection, rather than to the program’s capabilities in general. Likewise, any server may act as an origin server, proxy, gateway, or tunnel, switching behavior according to the nature of each request.

stale A response is stale if its age has passed its freshness lifetime.

tunnel An intermediary program which is acting as a blind relay between two connections. Once active, a tunnel is not considered a party to the HTTP communication, although the tunnel may have been initiated by an HTTP request. The tunnel ceases to exist when both ends of the relayed connections are closed.

user agent The client which initiates a request. These are often browsers, editors, spiders (Web-traversing robots), or other end-user tools.

validator A protocol element (e.g., an entity tag or a `Last-Modified` time) that is used to find out whether a cache entry is an equivalent copy of an entity.

variant A resource may have one, or more than one, representation(s) associated with it at any given instant. Each of these representations is termed a *variant.* Use of the term *variant* does not necessarily imply that the resource is subject to content negotiation.

8.4.3 HTTP Overall Operation

HTTP protocol is a request/response protocol. A client sends a request to the server in the form of a request method, URI, and protocol version, followed by a MIME-like message containing request modifiers, client information, and possible body content over a connection with a server. The server responds with a status line, including the message's protocol version and a success or error code, followed by a MIME-like message containing server information, entity metainformation, and possible entity-body content.

Most HTTP communication is initiated by a user agent and consists of a request to be applied to a resource on some origin server. In the simplest case, this may be accomplished via a single connection (v) between the user agent (UA) and the origin server (O).

```
request chain --------------->
UA ------------- v---------- O
<-------------- response chain
```

A more complicated situation occurs when one or more intermediaries are present in the request/response chain. There are three common forms of intermediary: proxy, gateway, and tunnel. A *proxy* is a forwarding agent, receiving requests for a URI in its absolute form, rewriting all or part of the message, and forwarding the reformatted request toward the server identified by the URI. A *gateway* is a receiving agent, acting as a layer above some other server(s) and, if necessary, translating the requests to the underlying server's protocol. A *tunnel* acts as a relay point between two connections without changing the messages; tunnels are used in a situation when the communication must pass through an intermediary (such as a firewall) even when the intermediary cannot understand the contents of the messages.

```
request chain ---------------->
UA --- v --- A --- v --- B --- v --- C --- v --- O
<----------------- response chain
```

This program fragment shows three intermediaries (A, B, and C) between the user agent and the origin server. A request or response message that travels the whole chain will pass through four separate connections. This distinction is important because some HTTP communication options may apply only to the connection with the nearest, nontunnel neighbor, only to the endpoints of the chain, or to all connections along the chain. Although the diagram is linear, each participant may be engaged in multiple, simultaneous communications. For example, B may be receiving requests from many clients other than A, and/or forwarding requests to servers other than C, at the same time that it is handling A's request.

Any party to the communication which is not acting as a tunnel may employ an internal cache for handling requests. The effect of a cache is that the request/response chain is shortened if one of the participants along the chain has a cached response applicable to that request. The following illustrates the resulting chain if B has a cached copy of an earlier response from O (via C) for a request which has not been cached by UA or A:

```
request chain ---------------->
UA ---- v ---- A ---- v ---- B ---- C ---- O
<---------------- response chain
```

Not all responses are usefully cacheable, and some requests may contain modifiers which place special requirements on cache behavior. In fact, a wide variety of architectures and configurations of caches and proxies are currently being experimented with or deployed across the World Wide Web; these systems include national hierarchies of proxy caches to save transoceanic bandwidth, systems that broadcast or multicast cache entries, organizations that distribute subsets of cached data via CD-ROM, and so on. HTTP systems are used in corporate intranets over high-bandwidth links, and for access via personal digital assistants (PDAs) with low-power radio links and intermittent connectivity. The goal of HTTP/1.1 is to support the wide diversity of configurations already deployed while introducing protocol constructs that meet the needs of those who build Web applications that require high reliability and, failing that, at least reliable indications of failure.

HTTP communication usually takes place over TCP/IP connections. The default port is TCP 80, but other ports can be used. This does not preclude HTTP from being implemented on top of any other protocol on the Internet, or on other networks. HTTP only presumes a reliable transport; any protocol that provides such guarantees can be used; the mapping of the HTTP/1.1 request and response structures onto the transport data units of the protocol in question is outside the scope of this specification.

In HTTP/1.0, most implementations used a new connection for each request/response exchange. In HTTP/1.1, a connection may be used for one or more request/response exchanges, although connections may be closed for a variety of reasons.

8.4.4 HTTP Notational Conventions and Generic Grammar

Augmented BNF

All the mechanisms specified in this document are described in both prose and an augmented Backus-Naur Form (BNF) similar to that used by RFC 822. Implementers will need to be familiar with the notation in order to understand this specification. The augmented BNF includes the following constructs:

`name = definition`	The name of a rule is simply the name itself (without any enclosing “<” and “>”) and is separated from its definition by the equal “ = ” character. Whitespace is only significant in that indentation of continuation lines is used to indicate a rule definition that spans more than one line. Certain basic rules are in uppercase, such as SP, LWS, HT, CRLF, DIGIT, and ALPHA. Angle brackets are used within definitions whenever their presence will facilitate discerning the use of rule names.
`"literal"`	Quotation marks surround literal text. Unless stated otherwise, the text is case-insensitive.
`rule1 | rule2`	Elements separated by a bar (“|”) are alternatives; for instance, yes | no will accept yes or no answers.
`(rule1 rule2)`	Elements enclosed in parentheses are treated as a single element. Thus, (elem (foo | bar) elem) allows the token sequences elem foo elem and elem bar elem.

`*rule`	The asterisk character (*) preceding an element indicates repetition. The full form is <n>*<m>element indicating at least <n> and at most <m> occurrences of element. Default values are 0 and infinity so that *(element) allows any number, including zero; 1*element requires at least one; and 1*2element allows one or two.
`[rule]`	Square brackets enclose optional elements; "[foo bar]" is equivalent to *1(foo bar).
`N rule`	Specific repetition: <n>(element) is equivalent to <n>*<n>(element); that is, exactly <n> occurrences of (element). Thus 2DIGIT is a two-digit number, and 3ALPHA is a string of three alphabetic characters.
`#rule`	A construct "#" is defined, similar to the asterisk (*) for defining lists of elements. The full form is <n>#<m>element indicating at least <n> and at most <m> elements, each separated by one or more commas (,) and optional linear whitespace (LWS). This makes the usual form of lists very easy; a rule such as (*LWS element *(*LWS "," *LWS element)) can be shown as 1#element. Wherever this construct is used, null elements are allowed, but do not contribute to the count of elements present. That is, (element), , (element) is permitted, but counts as only two elements. Therefore, where at least one element is required, at least one nonnull element must be present. Default values are 0 and infinity so that #element allows any number, including zero; 1#element requires at least one; and 1#2element allows one or two.
`; comment`	A semicolon, set off some distance to the right of rule text, starts a comment that continues to the end of line. This is a simple way of including useful notes in parallel with the specifications.
`implied *LWS`	The grammar described by this specification is word-based. Except where noted otherwise, linear whitespace (LWS) can be included between any two adjacent words (token or quoted-string), and between adjacent tokens and delimiters (tspecials), without changing the interpretation of a field. At least one delimiter (tspecials) must exist between any two tokens, since they would otherwise be interpreted as a single token.

8.4.5 HTTP Version Protocol Parameters

HTTP uses a <major>.<minor> numbering scheme to indicate versions of the protocol. The protocol versioning policy is intended to allow the sender to indicate the format of a message and its capacity for understanding further HTTP communication, rather than the features obtained via that communication. No change is made to the version number for the addition of message components which do not affect communication behavior or which only add to extensible field values.

The <minor> number is incremented when the changes made to the protocol add features that do not change the general message parsing algorithm, but which may add to the message semantics and imply additional capabilities of the sender. The <major> number is incremented when the format of a message within the protocol is changed.

The version of an HTTP message is indicated by an `HTTP-Version` field in the first line of the message:

HTTP-Version = “HTTP” “/” 1*DIGIT “.” 1*DIGIT

NOTE: *The major and minor numbers must be treated as separate integers and each may be incremented higher than a single digit. Thus, HTTP/2.4 is a lower version than HTTP/2.13, which, in turn, is lower than HTTP/12.3. Leading zeros must be ignored by recipients and must not be sent.*

Applications sending request or response messages, as defined by this specification, *must* include an HTTP version of HTTP/1.1. Use of this version number indicates that the sending application is at least conditionally compliant with this specification. The HTTP version of an application is the highest HTTP version for which the application is at least conditionally compliant.

Proxy and gateway applications must be careful when forwarding messages in protocol versions different from that of the application. Since the protocol version indicates the protocol capability of the sender, a proxy/gateway *must never* send a message with a version indicator which is greater than its actual version; if a higher version request is received, the proxy/gateway *must* either downgrade the request version, respond with an error, or switch to tunnel behavior. Requests with a version lower than that of the proxy/gateway’s version may be upgraded before being forwarded; the proxy/gateway’s response to that request *must* be in the same major version as the request.

Converting between versions of HTTP may involve modification of the header fields that are required or forbidden by the versions involved.

8.4.6 Uniform Resource Identifiers in HTTP

URIs have been known by many names: WWW addresses, Universal Document Identifiers, Universal Resource Identifiers, and finally the combination of Uniform Resource Locators (URL) and Names (URN). As far as HTTP is concerned, Uniform Resource Identifiers are simply formatted strings which identify—by name, location, or any other characteristic—a resource.

General Syntax

URIs in HTTP can be represented in absolute form or relative to some known base URI, depending on the context of their use. The two forms are differentiated by the fact that absolute URIs always begin with a scheme name followed by a colon.

```
URI                 = ( absoluteURI | relativeURI ) [ “#” fragment ]
absoluteURI         = scheme “:” *( uchar | reserved )
relativeURI         = net_path | abs_path | rel_path
net_path            = “//” net_loc [ abs_path ]
abs_path            = “/” rel_path
rel_path            = [ path ] [ “;” params ] [ “?” query ]
path                = fsegment *( “/” segment )
fsegment            = 1*pchar
segment             = *pchar
params              = param *( “;” param )
param               = *( pchar | “/” )
scheme              = 1*( ALPHA | DIGIT | “+” | “-” | “.” )
net_loc             = *( pchar | “;” | “?” )
query               = *( uchar | reserved )
fragment            = *( uchar | reserved )
pchar               = uchar | “:” | “@” | “&” | “ = ” | “+”
uchar               = unreserved | escape
unreserved          = ALPHA | DIGIT | safe | extra | national
escape              = “%” HEX HEX
reserved            = “;” | “/” | “?” | “:” | “@” | “&” | “ = ” | “+”
extra               = “!” | “*” | ”””” | “(” | “)” | “,”
safe                = “$” | “-” | “_” | “.”
unsafe              = CTL | SP | <”> | “#” | “%” | “<” | “>”
national            = <any OCTET excluding ALPHA, DIGIT, reserved, extra, safe, and
                       unsafe>
```

For definitive information on URL syntax and semantics, see RFC 1738 and RFC 1808. The BNF presented above includes national characters not allowed in valid URLs as specified by RFC 1738, since HTTP servers are not restricted in the set of unreserved characters allowed to represent the `rel_path` part of addresses, and HTTP proxies may receive requests for URIs not defined by RFC 1738.

The HTTP protocol does not place any a priori limit on the length of a URI. Servers *must* be able to handle the URI of any resource they serve, and should be able to handle URIs of unbounded length if they provide GET-based forms that could generate such URIs. A server should return 414 (`Request-URI Too Long`) status if a URI is longer than the server can handle.

NOTE: *Servers should be cautious about depending on URI lengths above 255 bytes, because some older client or proxy implementations may not properly support these lengths.*

HTTP URL Syntax and Semantics.

The “http” scheme is used to locate network resources via the HTTP protocol. This section defines the scheme-specific syntax and semantics for HTTP URLs.

`http_URL`	= “http: ” “//” host [“:” port] [abs_path]
`host`	= <A legal Internet host domain name or IP address (in dotted-decimal form). (Defined in RFC 1123)>
`port`	= *DIGIT

If the port is empty or not given, port 80 is assumed. The semantics are that the identified resource is located at the server listening for TCP connections on that port of that host, and the `Request-URI` for the resource is `abs_path.` The use of IP addresses in URLs should be avoided whenever possible (refer to RFC 1900). If the `abs_path` is not present in the URL, it *must* be given as a slash (/) when used as a `Request-URI` for a resource.

URI Comparison in HTTP

When comparing two URIs to decide whether they match, a client should use a case-sensitive octet-by-octet comparison of the entire URIs, except when a port that is empty or not given is equivalent to the default port for that URI; comparisons of host names or scheme names *must* be case-insensitive, or an empty `abs_path` is equivalent to an `abs_path` of “/”.

Characters other than those in the “reserved” and “unsafe” sets are equivalent to their `""%" HEX HEX"` encoding. For example, the following three URIs are equivalent:

http://abc.com:80/~smith/home.html
http://ABC.com/%7Esmith/home.html
http://ABC.com/%7esmith/home.html

8.4.7 Date and Time Formats in HTTP

Full Date

HTTP applications have historically allowed three different formats for the representation of date/timestamps:

Sun, 06 Nov 1994 08:49:37 GMT; RFC 1123
Sunday, 06-Nov-94 08:49:37 GMT; RFC 1036
Sun Nov 6 08:49:37 1994; ANSI C’s `asctime()` format

The first format is preferred as an Internet standard and represents a fixed-length subset of that defined by RFC 1123 (an update to RFC 822). The second format is in common use, but is based on the obsolete RFC 850 date format and lacks a four-digit year. HTTP/1.1 clients and servers that parse the date value *must* accept all three formats (for compatibility with HTTP/1.0), although they *must* generate the RFC 1123 format only for representing HTTP-date values in header fields.

Recipients of date values are encouraged to be robust in accepting date values that may have been sent by non-HTTP applications, as is sometimes the case when retrieving or posting messages via proxies/gateways to SMTP or NNTP.

All HTTP date/timestamps *must* be represented in Greenwich Mean Time (GMT), without exception. This is indicated in the first two formats by the inclusion of "GMT" as the three-letter abbreviation for time zone, and *must* be assumed when reading the `asctime` format.

HTTP-date = rfc1123-date | rfc850-date | asctime-date
rfc1123-date = wkday "," SP date1 SP time SP "GMT"
rfc850-date = weekday "," SP date2 SP time SP "GMT"
asctime-date = wkday SP date3 SP time SP 4DIGIT
date1 = 2DIGIT SP month SP 4DIGIT; day month year (02 Jun 1982)
date2 = 2DIGIT "-" month "-" 2DIGIT; day-month-year (e.g., 02-Jun-2)
date3 = month SP (2DIGIT | (SP 1DIGIT)); month day (e.g., Jun 2)
time = 2DIGIT ":" 2DIGIT ":" 2DIGIT; 00:00:00 - 23:59:59
wkday = "Mon" | "Tue" | "Wed" | "Thu" | "Fri" | "Sat" | "Sun"
weekday =
"Monday"|"Tuesday"|"Wednesday"|"Thursday"|"Friday"|"Saturday"|"Sunday"
month = "Jan"|"Feb"|"Mar"|"Apr"|"May"
|"Jun"|"Jul"|"Aug"|"Sep"|"Oct"|"Nov"|"Dec"
delta-seconds = 1*DIGIT

HTTP requirements for the date/timestamp format apply only to their usage within the protocol stream. Clients and servers are not required to use these formats for user presentation, request logging, or similar functions.

8.4.8 Character-Set Use in HTTP

HTTP uses the same definition of the term "character set" as that described for MIME, presented here for the sake of the reader:

> The term "character set" is used in this document to refer to a method used with one or more tables to convert a sequence of octets into a sequence of characters. Note that unconditional conversion in the other direction is not required, in that not all characters may be available in a given character set and a character set may provide more than one sequence of octets to represent a particular character. This definition is intended to allow various kinds of character encoding, from simple single-table mappings such as US-ASCII to complex table switching methods such as those that use ISO 2022's techniques. However, the definition associated with a MIME character set name *must* fully specify the mapping to be performed from octets to characters. In particular, use of external profiling information to determine the exact mapping is not permitted.

NOTE: *Use of the term "character set" is more commonly referred to as a "character encoding." However, since HTTP and MIME share the same registry, it is important that the terminology also be shared.*

HTTP character sets are identified by case-insensitive tokens. The complete set of tokens is defined by the IANA character-set registry.

charset = token

Although HTTP allows an arbitrary token to be used as a character-set value, any token that has a predefined value within the IANA character-set registry *must* represent the character set defined by that registry. Applications should limit their use of character sets to those defined by the IANA registry.

8.4.9 Content Codings in HTTP

Content coding values indicate an encoding transformation that has been or can be applied to an entity. Content codings are primarily used to allow a document to be compressed or otherwise usefully transformed without losing the identity of its underlying media type and without loss of information. Frequently, the entity is stored in coded form, transmitted directly, and decoded only by the recipient.

content-coding = token

All content-coding values are case-insensitive. HTTP/1.1 uses content-coding values in the accept-encoding and content-encoding header fields. Although the value describes the content coding, what is more important is that it indicates what decoding mechanism will be required to remove the encoding.

The *Internet Assigned Number Authority* (IANA) acts as a registry for content-coding value tokens. Initially, the registry contains the following tokens:

`gzip`	An encoding format produced by the file compression program gzip (GNU zip) as described in RFC 1952. This format is a Lempel-Ziv coding (LZ77) with a 32-bit CRC (cyclical redundancy check).
`compress`	The encoding format produced by the common UNIX file compression program compress. This format is an adaptive Lempel-Ziv-Welch coding (LZW). (Use of program names for the identification of encoding formats is not desirable and should be discouraged for future encodings. Their use here is representative of historical practice, not good design. For compatibility with previous implementations of HTTP, applications should consider x-gzip and x-compress to be equivalent to gzip and compress, respectively.)
`deflate`	The zlib format defined in RFC 1950[31] in combination with the deflate compression mechanism described in RFC 1951.

New content-coding value tokens should be registered; to allow interoperability between clients and servers, specifications of the content coding algorithms needed to implement a new value should be publicly available and adequate for independent implementation, and conform to the purpose of content coding.

8.4.10 Transfer Codings in HTTP

Transfer coding values are used to indicate an encoding transformation that has been, can be, or may need to be applied to an entity-body in order to ensure *safe transport* through the network. This differs from a content coding in that the transfer coding is a property of the message, not of the original entity.

transfer-coding = “chunked” | transfer-extension
transfer-extension = token

All transfer-coding values are case-insensitive. HTTP/1.1 uses transfer-coding values in the transfer-encoding header field.

Transfer codings are analogous to the `Content-Transfer-Encoding` values of MIME, which were designed to enable safe transport of binary data over a 7-bit transport service. However, safe transport has a different focus for an 8-bit clean transfer protocol. In HTTP, the only unsafe characteristic of message-bodies is the difficulty in determining the exact body length, or the desire to encrypt data over a shared transport.

The chunked encoding modifies the body of a message in order to transfer it as a series of chunks, each with its own size indicator, followed by an optional footer containing entity-header fields. This allows dynamically produced content to be transferred along with the information necessary for the recipient to verify that it has received the full message.

```
Chunked-Body = *chunk
               "0" CRLF
                footer
                 CRLF
chunk = chunk-size [ chunk-ext ] CRLF
chunk-data CRLF
hex-no-zero = <HEX excluding "0">
chunk-size = hex-no-zero *HEX
chunk-ext = *(";"chunk-ext-name[" = "chunk-ext-value])
chunk-ext-name = token
chunk-ext-val = token | quoted-string =
chunk-data = chunk-size (OCTET)
footer = *entity-header
```

The chunked encoding is ended by a zero-sized chunk followed by the footer, which is terminated by an empty line. The purpose of the footer is to provide an efficient way to supply information about an entity that is generated dynamically; applications *must not* send header fields in the footer which are not explicitly defined as being appropriate for the footer, such as `Content-MD5` or future extensions to HTTP for digital signatures or other facilities.

All HTTP/1.1 applications *must* be able to receive and decode the "chunked" transfer coding, and *must* ignore transfer-coding extensions they do not understand. A server that receives an entity-body with a transfer coding it does not understand should return 501 (unimplemented), and close the connection. A server must not send transfer codings to an HTTP/1.0 client.

8.4.11 Media Types and HTTP

HTTP uses Internet Media Types in the `Content-Type` and `Accept` header fields in order to provide open and extensible data typing and type negotiation.

```
media-type  = type "/" subtype *(";" parameter )
type        = token
subtype     = token
```

Parameters may follow the type/subtype in the form of attribute/ value pairs.

```
parameter    = attribute " = " value
attribute    = token
value        = token | quoted-string
```

The type, subtype, and parameter attribute names are case-insensitive. Parameter values may or may not be case-sensitive, depending on the semantics of the parameter name. Linear whitespace (LWS) *must not* be used between the type and subtype, nor between an attribute and its value. User agents that recognize the media type *must* process (or arrange to be processed by any external applications used to process that type/subtype by the user agent) the parameters for that MIME type as described by that type/subtype definition to the and inform the user of any problems discovered.

Some older HTTP applications do not recognize media-type parameters. When sending data to older HTTP applications, implementations should use media-type parameters only when they are required by that type/subtype definition.

Media-type values are registered with the Internet Assigned Number Authority (IANA). The media-type registration process is outlined in RFC 2048. Use of non-registered media types is discouraged.

Canonicalization and Text Defaults

Internet media types are registered with a canonical form. In general, an entity-body transferred via HTTP messages *must* be represented in the appropriate canonical form prior to its transmission; the exception is `text` types, as defined in the next paragraph.

When in canonical form, media subtypes of the `text` type use `CRLF` as the text line break. HTTP relaxes this requirement and allows the transport of text media with plain `CR` or `LF` alone representing a line break when it is done consistently for an entire entity body. HTTP applications *must* accept `CRLF,` bare `CR,` and bare `LF` as being representative of a line break in text media received via HTTP. In addition, if the text is represented in a character set that does not use octets 13 and 10 for `CR` and `LF,` respectively, as is the case for some multibyte character sets, HTTP allows the use of whatever octet sequences are defined by that character set to represent the equivalent of `CR` and `LF` for line breaks. This flexibility regarding line breaks applies only to text media in the entity body; a bare `CR` or `LF` *must not* be substituted for `CRLF` within any of the HTTP control structures (such as header fields and multipart boundaries).

If an entity-body is encoded with a `Content-Encoding,` the underlying data *must* be in a form as defined above prior to being encoded. The `charset` parameter is used with some media types to define the character set of the data. When no explicit `charset` parameter is provided by the sender, media subtypes of the `text` type are defined to have a default `charset` value of ISO-8859-1 when received via HTTP. Data in character sets other than ISO-8859-1 or its subsets *must* be labeled with an appropriate `charset` value.

Some HTTP/1.0 software has interpreted a `Content-Type` header without a `charset` parameter incorrectly to mean “recipient should guess.” Senders wishing to defeat this behavior may include a `charset` parameter even when the character set is ISO-8859-1 and should do so when it is known that it will not confuse the recipient.

Unfortunately, some older HTTP/1.0 clients did not deal properly with an explicit `charset` parameter. HTTP/1.1 recipients *must* respect the `charset` label provided by the sender; and those user agents that have a provision to “guess” a `charset` *must* use the `charset` from the content-type field if they support that `charset,` rather than the recipient’s preference, when initially displaying a document.

Multipart Types

MIME provides for a number of “multipart” types—encapsulations of one or more entities within a single message-body. All multipart types share a common syntax, as defined in MIME, and must include a boundary parameter as part of the media-type value. The message-body is itself a protocol element and *must* therefore use only `CRLF` to represent line breaks between body-parts. Unlike in MIME, the epilog of any multipart message *must* be empty; HTTP applications must not transmit the epilog (even if the original multipart contains an epilog).

In HTTP, multipart body-parts can contain header fields which are significant to the meaning of that part. A `Content-Location` header field should be included in the body-part of each enclosed entity that can be identified by a URL.

In general, an HTTP user agent follows the same or similar behavior as a MIME user agent would on receipt of a multipart type. If an application receives an unrecognized multipart subtype, the application *must* treat it as being equivalent to `multipart/mixed.` The `multipart/form-data` type has been specifically defined for carrying form data suitable for processing via the POST request method, as described in RFC 1867.

Product Tokens

Product tokens are used to allow communicating applications to identify themselves by software name and version.

Most fields using product tokens also allow subproducts which form a significant part of the application to be listed, separated by whitespace. By convention, the products are listed in order of their significance for identifying the application.

```
    product    = token [“/” product-version]
    product-version = token
Examples: User-Agent: ACE-LineMode/2.15 mewww/2.17b3
Server: Joker/0.8.4
```

Product tokens are generally short and to the point. Use of them for advertising or other nonessential information is explicitly forbidden. Although any token character may appear in a `product-version,` this token should be used only for a version identifier; that is, a successive version of the same product should only differ in the `product-version` portion of the product value.

8.4.12 HTTP Quality Values

HTTP content negotiation uses short *floating-point* numbers to indicate the relative importance (weight) of various negotiable parameters. A weight is normalized to a real number in the range 0 to 1, where 0 is the minimum and 1 the maximum value. HTTP/1.1 applications must not generate more than three digits after the decimal point. User configuration of these values should also be limited in this fashion:

```
value  =  (“0” [“.” 0*3DIGIT ] )
          | (“1” [“.” 0*3(“0”) ] )
```

The term *quality values* is a misnomer, since these values merely represent relative degradation in desired quality.

8.4.13 HTTP Language Tags

A language tag identifies a natural language spoken, written, or otherwise conveyed by human beings for communication of information to other human beings. Computer languages are explicitly excluded. HTTP uses language tags within the `Accept-Language` and `Content-Language` fields. The syntax and registry of HTTP language tags is the same as that defined by RFC 1766. In summary, a language tag is composed of 1 or more parts: A primary language tag and a possibly empty series of subtags:

language-tag = primary-tag *("-" subtag)
primary-tag = 1*8ALPHA
subtag = 1*8ALPHA

Whitespace is not allowed within the tag and all tags are case-insensitive. The name space of language tags is administered by the IANA. Example tags include `en, en-US, en-cockney, i-cherokee,` and `x-younme-latin.` In this case any two-letter `primary-tag` is an ISO 639 language abbreviation and any two-letter initial subtag is an ISO 3166 country code. (The last three tags above are not registered tags; all except the last are examples of tags which could be registered in the future.)

8.4.14 HTTP Entity Tags

Entity tags are used for comparing two or more entities from the same requested resource. HTTP/1.1 uses entity tags in the `ETag, If-Match, If-None-Match,` and `If-Range` header fields. An entity tag consists of an opaque quoted string, possibly prefixed by a weakness indicator.

entity-tag = [weak] opaque-tag
weak = "W/"
opaque-tag = quoted-string

A *strong entity* tag may be shared by two entities of a resource only if they are equivalent by octet equality. A *weak entity* tag, indicated by the W/ prefix, may be shared by two entities of a resource only if the entities are equivalent and could be substituted for each other with no significant change in semantics. A *weak entity* tag can be used only for weak comparison. An entity tag must be unique across all versions of all entities associated with a particular resource. A given entity-tag value may be used for entities obtained by requests on different URIs without implying anything about the equivalence of those entities.

8.4.15 HTTP Range Units

HTTP/1.1 allows a client to request that only part (a range) of the response entity be included within the response. HTTP/1.1 uses range units in the `Range` and `Content-Range` header fields. An entity may be broken down into subranges according to various structural units.

range-unit = bytes-unit | other-range-unit
bytes-unit = "bytes"
other-range-unit = token

The only range unit defined by HTTP/1.1 is `"bytes".` HTTP/1.1 implementations may ignore ranges specified using other units. HTTP/1.1 has been designed to allow implementations of applications that do not depend on knowledge of ranges.

8.4.16 HTTP Message Types

HTTP messages consist of requests from client to server and responses from server to client.

HTTP-message = Request | Response ; HTTP/1.1 messages

`Request | Response` messages use the generic message format of RFC 822 to transfer entities (the “payload” of the message). Both types of message consist of a start line, one or more header fields (also known as “headers”), an empty line (i.e., a line with nothing preceding the `CRLF`) indicating the end of the header fields, and an optional message body.

```
generic-message = start-line
                *message-header
                 CRLF
                 [ message-body ]
start-line = Request-Line | Status-Line
```

In the interest of robustness, servers should ignore any empty line(s) received where a `Request-Line` is expected. In other words, if the server is reading the protocol stream at the beginning of a message and receives a `CRLF` first, it should ignore the `CRLF.`

Certain buggy HTTP/1.0 client implementations generate an extra `CRLF` after a POST request. To restate what is explicitly forbidden by the BNF, an HTTP/1.1 client must not preface or follow a request with an extra `CRLF.`

8.4.17 Message Headers and HTTP

HTTP header fields, which include general-header, request-header, response-header, and entity-header fields, follow the same generic format as that given in RFC 822. Each header field consists of a name followed by a colon (:) and the field value. Field names are case-insensitive. The field value may be preceded by any amount of LWS, although a single SP is preferred. Header fields can be extended over multiple lines by preceding each extra line with at least one SP or HT. Applications should follow “common form” when generating HTTP constructs, since there might exist some implementations that fail to accept anything beyond the common forms.

```
message-header = field-name “:” [ field-value] CRLF
field-name = token
field-value = *( field-content | LWS )
field-content = <the OCTETs making up the field-value
and consisting of either *TEXT or combinations
of token, tspecials, and quoted-string>
```

The order in which header fields with differing field names are received is not significant. However, it is good practice to send general-header fields first, followed by request-header or response-header fields, and ending with the entity-header fields.

Multiple message-header fields with the same field-name may be present in a message if and only if the entire field value for that header field is defined as a comma-separated list [i.e., `#(values)`]. It *must* be possible to combine the multiple header fields into one “field name-field value” pair, without changing the semantics of the message, by appending each subsequent field value to the first, each separated by a comma. The order in which header fields with the same field-name are received is therefore significant to the interpretation of the combined field value, and thus a proxy *must not* change the order of these field values when a message is forwarded.

8.4.18 HTTP Message Body

The message-body (if any) of an HTTP message is used to carry the entity body associated with the request or response. The message body differs from the entity body only when a transfer coding has been applied, as indicated by the `Transfer-Encoding` header field.

```
message-body        = entity-body
                    |<entity-body encoded as Transfer-Encoding>
```

`Transfer-Encoding` *must* be used to indicate any transfer codings applied by an application to ensure safe and proper transfer of the message. `Transfer-Encoding` is a property of the message, not of the entity, and thus can be added or removed by any application along the request/response chain.

The rules for when a message body is allowed in a message differ for requests and responses. The presence of a message body in a request is signaled by the inclusion of a `Content-Length` or `Transfer-Encoding` header field in the request's message headers. A message body may be included in a request only when the request method allows an entity body.

For response messages, whether a message body is or is not included with a message is dependent on both the request method and the response status code. All responses to the HEAD request method *must not* include a message body, even though the presence of entity-header fields might lead one to believe they do.

8.4.19 HTTP Message Length

When a message body is included with a message, the length of that body is determined by one of the following (in order of precedence):

1. Any response message which *must not* include a message body is always terminated by the first empty line after the header fields, regardless of the entity-header fields present in the message.

2. If a `Transfer-Encoding` header field is present and indicates that the "chunked" transfer coding has been applied, then the length is defined by the chunked encoding.

3. If a `Content-Length` header field is present, its value in bytes represents the length of the message-body.

4. If the message uses the media type `multipart/byteranges,` which is self-delimiting, then that defines the length. This media type *must not* be used unless the sender knows that the recipient can parse it; the presence in a request of a `Range` header with multiple byte-range specifiers implies that the client can parse `multipart/byteranges` responses.

5. By the server closing the connection. (Closing the connection cannot be used to indicate the end of a request body, since that would leave no possibility for the server to send back a response.)

For compatibility with HTTP/1.0 applications, HTTP/1.1 requests containing a message body *must* include a valid `Content-Length` header field unless the server is known to be HTTP/1.1 compliant. If a request contains a message body and a `Content-Length` is not given, the server *should* respond with " it cannot determine the length of the message," or with "it wishes to insist on receiving a valid `Content-Length."`

All HTTP/1.1 applications that receive entities *must* accept the "chunked" transfer coding, thus allowing this mechanism to be used for messages when the message length cannot be determined in advance. Messages *must not* include both a `Content-Length` header field and the "chunked" transfer coding. If both are received, the `Content-Length` *must* be ignored.

When a `Content-Length` is given in a message where a message body is allowed, its field value must exactly match the number of `OCTETs` in the message body. HTTP/1.1 user agents MUST notify the user when an invalid length is received and detected.

8.4.20 HTTP General Header Fields

A few header fields have general applicability for both request and response messages but do not apply to the entity being transferred. These header fields apply only to the message being transmitted.

```
general-header = Cache-Control
            | Connection
            | Date
            | Pragma
            | Transfer-Encoding
            | Upgrade
            | Via
```

`General-header` field names can be extended reliably only in combination with a change in the protocol version. However, new or experimental header fields may be given the semantics of general-header fields if all parties in the communication recognize them to be general-header fields. Unrecognized header fields are treated as entity-header fields.

8.4.21 HTTP Request

A request message from a client to a server includes, within the first line of that message, the method to be applied to the resource, the identifier of the resource, and the protocol version in use.

```
Request  =  Request-Line
         *( general-header
         | request-header
         | entity-header )
         CRLF
         [ message-body ]
```

Request-Line

The `Request-Line` begins with a method token, followed by the `Request-URI` and the protocol version, and ending with `CRLF.` The elements are separated by SP characters. No `CR` or `LF` are allowed except in the final `CRLF` sequence.

Request-Line = Method SP Request-URI SP HTTP-Version CRLF

Method

The `Method` token indicates the method to be performed on the resource identified by the `Request-URI.` The method is case-sensitive.

```
Method =      "OPTIONS"
            | "GET"
            | "HEAD"
            | "POST"
            | "PUT"
            | "DELETE"
            | "TRACE"
            | extension-method
            extension-method = token
```

The list of methods allowed by a resource can be specified in an `Allow` header field. The return code of the response always notifies the client whether a method is currently allowed on a resource, since the set of allowed methods can change dramatically. Servers should return the status code 405 (“method not allowed”) if the method is known by the server but not allowed for the requested resource, and 501 (“not implemented”) if the method is unrecognized or not implemented by the server. The list of methods known by a server can be listed in a `Public response-header` field. The methods `GET` and `HEAD` must be supported by all general-purpose servers.

HTTP Request-URI

The `Request-URI` is a Uniform Resource Identifier and identifies the resource on which to apply the request.

Request-URI = “*” | absoluteURI | abs_path

The three options for `Request-URI` are dependent on the nature of the request. The asterisk “*” means that the request does not apply to a particular resource, but to the server itself, and is only allowed when the method used does not necessarily apply to a resource. One example would be `OPTIONS * HTTP/1.1/`.

The absolute URI form is required when the request is being made to a proxy. The proxy is requested to forward the request or service it from a valid cache, and return the response. Note that the proxy may forward the request on to another proxy or directly to the server specified by the `absoluteURI.` In order to avoid request loops, a proxy must be able to recognize all of its server names, including any aliases, local variations, and the numeric IP address. Consider the following request-line example:

GET http://www.w3.org/pub/WWW/TheProject.html HTTP/1.1

To allow for transition to `absoluteURIs` in all requests in future versions of HTTP, all HTTP/1.1 servers *must* accept the absolute URI form in requests, even though HTTP/1.1 clients will generate them only in requests to proxies.

The most common form of `Request-URI` is that used to identify a resource on an origin server or gateway. In this case the absolute path of the URI must be transmitted as the `Request-URI,` and the network location of the URI (`net_loc`) must be transmitted in a `Host` header field. For example, a client wishing to retrieve the resource above directly from the origin server would create a TCP connection to port 80 of the host www.w3.org and send the lines:

GET /pub/WWW/TheProject.html HTTP/1.1

Host: www.w3.org

Then, it is followed by the remainder of the `Request.` Note that the absolute path cannot be empty; if none is present in the original URI, it must be given as `/` (the server root). If a proxy receives a request without any path in the `Request-URI` and the method specified is capable of supporting the asterisk form of request, then the last proxy on the request chain *must* forward the request with an askerisk (*) as the final `Request-URI.` For example, the request

OPTIONS http://www.ics.uci.edu:8001 HTTP/1.1

would be forwarded by the proxy as

OPTIONS * HTTP/1.1

Host: www.ics.uci.edu:8001

after connecting to port 8001 of host www.ics.uci.edu.

The origin server must decode the `Request-URI` in order to properly interpret the request. Servers should respond to invalid `Request-URIs` with an appropriate status code.

In requests that they forward, proxies must not rewrite the `abs_path` part of a `Request-URI` in any way except as noted above to replace a null `abs_path` with an asterisk, no matter what the proxy does in its internal implementation.

NOTE: *The no rewrite rule prevents the proxy from changing the meaning of the request when the origin server is improperly using a nonreserved URL character for a reserved purpose. Implementers should be aware that some pre-HTTP/1.1 proxies have been known to rewrite the `Request-URI`.*

The Resource Identified by a Request

HTTP/1.1 origin servers should be aware that the exact resource identified by an Internet request is determined by examining both the `Request-URI` and the `Host` header field. An origin server that does not allow resources to differ by the requested host may ignore the `Host` header field value. An origin server that does differentiate resources on the basis of the host requested (virtual hosts) must use the following rules for determining the requested resource on an HTTP/1.1 request:

1. If `Request-URI` is an absolute URI, the host is part of the `Request-URI.` Any `Host` header field value in the request *must* be ignored.

2. If the `Request-URI` is not an absolute URI, and the request includes a `Host` header field, the host is determined by the `Host` header field value.

3. If the host as determined by rule 1 or 2 is not a valid host on the server, the response `must` be a 400 ("bad request") error message.

Recipients of an HTTP/1.0 request that lacks a `Host` header field may attempt to use heuristics (e.g., examination of the URI path for something unique to a particular host) in order to determine what exact resource is being requested.

HTTP request-header fields.

The request-header fields allow the client to pass additional information about the request, and about the client itself, to the server. These fields act as request modifiers, with semantics equivalent to the parameters on a programming language method invocation.

```
request-header          = Accept
                        | Accept-Charset
                        | Accept-Encoding
                        | Accept-Language
                        | Authorization
                        | From
                        | Host
                        | If-Modified-Since
                        | If-Match
                        | If-None-Match
                        | If-Range
                        | If-Unmodified-Since
                        | Max-Forwards
                        | Proxy-Authorization
                        | Range
                        | Referer
                        | User-Agent
```

Request-header field names can be extended reliably only in combination with a change in the protocol version. However, new or experimental header fields may be given the semantics of request-header fields if all parties in the communication recognize them to be request-header fields. Unrecognized header fields are treated as entity-header fields.

8.4.22 HTTP Message Response

After receiving and interpreting a request message, a server responds with an HTTP response message.

```
Response          = Status-Line
                    *( general-header
                    | response-header
                    | entity-header )
                    CRLF
                    [ message-body ]
```

HTTP Status-Line

The first line of a response message is the `Status-Line,` consisting of the protocol version followed by a numeric status code and its associated textual phrase, with each element separated by `SP` characters. No `CR` or `LF` is allowed except in the final `CRLF` sequence.

Status-Line = HTTP-Version SP Status-Code(SP)Reason-Phrase CRLF

HTTP Status-Code and Reason-Phrase

The `Status-Code` element is a three-digit integer result code of the attempt to understand and satisfy the request. The `Reason-Phrase` is intended to give a short textual description of the `Status-Code.` The `Status-Code` is intended for use by automata and the `Reason-Phrase` is intended for the human user. The client is not required to examine or display the `Reason-Phrase.` The first digit of the `Status-Code` defines the class of response. The last two digits do not have any categorization role. There are five values for the first digit:

`1xx: Informational`	Request received, continuing process.
`2xx: Success`	The action was successfully received, understood, and accepted.
`3xx: Redirection`	Further action must be taken in order to complete the request.
`4xx: Client Error`	The request contains bad syntax or cannot be fulfilled.
`5xx: Server Error`	The server failed to fulfill an apparently valid request.

The individual values of the numeric status codes defined for HTTP/1.1, and an example set of corresponding `Reason-Phrases,` are presented below. The reason phrases listed here are only recommended is that they may be replaced by local equivalents without affecting the protocol.

Status-Code = “100” ; Continue
| “101” ; Switching Protocols
| “200” ; OK
| “201” ; Created
| “202” ; Accepted
| “203” ; Non-Authoritative Information
| “204” ; No Content

| “205”; Reset Content
| “206”; Partial Content
| “300”; Multiple Choices
| “301”; Moved Permanently
| “302”; Moved Temporarily
| “303”; See Other
| “304”; Not Modified
| “305”; Use Proxy
| “400”; Bad Request
| “401”; Unauthorized
| “402”; Payment Required
| “403”; Forbidden
| “404”; Not Found
| “405”; Method Not Allowed
| “406”; Not Acceptable
| “407”; Proxy Authentication Required
| “408”; Request Time-out
| “409”; Conflict
| “410”; Gone
| “411”; Length Required
| “412”; Precondition Failed
| “413”; Request-Entity Too Large
| “414”; Request-URI Too Large
| “415”; Unsupported Media Type
| “500”; Internal Server Error
| “501”; Not Implemented

```
                | “502”; Bad Gateway
                | “503”; Service Unavailable
                | “504”; Gateway Time-out
                | “505”; HTTP Version not supported
                | extension-code
                    extension-code = 3DIGIT
                    Reason-Phrase = *<TEXT, excluding CR, LF>
```

HTTP status codes are extensible. HTTP applications are not required to understand the meaning of all registered status codes, although such understanding is obviously desirable. Nevertheless, applications must understand the class of any status code, as indicated by the first digit, and treat any unrecognized response as being equivalent to the x00 status code of that class; however, an unrecognized response must not be cached. For example, if an unrecognized status code of 431 is received by the client, it can safely assume that there was something wrong with its request and treat the response as if it had received a 400 status code. In such cases, user agents should present to the user the entity returned with the response, since that entity is likely to include human-readable information which will explain the unusual status.

HTTP response-header fields

The `response-header` fields allow the server to pass additional information about the response which cannot be placed in the `Status-Line.` These header fields give information about the server and about further access to the resource identified by the `Request-URI.`

```
response-header = Age
            | Location
            | Proxy-Authenticate
            | Public
            | Retry-After
            | Server
            | Vary
            | Warning
              | WWW-Authenticate
```

Response-header field names can be extended reliably only in combination with a change in the protocol version. However, new or experimental header fields may be given the semantics of response-header fields if all parties in the communication recognize them to be response-header fields. Unrecognized header fields are treated as entity-header fields.

8.4.23 HTTP Entity

Request and response messages may transfer an entity if not otherwise restricted by the request method or response status code. An entity consists of entity-header fields and an entity body, although some responses will only include the entity headers.

Entity-header Fields

Entity-header fields define optional metainformation about the entity body or, if no body is present, about the resource identified by the request.

```
entity-header = Allow
            | Content-Base
            | Content-Encoding
            | Content-Language
            | Content-Length
            | Content-Location
            | Content-MD5
            | Content-Range
            | Content-Type
            | ETag
            | Expires
            | Last-Modified
            | extension-header
extension-header = message-header
```

The extension-header mechanism allows additional entity-header fields to be defined without changing the protocol, but these fields can not be assumed to be recognizable by the recipient. Unrecognized header fields should be ignored by the recipient and forwarded by proxies.

HTTP Entity Body

The entity body (if any) sent with an HTTP request or response is in a format and encoding defined by the entity-header fields.

entity-body = *OCTET

An entity body is present in a message only when a message body is present. The entity body is obtained from the message body by decoding any `Transfer-Encoding` that may have been applied to ensure safe and proper transfer of the message.

When an entity body is included with a message, the *data type* of that body is determined via the header fields `Content-Type` and `Content-Encoding.` These define a two-layer, ordered encoding model:

entity-body : = Content-Encoding(Content-Type(data))

`Content-Type` specifies the media type of the underlying data. `Content-Encoding` may be used to indicate any additional content codings applied to the data, usually for the purpose of data compression, that are a property of the requested resource. There is no default encoding.

Any HTTP/1.1 message containing an entity body should include a `Content-Type` header field defining the media type of that body. If and only if the media type is not given by a `Content-Type` field, the recipient may attempt to guess the media type via inspection of its content and/or the name extension(s) of the URL used to identify the resource. If the media type remains unknown, the recipient should treat it as type `application/octet-stream.`

The *length* of an entity body is the length of the message body after any transfer codings have been removed.

8.4.24 HTTP Persistent Connections

Prior to persistent connections, a separate TCP connection was established to fetch each URL, increasing the load on HTTP servers and causing congestion on the Internet. The use of inline images and other associated data often requires a client to make multiple requests of the same server in a short amount of time. Analyses of these performance problems are available; analysis and results from a prototype implementation are in. Persistent HTTP connections have a number of advantages:

• By opening and closing fewer TCP connections, CPU time is saved, and memory used for TCP protocol control blocks is also saved.

• HTTP requests and responses can be pipelined on a connection. Pipelining allows a client to make multiple requests without waiting for each response, allowing a single TCP connection to be used much more efficiently, with much lower elapsed time.

• Network congestion is reduced by reducing the number of packets caused by TCP opens, and by allowing TCP sufficient time to determine the congestion state of the network.

HTTP can evolve more gracefully; since errors can be reported without the penalty of closing the TCP connection. Clients using future versions of HTTP might optimistically try a new feature, but if communicating with an older server, retry with old semantics after an error is reported.

HTTP Persistent Connection Operation

A significant difference between HTTP/1.1 and earlier versions of HTTP is that persistent connections are the default behavior of any HTTP connection. That is, unless otherwise indicated, the client may assume that the server will maintain a persistent connection.

Persistent connections provide a mechanism by which a client and a server can signal the close of a TCP connection. This signaling takes place using the `Connection` header field. Once a close has been signaled, the client *must* not send any more requests on that connection.

1. *Negotiation.* An HTTP/1.1 server may assume that an HTTP/1.1 client intends to maintain a persistent connection unless a `Connection` header including the `Connection-token` “close” was sent in the request. If the server chooses to close the connection immediately after sending the response, it should send a `Connection` header including the `Connection-token` close. An HTTP/1.1 client may expect a connection to remain open, but would decide to keep it open if the response from a server contains a `Connection` header with the `Connection-token` close. In case the client does not want to maintain a connection for more than that request, it should send a `Connection` header including the `Connection-token` close.

If either the client or the server sends the close token in the `Connection` header, that request becomes the last one for the connection. Clients and servers should not assume that a persistent connection is maintained for HTTP versions less than 1.1 unless it is explicitly signaled. In order to remain persistent, all messages on the `Connection` must have a self-defined message length; that is, one not defined by closure of the connection.

2. *Pipelining.* A client that supports persistent connections may “pipeline” its requests (i.e., send multiple requests without waiting for each response). A server must send its responses to those requests in the same order that the requests were received.

Clients which assume persistent connections and pipeline immediately after connection establishment should be prepared to retry their connection if the first pipelined attempt fails. If a client does such a retry, it must not pipeline before it knows the connection is persistent. Clients *must* also be prepared to resend their requests if the server closes the connection before sending all of the corresponding responses.

HTTP Proxy Servers

It is especially important that proxies correctly implement the properties of the `Connection` header field. The proxy server must signal persistent connections separately with its clients and the origin servers (or other proxy servers) that it connects to. Each persistent connection applies to only one transport link. A proxy server must not establish a persistent connection with an HTTP/1.0 client.

8.4.25 HTTP Caching

HTTP is typically used for distributed information systems, where performance can be improved by the use of response caches. The HTTP/1.1 protocol includes a number of elements intended to make caching work as well as possible. Because these elements are inextricable from other aspects of the protocol, and because they interact with each other, it is useful to describe the basic caching design of HTTP separately from the detailed descriptions of methods, headers, response codes, and so on.

Caching would be useless if it did not significantly improve performance. The goal of caching in HTTP/1.1 is to eliminate the need to send requests in many cases, and to eliminate the need to send full responses in many other cases. The former reduces the number of network round trips required for many operations; we use an expiration mechanism for this purpose. The latter reduces network bandwidth requirements; we use a “validation” mechanism for this purpose.

Requirements for performance, availability, and disconnected operation require us to be able to relax the goal of semantic transparency. The HTTP/1.1 protocol allows origin servers, caches, and clients to explicitly reduce transparency when necessary. However, because nontransparent operation may confuse nonexpert users, and may be incompatible with certain server applications (such as those for ordering merchandise), the protocol requires that transparency be relaxed only by an explicit protocol-level request when relaxed by client or origin server; or the other case is only with an explicit warning to the end user when relaxed by cache or client.

Therefore, the HTTP/1.1 protocol provides these important elements:

1. Protocol features that provide full semantic transparency when this is required by all parties.

2. Protocol features that allow an origin server or user agent to explicitly request and control nontransparent operation.

3. Protocol features that allow a cache to attach warnings to responses that do not preserve the requested approximation of semantic transparency.

A basic principle is that it must be possible for the clients to detect any potential relaxation of semantic transparency. The server, cache, or client implementers may be faced with design decisions not explicitly discussed in this specification. If a decision may affect semantic transparency, the implementers ought to err on the side of maintaining transparency unless a careful and complete analysis shows significant benefits in breaking transparency.

8.4.26 HTTP History Lists

User agents often have history mechanisms, such as BACK buttons and history lists, which can be used to redisplay an entity retrieved earlier in a session. History mechanisms and caches are different. In particular, history mechanisms should not try to show a semantically transparent view of the current state of a resource. Rather, a history mechanism is meant to show exactly what the user saw at the time when the resource was retrieved.

By default, an expiration time does not apply to history mechanisms. If the entity is still in storage, a history mechanism should display it even if the entity has expired, unless the user has specifically configured the agent to refresh expired history documents.

This should not be construed to prohibit the history mechanism from telling the user that a view may be stale. If history-list mechanisms unnecessarily prevent users from viewing stale resources, this will tend to force service authors to avoid using HTTP expiration controls and cache controls when they would otherwise like to. Service authors may consider it important that users not be presented with error messages or warning messages when they use navigation controls (such as BACK) to view previously fetched resources. Even though sometimes such resources ought not to be cached, or ought to expire quickly, user interface considerations may force service authors to resort to other means of preventing caching (e.g., “once-only” URLs) to avoid the effects of improperly functioning history mechanisms.

8.4.27 HTTP Header Field Definitions

This section defines the syntax and semantics of all standard HTTP/1.1 header fields. For entity-header fields, both sender and recipient refer to either the client or the server, depending on who sends and who receives the entity.

Accept

The `Accept request-header` field can be used to specify certain media types which are acceptable for the response. `Accept` headers can be used to indicate that the request is specifically limited to a small set of desired types, as in the case of a request for an in-line image.

```
Accept    =  “Accept” “:”
        #( media-range [ accept-params ] )
media-range           = (“*/*”
        | ( type “/” “*” )
        | ( type “/” subtype )
        ) *( “;” parameter )
accept-params = “;” “q” “ = ”qvalue *(accept-extension)
accept-extension = “;” token [“ = ” (token|quoted-string)]
```

The asterisk (*) character is used to group media types into ranges, with the slash bar (`/`) indicating all media types and `type/*` indicating all subtypes of that type. The media range may include media-type parameters that are applicable to that range. Each media range may be followed by one or more `accept-params,` beginning with the `q` parameter for indicating a relative quality factor. The first `q` parameter (if any) separates the media-range parameter(s) from the `accept-params.` Quality factors allow the user or user agent to indicate the relative degree of preference for that media range, using the `q value` scale from 0 to 1. The default value is `q` = 1. Use of the `q` parameter name to separate media-type parameters from `Accept` extension parameters is due to historical practice. Although this prevents any media- type parameter named `q` from being used with a media range, such an event is believed to be unlikely given the lack of any `q` parameters in the IANA media-type registry and the rare usage of any media-type parameters in `Accept.` Future media types should be discouraged from registering any parameter named `q.` Consider the following example:

Accept: audio/*; q = 0.2, audio/basic

This should be interpreted as “I prefer `audio/basic,` but send me any audio type if it is the best available after an 80 percent markdown in quality.” If no `Accept` header field is present, then it is assumed that the client accepts all media types. If an `Accept` header field is present, and if the server cannot send a response which is acceptable according to the combined `Accept` field value, then the server should send a 406 (not acceptable) response. A more elaborate example is

Accept: text/plain; q = 0.5, text/html
text/x-dvi; q = 0.8, text/x-c

Verbally, this would be interpreted as “`text/html` and `text/x-c` are the preferred media types, but if they do not exist, then send the `text/x-dvi` entity, and if that does not exist, send the `text/plain` entity.”

Media ranges can be overridden by more specific media ranges or specific media types. If more than one media range applies to a given type, the most specific reference has precedence. Consider the following:

Accept: text/*, text/html, text/html;level = 1, */*

The following apply to the previous example and have precedence: (1) `Text/html;level` = 1, (2) `text/html`, (3) `text/*`, and (4) `*/*`.

The media-type quality factor associated with a given type is determined by finding the media range with the highest precedence which matches that type. Consider the following:

Accept: text/*;q = 0.3, text/html;q = 0.7, text/html;level = 1
text/html;level = 2;q = 0.4, */*;q = 0.5

The previous example would cause the following values to be associated:

text/html;level = 1	= 1
text/html	= 0.7
text/plain	= 0.3
image/jpeg	= 0.5
text/html;level = 2	= 0.4
text/html;level = 3	= 0.7

A user agent may be provided with a default set of quality values for certain media ranges, unlessser agent is a closedm which cannot interact with other rendering agents.

Accept-Charset

The `Accept-Charset request-header` field can be used to indicate what character sets are acceptable for the response. This field allows clients capable of understanding more comprehensive or special-purpose character sets to signal that capability to a server which is capable of representing documents in those character sets. The ISO-8859-1 character set can be assumed to be acceptable to all user agents.

Accept-Charset = "Accept-Charset" ":"
    1#(charset [";" "q" " = " qvalue])

Each `charset` may be given an associated quality value which represents the user's preference for that `charset`. The default value is `q` = 1. An example is the following:

Accept-Charset: iso-8859-5, unicode-1-1;q = 0.8

If no `Accept-Charset` header is present, the default is that any character set is acceptable. If an `Accept-Charset` header is present, and if the server cannot send a response which is acceptable according to the `Accept-Charset` header, then the server should send an error response with the 406 ("not acceptable") status code, even though sending an unacceptable response is also allowed.

Accept-Encoding

The `Accept-Encoding request-header` field is similar to `Accept,` but restricts the content-coding values which are acceptable in the response.

Accept-Encoding = "Accept-Encoding" ":"
    #(content-coding)
Accept-Encoding: compress, gzip

If no `Accept-Encoding` header is present in a request, the server may assume that the client will accept any content coding. If an `Accept-Encoding` header is present, and if the server cannot send a response which is acceptable according to the `Accept-Encoding` header, then the server should send an error response with the 406 ("not acceptable") status code. An empty `Accept-Encoding` value indicates none are acceptable.

Accept-Language

The `Accept-Language request-header` field is similar to `Accept,` but restricts the set of natural languages that are preferred as a response to the request.

Accept-Language = "Accept-Language" ":"
    1#(language-range [";" "q" " = " qvalue])
language-range = ((1*8ALPHA *("-" 1*8ALPHA)) | "*")

Each `language-range` may be given an associated quality value which represents an estimate of the user's preference for the languages specified by that range. The quality value defaults to `q` = 1. For example:

Accept-Language: da, en-gb;q = 0.8, en;q = 0.7

The previous example means: “I prefer Danish, but will accept British English and other types of English.” A `language-range` matches a `language-tag` if it exactly equals the tag, or if it exactly equals a prefix of the tag such that the first tag character following the prefix is `"-".` The special range `"*",` if present in the `Accept-Language` field, matches every tag not matched by any other range present in the `Accept-Language` field. This use of a prefix matching rule does not imply that language tags are assigned to languages in such a way that it is always true that a user who understands a language with a certain tag will also understand all languages with tags for which this tag is a prefix. The prefix rule simply allows the use of prefix tags if this is the case.

The language quality factor assigned to a language tag by the `Accept-Language` field is the quality value of the longest `language-range` in the field that matches the language tag. If no `language-range` in the field matches the tag, the language quality factor assigned is 0. If no `Accept-Language` header is present in the request, the server should assume that all languages are equally acceptable. If an `Accept-Language` header is present, then all languages which are assigned a quality factor greater than 0 are acceptable.

It may be contrary to the privacy expectations of the user to send an `Accept-Language` header with the complete linguistic preferences of the user in every request. As intelligence is highly dependent on the individual user, it is recommended that client applications make the choice of linguistic preference available to the user. If the choice is not made available, then the `Accept-Language` header field must not be given in the request.

Accept-Ranges

The `Accept-Ranges response-header` field allows the server to indicate its acceptance of range requests for a resource:

Accept-Ranges = “Accept-Ranges” “:” acceptable-ranges
acceptable-ranges = 1#range-unit | “none”

Origin servers that accept byte-range requests may send the following:

Accept-Ranges: bytes

This may be sent but is not required. Clients may generate byte-range requests without having received this header for the resource involved. Servers that do not accept any kind of range request for a resource may send the following to advise the client to not attempt a range request:

Accept-Ranges: none

Age

The `Age` response-header field conveys the sender’s estimate of the amount of time since the response (or its revalidation) was generated at the origin server. A cached response is “fresh” if its age does not exceed its freshness lifetime.

Age = “Age” “:” age-value
age-value = delta-seconds

`Age` values are nonnegative decimal integers, representing time in seconds. If a cache receives a value larger than the largest positive integer it can represent, or if any of its age calculations overflow, it must transmit an `Age` header with a value of 2147483648. HTTP/1.1 caches *must* send an `Age` header in every response. Caches should use an arithmetic type of at least 31 bits of range.

Allow

The `Allow` entity-header field lists the set of methods supported by the resource identified by the `Request-URI.` The purpose of this field is strictly to inform the recipient of valid methods associated with the resource. An `Allow` header field is required in a 405 (“method not allowed”) response.

Allow = “Allow” “:” 1#method
Example of use: Allow: GET, HEAD, PUT

This field cannot prevent a client from trying other methods. However, the indications given by the `Allow` header field value should be followed. The actual set of allowed methods is defined by the origin server at the time of each request.

The `Allow` header field may be provided with a PUT request to recommend the methods to be supported by the new or modified resource. The server is not required to support these methods and include an `Allow` header in the response giving the actual supported methods.

A proxy cannot modify the `Allow` header field even if it does not understand all the methods specified, since the user agent may have other means of communicating with the origin server.

The `Allow` header field does not indicate what methods are implemented at the server level. Servers may use the `Public` response-header field to describe what methods are implemented on the server as a whole.

Authorization

A user agent that wishes to authenticate itself with a server, usually but not necessarily, after receiving a 401 response, may do so by including an `Authorization` request-header field with the request. The `Authorization` field value consists of credentials containing the authentication information of the user agent for the realm of the resource being requested.

Authorization = “Authorization” “:” credentials

If a request is authenticated and a realm specified, the same credentials should be valid for all other requests within this realm. When a shared cache receives a request containing an `Authorization` field, it *must not* return the corresponding response as a reply to any other request, unless one of the following specific exceptions holds:

1. If the response includes the *`proxy-revalidate`* `Cache-Control` directive, the cache may use that response in replying to a subsequent request, but a proxy cache must first revalidate it with the origin server, using the `request-headers` from the new request to allow the origin server to authenticate the new request.

2. If the response includes the *`must-revalidate`* `Cache-Control` directive, the cache may use that response in replying to a subsequent request, but all caches must first revalidate it with the origin server, using the request headers from the new request to allow the origin server to authenticate the new request.

3. If the response includes the *`public`* `Cache-Control` directive, it may be returned in reply to any subsequent request.

Cache-Control

The `Cache-Control` general-header field is used to specify directives that must be obeyed by all caching mechanisms along the request/response chain. The directives specify behavior intended to prevent caches from adversely interfering with the request or response. These directives typically override the default caching algorithms. Cache directives are unidirectional in that the presence of a directive in a request does not imply that the same directive should be given in the response. HTTP/1.0 caches may not implement `Cache-Control` and may only implement `Pragma`; that is, `no-cache.`

Cache directives must be passed through by a proxy or gateway application, regardless of their significance to that application, since the directives may be applicable to all recipients along the request/response chain. It is not possible to specify a `cache directive` for a specific cache.

```
Cache-Control = “Cache-Control” “:” 1#cache-directive
cache-directive = cache-request-directive
        | cache-response-directive
cache-request-directive = “no-cache” [“ = “<”>1#field-name<”>]
        | “no-store”
        | “max-age” “ = ”delta-seconds
        | “max-stale” [“ = ” delta-seconds ]
        | “min-fresh” “ = ” delta-seconds
        | “only-if-cached”
        | cache-extension



cache-response-directive = “public”
        | “private” [“ = ” <”> 1#field-name <”> ]
        | “no-cache” [“ = ” <”> 1#field-name <”> ]
        | “no-store”
        | “no-transform”
        | “must-revalidate”
        | “proxy-revalidate”
        | “max-age” “ = ” delta-seconds
        | cache-extension
cache-extension = token [“ = ” ( token | quoted-string ) ]
```

When a directive appears without any `1#field-name` parameter, the directive applies to the entire request or response. When such a directive appears with a `1#field-name` parameter, it applies only to the named field or fields, and not to the rest of the request or response. This mechanism supports extensibility; implementations of future versions of the HTTP protocol may apply these directives to header fields not defined in HTTP/1.1. The `cache-control` directives can be broken down into these general categories:

• Restrictions on what is cacheable; these may be imposed only by the origin server.

• Restrictions on what may be stored by a cache; these may be imposed by either the origin server or the user agent.

• Modifications of the basic expiration mechanism; these may be imposed by either the origin server or the user agent.

• Controls over cache revalidation and reload; these may only be imposed by a user agent.

• Control over transformation of entities.

• Extensions to the caching system.

8.4.28 HTTP Security Considerations

The basic authentication scheme is not a secure method of user authentication, nor does it in any way protect the entity, which is transmitted in clear text across the physical network used as the carrier. HTTP does not prevent additional authentication schemes and encryption mechanisms from being employed to increase security or the addition of enhancements (such as schemes to use one-time passwords) to basic authentication.

The most serious flaw in basic authentication is that it results in the essentially clear text transmission of the user's password over the physical network. It is this problem which digest authentication attempts to address.

Because basic authentication involves the clear text transmission of passwords, it should never be used (without enhancements) to protect sensitive or valuable information.

A common use of basic authentication is for identification purposes requiring the user to provide a user name and password as a means of identification, for example, for purposes of gathering accurate usage statistics on a server. When used in this way, it is tempting to think that there is no danger in its use if illicit access to the protected documents is not a major concern. This is only correct if the server issues both user name and password to the users and in particular does not allow the user to choose his or her own password. The danger arises because naive users frequently reuse a single password to avoid the task of maintaining multiple passwords.

If a server permits users to select their own passwords, then the threat is not only illicit access to documents on the server but also illicit access to the accounts of all users who have chosen to use their account password. If users are allowed to choose their own password that also means the server must maintain files containing the (presumably encrypted) passwords. Many of these may be the account passwords of users perhaps at distant sites. The owner or administrator of such a system could conceivably incur liability if this information is not maintained in a secure fashion.

Basic authentication is also vulnerable to spoofing by counterfeit servers. If a common user can be led to believe that she is connecting to a host containing information protected by basic authentication when in fact she is connecting to a hostile server or gateway, then the attacker can request a password, store it for later use, and feign an error. This type of attack is not possible with digest authentication. Server implementers should guard against the possibility of this sort of counterfeiting by gateways or Common Gateway Interface (CGI) scripts. In particular it is very dangerous for a server to simply turn over a connection to a gateway since that gateway can then use the persistent connection mechanism to engage in multiple transactions with the client while impersonating the original server in a way that is not detectable by the client.

An HTTP/1.1 server may return multiple challenges with a 401 (“authenticate”) response, and each challenge may use a different scheme. The challenges are returned to the user agent in the same order that the server would prefer they be chosen. The server should order its challenges with the “most secure” authentication scheme first. A user agent should choose as the challenge to be made to the user the first one that the user agent understands.

When the server offers choices of authentication schemes using the `WWW-Authenticate` header, the “security” of the authentication is such that only malicious users could capture the set of challenges and try to authenticate themselves using the weakest of the authentication schemes. Thus, the ordering serves more to protect the user's credentials than the server's information.

A possible person-in-the-middle (PITM) attack would be to add a weak authentication scheme to the set of choices, hoping that the client will use one that exposes the user's credentials (e.g., password). For this reason, the client should always use the strongest scheme that it understands from the choices accepted. An even better PITM attack would be to remove all offered choices, and to insert a challenge that requests basic authentication. For this reason, user agents that are concerned about this kind of attack could remember the strongest authentication scheme ever requested by a server and produce a warning message that requires user confirmation before using a weaker one. A particularly insidious way to mount such a PITM attack would be to offer a "free" proxy caching service to gullible users.

A server is in the position to save personal data about a user's requests which may identify their reading patterns or subjects of interest. This information is clearly confidential in nature and its handling may be constrained by law in certain countries. People using the HTTP protocol to provide data are responsible for ensuring that such material is not distributed without the permission of any individuals that are identifiable by the published results.

Like any generic data-transfer protocol, HTTP cannot regulate the content of the data that is transferred, nor is there any a priori method of determining the sensitivity of any particular piece of information within the context of any given request. Therefore, applications should supply as much control over this information as possible to the provider of that information. Four header fields are worth special mention in this context: `Server, Via, Referrer,` and `From.` Revealing the specific software version of the server may allow the server machine to become more vulnerable to attacks against software that is known to contain security holes. Implementers should make the `Server` header field a configurable option.

Proxies which serve as a portal through a network firewall should take special precautions regarding the transfer of header information that identifies the hosts behind the firewall. In particular, they should remove, or replace with sanitized versions, any `Via` fields generated behind the firewall.

The `Referrer` field allows reading patterns to be studied and reverse links drawn. Although it can be very useful, its power can be abused if user details are not separated from the information contained in the `Referrer.` Even when the personal information has been removed, the `Referrer` field may indicate a private document's URI whose publication would be inappropriate. The information sent in the `From` field might conflict with the user's privacy interests or their site's security policy, and hence it should not be transmitted without the user being able to disable, enable, and modify the contents of the field. The user *must* be able to set the contents of this field within a user preference or application defaults configuration.

We suggest, though do not require, that a convenient toggle interface be provided for the user to enable or disable the sending of `From` and `Referrer` information.

Implementations of HTTP origin servers should be careful to restrict the documents returned by HTTP requests to be only those that were intended by the server administrators. If an HTTP server translates HTTP URIs directly into file system calls, the server *must* take special care not to serve files that were not intended to be delivered to HTTP clients. For example, UNIX, Microsoft Windows, and other operating systems use ".." as a path component to indicate a directory level above the current one. On such a system, an HTTP server *must* disallow any such construct in the `Request-URI` if it would otherwise allow access to a resource outside those intended to be accessible via the HTTP server. Similarly, files intended for reference only internally to the server (such as access control files, configuration files, and script code) *must* be protected from inappropriate retrieval, since they might contain sensitive information. Experience has shown that minor bugs in such HTTP server implementations have turned into security risks.

HTTP clients are often privy to large amounts of personal information (the user's name, location, mail address, passwords, encryption keys, etc.), and should be very careful to prevent unintentional leakage of this information via the HTTP protocol to other sources. A convenient interface may be provided for the user to control dissemination of such information, and consequently designers and implementers should be particularly careful in this area. History shows that errors in this area are often both serious security and/or privacy problems, and often generate highly adverse publicity for the implementer's company.

`Accept request-headers` can reveal information about the user to all servers which are accessed. The `Accept-Language` header in particular can reveal information the user would consider to be of a private nature, because the understanding of particular languages is often strongly correlated to the membership of a particular ethnic group. User agents which offer the option to configure the contents of an `Accept-Language` header to be sent in every request are strongly encouraged to let the configuration process include a message which makes the user aware of the loss of privacy involved.

An approach that limits the loss of privacy would be for a user agent to omit the sending of `Accept-Language` headers by default, and to ask the user whether it should start sending `Accept-Language` headers to a server if it detects, by looking for any `Vary response-header` fields generated by the server, that such sending could improve the quality of service.

Elaborate user-customized `Accept` header fields sent in every request, in particular if these include quality values, can be used by servers as relatively reliable and long-lived user identifiers. Such user identifiers would allow content providers to do click-trail tracking, and would allow collaborating content providers to match cross-server click-trails or form submissions of individual users. Note that for many users not behind a proxy, the network address of the host running the user agent will also serve as a long-lived user identifier. In environments where proxies are used to enhance privacy, user agents should be conservative in offering `Accept` header configuration options to end users. As an extreme privacy measure, proxies could filter the `Accept` headers in relayed requests. General-purpose user agents which provide a high degree of header configurability should warn users about the loss of privacy which can be involved.

8.4.29 HTTP and DNS Spoofing

Clients using HTTP rely heavily on the Domain Name Service, and are thus generally prone to security attacks based on the deliberate misassociation of IP addresses and DNS names. Clients need to be cautious in assuming the continuing validity of an IP number/DNS name association.

In particular, HTTP clients should rely on their name resolver for confirmation of an IP number/DNS name association, rather than caching the result of previous host name lookups. Many platforms already can cache host name lookups locally when appropriate, and they should be configured to do so. These lookups should be cached, however, only when the TTL (time to live) information reported by the name server makes it likely that the cached information will remain useful.

If HTTP clients cache the results of host name lookups in order to achieve a performance improvement, they *must* observe the TTL information reported by DNS. If HTTP clients do not observe this rule, they could be spoofed when a previously accessed server's IP address changes. As network renumbering is expected to become increasingly common, the possibility of this form of attack will grow. Observing this requirement thus reduces this potential security vulnerability.

This requirement also improves the load-balancing behavior of clients for replicated servers using the same DNS name and reduces the likelihood of a user's experiencing failure in accessing sites which use that strategy. If a single server supports multiple organizations that do not trust one another, then it must check the values of `Location` and `Content-Location` headers in responses that are generated under control of said organizations to make sure that they do not attempt to invalidate resources over which they have no authority.

8.4.30 Additional information on HTTP

The following documents provide further information on HTTP:

Tags for the Identification of Languages, RFC 1766

The Internet Gopher Protocol, RFC 1436

Universal Resource Identifiers in WWW: A Unifying Syntax for the Expression of Names and Addresses of Objects on the Network as used in the World-Wide Web, RFC 1630

Uniform Resource Locators (*URL*), RFC 1738

HyperText Markup Language Specification 2.0, RFC 1866

Hypertext Transfer Protocol HTTP/1.0, RFC 1945

Multipurpose Internet Mail Extensions (*MIME*), Part One: *Format of Internet Message Bodies,* RFC 2045

Requirements for Internet Hosts, Applications, and Support, RFC 1123

Standard for the Format of ARPA Internet Text Messages, RFC 822

WAIS Interface Protocol Prototype Functional Specification Relative Uniform Resource Locators, RFC 1808

Standard for Interchange of USENET Messages, RFC 1036

Network News Transfer Protocol. A Proposed Standard for the Stream-Based Transmission of News, RFC 977

MIME (*Multipurpose Internet Mail Extensions*), Part Three: *Message Header Extensions for Non-ASCII Text,* RFC 2047

Form-Based File Upload in HTML, RFC 1867

Simple Mail Transfer Protocol, RFC 821

Media Type Registration Procedure, RFC 2048

File Transfer Protocol (FTP), RFC 959

Assigned Numbers, RFC 1700

Functional Requirements for Uniform Resource Names, RFC 1737

US-ASCII. Coded Character Set—7-Bit American Standard Code for Information Interchange. International Standard Information Processing 8-bit Single-Byte Coded Graphic Character Sets, Standard ANSI X3.4-1986, ANSI

ISO-8859.

Part 1:	*Latin Alphabet No. 1*, ISO 8859-1:1987
Part 2:	*Latin Alphabet No. 2*, ISO 8859-2, 1987
Part 3:	*Latin Alphabet No. 3*, ISO 8859-3, 1988
Part 4:	*Latin Alphabet No. 4*, ISO 8859-4, 1988
Part 5:	*Latin/Cyrillic Alphabet*, ISO 8859-5, 1988
Part 6:	*Latin/Arabic Alphabet*, ISO 8859-6, 1987
Part 7:	*Latin/Greek Alphabet*, ISO 8859-7, 1987
Part 8:	*Latin/Hebrew Alphabet*, ISO 8859-8, 1988
Part 9:	*Latin Alphabet No. 5*, ISO 8859-9, 1990

The Content-MD5 Header Field, RFC 1864

Renumbering Needs Work, RFC 1900

GZIP file format specification version 4.3, RFC 1952

Analysis of HTTP Performance, <URL: http://www.isi.edu/lsam/ ib/http-perf/>

Network Time Protocol, V.3, Specification, Implementation, & Analysis, RFC 1305

DEFLATE Compressed Data Format Specification V.1.3, RFC 1951

Analysis of HTTP Performance Problems, <URL: http:// sunsite.unc.edu/mdma-release/http-prob.html>

ZLIB Compressed Data Format Specification V.3.3, RFC 1950

An Extension to HTTP: Digest Access Authentication, RFC 2069

8.5 Simple Mail Transfer Protocol

The *Simple Mail Transfer Protocol* (SMTP) is another TCP application. It does not use a client and server, but the functionality is similar. SMTP utilizes what is called a user agent and a message transfer agent. Figure 8-6 is a simple example of how SMTP operates.

Sending mail is accomplished by invoking a user agent, which, in turn, causes an editor to appear on the user's display. After the mail message is created and sent from the user agent, it is transferred to the sending message transfer agent, who is responsible for establishing communication with the message transfer agent on the destination host. Once this is accomplished, the sending message transfer agent sends the message to the receiving message transfer agent, which then stores it in the appropriate queue for the user. The recipient of the mail only needs to invoke the user agent on that machine to read the mail.

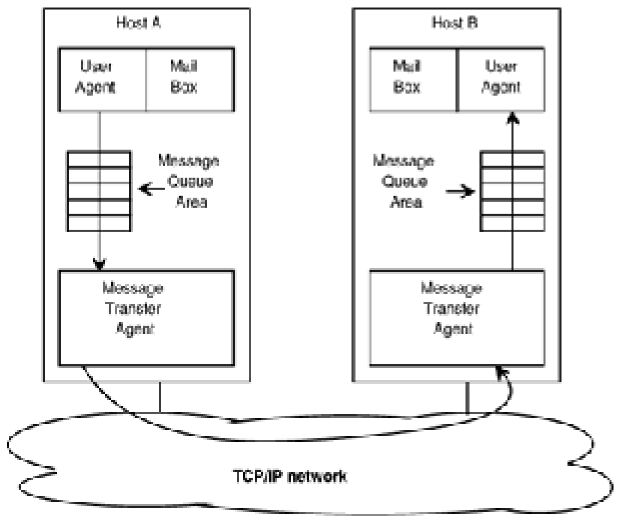


Figure 8-6
SMTP components.

8.6 Domain Name System

In the beginning, the Internet used *hosts* files to keep track of hosts on the Internet. This meant that when new hosts were added to the Internet, all participating hosts had to have their *hosts* file updated. As the Internet grew, this task of updating the *hosts* file became insurmountable. The Domain Name System (DNS) grew from the need to replace such a system.

The philosophy of DNS was to replace the need to FTP updated hosts files throughout the entire network. Thus, the foundation of DNS was built around a distributed database architecture.

DNS Structure

DNS is a hierarchical structure that in theory resembles an upside-down tree. The root is at the top and layers below. Figure 8-7 illustrates an example of how DNS is implemented on the Internet.

The legend for the DNS structure in Fig. 8-7 is ROOT—the root server contains information about itself and the top-level domains immediately beneath it; GOV—government entities; EDU—any educational institutions; ARPA—any ARPAnet (Internet) host ID; COM—any commercial organizations; MIL—military organizations; ORG—serves as a miscellaneous category for those not formally covered; CON—counties conforming to ISO standards.

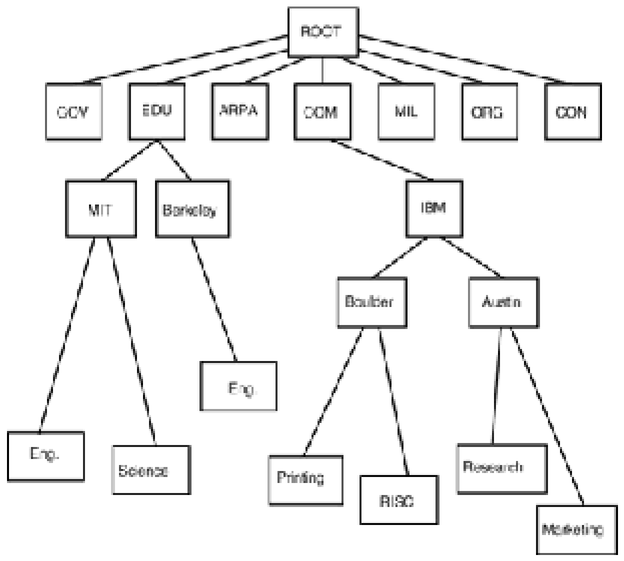


Figure 8-7
DNS structure.

Figure 8-7 shows the Internet implementation of DNS. Three examples are shown to clarify the structure. Notice that IBM is under COM (which is commercial), beneath IBM are Boulder and Austin, beneath Austin are research and marketing, and beneath Boulder are printing and RISC. The other examples are MIT and Berkeley. The example with MIT shows beneath it two *zones:* engineering and science. The Berkeley example has one layer beneath it entitled *engineering.*

At a local level, such as in a corporation, most sites follow the naming scheme and structure because it is consistent and if a connection is ever made to the Internet restructuring of DNS is not necessary.

DNS Components

To better understand DNS knowing the components that make it functional is helpful. These components include:

domain The last part in a domain name is considered the domain. An example is `eng.mit.edu`; here `edu` is the domain.

domain name Defined by the DNS as being the sequence of names and domain. For example, a domain name could be `eng.mit.edu.`

label The DNS identifies each part of a domain name as a label. For example, `eng.mit.edu` has three labels: `eng.mit.edu.`

name cache Storage used by the name resolver to store frequently used information.

name resolver This is software that functions as a client regarding its interaction with a name server. Sometimes referred to simply as the *client.*

name server A program operating on a host that translates names to addresses; it does this by mapping domain names to IP addresses. Also, name server may be used to refer to a dedicated processor running name server software.

zone A contiguous part of a domain.

Figure 8-8 shows a TCP/IP network with five hosts.

1. *Theory of operation.* Of the five hosts shown in Fig. 8-8, host B has been designated as the name server. It has a database with a list of aliases and IP addresses of participating hosts in the network. When the user on host A wants to communicate with host C, the name resolver checks its local cache; if no match is found, then the name resolver (client) sends a request (also known as a query) to the name server, which then checks its cache for a match. If no match is found, then the name server checks its database. Although not shown in this figure, if the name server were unable to locate the name in its cache or database, it would forward the request to another name server then return the response back to host A. In an Internet environment that implements DNS, some givens are assumed. For example, a name resolver is required, a name server, and usually a foreign name server is part of the network.

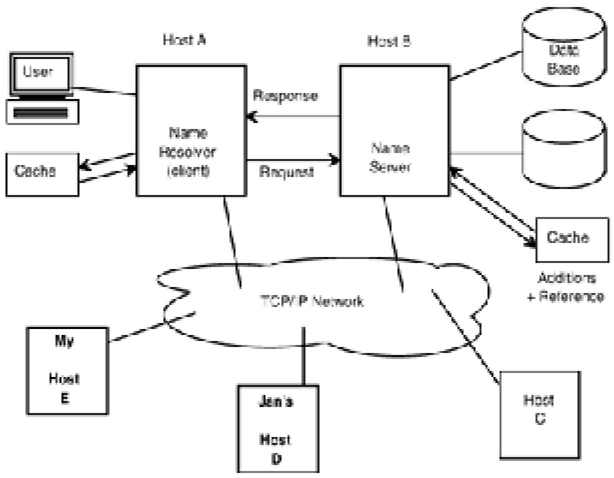


Figure 8-8
Conceptual view of DNS

2. *Implementation with UDP.* The DNS provides service for TCP and UDP; this is why figures have shown DNS residing above part of TCP and part of UDP. It serves the same purpose for both transport-layer protocols.

3. *Obtaining additional information.* Additional information can be obtained in any issue where DNS is implemented. RFCs 882, 883, 920, 973, 974, 1032-1035, and 1123 are a good starting point.

8.7 Popular UDP Applications

This section presents popular UDP applications, including a Simple Network Management Protocol (SNMP), Trivial File Transfer Protocol (TFTP), Network File System (NFS), Remote Procedure Call (RPC), custom applications, and Packet Internet Groper (PING) and FINGER.

8.8 Simple Network Management Protocol

The Simple Network Management Protocol (SNMP) is considered the de facto standard for managing TCP/IP networks as of this writing. SNMP uses agents and application managers (or simply managers). A user agent can reside on any node that supports SNMP, and each agent maintains status information about the node on which it operates. These nodes, which may be a host, gateway, router, or other type of network device, are called *network elements* in SNMP lingo. This term *element* is merely a generic reference to a node.

Normally, multiple elements exist in a TCP/IP network and each has its own agent. Typically, one node is designated as a network management node. Some refer to this node as the *network manager*. This host (network management node) has an application that communicates with each network element to obtain the status of a given element. The network management node and the element communicate via different message types. Some of these messages are:

GET REQUEST	This request is used by the network manager to communicate with an element to request a variable or list about that particular network element.
GET RESPONSE	This is a reply to a GET REQUEST, SET REQUEST, and GET NEXT REQUEST.
GET NEXT REQUEST	This request is used to sequentially read information about an element.
SET REQUEST	This request enables variable values to be set in an element.
TRAP	This message is designed to report information such as link status, whether a neighbor responds, whether a message is received, or status of the element.

Information stored on elements are maintained in a *management information base* (MIB), a database containing information about a particular element; each element has an MIB. MIB information includes statistical information regarding segments transferred to and from the manager application, a community name, interface type, and other element-specific information.

MIB information structure is defined by the *Structure of Management Information* (SMI) language, which is used to define a data structure and methods for identifying an element for the manager application. This information identifies object variables in the MIB. Object descriptions defined by SMI include at least the following:

access Object access control is maintained via this description.

definition This provides a textual description of an object.

names This term is also synonymous with *object identifiers*. This refers to a sequence of integers.

object descriptor This is a text name ascribed to the object.

object identifier This is a numeric ID used to identify the object.

status This describes the level of object support for status.

SNMP implementations use ASN.1 for defining data structures in network elements. Because this language is based on a data-type definition, it can be used to define practically any element on a network.

SNMP itself is event-oriented. An event is generated when a change occurs to an object. SNMP operation is such that approximately every 10 to 15 min, the manager application communicates with all network elements regarding their individual MIB data.

Additional information can be obtained from RFCs 1155 to 1157.

8.9 The Trivial File Transfer Protocol

The *Trivial File Transfer Protocol* (TFTP) is an application that uses UDP as a transport mechanism. The program itself is simpler than its counterpart FTP, which uses TCP as a transport mechanism. TFTP is sufficiently small in size so that it can be part of ROM on diskless workstations.

TFTP maximum packet size is 512 bytes. Because of this and the nature of operation, TFTP is popular with network devices such as routers and bridges. If implemented, it is normally used on initial device boots.

TFTP utilizes no security provision or authentication; however, it does have some basic timing and retransmission capabilities. TFTP uses five basic types of protocol data units (PDUs): acknowledgment, data, error, read request, and write request. These PDUs are used by TFTP during file transfer. The first packet TFTP establishes a session with the target TFTP program. It then requests a file transfer between the two. Next it identifies a filename and whether a file will be read or written. These five PDUs represent the operational ability of TFTP. It is straightforward and not as complex as FTP. Additional information can be obtained from RFCs 783 and 1068.

8.10 Remote Procedure Call

RPC is a protocol. Technically speaking, it can operate over TCP or UDP as a transport mechanism. Applications use RPC to call a *routine,* thus executing like a client and making a call against a server on a remote host. This type of application programming represents a high-level, peer-oriented relationship between an application and an RPC server. Consequently, this means that these applications are portable to the extent that RPC is implemented.

Within RPC is the eXternal Data Representation (XDR) protocol. XDR data description language can be used to define datatypes when heterogeneous hosts are integrated. Having the capability to overcome the inherent characteristics of different architectures lends RPC and XDR a robust solution for distributed application communication. This language permits parameter requests to be made against a file of an unlike type. In short, XDR permits datatype definition in the form of parameters and transmission of these encoded parameters.

XDR provides data transparency by way of encoding (or encapsulating) data at the application layer, so lower layers and hardware do not have to perform any conversions. A powerful aspect of XDR is automatic data conversion performed via declaration statements and the XDR compiler. The XDR compiler generates required XDR calls, thus somewhat automating the operation. Figure 8-9 shows an example of this type implementation.

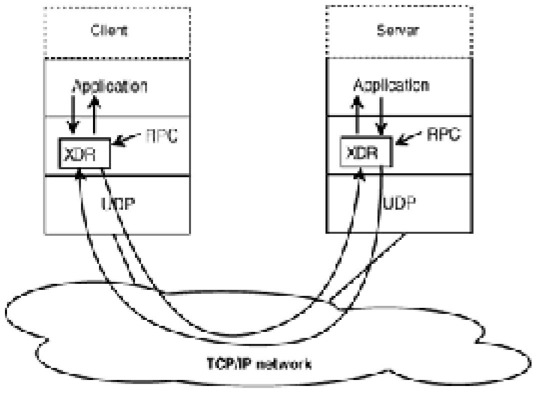


Figure 8-9
Conceptual view of RPC and XDR.

RPC implements a *port mapper,* which starts on RPC server initialization. When RPC services start, the operating system assigns a port number to each service. These services inform the port mapper of this port number, its program number, and other information required by the port mapper to know for it to match a service with a requestor.

Client applications issue a service request to a port mapper. The port mapper, in turn, identifies the requested service and returns the appropriate parameters to the requesting client application. In other words, the port mapper is similar in function to a manager knowing what services are available and their specific addressable location.

The port mapper can be used in a broadcast scenario. For example, a requesting RPC call can broadcast a call to all hosts on a network. Applicable port mappers report back to the information sought after by the client; hence, the term *remote procedure call* (RPC).

Additional information on RPC and related components can be found in RFCs 1057 and 1014.

8.11 Network File System

The *Network File System* (NFS) is a product of Sun Microsytems. It permits users to execute files without knowing the location of these files. They may be local or remote in respect to the user. Users can create, read, or remove a directory. Files themselves can be written to or deleted. NFS provides a distributed file system that permits a user to capitalize on access capabilities beyond their local file system.

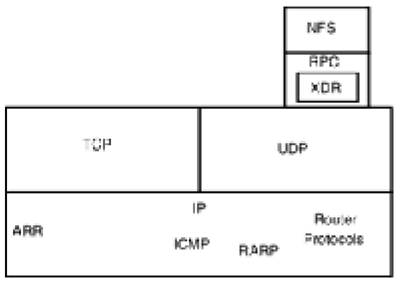


Figure 8-10
Conceptual view of NFS, RPC, and UDP.

NFS uses RPC to enable execution of a routine on a remote server. Conceptually, NFS, RPC, and UDP (which it typically uses) appear as shown in Fig. 8-10.

The idea behind NFS is to have one NFS copy of it on a server that all users on a network can access. As a consequence, software (and updates) can be installed on one server and not on multiple hosts in a networked environment. NFS is based on a client/server model. However, with NFS a single NFS server can function to serve the request of many client requests.

NFS originated in UNIX, where it is implemented in a hierarchical (tree) structure. However, NFS can operate with IBM’s VM and MVS operating systems. It can also operate with Digital Equipment’s VMS operating system.

NFS uses a *mount* protocol, which identifies a file system and remote host to a local user’s file system. The NFS mount is known by the port mapper of RPC; and thus is capable of being known by requesting client applications.

NFS also uses the NFS protocol; it performs file transfers among systems. NFS uses port number 2049 in many cases; however, this is not a well-known port number (at least at the time of this writing). Consequently, the best approach to the use of NFS is to use the NFS port number with the port mapper.

In some respects an NFS server operates with little information identified to it. A loose analogy of NFS operation is UDP. UDP assumes a custom application (or other entity operating on top of it) will perform requirements such as retransmissions (if required) and other procedures that would otherwise be performed by a connection-oriented transport protocol such as TCP. NFS assumes that required services are implemented in other protocols.

From a user perspective, NFS is transparent. Typical user commands are entered, and then passed to the NFS server, and in most cases a user does not know the physical location of a file in a networked environment.

Additional information about the Network File System and related components can be obtained through RFCs 1014, 1057, and 1094.

8.12 Custom Applications

Custom applications can be written and use UDP as a transport mechanism. One scenario could be where two hosts need peer program communication through a network. Writing a custom application using UDP can achieve this task as shown in Fig. 8-11.

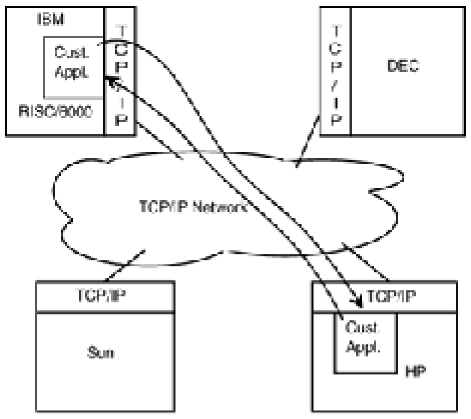


Figure 8-11
Custom applications using UDP.

8.13 PING and FINGER

Packet Internet Groper (PING) is actually a protocol that uses UDP as a transport mechanism to achieve its function. It is used to send a message to a host and wait for that host to respond to the message (if the target host is “alive”). PING uses ICMP echo messages along with the echo reply messages.

PING is a helpful tool on TCP/IP networks that is used to determine whether a device can be addressed. It is used in a network to determine whether a network itself can be addressed. A PING can also be issued against a remote host *name*. The purpose for this function is name verification and is generally used by individuals who troubleshoot TCP/IP networks.

FINGER is a command issued against a host which will cause the target host to return information about users logged onto that host. Some information retrievable via FINGER includes user name, user interface, and job name that the user is running.

Additional information on FINGER can be obtained in RFC 1288.

8.14 Summary

TCP/IP is an upper-layer protocol that has a proven track record. It began around 1975, public demonstration of its capabilities were presented in 1978, and in 1983 the DoD endorsed it as the protocol to use for connection to the Internet.

Many vendors supply TCP/IP products today. TCP/IP can operate on different hardware and software platforms. This flexibility, along with its cost-efficient pricing, does put it in a favorable position for those looking for a protocol that provides a variety of services such as remote logon, file transfer, electronic mail (email), a windowing system, and programmatic interface support.

Network management capabilities include distributed processing support and other offerings as well.

TCP/IP has two transport-layer protocols which make usability flexible; some need the reliability of transport such as TCP; others, a connectionless transport such as UDP. TCP/IP supports both.

TCP/IP has in many ways become a de facto standard in many different institutions. It is used in government, commercial business, educational, nonprofit organizations, and individual use as well. TCP/IP is dominant throughout the global marketplace today; most major vendors around the world support it to varying degrees. Its flexibility with data-link-layer protocols makes it attractive.

9
TCP/IP Routing IP V.4

9.1 IP V.4 Perspective and Functions

The *Internet Protocol* (IP), also called a *catenet,* was originally designed for use in interconnected systems of packet-switched computer communication networks. It provides for transmitting blocks of data called *datagrams* from sources to destinations, where sources and destinations are hosts identified by fixed-length addresses. IP also provides for fragmentation and reassembly of long datagrams, if necessary, for transmission through small packet networks.

IP was originally limited in purpose to provide the functions necessary to deliver a package of bits (*Internet datagrams*) from a source to a destination through an interconnected group of networks. IP has no mechanisms to augment end-to-end data reliability, flow control, sequencing, or other services commonly found in host-to-host protocols. However, IP can capitalize on the services of its supporting networks to provide various types and qualities of service.

IP is invoked by host-to-host protocols in the Internet environment. This protocol calls on local network protocols to carry the Internet datagram to the next gateway or destination host. In this case, a TCP module would call on the IP part to take a TCP segment (the TCP header and user data) as the *data* portion of an Internet datagram.

IP implements two basic functions: addressing and fragmentation. Internet modules use these addresses carried in the Internet header to transmit Internet datagrams toward their destinations. The selection of a path for transmission is called *routing.* Fields in the IP are used to fragment and reassemble Internet datagrams when necessary for transmission through small packet-oriented networks.

An Internet module resides in each host engaged in Internet communication and in each router that interconnects networks. These modules share common rules for interpreting address fields and for fragmenting and assembling Internet datagrams and have procedures for making routing decisions and other functions.

IP treats each Internet datagram as an independent entity unrelated to any other Internet datagram. Hence, there are no connections or logical circuits. IP uses four key mechanisms in providing its service: type of service, time to live, options, and header checksum.

1. *Type of service* is used to indicate the quality of the service desired. This is an abstract or generalized set of parameters to be used to determine the selection of the actual service parameters employed when transmitting a datagram through a particular network. This type-of-service indication is used by routers to select the actual transmission parameters for a particular network, the network for the next hop, or the next router when routing an Internet datagram.

2. *Time to live* indicates an upper boundary of an IP datagram lifetime. It is set by the datagram sender and reduced at points along the route where it is processed. If the time to live reaches zero before the Internet datagram reaches its destination, the Internet datagram is destroyed.

3. *Options* provide for control functions needed or useful in some situations but unnecessary for the most common communications. This makes timestamps, security, and special routing possible.

4. *Header checksum* provides a verification that the information used in processing an Internet datagram has been transmitted correctly. The data may contain errors. If the header checksum fails, the Internet datagram is discarded at once by the entity which detects the error. IP does not provide a reliable communication facility. There are no acknowledgments either end to end or hop by hop; nor is there any error control for data, only a header checksum. IP does not perform any retransmissions and has no flow control. Any errors detected are reported via the *Internet Control Message Protocol* (ICMP), a required component that accompanies IP.

9.2 IP Operation

Transmitting a datagram from one application program to another is illustrated by the following example.

Assume that an intermediate router is operating in the environment depicted in this example. A sending application program prepares its data and calls on its IP module to send the data as a datagram. It passes the destination address and other parameters as arguments to this call. The IP module prepares a datagram header and attaches the data to it. The IP module determines a local network address for this Internet address, in this case it is the address of a router. The IP module sends this datagram and the local network address to the network interface. The network interface, in turn, creates a local network header, and attaches the datagram to it. It then sends the result via the local network.

The datagram arrives at a router wrapped in the local network header, the local network interface strips off this header, and turns the datagram over to the Internet module. The Internet module determines from the Internet address that the datagram is to be forwarded to another host in a second network. The Internet module determines a local net address for the destination host. It calls on the local network interface for that network to send the datagram.

This local network interface creates a local network header and attaches the datagram sending the result to the destination host. Here the destination host datagram is stripped of the local net header by the local network Interface and handed to the Internet module. The Internet module determines that the datagram is intended for an application program in this host; it then passes the data to the application program in response to a system call, passing the source address and other parameters as a result of the call (see Fig. 9-1).

The purpose of IP is to move datagrams through a set of networks. Datagrams are passed from one Internet module to another until the destination is reached. The Internet modules reside in hosts and routers in the Internet. Datagrams are routed from one Internet module to another through individual networks according to the interpretation of an Internet address. Thus, one important mechanism of the Internet Protocol is the Internet address.

Because datagrams may have to traverse multiple internal networks within the Internet, the routing of messages from one Internet module to another may be achieved by *fragmentation,* which is necessary to traverse a network whose maximum packet size is smaller than the datagram.

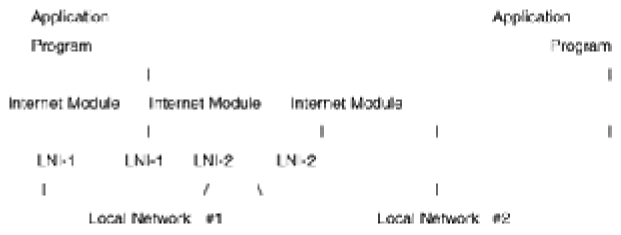


Figure 9-1
Using the Internet module to send a datagram.

9.2.1 Fragmentation

Fragmentation of an Internet datagram is necessary when the datagram originates in a local net that allows a large packet size but, to reach its destination, must traverse a local net that accepts smaller packets only.

Any Internet datagram marked “don’t fragment” cannot be fragmented under any circumstances and will be discarded if it cannot be delivered to its destination without fragmenting.

Fragmentation, transmission, and reassembly across a local network which is invisible to the IP module is called *intranet fragmentation.* The combined fragmentation-reassembly procedure must be able to break a datagram into an almost arbitrary number of pieces that can be reassembled later. The receiver of the fragments uses the identification field to ensure that fragments of different datagrams are not mixed. The fragment offset field informs the receiver of the position of a fragment in the original datagram. The fragment offset and length determine the portion of the original datagram covered by this fragment. The no-more-fragments flag indicates (by being reset) the last fragment. These fields provide sufficient information to reassemble datagrams.

The identification field is used to distinguish the fragments of one datagram from those of another. The originating protocol module of an Internet datagram sets the identification field to a value that must be unique for that source-destination pair and protocol for the time the datagram will be active in the Internet system. The originating protocol module of a complete datagram sets the more-fragments flag to zero and the fragment offset to zero.

To fragment a long Internet datagram, an IP module creates two new Internet datagrams and copies the contents of the Internet header fields from the long datagram into both new Internet headers. The data of the long datagram are divided into two portions on an 8-octet (64-bit) boundary (the second portion might not be an integral multiple of 8 octets, but the first must be). Call the number of 8-octet blocks in the first portion *number of fragment blocks* (NFB). The first portion of the data is placed in the first new Internet datagram, and the total length field is set to the length of the first datagram. The more-fragments flag is set to one. The second portion of the data is placed in the second new datagram, and the total length field is set to the length of the second datagram. The more-fragments flag carries the same value as the long datagram. The fragment offset field of the second new datagram is set to the value of that field in the long datagram plus NFB.

This procedure can be generalized for an *n*-way split, rather than the two-way split described. To assemble the fragments of an Internet datagram, an IP module (e.g., at a destination host) combines Internet datagrams that all have the same value for four fields: identification, source, destination, and protocol.

The combination is done by placing the data portion of each fragment in the relative position indicated by the fragment offset in that fragment’s Internet header. The first fragment will have the fragment offset zero, and the last fragment will have the more-fragments flag reset to zero.

9.2.2 Addressing

Names, addresses, and route are distinguished: a *name* indicates what we seek, an *address* indicates where it is, and a *route* indicates how to get there. IP deals primarily with addresses. It is the task of higher-level protocols to map from names to addresses. The Internet module maps Internet addresses to local net addresses. It is the task of lower-level (i.e., local net or gateways) procedures to map from local net addresses to routes.

Addresses are fixed length of 4 octets (32 bits). An address begins with a network number, followed by local address (called the *rest* field). There are three formats or classes of Internet addresses:

Class A	The high-order bit is zero, the next 7 bits are the network, and the last 24 bits are the local address.
Class B	The high-order 2 bits are one-zero, the next 14 bits are the network address, and the last 16 bits are the local address.
Class C	The high-order bits are one-one-zero, the next 21 bits are the network address, and the last 8 bits are the local address.

Care must be taken in mapping Internet addresses to local net addresses; a single physical host must be able to act as if it were several distinct hosts to the extent of using several distinct Internet addresses. Some hosts will also have several physical interfaces; this is also called a *multihomed host.*

A host must be provided with several physical interfaces to the network, each having several logical Internet addresses.

9.3 IP Terminology

IP-related terms have specific meanings. Sometimes these terms are misunderstood; for that reason I am including the following for your reference.

ARPAnet leader The control information on an ARPAnet message at the host-IMP interface.

ARPAnet message The unit of transmission between a host and an IMP in the ARPAnet. The maximum size is about 1012 octets (8096 bits).

ARPAnet packet A unit of transmission used internally in the ARPAnet between IMPs. The maximum size is about 126 octets (1008 bits).

Destination The destination address, an Internet header field.

DF The don’t-fragment bit carried in the flags field.

flags An Internet header field carrying various control flags.

fragment offset An Internet header field indicating where in the Internet datagram a fragment belongs.

GGP Gateway-to-Gateway Protocol, used primarily between gateways to control routing and other gateway functions.

Header Control information at the beginning of a message, segment, datagram, packet, or block of data.

ICMP Internet Control Message Protocol, implemented in the Internet module, and used from gateways to hosts and between hosts to report errors and make routing suggestions.

Identification An Internet header field carrying the identifying value assigned by the sender to aid in assembling the fragments of a datagram.

IHL The Internet header field length; length of the Internet header measured in 32-bit words.

IMP The Interface Message Processor, the packet switch of the ARPAnet.

Internet Address A 4-octet (32-bit) source or destination address consisting of a network field and a local address field.

Internet datagram The unit of data exchanged between a pair of Internet modules (includes the Internet header).

Internet fragment A portion of the data of an Internet datagram with an Internet header.

local address The address of a host within a network. The actual mapping of an Internet local address on to the host addresses in a network is quite general, allowing for many to one mappings.

MF The more-fragments flag carried in the Internet header flags field.

module An implementation, usually in software, of a protocol or other procedure.

more-fragments flag A flag indicating whether the Internet datagram in question contains the end of an Internet datagram, carried in the Internet header flags field.

NFB The number of fragment blocks in the data portion of an Internet fragment; the length of a portion of data measured in 8-octet units.

octet An 8-bit byte.

options An Internet header field that may contain several options, and each option may be several octets in length.

padding An Internet header padding used to ensure that the data begin on 32-bit word boundary. The padding is zero.

protocol In this document, the next-higher-level protocol identifier, an Internet header field.

rest The local-address portion of an Internet address.

source The source address, an Internet header field.

TCP Transmission Control Protocol; a host-to-host protocol for reliable communication in Internet environments.

TCP segment The unit of data exchanged between TCP modules (including the TCP header).

TFTP Trivial File Transfer Protocol; a simple file transfer protocol built on UDP.

time to live (TTL) An Internet header field which indicates the upper bound on how long this Internet datagram may exist.

total length An Internet header field indicating the length of the datagram in octets, including Internet header and data.

Type of service (TOS) An Internet header field indicating the type (or quality) of service for this Internet datagram.

UDP User Datagram Protocol; a user-level protocol for transaction-oriented applications.

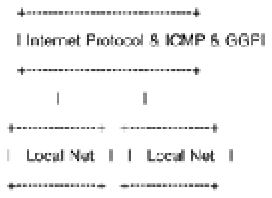
user The user of the Internet protocol. This may be a higher-level protocol module, an application program, or a gateway program.

version A field indicating the format of the Internet header.

9.4 Routers and IP

Routers (previously termed *gateways*) implement Internet protocols to forward datagrams between networks and implement the Gateway-to-Gateway Protocol (GGP) to coordinate routing and other Internet control information.

In a router the higher-level protocols need not be implemented and the GGP functions are added to the IP module. See Fig. 9-2.



```
+------------------------------+
|Internet Protocol & ICMP & GGP|
+------------------------------+
       |              |
+---------------+  +---------------+
|  Local Net  |  |  Local Net  |
+---------------+  +---------------+
```

Figure 9-2
GGP functions added to the IP module.

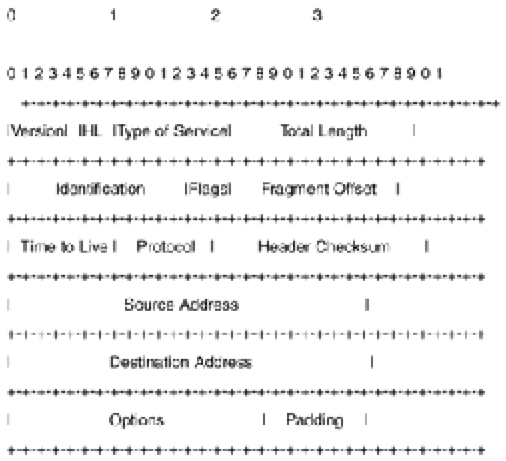
9.5 IP Header Format

The contents of the Internet header are shown in Fig. 9-3. The specifics of these fields are as follows:

1. *Version:* 4 bits. This field indicates the format of the Internet header. This document describes version 4.

2. *IHL:* 4 bits. The length of the Internet header is measured in 32-bit words, and thus points to the beginning of the data. Note that the minimum value for a correct header is 5.

3. *Type of service:* 8 bits. This feature provides an indication of the abstract parameters of the quality of service desired. These parameters are to be used to guide the selection of the actual service parameters used when transmitting a datagram through a particular network. Several networks offer service precedence, which somehow treats high-precedence traffic as more important than other traffic (generally by accepting only traffic above a certain precedence at time of high load). The major choice is a three-way tradeoff between low delay, high reliability, and high throughput. Bit priority (Fig. 9-4) is as follows: bits 0 to 2—precedence; bit 3—0 = normal delay, 1 = low delay; bit 4—0 = normal throughput, 1 = high throughput; bit 5—0 = normal reliability, 1 = high reliability; bits 6 and 7—reserved for future use.



```
0               1               2               3

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version|  IHL  |Type of Service|          Total Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Identification        |Flags|      Fragment Offset    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time to Live |    Protocol   |         Header Checksum       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source Address                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination Address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 9-3
The Internet header.

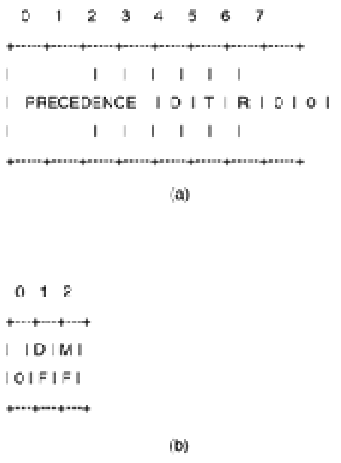


Figure 9-4
Bit priority.

Precedence is broken down as follows: 111—network control; 110—internetwork control; 101—CRITIC/ECP; 100—flash override; 011—flash; 010—immediate; 001—priority; 000—routine. Delay, throughput, and reliability indications may increase the cost (in some sense) of the service. In many networks better performance for one of these parameters is coupled with worse performance on another. Except for very unusual cases, only two of these three indications should be set. The type of service is used to specify the treatment of the datagram during its transmission through the Internet system. The *network control* precedence designation is intended for use within a network only. The actual use and control of that designation is up to each network. The *internetwork control* designation is intended for use by gateway control originators only. If the actual use of these precedence designations is of concern to a particular network, that network must control the access to, and use of, those precedence designations.

4. *Total length:* 16 bits. This is the length of the datagram, measured in octets, including Internet header and data. This field allows the length of a datagram to be up to 65,535 octets. Such long datagrams are impractical for most hosts and networks. All hosts must be prepared to accept datagrams of up to 576 octets (whether they arrive whole or in fragments). It is recommended that hosts send datagrams larger than 576 octets only if they have assurance that the destination is prepared to accept the larger datagrams. The number 576 is selected to allow a reasonable-sized data block to be transmitted in addition to the required header information. For example, this size allows a data block of 512 octets plus 64 header octets to fit in a datagram. The maximum Internet header is 60 octets, and a typical Internet header is 20 octets, allowing a margin for headers of higher-level protocols.

5. *Identification:* 16 bits. An identifying value assigned by the sender to aid in assembling the fragments of a datagram.

6. *Flags:* 3 bits. Various control flags. Breakdown is as follows: bit 0—reserved, must be zero; bit 1—(DF) 0 = may fragment, 1 = don't fragment; bit 2—(MF) 0 = last fragment, 1 = more fragments.

0 1 2

+—+—+—+

| | D | M |

```
| 0 |  F  |  F |

+—+—+—+
```

7. *Fragment offset:* 13 bits. This field indicates where in the datagram this fragment belongs. The fragment offset is measured in units of 8 octets (64 bits). The first fragment has offset zero.

8. *Time to live:* 8 bits. This field indicates the maximum time the datagram is allowed to remain in the Internet system. If this field contains the value zero, then the datagram must be destroyed. This field is modified in Internet header processing. The time is measured in units of seconds, but since every module that processes a datagram must decrease the TTL by at least one even if it processes the datagram in less than a second, the TTL must be regarded only as an upper bound on the time a datagram may exist. The intention is to cause undeliverable datagrams to be discarded, and to bound the maximum datagram lifetime.

9. *Protocol:* 8 bits. This field indicates the next-level protocol used in the data portion of the Internet datagram.

10. *Header checksum:* 16 bits. A checksum on the header only. Since some header fields change (e.g., time to live), this is recomputed and verified at each point that the Internet header is processed. The checksum field is the 16-bit one's complement of the one's complement sum of all 16-bit words in the header. For purposes of computing the checksum, the value of the checksum field is zero. This checksum is simple to compute, and experimental evidence indicates it to be adequate, but it is provisional and may be replaced by a CRC procedure, depending on further experience.

11. *Source address:* 32 bits.

12. *Destination address:* 32 bits.

13. *Options:* variable. The options may or may not appear in datagrams. They must be implemented by all IP modules (host and routers). What is optional is their transmission in any particular datagram, not their implementation. In some environments the security option may be required in all datagrams. The option field is variable in length, containing zero or more options. There are two cases for the format of an option: (1) a single octet of option type and (2) an option-type octet, an option-length octet, and the actual option-data octets. The option-length octet counts the option-type octet and the option-length octet as well as the option-data octets. The option-type octet is viewed as having three fields: 1 bit, copied flag; 2 bits, option class; 5 bits, option number. The copied flag indicates that this option is copied into all fragments on fragmentation: 0—not copied; 1—copied. Option classes are 0—control; 1—reserved for future use; 2—debugging and measurement; 3—reserved for future use. Internet options are defined as shown in Table 9-1 and Fig. 9-5.

This option indicates the end of the option list. This might not coincide with the end of the Internet header according to the Internet header length. This is used at the end of all options, not the end of each option, and need be used only if the end of the options would not otherwise coincide with the end of the Internet header. This option may be copied, introduced, or deleted on fragmentation, or for any other reason.

The *no-operation option* (see Fig. 9-6) may be used between options, for example, to align the beginning of a subsequent option on a 32-bit boundary. It may be copied, introduced, or deleted on fragmentation, or for any other reason.

The *security* option provides a way for hosts to send security, compartmentation, handling restrictions, and TCC (closed user group) parameters. The format for the security option is shown in Fig. 9-7.

Table 9-1 Internet Options

Class	**Number**	**Length**	**Description**
0	0	—	*End of option list.* This option occupies only 1 octet; it has no length octet.
0	1	—	*No operation.* This option occupies only 1 octet; it has no length octet.
0	2	11	*Security.* Used to carry security, compartmentation, user group (TCC), and handling restriction codes compatible with DoD requirements.
0	3	Variable	*Loose source routing.* Used to route the Internet datagram per information supplied by the source.
0	9	Variable	*Strict source routing.* Used to route the Internet datagram per information supplied by the source.
0	7	Variable	*Record route.* Used to trace the route taken by an Internet datagram.
0	8	4	*Stream ID.* Used to carry the stream identifier.
2	4	Variable	*Internet timestamp.*

Specific option definitions
End of option list

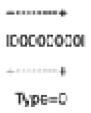


Figure 9-5
Internet options.

|00000001|
Type=1

Figure 9-6
The no-operation option.

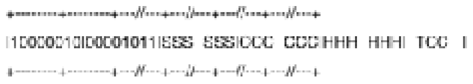


Figure 9-7
The security option.

1. *Security (S field)*: 16 bits. Specifies one of 16 levels of security (eight of which are reserved for future use):

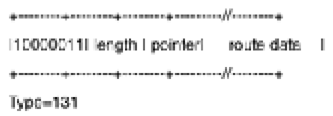
00000000 00000000	Unclassified
11110001 00110101	Confidential
01111000 10011010	EFTO
10111100 01001101	MMMM
01011110 00100110	PROG
10101111 00010011	Restricted
11010111 10001000	Secret
01101011 11000101	Top secret
00110101 11100010	(Reserved for future use)
10011010 11110001	(Reserved for future use)
01001101 01111000	(Reserved for future use)
00100100 10111101	(Reserved for future use)
00010011 01011110	(Reserved for future use)
10001001 10101111	(Reserved for future use)
11000100 11010110	(Reserved for future use)
11100010 01101011	(Reserved for future use)

2. *Compartments (C field)*: 16 bits. An all-zero value is used when the information transmitted is not compartmented. Other values for the compartments field may be obtained from the Defense Intelligence Agency (DIA).

3. *Handling restrictions (H field)*: 16 bits. The values for the control and release markings are alphanumeric digraphs and are defined in the DIA manual DIAM 65-19, *Standard Security Markings.*

4. *Transmission control code (TCC field)*: 24 bits. Provides a means to segregate traffic and define controlled communities of interest among subscribers. The TCC values are trigraphs, and are available from HQ DCA Code 530. Must be copied on fragmentation. This option appears at most once in a datagram.

The *loose source and record route* (LSRR) option (see Fig. 9-8) provides a means for the source of an Internet datagram to supply routing information to be used by the gateways in forwarding the datagram to the destination, and to record the route information. The option begins with the option type code. The second octet is the option length which includes the option type code and the length octet, the pointer octet, and length-3 octets of route data. The third octet is the pointer into the route data indicating the octet which begins the next source address to be processed. The pointer is relative to this option, and the smallest legal value for the pointer is 4. A route datum is composed of a series of Internet addresses. Each Internet address is 32 bits or 4 octets. If the pointer is greater than the length, the source route is empty (and the recorded route full) and the routing is to be based on the destination address field.



```
+--------+--------+--------+---------//--------+
|10000011| length | pointer|     route data    |
+--------+--------+--------+---------//--------+
Type=131
```

Figure 9-8
The loose source and record route option.

If the address in the destination address field has been reached and the pointer is not greater than the length, the next address in the source route replaces the address in the destination address field, and the recorded route address replaces the source address just used, and the pointer is increased by 4. The recorded route address is the Internet module’s own Internet address as known in the environment into which this datagram is being forwarded. This procedure of replacing the source route with the recorded route (although it is in reverse order of the order it must be in to be used as a source route) means that the option (and the IP header as a whole) remains a constant length as the datagram progresses through the Internet.

This option is a loose source route because the gateway or host IP is allowed to use any route of any number of other intermediate gateways to reach the next address in the route. *Must be copied on fragmentation* appears at most once in a datagram.

The *strict source and record route* (SSRR) option (see Fig. 9-9) provides a means for the source of an Internet datagram to supply routing information to be used by the gateways in forwarding the datagram to the destination, and to record the route information.

The option begins with the option type code. The second octet is the option length which includes the option type code and the length octet, the pointer octet, and length-3 octets of route data. The third octet is the pointer into the route data indicating the octet which begins the next source address to be processed. The pointer is relative to this option, and the smallest legal value for the pointer is 4.

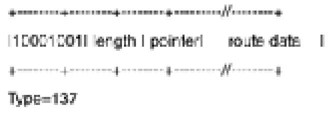


Figure 9-9
The strict source and record route option.

A route datum is composed of a series of Internet addresses, each Internet address is 32 bits or 4 octets. If the pointer is greater than the length, the source route is empty (and the recorded route full) and the routing is to be based on the destination address field.

If the address in destination address field has been reached and the pointer is not greater than the length, the next address in the source route replaces the address in the destination address field, and the recorded route address replaces the source address just used, and the pointer is increased by 4. The recorded route address is the Internet module’s own Internet address as known in the environment into which this datagram is being forwarded. This procedure of replacing the source route with the recorded route (although it is in reverse order of the order it must be in to be used as a source route) means the option (and the IP header as a whole) remains a constant length as the datagram progresses through the Internet.

This option is a strict source route because the gateway or host IP must send the datagram directly to the next address in the source route through only the directly connected network indicated in the next address to reach the next gateway or host specified in the route. *Must be copied on fragmentation* appears at most once in a datagram.

The record route option (Fig. 9-10) provides a means to record the route of an Internet datagram. The option begins with the option-type code. The second octet is the option length which includes the option type code and the length octet, the pointer octet, and length-3 octets of route data. The third octet is the pointer into the route data indicating the octet which begins the next area to store a route address. The pointer is relative to this option, and the smallest legal value for the pointer is 4.

A *recorded route* is composed of a series of Internet addresses. Each Internet address is 32 bits or 4 octets in length. The pointer in the orig inating host must compose this option with a route data area large enough to hold all the addresses expected. The size of the option does not change when addresses are added. The initial contents of the route data area must be zero.

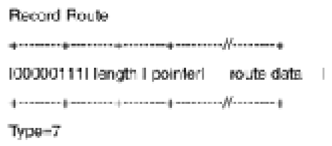


Figure 9-10
The record route option.

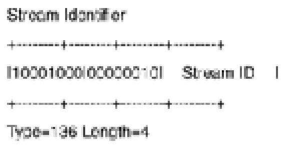


Figure 9-11
The stream identifier option.

When an Internet module routes a datagram, it checks to see if the record route option is present. If it is, it inserts its own Internet address as known in the environment into which this datagram is being forwarded into the recorded route beginning at the octet indicated by the pointer, and increments the pointer by 4.

If the route data area is already full (the pointer exceeds the length), the datagram is forwarded without inserting the address into the recorded route. If there is not enough room for a full address to be inserted, the original datagram is considered to be in error and is discarded. In either case an ICMP parameter problem message may be sent to the source host. *Not copied on fragmentation, goes in first fragment only* appears at most once in a datagram.

The *stream identifier option* (Fig. 9-11) provides a way for the 16-bit SATNET stream identifier to be carried through networks that do not support the stream concept. *Must be copied on fragmentation* appears at most once in a datagram.

9.6 Internet Timestamp

The Internet timestamp, an integral tool in the implementation and use of IP, appears as shown in Fig. 9-12.

The *option length* is the number of octets in the option counting the type, length, pointer, and overflow/flag octets (maximum length 40). The *pointer* is the number of octets from the beginning of this option to the end of timestamps plus one (i.e., it points to the octet beginning the space for the next timestamp). The smallest legal value is 5. The timestamp area is full when the pointer is greater than the length. The *overflow* (oflw) (4 bits) is the number of IP modules that cannot register timestamps because of lack of space. The *flag* (flg) (4 bits) values are

0	Timestamps only, stored in consecutive 32-bit words.
1	Each timestamp is preceded with the Internet address of the registering entity.
3	The Internet address fields are prespecified.

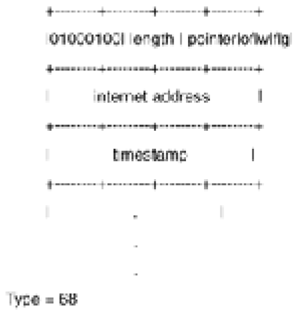


Figure 9-12
The Internet timestamp.

An IP module registers its timestamp only if it matches its own address with the next specified Internet address. The timestamp is a right-justified, 32-bit timestamp in milliseconds since midnight UT (universal time). If the time is not available in milliseconds or cannot be provided with respect to midnight UT, then any time may be inserted as a timestamp provided the high-order bit of the timestamp field is set to one to indicate the use of a nonstandard value.

The originating host must compose this option with a sufficiently large timestamp data area to hold all the timestamp information expected. The size of the option does not change with the addition of timestamps. The initial contents of the timestamp data area must be zero or Internet address/zero pairs.

If the timestamp data area is already full (the pointer exceeds the length), the datagram is forwarded without inserting the timestamp, but the overflow count is incremented by one. If there is some room but not enough for a full timestamp to be inserted, or if the overflow count itself overflows, the original datagram is considered to be in error and is discarded. In either case an ICMP parameter problem message may be sent to the source host.

The timestamp option is not copied on fragmentation. It is carried in the first fragment. It appears at most once in a datagram.

The Internet header padding is variable and is used to ensure that the Internet header ends on a 32-bit boundary. The padding is zero.

Fragmentation and Reassembly

The Internet *identification field* (ID) is used together with the source and destination address, and the protocol fields, to identify datagram fragments for reassembly. The *more-fragments flag bit* (MF) is set if the datagram is not the last fragment. The *fragment-offset field* identifies the fragment location relative to the beginning of the original unfragmented datagram. Fragments are counted in units of 8 octets. The fragmentation strategy is designed so that an unfragmented datagram has all-zero fragmentation information (MF = 0, fragment offset = 0). If an Internet datagram is fragmented, its data portion must be broken on 8 octet boundaries.

This format allows 2**13 = 8192 fragments of 8 octets each for a total of 65,536 octets. Note that this is consistent with the datagram total length field (of course, the header is counted in the total length and not in the fragments).

When fragmentation occurs, some options are copied but others remain with the first fragment only. Every Internet module must be able to forward a datagram of 68 octets without further fragmentation. This is because an Internet header may be up to 60 octets, and the minimum fragment is 8 octets. Every Internet destination must be able to receive a datagram of 576 octets either in one piece or in fragments to be reassembled.

The fields which may be affected by fragmentation include the (1) options field, (2) more-fragments flag, (3) fragment offset, (4) Internet header-length field, (5) total-length field, and (6) header checksum.

If the *don't-fragment flag* (DF) bit is set, then Internet fragmentation of this datagram is *not* permitted, although it may be discarded. This can be used to prohibit fragmentation in cases where the receiving host does not have sufficient resources to reassemble Internet fragments. One example of use of the don't-fragment feature is to download a small host. A small host could have a bootstrap program that accepts a datagram, stores it in memory, and then executes it.

The fragmentation and reassembly procedures are most easily described by examples. The following procedures are example implementations.

General notation in the following pseudo programs is as follows: "=<" means *less than or equal,* "#" means *not equal,* "=" means *equal,* "<-" means *is set to.* Also, "*x* to *y*" includes *x* and excludes *y*; for example, "4 to 7" would include 4, 5, and 6 (but not 7).

Fragmentation procedure example. The largest datagram that can be transmitted through the next network is called the *maximum transmission unit* (MTU).

If the total length is less than or equal to the maximum transmission unit, then submit this datagram to the next step in datagram processing; otherwise cut the datagram into two fragments: (1) the largest fragment and (2) the rest of the datagram. The first fragment is submitted to the next step in datagram processing, while the second fragment is submitted to this procedure in case it is still too large.

1. *Notation.* Notation is as follows: FO—fragment offset; IHL—Internet header length; DF—don't-fragment flag; MF—more-fragments flag; TL—total length; OFO—old fragment offset; OIHL—old Internet header length; OMF—old more-fragments flag; OTL—old total length; NFB—number of fragment blocks; MTU—maximum transmission unit.

2. *Procedure.* IF TL = <MTU THEN submit this datagram to the next step in datagram processing ELSE IF DF = 1 THEN discard the datagram ELSE). To produce the first fragment: (*a*) copy the original Internet header; (*b*) OIHL <- IHL; OTL <- TL; OFO <- FO; OMF <- MF; (*c*) NFB <- (MTU-IHL*4)/8; (*d*) attach the first NFB*8 data octets; (*e*) correct the header, MF <- 1; TL <- (IHL*4)+(NFB*8); (Recomputed checksum); (*f*) submit this fragment to the next step in datagram processing; then, to produce the second fragment (*g*) selectively copy the Internet header (some options are not copied, see option definitions); (*h*) append the remaining data; (*i*) correct the header, IHL <- (((OIHL*4)-(length of options not copied))+3)/4; TL <- OTL - NFB*8 - (OIHL-IHL)*4); FO <- OFO + NFB; MF <- OMF (recomputed checksum); (*j*) submit this fragment to the fragmentation test; DONE. In this procedure each fragment (except the last) was made the maximum allowable size. An alternative might produce less than the maximum-size datagrams. For example, one could implement a fragmentation procedure that repeatedly divided large datagrams in half until the resulting fragments were less than the maximum transmission unit size.

Reassembly procedure example. For each datagram the buffer identifier is computed as the concatenation of the source, destination, protocol, and identification fields. If this is a whole datagram (i.e, both the fragment-offset and the more-fragments fields are zero), then any reassembly resources associated with this buffer identifier are released and the datagram is forwarded to the next step in datagram processing.

If no other fragment with this buffer identifier is on hand, then reassembly resources are allocated. The reassembly resources consist of a data buffer, a header buffer, a fragment block bit table, a total-data-length field, and a timer. The data from the fragment are placed in the data buffer according to its fragment offset and length, and bits are set in the fragment block bit table corresponding to the fragment blocks received.

If this is the first fragment (i.e., the fragment offset is zero), this header is placed in the header buffer. If this is the last fragment (i.e., the more-fragments field is zero), the total data length is computed. If this fragment completes the datagram (tested by checking the bits set in the fragment block table), then the datagram is sent to the next step in datagram processing; otherwise the timer is set to the maximum of the current timer value and the value of the time-to-live field from this fragment; and the reassembly routine relinquishes control.

If the timer runs out, then all reassembly resources for this buffer identifier are released. The initial setting of the timer is a lower bound on the reassembly waiting time. This is because the waiting time will be increased if the time-to-live value in the arriving fragment is greater than the current timer value but will not be decreased if it is less. The maximum this timer value could reach is the maximum time to live (approximately 4.25 min). The current recommendation for the initial timer setting is 15 s. This may be changed as experience with this protocol accumulates. Note that the choice of this parameter value is related to the buffer capacity available and the data rate of the transmission medium; that is, data rate times timer value equals buffer size (e.g., 10 kbits/s × 15 s = 150 kbits).

1. *Notation reference:* FO—fragment offset; IHL—Internet header length; MF—more-fragments flag; TTL—time to live; NFB—number of fragment blocks; TL—total length; TDL—total data length; BUFID—buffer identifier; RCVBT—fragment-received bit table; TLB—timer lower bound.

2. *Procedure: (a*) BUFID ,- source | destination | protocol | identification; (*b*) IF FO 5 0 AND MF 5 0; (*c*) THEN IF buffer with BUFID is allocated; (*d*) THEN flush all reassembly for this BUFID; (*e*) submit datagram to next step; DONE; (*f*) ELSE IF no buffer with BUFID is allocated; (*g*) THEN allocate reassembly resources with BUFID; TIMER ,- TLB; TDL ,- 0; (*h*) put data from fragment into data buffer with BUFID from octet FO*8 to octet (TL-(IHL*4))+FO*8; (*i*) set RCVBT bits from FO to FO+((TL-(IHL*4)+7)/8); (*j*) IF MF 5 0 THEN TDL ,- TL-(IHL*4)+(FO*8); (*k*) IF FO 5 0 THEN put header in header buffer; (*l*) IF TDL # 0; (*m*) AND all RCVBT bits from 0 to (TDL 1 7)/8 are set; (*n*) THEN TL ,- TDL+(IHL*4); (*o*) submit datagram to next step; (*p*) free all reassembly resources for this BUFID; DONE; (*q*) TIMER ,- MAX(TIMER,TTL); (*r*) give up until next fragment or timer expires; (*s*) timer expires: flush all reassembly with this BUFID; DONE. If two or more fragments contain the same data either identically or through a partial overlap, this procedure will use the more recently arrived copy in the data buffer and datagram delivered.

Identification

The choice of the Identifier for a datagram is based on the need to provide a way to uniquely identify the fragments of a par ticular datagram. The protocol module assembling fragments judges fragments to belong to the same datagram if they have the same source, destination, protocol, and Identifier. Thus, the sender must choose the Identifier to be unique for this source, destination pair, and protocol for the time the datagram (or any fragment of it) could be alive in the Internet.

It seems, then, that a sending protocol module needs to keep a table of Identifiers, one entry for each destination with which it has communicated in the last maximum packet lifetime for the Internet. However, since the Identifier field allows 65,536 different values, some host may be able to simply use unique identifiers independent of destination.

It is appropriate for some higher-level protocols to choose the identifier. For example, TCP protocol modules may retransmit an identical TCP segment, and the probability for correct reception would be enhanced if the retransmission carried the same identifier as the original transmission since fragments of either datagram could be used to construct a correct TCP segment.

Type of Service

The type of service (TOS) is for Internet service quality selection. The type of service is specified along the abstract parameters of precedence, delay, throughput, and reliability. These abstract parameters are to be mapped into the actual service parameters of the particular networks the datagram traverses.

Precedence—this is an independent measure of the importance of this datagram.

Delay—prompt delivery is important for datagrams with this indication.

Throughput—high data rate is important for datagrams with this indication.

Reliability—a higher level of effort to ensure delivery is important for datagrams with this indication.

Time to Live

The time to live is set by the sender to the maximum time the datagram is allowed to be in the Internet system. If the datagram is in the Internet system longer than the time to live, then the datagram must be destroyed.

This field must be decreased at each point that the Internet header is processed to reflect the time spent processing the datagram. Even if no local information is available on the time actually spent, the field must be decremented by 1. The time is measured in units of seconds (i.e., the value 1 means 1 s). Thus, the maximum time to live is 255 s or 4.25 min. Since every module that processes a datagram must decrease the TTL by at least one even if it processes the datagram in less than a second, the TTL must be regarded only as an upper bound on the time a datagram may exist. The intention is to cause undeliverable datagrams to be discarded, and to bound the maximum datagram lifetime.

Some higher-level reliable connection protocols are based on assumptions that old duplicate datagrams will not arrive after a certain time elapses. The TTL is a way for such protocols to have an assurance that their assumption is met.

Options

These are optional in each datagram, but required in implementations. In other words, the presence or absence of an option is the choice of the sender, but each Internet module must be able to parse every option. Several options can be present in the option field.

The options might not end on a 32-bit boundary. The Internet header must be filled out with octets of zeros. The first of these would be interpreted as the end-of-options option; the remainder, as Internet header padding.

Every Internet module must be able to act on every option. The security option is required if classified or restricted, or if compartmented traffic is to be passed.

Checksum

The Internet header *checksum* is recomputed if the Internet header is changed, as a result of, for example, a reduction in the time to live, additions or changes to Internet options, or fragmentation. This checksum at the Internet level is intended to protect the Internet header fields from transmission errors.

In certain applications a few data bit errors are acceptable while retransmission delays are not. If the Internet protocol enforced data correctness, such applications could not be supported.

Errors

Internet protocol errors may be reported via the ICMP messages.

9.7 Interfaces and IPv4

The functional description of user interfaces to the IP is, at best, fictional, since every operating system will have different facilities. Consequently, we must warn readers that different IP implementations may have different user interfaces. However, all IPs must provide a certain minimum set of services to guarantee that all IP implementations can support the same protocol hierarchy. This section specifies the functional interfaces required of all IP implementations.

Internet Protocol interfaces on one side to the local network and on the other side to either a higher-level protocol or an application program. In the following, the higher-level protocol or application program (or even a gateway program) will be called the “user” since it is using the Internet module. Since IP is a datagram protocol, there is minimal memory or state maintained between datagram transmissions, and each call on the Internet Protocol module by the user supplies all information necessary for the IP to perform the service requested.

The following two upper-level example calls satisfy the requirements for the user for IP module communication (“ = >” means returns):

SEND: `src, dst, prot, TOS, TTL, BufPTR, len, Id, DF, opt = >` result, where `src =` source address, `dst =` destination address, `prot =` protocol, `TOS =` type of service, `TTL =` time to live, `BufPTR =` uffer pointer, `len =` length of buffer, `Id =` identifier, `DF =` don’t fragment, `opt =` option data, `result =` response, `OK =` datagram sent ok, and `Error =` error in arguments or local network error. Note that the precedence is included in the TOS and the security/compartment is passed as an option.

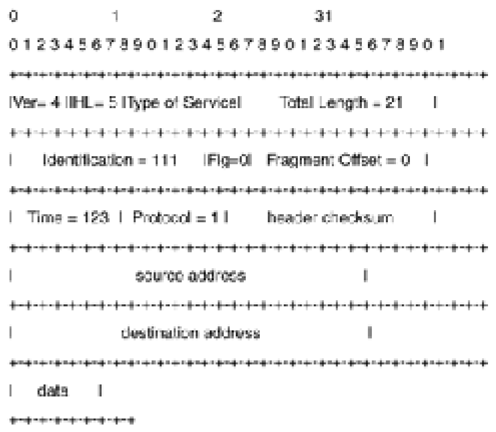
RECV: `BufPTR, prot, = >` result, src, dst, TOS, len, opt, where `BufPTR =` buffer pointer, `prot =` protocol, `result =` response, `OK =` datagram received ok, `Error` = error in arguments, `len` length of buffer, `src` source address, `dst` = destination address, `TOS` = type of service, and `opt` = option data.

When the user sends a datagram, it executes the SEND call supplying all the arguments. The IP module, on receiving this call, checks the arguments and prepares and sends the message. If the arguments are good and the datagram is accepted by the local network, the call returns successfully. If either the arguments are bad, or the datagram is not accepted by the local network, the call returns unsuccessfully. On unsuccessful returns, a reasonable report must be made as to the cause of the problem, but the details of such reports are up to individual implementations.

When a datagram arrives at the IP module from the local network, there either is or is not a pending RECV call from the user addressed. In the first case, the pending call is satisfied by passing the information from the datagram to the user. In the second case, the user addressed is notified of a pending datagram. If the user addressed does not exist, an ICMP error message is returned to the sender, and the data are discarded.

The notification of a user may be via a pseudointerrupt or similar mechanism, as appropriate in the particular operating system environment of the implementation.

A user's `RECV` call may then either be immediately satisfied by a pending datagram, or the call may be pending until a datagram arrives. The source address is included in the SEND call in case the sending host has several addresses (multiple physical connections or logical addresses). The Internet module must check to see that the source address is one of the legal addresses for this host.



```
0                   1                   2                   31
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Ver= 4 |IHL= 5 |Type of Service|        Total Length = 21      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Identification = 111     |Flg=0|   Fragment Offset = 0   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Time = 123  |  Protocol = 1 |        header checksum        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         source address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      destination address                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     data      |
+-+-+-+-+-+-+-+-+
```

Figure 9-13
Minimal data-carrying Internet datagram.

An implementation may also allow or require a call to the Internet module to indicate interest in or reserve exclusive use of a class of datagrams (e.g., all those with a certain value in the protocol field).

This section functionally characterizes a user/IP interface. The notation used is similar to most function call procedures in high-level languages, but this usage is not meant to rule out trap-type service calls [e.g., switched virtual circuits (SVCs), UUOs, EMTs], or any other form of interprocess communication.

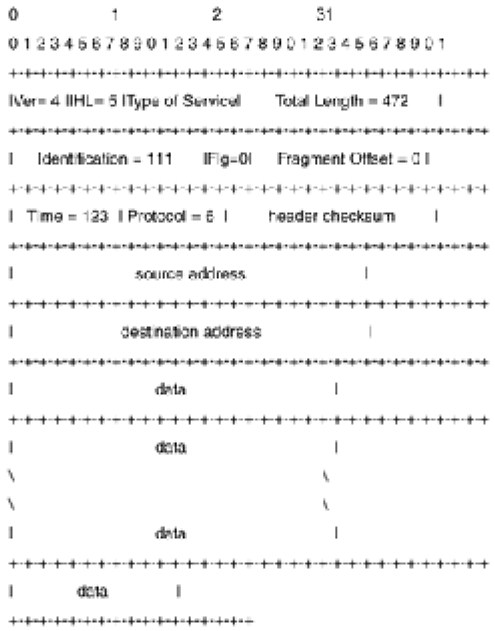
IPv4 Datagram

Figure 9-13 is an example of the minimal data-carrying Internet datagram (where each tickmark represents one bit position).

This Internet datagram reflects version 4 of IP. The IP header consists of five 32-bit words, and the total length of the datagram is 21 octets. This datagram is a complete datagram (not a fragment).

IPv4 Datagram Fragment

Figure 9-14 shows a moderate size IP datagram (452 data octets), then two Internet fragments that might result from the fragmentation of this datagram if the maximum-sized transmission allowed were 280 octets.



```
0                   1                   2                   31
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Ver= 4 |IHL= 5 |Type of Service|       Total Length = 472      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Identification = 111      |Flg=0|     Fragment Offset = 0 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Time = 123  | Protocol = 6  |        header checksum        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         source address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      destination address                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             data                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             data                              |
\                                                               \
\                                                               \
|                             data                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             data              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 9-14
Moderate-size IP diagram.

IPv4 First Datagram Fragment

Figure 9-15 is an example of the first fragment that results from splitting the datagram after 256 data octets.

IPv4 Second Datagram Fragment

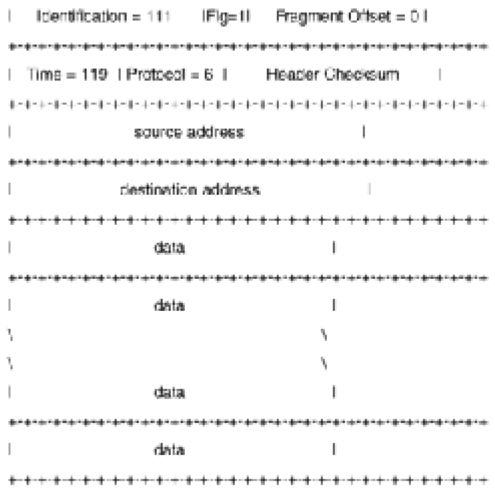
Figure 9-16 is an example of the second fragment.

IPv4 Datagram with Options

Figure 9-17 is an example of a datagram with options.

Order of IP Data Transmission

The order of transmission of the header and data described in this document is resolved to the octet level.



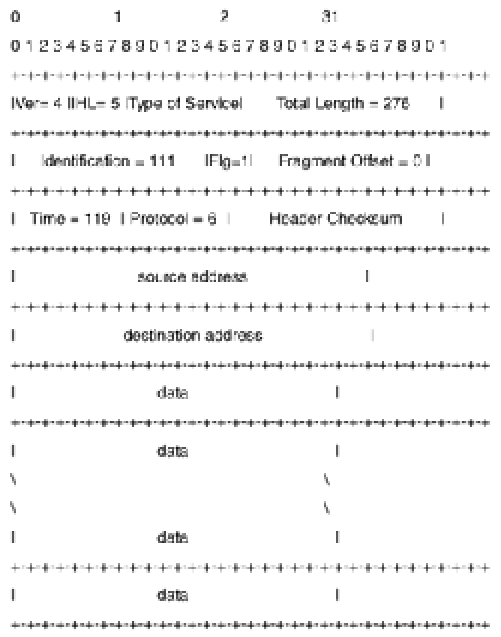
```
|     Identification = 111      |Flg=1|     Fragment Offset = 0 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Time = 119  | Protocol = 6  |        Header Checksum        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         source address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      destination address                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             data                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             data                              |
\                                                               \
\                                                               \
|                             data                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             data                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 9-15
IPv4 first datagram fragment.

Whenever a diagram shows a group of octets, the order of transmission of those octets is the normal order in which they are read in English. For example, in Fig. 9-18 the octets are transmitted in the order they are numbered.

Whenever an octet represents a numeric quantity, the leftmost bit in the diagram is the high-order or most-significant bit; that is, the bit labeled 0 is the most-significant bit. For example, Fig. 9-19 represents the value 170 (decimal).

Whenever a multioctet field represents a numeric quantity, the leftmost bit of the whole field is the most-significant bit. When a multioctet quantity is transmitted, the most-significant octet is transmitted first.



```
0                   1                   2                   31
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Ver= 4 |IHL= 5 |Type of Service|     Total Length = 276      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Identification = 111      |Flg=1|   Fragment Offset = 0 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time = 119  | Protocol = 6  |        Header Checksum       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        source address                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      destination address                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             data                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             data                            |
\                                                             \
\                                                             \
|                             data                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             data                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 9-16
IPv4 second datagram fragment.

```
0                   1                   2                   30
01 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Ver= 4 |IHL= 8 |Type of Service|       Total Length = 576       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Identification = 111    |Flg=0|     Fragment Offset = 0 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Time = 123  |  Protocol = 6 |        Header Checksum         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        source address                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      destination address                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Opt. Code = x | Opt.  Len.= 3 | option value  | Opt. Code = x |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Opt. Len. = 4 |           option value        | Opt. Code = 1 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Opt. Code = y | Opt. Len. = 3 |  option value | Opt. Code = 0 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             data                              |
\                                                               \
\                                                               \
|                             data                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             data                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 9-17
Datagram with options.

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       1       |       2       |       3       |       4       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       5       |       6       |       7       |       8       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       9       |      10       |      11       |      12       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 9-18
IP data transmission order.

```
0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+
|1 0 1 0 1 0 1 0|
+-+-+-+-+-+-+-+-+
```

Figure 9-19
The value 170 (decimal).

10
Internet Protocol Version 6

10.1 Perspective

Internet Protocol version 6 (IPv6) is a new version of the Internet Protocol, designed as a successor to IP version 4 (IPv4) (RFC 791). The changes from IPv4 to IPv6 fall primarily into the following categories:

Expanded Addressing Capabilities

IPv6 increases the IP address size from 32 to 128 bits, to support more levels of addressing hierarchy, a much greater number of addressable nodes, and simpler autoconfiguration of addresses. The scalability of multicast routing is improved by adding a “scope” field to multicast addresses. And a new type of address called an “anycast address” is defined, and is used to send a packet to any one of a group of nodes.

Header Format Simplification

Some IPv4 header fields have been dropped or rendered optional, to reduce the common-case processing cost of packet handling and to limit the bandwidth cost of the IPv6 header.

Improved Support for Extensions and Options

Changes in the way IP header options are encoded allows for more efficient forwarding, less stringent limits on the length of options, and greater flexibility for introducing new options in the future.

Flow Labeling Capability

A new capability is added to enable the labeling of packets belonging to particular traffic “flows” for which the sender requests special handling, such as nondefault quality of service or real-time service.

Authentication and Privacy Capabilities

Extensions to support authentication, data integrity, and data confidentiality are specified for IPv6. This document specifies the basic IPv6 header and the initially defined IPv6 extension headers and options. It also discusses packet size issues, the semantics of flow labels and priority, and the effects of IPv6 on upper-layer protocols. The format and semantics of IPv6 addresses are specified separately in RFC 1884. The IPv6 version of ICMP, which all IPv6 implementations are required to include, is specified in RFC 1885.

10.2 IPv6 Terminology

address In IPv6-layer identifier for an interface or a set of interfaces.

host Any node that is not a router.

interface A node’s attachment to a link.

link A communication facility or medium over which nodes can communicate at the link layer (i.e., the layer immediately below IPv6). Examples are Ethernets (simple or bridged); PPP links; X.25, frame relay, or ATM networks; and Internet (or higher)-layer *tunnels,* such as tunnels over IPv4 or IPv6 itself.

link MTU The maximum transmission unit, that is, maximum packet size in octets, that can be conveyed in one piece over a link.

neighbors Nodes attached to the same link.

node A device that implements IPv6.

packet An IPv6 header plus payload.

path MTU The minimum link MTU of all the links in a path between a source node and a destination node.

router A node that forwards IPv6 packets not explicitly addressed to itself.

upper layer A protocol layer immediately above IPv6. Examples are transport protocols such as TCP and UDP, control protocols such as ICMP, routing protocols such as OSPF, and Internet or lower-layer protocols being “tunneled” over (i.e., encapsulated in) IPv6 such as IPX, AppleTalk, or IPv6 itself.

It is possible for a device with multiple interfaces to be configured to forward non-self-destined packets arriving from some set (fewer than all) of its interfaces, and to discard non-self-destined packets arriving from its other interfaces. Such a device must obey the protocol requirements for routers when receiving packets from, and interacting with, neighbors over the former (forwarding) interfaces. It must obey the protocol requirements for hosts when receiving packets from, and interacting with, neighbors over the latter (nonforwarding) interfaces.

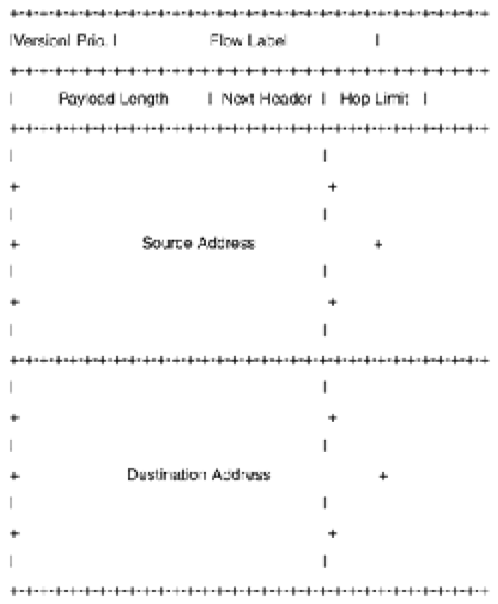


Figure 10-1
IPv6 header.

10.3 IPv6 Header Format

Figure 10-1 is a representation of the IPv6 header.

`Version`	4-bit Internet Protocol version number = 6.
`Priority`	4-bit priority value.
`Flow label`	24-bit flow label.
`Payload length`	16-bit unsigned integer. Length of payload, i.e., the rest of the packet following the IPv6 header, in octets. If zero, indicates that the payload length is carried in a Jumbo payload hop-by-hop option.
`Next header`	8-bit selector. Identifies the type of header immediately following the IPv6 header. Uses the same values as the IPv4 protocol field.
`Hop limit`	8-bit unsigned integer. Decremented by 1 by each node that forwards the packet. The packet is discarded if hop limit is decremented to zero.
`Source address`	128-bit address of the originator of the packet.
`Destination address`	128-bit address of the intended recipient of the packet (possibly not the ultimate recipient, if a routing header is present).

10.4 IPv6 Extension Headers

In IPv6, optional Internet-layer information is encoded in separate headers that may be placed between the IPv6 header and the upper-layer header in a packet. There are a small number of such extension headers, each identified by a distinct `Next Header` value. An IPv6 packet may carry zero, one, or more extension headers, each identified by the `Next Header` field of the preceding header. See Fig. 10-2.

```
+---------------+------------------------
|  IPv6 header  | TCP header + data
|               |
| Next Header = |
|      TCP      |
+---------------+------------------------
                              (a)

+---------------+----------------+------------------------
|  IPv6 header  | Routing header | TCP header + data
|               |                |
| Next Header = |  Next Header = |
|    Routing    |      TCP       |
+---------------+----------------+------------------------
                              (b)

+---------------+----------------+-----------------+-----------------
|IPv6 header    | Routing header | Fragment header | fragment of TCP
|               |                |                 |  header + data
| Next Header = |  Next Header = |  Next Header =  |
|    Routing    |    Fragment    |       TCP       |
+---------------+----------------+-----------------+-----------------
                              (c)
```

Figure 10-2
IPv6 extension header examples.

With one exception, extension headers are not examined or processed by any node along a packet's delivery path, until the packet reaches the node (or each of the set of nodes, in the case of multicast) identified in the `Destination Address` field of the IPv6 header. There, normal demultiplexing on the `Next Header` field of the IPv6 header invokes the module to process the first extension header, or the upper-layer header if no extension header is present. The contents and semantics of each extension header determine whether to proceed to the next header. Therefore, extension headers must be processed strictly in the order they appear in the packet; a receiver must not, for example, scan through a packet looking for a particular kind of extension header and process that header prior to processing all preceding ones.

The exception referred to in the preceding paragraph is the `Hop-by-Hop Options` header, which carries information that must be examined and processed by every node along a packet's delivery path, including the source and destination nodes. The `Hop-by-Hop Options` header, when present, must immediately follow the IPv6 header. Its presence is indicated by the value zero in the `Next Header` field of the IPv6 header.

If, as a result of processing a header, a node is required to proceed to the next header but the `Next Header` value in the current header is unrecognized by the node, it should discard the packet and send an ICMP Parameter Problem message to the source of the packet, with an ICMP code value of 2 ("unrecognized `Next Header` type encountered") and the ICMP `Pointer` field containing the offset of the unrecognized value within the original packet. The same action should be taken if a node encounters a `Next Header` value of zero in any header other than an IPv6 header.

Each extension header is an integer multiple of 8 octets long, in order to retain 8-octet alignment for subsequent headers. Multioctet fields within each extension header are aligned on their natural boundaries, that is, fields of width *n* octets are placed at an integer multiple of *n* octets from the start of the header, for *n* = 1, 2, 4, or 8.

A full implementation of IPv6 includes implementation of the following extension headers: `Hop-by-Hop Options, Routing (Type 0), Fragment, Destination Options, Authentication,` and `Encapsulating Security Payload.`

10.5 Extension Header Order

When more than one extension header is used in the same packet, it is recommended that those headers appear in the following order:

IPv6 header

`Hop-by-Hop Options` header

`Destination Options` header

`Routing` header

`Fragment` header

`Authentication` header

`Encapsulating Security Payload` header

`Destination Options` header

upper-layer header

Additional recommendations regarding the relative order of the `Authentication` and `Encapsulating` headers are for options to be processed by the first destination that appears in the IPv6 `Destination Address` field plus subsequent destinations listed in the `Routing` header and for options to be processed only by the final destination of the packet.

Each extension header should occur at most once, except for the `Destination Options` header, which should occur at most twice (once before a `Routing` header and once before the upper-layer header). If the upper-layer header is another IPv6 header (if IPv6 is tunneled over or encapsulated in IPv6), it may be followed by its own extensions headers, which are separately subject to the same ordering recommendations.

If and when other extension headers are defined, their ordering constraints relative to the headers listed above must be specified.

IPv6 nodes must accept and attempt to process extension headers in any order and occurring any number of times in the same packet, except for the `Hop-by-Hop Options` header, which is restricted to appear immediately after an IPv6 header only. Nonetheless, it is strongly advised that sources of IPv6 packets adhere to the recommended order shown above until and unless subsequent specifications revise that recommendation.

Options

Two of the currently defined extension headers—the `Hop-by-Hop Options` header and the `Destination Options` header—carry a variable number of type-length-value (TLV)-encoded options, of the format shown in Fig. 10-3, where

Option Type	8-bit identifier of the type of option.
Optional Data Length	8-bit unsigned integer. Length of the Option. Data field of this option, in octets.
Option Data	Variable-length field. Option-Types specific data.

The sequence of options within a header must be processed strictly in the order they appear in the header; a receiver must not, for example, scan through the header looking for a particular kind of option and process that option prior to processing all preceding ones.

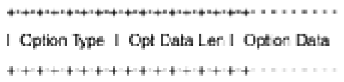


Figure 10-3
TLV-encoded options.

The option-type identifiers are internally encoded such that their highest-order two bits specify the action that must be taken if the processing IPv6 node does not recognize the option type:

00	Skip over this option and continue processing the header.
01	Discard the packet.
10	Discard the packet and, regardless of whether the packet's destination address was a multicast address, send an ICMP Parameter Problem, Code 2, message to the packet's Source Address, pointing to the unrecognized option type.
11	Discard the packet and, only if the packet's destination address was not a multicast address, send an ICMP Parameter Problem, Code 2, message to the packet's Source Address, pointing to the unrecognized option type.

The third-highest-order bit of the option type specifies whether the option data of that option can change en route to the packet's final destination. When an `Authentication` header is present in the packet, for any option whose data may change en route, its entire option-data field must be treated as zero-valued octets when computing or verifying the packet's authenticating value: 0—option data does not change en route, 1—option data may change en route.

Individual options may have specific alignment requirements, to ensure that multioctet values within option-data fields fall on natural boundaries. The alignment requirement of an option is specified using the notation *xn* + *y*, meaning that the option type must appear at an integer multiple of *x* octets from the start of the header, plus *y* octets. For example, 2*n* means any 2-octet offset from the start of the header, and, 8*n*+2 means any 8-octet offset from the start of the header, plus 2 octets.

Two padding options are used when necessary to align subsequent options and to pad out the containing header to a multiple of 8 octets in length. These padding options must be recognized by all IPv6 implementations (see Fig. 10-4).

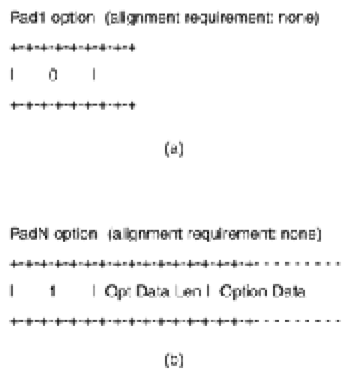

Pad1 option (alignment requirement: none)
0
(a)
PadN option (alignment requirement: none)
1
Opt Data Len
Option Data
(b)

Figure 10-4
IPv6 option recognition. (*a*) Pad1 option,
(*b*) PadN option.

The format of the `Pad1` option (Fig. 10-4*a*) is a special case—it does not have length and value fields. `Pad1` option is used to insert one octet of padding into the Options area of a header. If more than one octet of padding is required, the `PadN` option, described next, should be used, rather than multiple `Pad1` options.

The `PadN` option (Fig. 10-4*b*) is used to insert two or more octets of padding into the Options area of a header. For *N* octets of padding, the `Opt Data Len` field contains the value *N* – 2, and the Option Data consists of *N* – 2 zero-valued octets.

10.6 IPv6 Options Header (Hop-by-Hop)

The hop-by-hop options header is used to carry optional information that must be examined by every node along a packet's delivery path and is identified by a `Next Header` value of 0 in the IPv6 header, and has the format shown in Fig. 10-5, where

`Next Header`	8-bit selector. Identifies the type of header immediately following the hop-by-hop options header. Uses the same values as the IPv4 protocol field.
`Hdr Ext LenG`	8-bit unsigned integer. Length of the hop-by-hop options header in 8-octet units, not including the first 8 octets.
`Options`	Variable-length field, of length such that the complete hop-by-hop options header is an integer multiple of 8 octets long. Contains one or more TLV-encoded options.

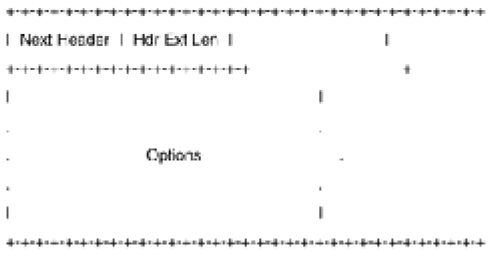


Figure 10-5
IPv6 options header.

In addition to the `Pad1` and `PadN` options the hop-by-hop option, the `Jumbo` Payload option, is defined as shown in Fig. 10-6.

The `Jumbo Payload` option is used to send IPv6 packets with payloads longer than 65,535 octets. The `Jumbo Payload Length` is the length of the packet in octets, excluding the IPv6 header but including the hop-by-hop options header; it must be greater than 65,535. If a packet is received with a `Jumbo Payload` option containing a `Jumbo Payload Length` less than or equal to 65,535, an ICMP Parameter Problem message, Code 0, should be sent to the packet's source, pointing to the high-order octet of the invalid `Jumbo Payload Length` field.

The `Payload Length` field in the IPv6 header must be set to zero in every packet that carries the `Jumbo Payload` option. If a packet is received with a valid `Jumbo Payload` option present and a nonzero IPv6 `Payload Length` field, an ICMP Parameter Problem message, Code 0, should be sent to the packet's source, pointing to the Option Type field of the `Jumbo Payload` option. The `Jumbo Payload` option must not be used in a packet that carries a `Fragment` header. If a `Fragment` header is encountered in a packet that contains a valid `Jumbo Payload` option, an ICMP Parameter Problem message, Code 0, should be sent to the packet's source, pointing to the first octet of the `Fragment` header.

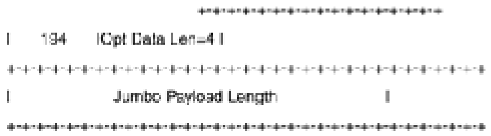


Figure 10-6
Jumbo Payload option.

An implementation that does not support the `Jumbo Payload` option cannot have interfaces to links whose link MTU is greater than 65,575 (40 octets of IPv6 header plus 65,535 octets of payload).

10.7 IPv6 Routing Header

The routing header is used by an IPv6 source to list one or more intermediate nodes to be "visited" on the way to a packet's destination. This function is very similar to IPv4's source route options. The routing header is identified by a `Next Header` value of 43 in the immediately preceding header, and has the format shown in Fig. 10-7, where

`Next Header`	8-bit selector. Identifies the type of header immediately following the routing header. Uses the same values as the IPv4 protocol field.
`Hdr Ext Len`	8-bit unsigned integer. Length of the routing header in 8-octet units, not including the first 8 octets.
`Routing Type`	8-bit identifier of a particular routing header variant.
`Segments Left`	8-bit unsigned integer. Number of route segments remaining, i.e., number of explicitly listed intermediate nodes still to be visited before reaching the final destination.
`type-specific data`	Variable-length field, of format determined by the routing type, and of length such that the complete routing header is an integer multiple of 8 octets long.

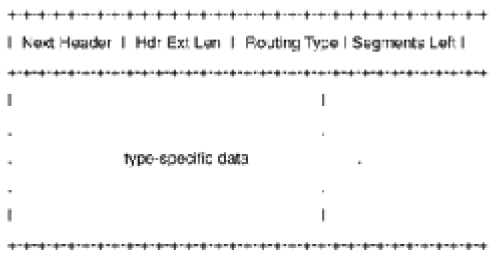


Figure 10-7
IPv6 routing header.

If, while processing a received packet, a node encounters a routing header with an unrecognized routing-type value, the required behavior of the node depends on the value of the `Segments Left` field, as follows. If `Segments Left` is zero, the node must ignore the routing header and proceed to process the next header in the packet, whose type is identified by the `Next Header` field in the routing header. If `Segments Left` is nonzero, the node must discard the packet and send an ICMP Parameter Problem, Code 0, message to the packet's Source Address, pointing to the unrecognized routing type. Figure 10-8 shows the Type 0 routing header format, where

`Next Header`	8-bit selector. Identifies the type of header immediately following the routing header. Uses the same values as the IPv4 protocol field.
`Hdr Ext Len`	8-bit unsigned integer. Length of the routing header in 8-octet units, not including the first 8 octets. For the type 0 Routing header, Hdr Ext Len is equal to twice the number of addresses in the header, and must be an even number less than or equal to 46.
`Routing Type`	0.
`Segments Left`	8-bit unsigned integer. Number of route segments remaining, i.e., number of explicitly listed intermediate nodes still to be visited before reaching the final destination. Maximum legal value = 23.
`Reserved`	8-bit reserved field. Initialized to zero for transmission; ignored on reception.
`Strict/Loose Bit Map`	24-bit bitmap, numbered 0 to 23, left to right.

This indicates, for each segment of the route, whether the next destination address must be a neighbor of the preceding address: 1 means strict (must be a neighbor), 0 means loose (need not be a neighbor).

Address[1..n] Vector of 128-bit addresses, numbered 1 to n.

Multicast addresses must not appear in a routing header of type 0, or in the IPv6 `Destination Address` field of a packet carrying a routing header of type 0. If bit number 0 of the `Strict/Loose Bit Map` has value 1, the Destination Address field of the IPv6 header in the original packet must identify a neighbor of the originating node. If bit number 0 has value 0, the originator may use any legal, nonmulticast address as the initial destination address. Bits numbered greater than *n*, where *n* is the number of addresses in the routing header, must be set to 0 by the originator and ignored by receivers.

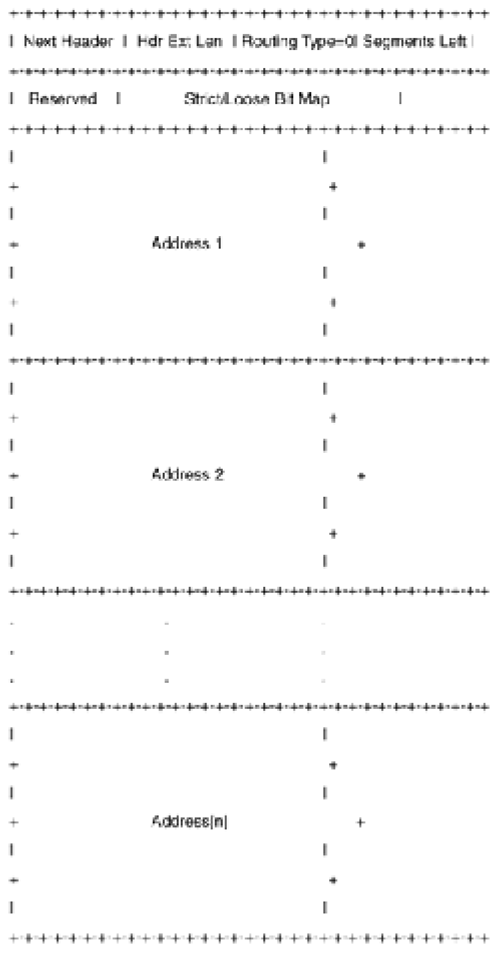


Figure 10-8
Type 0 routing header.

A routing header is not examined or processed until it reaches the node identified in the destination address field of the IPv6 header. In that node, dispatching on the `Next Header` field of the immediately preceding header causes the routing header module to be invoked, which, in the case of routing type 0, performs the following algorithm:

```
if Segments Left = 0
{
    proceed to process the next header in the packet, whose type is identified by the
    Next Header field in the routing header
}
else if Hdr Ext Len is odd or greater than 46
{
    send an ICMP Parameter Problem, Code 0, message to the Source Address, pointing
    to the Hdr Ext Len field, and discard the packet
}
else
{
    compute n, the number of addresses in the Routing header, by dividing Hdr Ext Len
    by 2
    if Segments Left is greater than n
    {
        send an ICMP Parameter Problem, Code 0, message to the Source Address, pointing
        to the Segments Left field, and discard the packet
    }
    else
    {
        decrement Segments Left by 1;
        compute i, the index of the next address to be visited
        in the address vector, by subtracting Segments Left from n
        if Address [i] or the IPv6 Destination Address is multicast
        {
            discard the packet
        }
        else
        {
            swap the IPv6 Destination Address and Address[i]
            if bit i of the Strict/Loose Bit map
            has value 1 and the new Destination Address is not the address of a neighbor of this
            node
            {
                send an ICMP Destination Unreachable — Not a Neighbor message to the Source
                Address and discard the packet
            }
            else if the IPv6 Hop Limit is less than or equal to 1
            {
                send an ICMP Time Exceeded — Hop Limit Exceeded in Transit message to the
                Source Address and discard the packet
            }
            else
            {
                decrement the Hop Limit by 1 and resubmit the packet to the IPv6 module for trans-
                mission to the new destination
            }
        }
    }
}
```

Consider the case of a source node S sending a packet to destination node D, using a Routing header to cause the packet to be routed via intermediate nodes I1, I2, and I3. The values of the relevant IPv6 header and routing header fields on each segment of the delivery path would be as follows, as the packet travels from S to I1:

Source Address = S   Hdr Ext Len = 6
Destination Address = I1   Segments Left = 3
          Address[1] = I2
(if bit 0 of the Bit Map is 1,   Address[2] = I3
S and I1 must be neighbors;   Address[3] = D
this is checked by S)
As the packet travels from I1 to I2:
Source Address = S   Hdr Ext Len = 6
Destination Address = I2   Segments Left = 2
          Address[1] = I1
(if bit 1 of the Bit Map is 1,   Address[2] = I3
I1 and I2 must be neighbors;   Address[3] = D
this is checked by I1)
As the packet travels from I2 to I3:
Source Address = S   Hdr Ext Len = 6
Destination Address = I3   Segments Left = 1
          Address[1] = I1
(if bit 2 of the Bit Map is 1,   Address[2] = I2
I2 and I3 must be neighbors;   Address[3] = D
this is checked by I2)
As the packet travels from I3 to D:
Source Address = S   Hdr Ext Len = 6
Destination Address = D   Segments Left = 0
          Address[1] = I1
(if bit 3 of the Bit Map is 1,   Address[2] = I2
I3 and D must be neighbors;   Address[3] = I3
this is checked by I3)

10.8 IPv6 Fragment Header

The *fragment header* is used by an IPv6 source to send packets larger than would fit in the path MTU to their destinations. (Unlike IPv4, fragmentation in IPv6 is performed only by source nodes, not by routers along a packet's delivery path. The fragment header is identified by a Next Header value of 44 in the immediately preceding header, and has the format shown in Fig. 10-9, where

`Next Header`	8-bit selector. Identifies the initial header type of the fragmental part of the original packet. Uses the same values as the IPv4 protocol field.
`Reserved`	8-bit reserved field. Initialized to zero for transmission; ignored on reception.

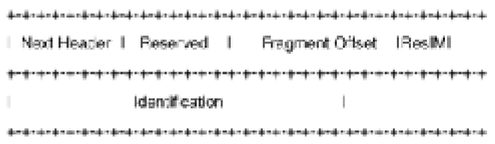


Figure 10-9
IPv6 fragment header.

`Fragment Offset`	13-bit unsigned integer. The offset, in 8-octet units, of the data following this header, relative to the start of the fragmental part of the original packet.
`Res`	2-bit reserved field. Initialized to zero for transmission; ignored on reception (M flag 1 = more fragments; 0 = last fragment).
`Identification`	32 bits. See description in text below.

In order to send a packet that is too large to fit in the MTU of the path to its destination, a source node may divide the packet into fragments and send each fragment as a separate packet, to be reassembled at the receiver. For every packet that is to be fragmented, the source node generates an identification value. The identification must be different from that of any other fragmented packet sent recently (i.e., within the maximum likely lifetime of a packet, including transit time from source to destination and time spent awaiting reassembly with other fragments of the same packet) with the same source address and destination address. If a Routing header is present, the destination address of concern is that of the final destination. However, it is not required that a source node know the maximum packet lifetime. Rather, it is assumed that the requirement can be met by maintaining the Identification value as a simple, 32-bit, "wraparound" counter, incremented each time a packet must be fragmented. It is an implementation choice whether to maintain a single counter for the node or multiple counters, for instance, one for each of the node's possible source addresses, or one for each active (source address, destination address) combination.

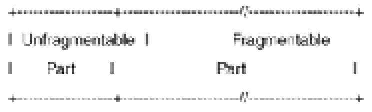


Figure 10-10
Original packet.

The initial, unfragmented packet is referred to as the original packet, and it is considered to consist of two parts, an unfragmentable and a fragmentable part (Fig. 10-10). The *unfragmentable part* consists of the IPv6 header plus any extension headers that must be processed by nodes en route to the destination, that is, all headers up to and including the routing header if present, else the hop-by-hop options header if present, else no extension headers. The *fragmentable part* consists of the rest of the packet, that is, any extension headers that need be processed only by the final destination node(s), plus the upper-layer header and data. The fragmentable part of the original packet is divided into fragments; each, except possibly the last ("rightmost") one, is an integer multiple of 8 octets long. The fragments are transmitted in separate "fragment packets" as illustrated in Fig. 10-11. Each fragment packet is composed of

1. The *unfragmentable part* of the original packet, with the payload length of the original IPv6 header changed to contain the length of this fragment packet only (excluding the length of the IPv6 header itself), and the `Next Header` field of the last header of the unfragmentable part changed to 44.

2. A fragment header containing (*a*) the `Next Header` value that identifies the first header of the fragmentable part of the original packet; (*b*) a Fragment Offset containing the offset of the fragment, in 8-octet units, relative to the start of the fragmentable part of the original packet—the fragment offset of the first ("leftmost") fragment is 0; (*c*) an *M* flag value of 0 if the fragment is the last rightmost one, else an *M* flag value of 1; and (*d*) the identification value generated for the original packet.

3. The fragment itself. The lengths of the fragments must be chosen such that the resulting fragment packets fit within the MTU of the path to the packets' destination(s).

At the destination, fragment packets are reassembled into their original, unfragmented form, as illustrated in Fig. 10-12. The following rules govern reassembly. An original packet is reassembled only from fragment packets that have the same source address, destination address, and fragment Identification.

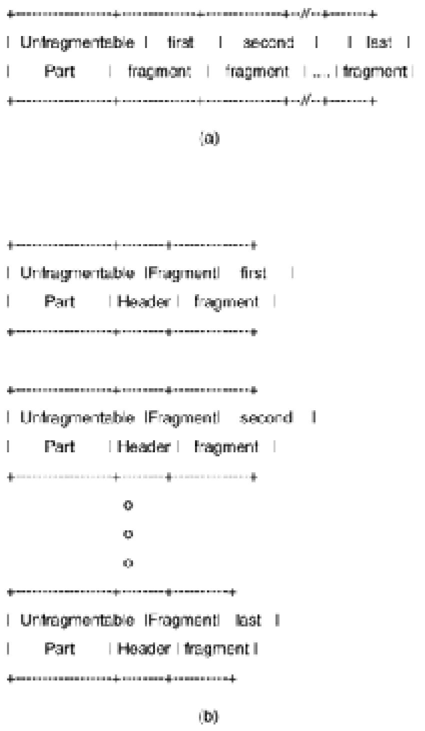


Figure 10-11
Transmitting the (*a*) original packet in (*b*) fragment packages.

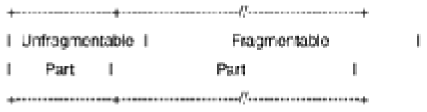


Figure 10-12
Reassembling the fragment packets.

The Unfragmentable Part of the reassembled packet consists of all headers up to, but not including, the fragment header of the first fragment packet (i.e., the packet whose Fragment Offset is zero), with the following two changes: (1) the `Next Header` field of the last header of the Unfragmentable Part is obtained from the `Next Header` field of the first fragment's fragment header, and (2) the payload length of the reassembled packet is computed from the length of the unfragmentable part and the length and offset of the last fragment. For example, a formula for computing the payload length of the reassembled original packet is

PL.orig = PL.first - FL.first - 8 1 (8 * FO.last) + FL.last

where `PL.orig` = payload-length field of reassembled packet; `PL.first` = payload-length field of first fragment packet; `FL.first` = length of fragment following fragment header of first fragment packet; `FO.last` = fragment-offset field of fragment header of last fragment packet; `FL.last` = length of fragment following fragment header of last fragment packet.

The fragmentable part of the reassembled packet is constructed from the fragments following the Fragment headers in each fragment packet. The length of each fragment is computed by subtracting from the packet's payload length the length of the headers between the IPv6 header and fragment itself; its relative position in fragmentable part is computed from its fragment-offset value.

The Fragment header is not present in the final, reassembled packet. The following error conditions may arise when reassembling fragmented packets. If insufficient fragments are received to complete reassembly of a packet within 60 s of the reception of the first-arriving fragment of that packet, reassembly of that packet must be abandoned and all the fragments that have been received for that packet must be discarded. If the first fragment (i.e., the one with a fragment offset of zero) has been received, an ICMP "time exceeded—fragment reassembly time exceeded" message should be sent to the source of that fragment.

If the length of a fragment, as derived from the fragment packet's payload-length field, is not a multiple of 8 octets and the *M* flag of that fragment is 1, then that fragment must be discarded and an ICMP Parameter Problem, Code 0, message should be sent to the source of the fragment, pointing to the Payload Length field of the fragment packet.

If the length and offset of a fragment are such that the payload length of the packet reassembled from that fragment would exceed 65,535 octets, then that fragment must be discarded and an ICMP Parameter Problem, Code 0, message should be sent to the source of the fragment, pointing to the fragment-offset field of the fragment packet.

The following conditions are not expected to occur, but are not considered errors if they do. The number and content of the headers preceding the fragment header of different fragments of the same original packet may differ. Whatever headers are present, preceding the fragment header in each fragment packet, are processed when the packets arrive, prior to queuing the fragments for reassembly. Only those headers in the offset zero-fragment packet are retained in the reassembled packet.

The `Next Header` values in the Fragment headers of different fragments of the same original packet may differ. Only the value from the offset zero-fragment packet is used for reassembly.

10.9 IPv6 Destination Options Header

The destination-options header is used to carry optional information that need be examined only by a packet's destination node(s). This header is identified by a Next Header value of 60 in the immediately preceding header, and has the format shown in Fig. 10-13, where

`Next Header`	8-bit selector. Identifies the type of header immediately following the destination-options header. Uses the same values as the IPv4 Protocol field.
`Hdr Ext Len`	8-bit unsigned integer. Length of the destination-options header in 8-octet units, not including the first 8 octets.
`Options`	Variable-length field, of length such that the complete destination-options header is an integer multiple of 8 octets long. Contains one or more TLV-encoded options.

The only destination options defined in this document are the `Pad1` and `PadN` options.

Note that there are two possible ways to encode optional destination information in an IPv6 packet: either as an option in the destination-options header or as a separate extension header. The fragment header and the authentication header are examples of the latter approach. Which approach can be used depends on what action is desired of a destination node that does not understand the optional information

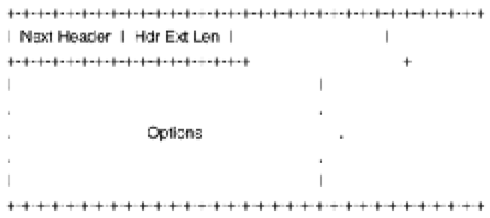


Figure 10-13
IPv6 destination options header.

If the desired action is for the destination node to discard the packet and, only if the packet's destination address is not a multicast address, send an ICMP unrecognized-type message to the packet's source address, then the information may be encoded either as a separate header or as an option in the destination-options header whose Option Type has the value 11 in its highest-order 2 bits. The choice may depend on such factors as which takes fewer octets, or which yields better alignment or more efficient parsing.

If any other action is desired, the information must be encoded as an option in the destination-options header whose option type has the value 00, 01, or 10 in its highest-order two bits, specifying the desired action.

10.10 IPv6 No Next Header

The value 59 in the Next Header field of an IPv6 header or any extension header indicates that there is nothing following that header. If the `Payload Length` field of the IPv6 header indicates the presence of octets past the end of a header whose `Next Header` field contains 59, those octets must be ignored, and passed on unchanged if the packet is forwarded.

10.11 IPv6 Packet Size Considerations

IPv6 requires that every link in the Internet have an MTU of 576 octets or greater. On any link that cannot convey a 576-octet packet in one piece, link-specific fragmentation and reassembly must be provided at a layer below IPv6. From each link to which a node is directly attached, the node must be able to accept packets as large as that link's MTU. Links that have a configurable MTU (e.g., PPP links must be configured to have an MTU of at least 576 octets; it is recommended that a larger MTU be configured, to accommodate possible encapsulations (i.e., tunneling) without incurring fragmentation.

It is strongly recommended that IPv6 nodes implement Path MTU Discovery, in order to discover and take advantage of paths with MTU greater than 576 octets. However, a minimal IPv6 implementation (e.g., in a boot ROM) may simply restrict itself to sending packets no larger than 576 octets, and omit implementation of Path MTU Discovery.

In order to send a packet larger than a path's MTU, a node may use the IPv6 fragment header to fragment the packet at the source and have it reassembled at the destination(s). However, the use of such fragmentation is discouraged in any application that is able to adjust its packets to fit the measured path MTU (i.e., down to 576 octets).

A node must be able to accept a fragmented packet that, after reassembly, is as large as 1500 octets, including the IPv6 header. A node is permitted to accept fragmented packets that reassemble to more than 1500 octets. However, a node must not send fragments that reassemble to a size greater than 1500 octets unless it has explicit knowledge that the destination(s) can reassemble a packet of that size.

In response to an IPv6 packet that is sent to an IPv4 destination (i.e., a packet that undergoes translation from IPv6 to IPv4), the originating IPv6 node may receive an "ICMP packet too big" message reporting a next-hop MTU less than 576. In that case, the IPv6 node is not required to reduce the size of subsequent packets to less than 576, but must include a Fragment header in those packets so that the IPv6-to-IPv4 translating router can obtain a suitable identification value to use in resulting IPv4 fragments. Note that this means the payload may have to be reduced to 528 octets (576 minus 40 for the IPv6 header and 8 for the Fragment header), and smaller still if additional extension headers are used.

The Path MTU Discovery must be performed even in cases where a host "thinks" a destination is attached to the same link as itself. Unlike IPv4, it is unnecessary in IPv6 to set a "Don't Fragment" flag in the packet header in order to perform Path MTU Discovery; that is an implicit attribute of every IPv6 packet. Also, those parts of the RFC-1191 procedures that involve use of a table of MTU "plateaus" do not apply to IPv6, because the IPv6 version of the "datagram too big" message always identifies the exact MTU to be used.

10.12 IPv6 Flow Labels

The 24-bit flow-label field in the IPv6 header may be used by a source to label those packets for which it requests special handling by the IPv6 routers, such as nondefault quality of service or *real-time* service. This aspect of IPv6 is, at the time of writing, still experimental and subject to change as the requirements for flow support in the Internet become clearer. Hosts or routers that do not support the functions of the flow-label field are required to set the field to zero when originating a packet, pass the field on unchanged when forwarding a packet, and ignore the field when receiving a packet.

A flow is a sequence of packets sent from a particular source to a particular (unicast or multicast) destination for which the source desires special handling by the intervening routers. The nature of that special handling might be conveyed to the routers by a control protocol, such as a resource reservation protocol, or by information within the flow's packets themselves, such as in a hop-by-hop option. The details of such control protocols or options are beyond the scope of this document.

There may be multiple active flows from a source to a destination, as well as traffic that is not associated with any flow. A flow is uniquely identified by the combination of a source address and a nonzero flow label. Packets that do not belong to a flow carry a flow label of zero. A flow label is assigned to a flow by the flow's source node. New flow labels must be chosen (pseudo-)randomly and uniformly from the range 1 to FFFFFF hex. The purpose of the random allocation is to make any set of bits within the flow-label field suitable for use as a hash key by routers, for looking up the state associated with the flow.

All packets belonging to the same flow must be sent with the same source address, destination address, priority, and flow label. If any of those packets includes a hop-by-hop options header, then they all must be originated with the same hop-by-hop options header contents (excluding the `Next Header` field of the hop-by-hop options header). If any of those packets includes a routing header, then they all must be originated with the same contents in all extension headers up to and including the routing header (excluding the `Next Header` field in the routing header). The routers or destinations are permitted, but not required, to verify that these conditions are satisfied. If a violation is detected, it should be reported to the source by an ICMP Parameter Problem message, Code 0, pointing to the high-order octet of the flow-label field (i.e., offset 1 within the IPv6 packet).

Routers are free to "opportunistically" set up a flow-handling state for any flow, even when no explicit flow establishment information has been provided to them via a control protocol, a hop-by-hop option, or other means. For example, on receiving a packet from a particular source with an unknown, nonzero flow label, a router may process its IPv6 header and any necessary extension headers as if the flow label were zero. That processing would include determining the next-hop interface, and possibly other actions, such as updating a hop-by-hop option, advancing the pointer and addresses in a Routing header, or deciding on how to queue the packet based on its `Priority` field. The router may then choose to "remember" the results of those processing steps and cache that information, using the source address plus the flow label as the cache key. Subsequent packets with the same source address and flow label may then be handled by referring to the cached information rather than examining all those fields that, according to the requirements of the previous paragraph, can be assumed unchanged from the first packet seen in the flow.

A cached flow-handling state that is set up opportunistically, as discussed in the preceding paragraph, must be discarded no more than 6 s after it is established, regardless of whether packets of the same flow continue to arrive. If another packet with the same source address and flow label arrives after the cached state has been discarded, the packet undergoes full, normal processing (as if its flow label were zero), which may result in the re-creation of cached flow state for that flow.

The lifetime of flow-handling state that is set up explicitly, for example, by a control protocol or a hop-by-hop option, must be specified as part of the specification of the explicit setup mechanism; it may exceed 6 s. A source must not reuse a flow label for a new flow within the lifetime of any flow-handling state that might have been established for the prior use of that flow label. Since flow-handling state with a lifetime of 6 s may be established opportunistically for any flow, the minimum interval between the last packet of one flow and the first packet of a new flow using the same flow label is 6 s. Flow labels used for explicitly setup flows with longer flow-state lifetimes must remain unused for those longer lifetimes before being reused for new flows.

When a node stops and restarts as a result of a *crash,* it must be careful not to use a flow label that it might have used for an earlier flow whose lifetime may not have expired yet. This may be accomplished by recording flow label usage on stable storage so that it can be remembered across crashes, or by refraining from using any flow labels until the maximum lifetime of any possible previously established flows has expired (at least 6 s; more if explicit flow setup mechanisms with longer lifetimes might have been used). If the minimum time for rebooting the node is known (often more than 6 s), that time can be deducted from the necessary waiting period before starting to allocate flow labels.

There is no requirement that all, or even most, packets belong to flows, that is, carry nonzero flow labels. This observation is placed here to remind protocol designers and implementers not to assume otherwise. For example, it would be unwise to design a router whose performance would be adequate only if most packets belonged to flows, or to design a header compression scheme that worked only on packets that belonged to flows.

10.13 IPv6 Packet Priority Field

The 4-bit `Priority` field in the IPv6 header enables a source to identify the desired delivery priority of its packets, relative to other packets from the same source. The priority values are divided into two ranges: Values 0 through 7 are used to specify the priority of traffic for which the source is providing congestion control, specifically, traffic that "backs off" in response to congestion, such as TCP traffic. Values 8 through 15 are used to specify the priority of traffic that does not back off in response to congestion, such as "real-time" packets being sent at a constant rate.

For congestion-controlled traffic, the following Priority values are recommended for particular application categories:

0	Uncharacterized traffic
1	"Filler" traffic (e.g., Netnews)
2	Unattended data transfer (e.g., email)
3	(Reserved)
4	Attended bulk transfer (e.g., FTP, NFS)
5	(Reserved)
6	Interactive traffic (e.g., TELNET, X)
7	Internet control traffic (e.g., routing protocols, SNMP)

For non-congestion-controlled traffic, the lowest priority value (8) should be used for those packets that the sender is most willing to have discarded under conditions of congestion (e.g., high-fidelity video traffic), and the highest value (15) should be used for those packets that the sender is least willing to have discarded (e.g., low-fidelity audio traffic). There is no relative ordering implied between the congestion-controlled priorities and the non-congestion-controlled priorities.

10.14 IPv6 and Upper-Layer Protocols

10.14.1 Upper-Layer Checksums

Any transport or other upper-layer protocol that includes the addresses from the IP header in its checksum computation must be modified for use over IPv6, to include the 128-bit IPv6 addresses instead of 32-bit IPv4 addresses. In particular, the following program fragment in Fig. 10-14 shows the TCP and UDP "pseudoheader" for IPv6.

1. If the packet contains a Routing header, the destination address used in the pseudoheader is that of the final destination. At the orig inating node, that address will be in the last element of the routing header; at the recipient(s), that address will be in the destination-address field of the IPv6 header.

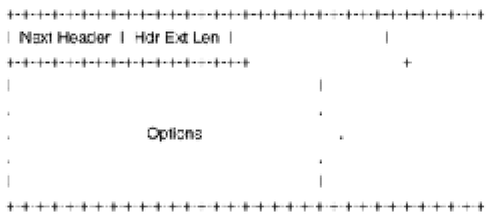


Figure 10-14
IPv6 TCP and UDP "pseudoheader."

2. The `Next Header` value in the pseudoheader identifies the upper-layer protocol (6 for TCP or 17 for UDP). It will differ from the Next Header value in the IPv6 header if there are extension headers between the IPv6 header and the upper-layer header.

3. The `Payload Length` used in the pseudoheader is the length of the upper-layer packet, including the upper-layer header. It will be less than the `Payload Length` in the IPv6 header (or in the `Jumbo Payload` option) if there are extension headers between the IPv6 header and the upper-layer header.

4. Unlike IPv4, when UDP packets are originated by an IPv6 node, the UDP checksum is not optional. That is, whenever originating a UDP packet, an IPv6 node must compute a UDP checksum over the packet and the pseudoheader, and, if that computation yields a result of zero, it must be changed to hex FFFF for placement in the UDP header. IPv6 receivers must discard UDP packets containing a zero checksum, and should log the error.

The IPv6 version includes this pseudoheader in its checksum computation; this is a change from the IPv4 version of ICMP, which does not include a pseudoheader in its checksum. The reason for the change is to protect ICMP from misdelivery or corruption of those fields of the IPv6 header on which it depends, which, unlike IPv4, are not covered by an Internet-layer checksum. The Next Header field in the pseudoheader for ICMP contains the value 58, which identifies the IPv6 version of ICMP.

Maximum Packet Lifetime

Unlike IPv4, IPv6 nodes are not required to enforce maximum packet lifetime; that is why the IPv4 time-to-live field was renamed "hop limit" in IPv6. In practice, very few, if any, IPv4 implementations conform to the requirement that they limit packet lifetime, so this is not a change in practice. Any upper-layer protocol that relies on the Internet layer (whether IPv4 or IPv6) to limit packet lifetime ought to be upgraded to provide its own mechanisms for detecting and discarding obsolete packets.

Maximum Upper-Layer Payload Size

When computing the maximum payload size available for upper-layer data, an upper-layer protocol must take into account the larger size of the IPv6 header relative to the IPv4 header. For example, in IPv4, TCP's MSS option is computed as the maximum packet size (a default value or a value learned through Path MTU Discovery) minus 40 octets (20 octets for the minimum-length IPv4 header and 20 octets for the minimum-length TCP header). When using TCP over IPv6, the MSS must be computed as the maximum packet size minus 60 octets, because the minimum-length IPv6 header (i.e., an IPv6 header with no extension headers) is 20 octets longer than a minimum-length IPv4 header.

10.14.2 Formatting Guidelines for Options

Following are some guidelines on how to lay out the fields when designing new options to be used in the hop-by-hop options header or the destination-options header. These guidelines are based on the following assumptions:

• One desirable feature is that any multioctet fields within the option-data area of an option be aligned on their natural boundaries, that is, fields of width *n* octets should be placed at an integer multiple of *n* octets from the start of the hop-by-hop or destination-options header, for *n* = 1, 2, 4, or 8.

• Another desirable feature is that the hop-by-hop or destination-options header take up as little space as possible, subject to the requirement that the header be an integer multiple of 8 octets long.

• It may be assumed that, when either of the option-bearing headers are present, these headers carry a very small number of options, usually only one.

These assumptions suggest the following approach to laying out the fields of an option: order the fields from smallest to largest, with no interior padding, then derive the alignment requirement for the entire option based on the alignment requirement of the largest field (up to a maximum alignment of 8 octets). This approach is illustrated in the following examples.

If an option X required two data fields, one of length 8 octets and one of length 4 octets, it would be laid out in Fig. 10-15. Its alignment requirement is 8*n* + 2, to ensure that the 8-octet field starts at a multiple-of-8 offset from the start of the enclosing header. A complete hop-by-hop or destination-options header containing this one option is shown in Fig. 10-16.

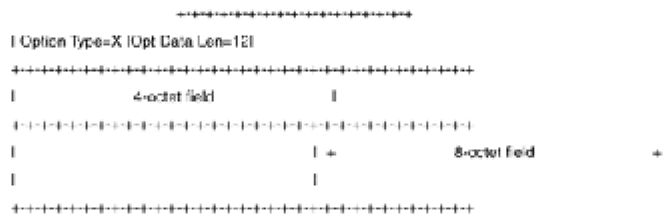


Figure 10-15
Option X with two data fields.

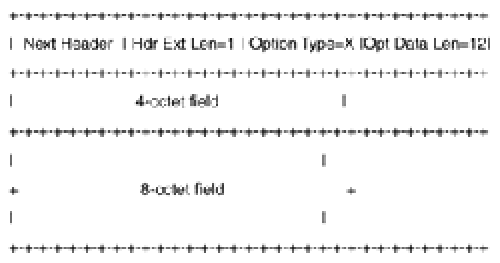


Figure 10-16
Hop-by-hop head for option X.

If an option Y required three data fields, one of length 4 octets, one of length 2 octets, and one of length 1 octet, it would be laid out as shown in Fig. 10-17.

Its alignment requirement is 4*n* + 3, to ensure that the 4-octet field starts at a multiple-of-4 offset from the start of the enclosing header. A complete hop-by-hop or destination-options header containing this one option is shown in Fig. 10-18.

A hop-by-hop or destination-options header containing both options X and Y from the first two examples would have one of the two formats shown in Fig. 10-19.

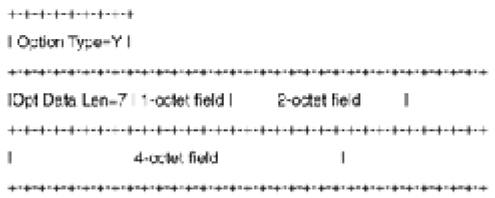


Figure 10-17
Option Y with three data fields.

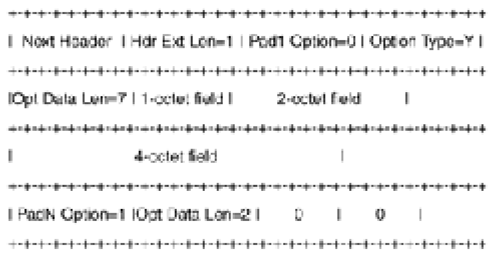


Figure 10-18
Hop-by-hop head for option Y.

10.15 IPv6 Addresses

10.15.1 Address Architecture

The remainder of this chapter focuses upon the address architecture of the IP version 6 protocol. It includes IPv6 addressing model, text representations of IPv6 addresses, definition of IPv6 unicast addresses, anycast addresses, and multicast addresses, and IPv6 nodes required addresses.

10.15.2 A Perspective

IPv6 addresses are 128-bit. There are three types of addresses:

1. *Unicast:* An identifier for a single interface. A packet sent to a unicast address is delivered to the interface identified by that address.

2. *Anycast:* An identifier for a set of interfaces (typically belonging to different nodes). A packet sent to an anycast address is delivered to one of the interfaces identified by that address (the “nearest” one, according to the routing protocols’ measure of distance).

3. *Multicast:* An identifier for a set of interfaces (typically belonging to different nodes). A packet sent to a multicast address is delivered to all interfaces identified by that address.

There are no broadcast addresses in IPv6. This type of address is superseded by one or more multicast addresses. Here address fields are given a specific name, for example, *subscriber*. When this name is used with the *ID* for identifier after the name subscriber ID, it refers to the contents of that field. When it is used with the term prefix, it refers to all the address up to and including this field. In IPv6, all zeros and ones are legal values for any field unless specifically excluded. Specifically, prefixes may contain zero-valued fields or end in zeros.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Next Header  | Hdr Ext Len=3 | Option Type=X |Opt Data Len=12|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         4-octet field                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                         8-octet field                         +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| PadN Option=1 |Opt Data Len=1 |       0       | Option Type=Y |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Opt Data Len=7 | 1-octet field |         2-octet field         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         4-octet field                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| PadN Option=1 |Opt Data Len=2 |       0       |       0       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Next Header  | Hdr Ext Len=3 | Pad1 Option=0 | Option Type=Y |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Opt Data Len=7 | 1-octet field |         2-octet field         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         4-octet field                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| PadN Option=1 |Opt Data Len=4 |       0       |       0       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       0       |       0       | Option Type=X |Opt Data Len=12|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         4-octet field                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                         8-octet field                         +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 10-19
Hop-by-hop head for both options X and Y.

10.15.3 Address Assignment

IPv6 addresses of all types are assigned to interfaces, *not nodes.* Since each interface belongs to a single node, any of that node's interfaces' unicast addresses may be used as an identifier for the node.

An IPv6 unicast address refers to a single interface. A single interface may be assigned multiple IPv6 addresses of any type (unicast, anycast, and multicast). There are two exceptions to this model: (1) a single address may be assigned to multiple physical interfaces if the implementation treats the multiple physical interfaces as one interface when presenting it to the Internet layer, or (2) routers may have unnumbered interfaces on point-to-point links to eliminate the necessity to manually configure and advertise the addresses.

Addresses are not required for point-to-point interfaces on routers if those interfaces are not to be used as the origins or destinations of any IPv6 datagrams.

IPv6 continues the IPv4 subnet model that is associated with one link. Multiple subnets may be assigned to that link. There are three conventional forms for representing IPv6 addresses as text strings:

1. The preferred form is x:x:x:x:x:x:x:x, where the "x"s are the hexadecimal values of the eight 16-bit pieces of the address. Examples are FEDC:BA98:7654:3210:FEDC:BA98:7654:3210 and 1080:0:0:0:8:800: 200C:417A.

2. Because of the method of allocating certain styles of IPv6 addresses, it is common for addresses to contain long strings of zero bits. To facilitate writing of addresses containing zero bits, a special syntax is available to compress the zeros. The use of “::” indicates multiple groups of 16 bits of zeros. The “::” can appear only once in an address. The “::” can also be used to compress the leading and/or trailing zeros in an address. For example, the following addresses 1080:0:0:0:8:800:200C:417A (a unicast address), FF01:0:0:0:0:0:0:43 (a multicast address), and 0:0:0:0:0:0:0:1 (the loopback address), and 0:0:0:0:0:0:0:0 (the unspecified addresses) may be represented as 1080::8:800:200C:417A (a unicast address), or FF01::43 (a multicast address), ::1 indicating the loopback address and :: indicating the unspecified addresses

3. An alternative form that is sometimes more convenient when dealing with a mixed environment of IPv4 and IPv6 nodes is x:x:x:x:x:x:d.d.d.d, where the “x”s are the hexadecimal values of the six high-order 16-bit pieces of the address, and the “d” s are the decimal values of the four low-order 8-bit pieces of the address (standard IPv4 representation). Examples are 0:0:0:0:0:0:13.1.68.3 and 0:0:0:0:0: FFFF:129.144.52.38. In compressed form these appear as ::13.1.68.3 and ::FFFF:129.144.52.38, respectively.

10.15.4 Address-Type Representation

The specific type of an IPv6 address is indicated by the address’s leading bits. The variable-length field comprising these leading bits is called the *format prefix* (FP). The initial allocations of these prefixes are listed in Table 10-1.

The “unspecified address,” the loopback address, and the IPv6 addresses with embedded IPv4 addresses are assigned out of the 0000 0000 format prefix space. This allocation supports the direct provider address allocation, local-use addresses, and multicast addresses. Space is reserved for NSAP addresses, IPX addresses, and geographic addresses. The remainder of the address space is unassigned for future use (this can be used for expansion of existing use or new uses); 15 percent of the address space is initially allocated. The remaining 85 percent is reserved for future use.

Table 10-1 Address Format Prefix Allocations

Allocation space	**Prefix (binary)**	**Fraction of address space**
Reserved	0000 0000	1/256
Unassigned	0000 0001	1/256
Reserved for NSAP Allocation	0000 001	1/128
Reserved for IPX Allocation	0000 010	1/128
Unassigned	0000 011	1/128
Unassigned	0000 1	1/32
Unassigned	0001	1/16
Unassigned	001	1/8
Provider-based unicast address	010	1/8

Unassigned	011	1/8
Reserved for geographically based unicast addresses	100	1/8
Unassigned	101	1/8
Unassigned	110	1/8
Unassigned	1110	1/16
Unassigned	1111 0	1/32
Unassigned	1111 10	1/64
Unassigned	1111 110	1/128
Unassigned	1111 1110 0	1/512
Link local-use addresses	1111 1110 10	1/1024
Site local-use addresses	1111 1110 11	1/1024
Multicast addresses	1111 1111	1/256

Unicast addresses are distinguished from multicast addresses by the value of the high-order octet of the addresses: a value of FF (11111111) identifies an address as a multicast address; any other value identifies an address as a unicast address. Anycast addresses are taken from the unicast address space, and are not syntactically distinguishable from unicast addresses.

10.15.5 Unicast Addresses

The IPv6 unicast address is contiguous bitwise maskable, similar to IPv4 addresses under *classless interdomain routing* (CIDR). There are several forms of unicast address assignment in IPv6, including the global-provider-based unicast address, the geographically based unicast address, the NSAP address, the IPX hierarchical address, the site-local-use address, the link-local-use address, and the IPv4-capable host address. Additional address types can be defined in the future.

IPv6 nodes (Fig. 10-20) may have considerable or little knowledge of the internal structure of the IPv6 address, depending on what the host does. Remember that a host is not necessarily a computer in the sense that a user does work on it; it could be any valid network device. At a minimum, a node may consider that unicast addresses (including its own) have no internal structure (Fig. 10-20*a*). A slightly sophisticated host (but still rather simple) may additionally be aware of subnet pre fix(es) for the link(s) it is attached to and different addresses can have different *n* values (Fig. 10-20*b*).

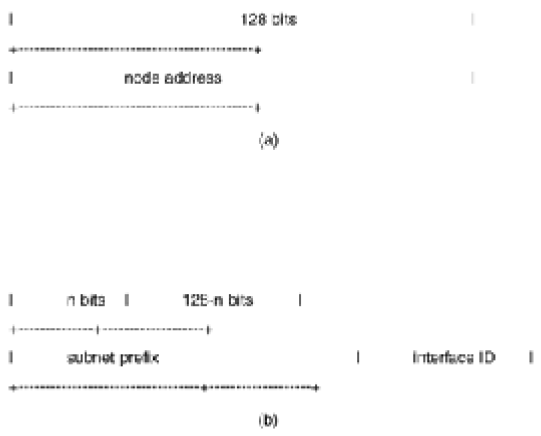


Figure 10-20
IPv6 nodes: (*a*) simple host. (*b*) slightly sophisticated host.

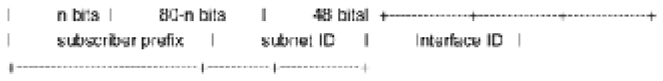


Figure 10-21
Unicast address format for IEEE 802 MAC addresses.

More sophisticated hosts may be aware of other hierarchical boundaries in the unicast address. Although a very simple router may have no knowledge of the internal structure of IPv6 unicast addresses, routers will more generally have knowledge of one or more of the hierarchical boundaries for the operation of routing protocols. The known boundaries will differ from router to router, depending on what positions the router holds in the routing hierarchy.

Figure 10-21 shows a unicast address format which will likely be common on LANs and other environments where IEEE 802 MAC addresses are available.

In this last algorithm the 48-bit Interface ID is an IEEE 802 MAC address. The use of IEEE 802 MAC addresses as a interface ID is expected to be very common in environments where nodes have an IEEE 802 MAC address. In other environments, where IEEE 802 MAC addresses are not available, other types of link-layer addresses can be used, such as E.164 addresses, for the interface ID.

The inclusion of a unique global interface identifier, such as an IEEE MAC address, makes possible a very simple form of autoconfiguration of addresses. A node may discover a subnet ID by listening to router advertisement messages sent by a router on its attached link(s), and then fabricating an IPv6 address for itself by using its IEEE MAC address as the interface ID on that subnet.

Figure 10-22 shows another unicast address format example where a site or organization requires additional layers of internal hierarchy. In this example the subnet ID is divided into an area ID and a subnet ID. This technique can be continued to allow a site or organization to add additional layers of internal hierarchy. It may be desirable to use an interface ID smaller than a 48-bit IEEE 802 MAC address to allow more space for the additional layers of internal hierarchy. These could be interface IDs which are administratively created by the site or organization.

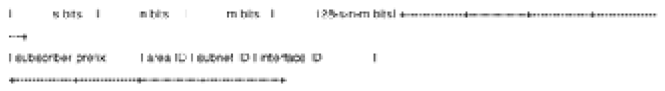


Figure 10-22
Unicast address format for internal hierarchy.

The address 0:0:0:0:0:0:0:0 is called the *unspecified address.* It must never be assigned to any node. It indicates the absence of an address. One example of its use is in the Source Address field of any IPv6 datagrams sent by an initializing host before it has learned its own address. The unspecified address must not be used as the destination address of IPv6 datagrams or in IPv6 Routing Headers. The unicast address 0:0:0:0:0:0:0:1 is called the *loopback address.* It may be used by a node to send an IPv6 datagram to itself. It may never be assigned to any interface. The loopback address must not be used as the source address in IPv6 datagrams that are sent outside a single node. An IPv6 datagram with a destination address of loopback must never be sent outside of a single node.

10.15.6 IPv6 Versus IPv4 Addresses

The IPv6 transition mechanisms include a technique for hosts and routers to dynamically tunnel IPv6 packets over IPv4 routing infrastructure. IPv6 nodes that utilize this technique are assigned special IPv6 unicast addresses that carry an IPv4 address in the low-order 32 bits. This type of address is termed an “IPv4-compatible IPv6 address” and has the format shown in Fig. 10-23.

A second type of IPv6 address which holds an embedded IPv4 address is also defined (see Fig. 10-24). This address is used to represent the addresses of IPv4-only nodes (those that *do not* support IPv6) as IPv6 addresses. This type of address is termed an “IPv4-mapped IPv6 address.”

Figures 10.25 and 10.26 show the mapping of NSAP addresses and IPX addresses, respectively, into IPv6 addresses. This initial assignment plan for global unicast addresses is similar to assignment of IPv4 addresses under the CIDR scheme. The IPv6 global-provider-based unicast address format is shown in Fig. 10-27.

The high-order bit part of the address is assigned to registries, which assigns portions of the address space to providers, which assigns portions of the address space to subscribers, and so forth.

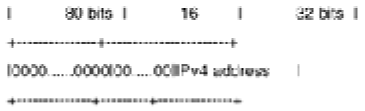


Figure 10-23
IPv4-compatible IPv6 address.

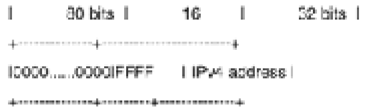


Figure 10-24
IPv4-mapped IPv6 address.

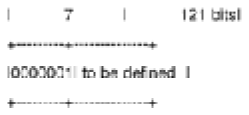


Figure 10-25
Mapping an NSAP address into an IPv6 address.

```
|     7     |     121 bits |
+-------+------------+
|0000010  | to be defined |
+-------+------------+
```

Figure 10-26
Mapping an IPX address into an IPv6 address.

```
| 3 | n bits | m bits | o bits | 125-n-m-o bits |
+---+--------+-------+-------+---------------+
|010|registry ID|provider ID|subscriber ID|intra-subscriber |
+---+----------+----------+-----------+--------------+
```

Figure 10-27
IPv6 global-provider-based unicast address.

The registry ID identifies the registry which assigns the provider portion of the address. The term *registry prefix* refers to the high-order part of the address up to and including the registry ID. The provider ID identifies a specific provider which assigns the subscriber portion of the address. The term *provider prefix* refers to the high-order part of the address up to and including the provider ID. The subscriber ID distinguishes among multiple subscribers attached to the provider identified by the provider ID. The term *subscriber prefix* refers to the high-order part of the address up to and including the subscriber ID.

The intrasubscriber portion of the address is defined by an individual subscriber and is organized according to the subscriber’s local Internet topology. It is likely that many subscribers will choose to divide the intrasubscriber portion of the address into a subnet ID and an interface ID. In this case the subnet ID identifies a specific physical link and the interface ID identifies a single interface on that subnet.

IPv6 Unicast Addresses

Two types of local-use unicast addresses are defined: link-local and site-local (see Fig. 10-28). The *link-local* address is for use on a single link; the *site-local* address is for use in a single site.

Link-local addresses (Fig. 10-28*a*) are designed for addressing on a single link for purposes such as autoaddress configuration, neighbor discovery, or when no routers are present. Routers are not permitted to forward any packets with link-local source addresses.

Site-local addresses have the format shown in Fig. 10-28*b.* Site-local addresses may be used for sites or organizations that are not (yet) connected to the global Internet. They do not need to request or “steal” an address prefix from the global Internet address space. IPv6 site-local addresses can be used instead. When the organization connects to the global Internet, it can then form global addresses by replacing the site-local prefix with a subscriber prefix.

```
| 10 |  bits |  n bits |       118-n bits      |
+----+---------+-----------------------------+
|1111111010   |  0       | interface ID  |
+----+---------+-----------------------------+
                  (a)


| 10 | bits | n  bits | m bits | 118-n-m bits    |
+----+------+---------+-------------------------+
|1111111011 |   0     | subnet ID | interface ID |
+---------+--------+----------+-------------+
                  (b)
```

Figure 10-28
Local-use unicast addresses: (*a* link-local
and (*b*) site-local.

Routers *must not* forward any packets with site-local source addresses outside of the site.

10.15.7 Anycast Addresses

An IPv6 *anycast address* is assigned to more than one interface (typically belonging to different nodes), with the property that a packet sent to an anycast address is routed to the *nearest* interface having that address, according to the routing protocols' measure of distance.

Anycast addresses are allocated from the unicast address space, using any of the defined unicast address formats. Thus, anycast addresses are syntactically indistinguishable from unicast addresses. When a unicast address is assigned to more than one interface, thus turning it into an anycast address, the nodes to which the address is assigned must be explicitly configured to know that it is an anycast address.

For any assigned anycast address, there is a longest address prefix P that identifies the topological region in which all interfaces belonging to that anycast address reside. Within the region identified by P, each member of the anycast set must be advertised as a separate entry in the routing system (referred to as a *host route*); outside the region identified by P, the anycast address may be aggregated into the routing advertisement for prefix P.

Note that in, the worst case, the prefix P of an anycast set may be the null prefix; thus the members of the set may have no topological locality. In that case, the anycast address must be advertised as a separate routing entry throughout the entire Internet, thus presenting a severe scaling limit on how many such "global" anycast sets may be supported. Therefore, it is expected that support for global anycast sets may be unavailable or very restricted.

One expected use of anycast addresses is to identify the set of routers belonging to an Internet service provider. Such addresses could be used as intermediate addresses in an IPv6 routing header, to cause a packet to be delivered via a particular provider or sequence of providers. Some other possible uses are to identify the set of routers attached to a particular subnet, or the set of routers providing entry into a particular routing domain.

There is little experience with widespread, arbitrary use of Internet anycast addresses, and some known complications and hazards when using them in their full generality [ANYCST]. Until more experience has been gained and solutions agreed on for those problems, the following restrictions are imposed on IPv6 anycast addresses. An anycast address *must not* be (1) used as the source address of an IPv6 packet or (2) be assigned to an IPv6 host—that is, it may be assigned to an IPv6 router only.

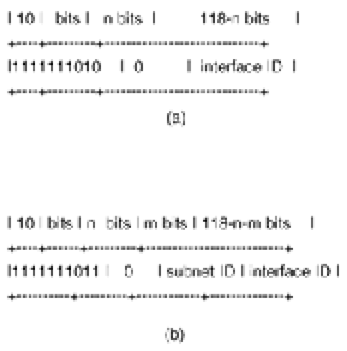

| 10 | bits | n bits | 118-n bits |
|1111111010 | 0 | interface ID |
(a)
| 10 | bits | n bits | m bits | 118-n-m bits |
|1111111011 | 0 | subnet ID | interface ID |
(b)

Figure 10-29
Subnet-router anycast address.

The subnet-router anycast address (Fig. 10-29) is predefined. The *subnet prefix* in an anycast address is the prefix which identifies a specific link. This anycast address is syntactically the same as a unicast address for an interface on the link with the interface identifier set to zero.

Packets sent to the subnet-router anycast address will be delivered to one router on the subnet. All routers are required to support the subnet-router anycast addresses for the subnets with which they interface.

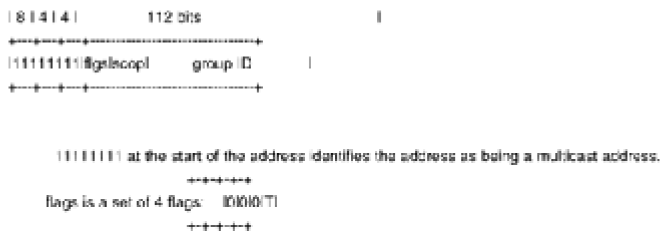
The subnet-router anycast address is intended for use in applications where a node needs to communicate with one of a set of routers on a remote subnet, such as when a mobile host needs to communicate with one of the mobile agents on its "home" subnet.

10.15.8 Multicast Addresses

An IPv6 multicast address (Fig. 10-30) is an identifier for a group of nodes. A node may belong to any number of multicast groups.

The high-order three flags are reserved, and must be initialized to 0. T = 0 indicates a permanently assigned ("well-known") multicast address, assigned by the global Internet numbering authority; T = 1 indicates a non–permanently assigned multicast address (it is also referred to as *transient*).

Scope is a 4-bit multicast scope value used to limit the scope of the multicast group. The values are 0—reserved, 1—node-local scope, 2—link-local scope, 3—(unassigned), 4—(unassigned), 5—site-local scope, 6—(unassigned), 7—(unassigned), 8—organization-local scope, 9—(unassigned), A—(unassigned), B—(unassigned), C—(unassigned), D—(unassigned), E—global scope, F—reserved.



```
| 8 | 4 | 4 |          112 bits                    |
+------+----+----+-----------------------------------+
|11111111|flgs|scop|        group ID            |
+------+----+----+-----------------------------------+

   11111111 at the start of the address identifies the address as being a multicast address.
                         +-+-+-+-+
flags is a set of 4 flags:   |0|0|0|T|
                         +-+-+-+-+
```

Figure 10-30
IPv6 multicast address format.

Group ID identifies the multicast group, either permanent or transient, within the given scope. The "meaning" of a permanently assigned multicast address is independent of the scope value. For example, if the "NTP servers group" is assigned a permanent multicast address with a group ID of 43 (hex), then

FF01:0:0:0:0:0:0:43 means all NTP servers on the same node as the sender.

FF02:0:0:0:0:0:0:43 means all NTP servers on the same link as the sender.

FF05:0:0:0:0:0:0:43 means all NTP servers at the same site as the sender.

FF0E:0:0:0:0:0:0:43 means all NTP servers in the Internet.

Non-permanently assigned multicast addresses are meaningful only within a given scope. For example, a group identified by the nonpermanent, site-local multicast address FF15:0:0:0:0:0:0:43 at one site bears no relationship to a group using the same address at a different site, nor to a nonpermanent group using the same group ID with different scope, nor to a permanent group with the same group ID.

Multicast addresses must not be used as source addresses in IPv6 datagrams or appear in any routing header.

Predefined Multicast Addresses

The following well-known multicast addresses are predefined:

1. Reserved Multicast Addresses:

`FF00:0:0:0:0:0:0:0`	`FF04:0:0:0:0:0:0:0`	`FF08:0:0:0:0:0:0:0`
`FF0C:0:0:0:0:0:0:0`	`FF01:0:0:0:0:0:0:0`	`FF06:0:0:0:0:0:0:0`
`FF09:0:0:0:0:0:0:0`	`FF0D:0:0:0:0:0:0:0`	`FF02:0:0:0:0:0:0:0`
`FF06:0:0:0:0:0:0:0`	`FF0A:0:0:0:0:0:0:0`	`FF0E:0:0:0:0:0:0:0`
`FF03:0:0:0:0:0:0:0`	`FF07:0:0:0:0:0:0:0`	`FF0B:0:0:0:0:0:0:0`
`FF0F:0:0:0:0:0:0:0`		

These multicast addresses are reserved and shall never be assigned to any multicast group.

2. *All-nodes Addresses:* FF01:0:0:0:0:0:0:1 and FF02:0:0:0:0:0:0:1. These multicast addresses identify the group of all IPv6 nodes, within scope 1 (node-local) or 2 (link-local).

3. *All-routers addresses:* FF01:0:0:0:0:0:0:2 and FF02:0:0:0:0:0:0:2. These multicast addresses identify the group of all IPv6 routers, within scope 1 (node-local) or 2 (link-local).

4. *DHCP server/relay agent:* FF02:0:0:0:0:0:0:C. These multicast addresses identify the group of all IPv6 DHCP servers and relay agents within scope 2 (link-local).

5. *Solicited-node address:* FF02:0:0:0:0:1:XXXX:XXXX. This multicast address is computed as a function of a node's unicast and anycast addresses. The solicited-node multicast address is formed by taking the low-order 32 bits of the address (unicast or anycast) and appending those bits to the 96-bit prefix FF02:0:0:0:0:1, resulting in a multicast address in the range FF02:0:0:0:0:1:0000:0000 to FF02:0:0:0:0:1:FFFF:FFFF. For example, the solicited-node multicast address corresponding to the IPv6 address 4037::01:800: 200E:8C6C is FF02::1:200E:8C6C. IPv6 addresses that differ only in the high-order bits, for example, due to multiple high-order prefixes associated with different providers, will map to the same solicited-node address thereby reducing the number of multicast addresses a node must join. A node is required to compute and support a solicited-node multicast address for every unicast and anycast address to which it is assigned.

10.15.9 Node Address Requirement

A host is required to recognize the following addresses as identifying itself: (1) its link-local address for each interface, (2) assigned unicast addresses, (3) loopback address, (4) all-nodes multicast address, (5) solicited-node multicast address for each of its assigned unicast and anycast addresses, and (6) multicast addresses of all other groups to which the host belongs.

A router is required to recognize the following addresses as identify ing itself: (1) its link-local address for each interface, (2) assigned unicast addresses, (3) loopback address, (4) the Subnet-Router anycast addresses for the links with which it interfaces, (5) all other Anycast addresses with which the router has been configured, (6) all-nodes multicast address, (7) all-router multicast address, (8) solicited-node multicast address for each of its assigned unicast and anycast addresses, and (9) multicast addresses of all other groups to which the router belongs.

The only address prefixes which should be predefined in an implementation are the (1) unspecified address, (2) loopback address, (3) multicast prefix (FF), (4) local-use prefixes (link-local and site-local), (5) predefined multicast addresses, and (6) IPv4-compatible prefixes.

Implementations should assume that all other addresses are unicast unless specifically configured (e.g., anycast addresses).

10.16 Summary

IP version 6 is more robust than its predecessor. Its ability to accommodate more networks and host addresses is only one of its many advantages.

11
Transmission Control Protocol

The *Transmission Control Protocol* (TCP) is intended for use as a reliable host-to-host protocol between hosts in packet-switched computer communication networks, and in interconnected systems of such networks. As strategic and tactical computer communication networks increase, it is essential to provide means of interconnecting them and to provide standard interprocess communication protocols which can support a broad range of applications.

11.1 TCP: A Perspective

TCP is a connection-oriented, end-to-end reliable protocol designed to fit into a layered hierarchy of protocols which support multinetwork applications. The TCP provides for reliable interprocess communication between pairs of processes in host computers attached to distinct but interconnected computer communication networks. Very few assumptions are made as to the reliability of the communication protocols below the TCP layer. TCP assumes that it can obtain a simple, potentially unreliable datagram service from the lower-level protocols. In principle, the TCP should be able to operate above a wide spectrum of communication systems ranging from hard-wired connections to packet-switched or circuit-switched networks.

TCP interfaces on one side to user or application processes and on the other side to a lower-level protocol such as the Internet Protocol (IP). The interface between an application process and TCP consists of a set of calls much like the calls that an operating system provides to an application process for manipulating files. For example, there are calls to open and close connections and to send and receive data on established connections. It is also expected that the TCP can asynchronously communicate with application programs. Although considerable freedom is permitted to TCP implementers to design interfaces which are appropriate to a particular operating system environment, a minimum functionality is required at the TCP/user interface for any valid implementation.

The interface between TCP and lower-level protocol is essentially unspecified except that it is assumed that there is a mechanism by which the two levels can asynchronously pass information to each other. Typically, one expects the lower-level protocol to specify this interface. TCP is designed to work in a very general environment of interconnected networks.

11.2 TCP Operation

As noted above, the primary purpose of the TCP is to provide reliable, securable logical circuit or connection service between pairs of processes. To provide this service on top of a less reliable Internet communication system requires facilities in the following areas:

1. *Basic data transfer*. TCP is able to transfer a continuous stream of octets in each direction between its users by packaging some number of octets into segments for transmission through the Internet system. In general, TCP decides when to block and forward data at their own convenience. Sometimes users need to ensure that all the data they have submitted to TCP have been transmitted. For this purpose a push function is defined. To assure that data submitted to TCP is actually transmitted the sending user indicates that they should be pushed through to the receiving user. A push causes the TCPs to promptly forward and deliver data up to that point to the receiver. The exact push point might not be visible to the receiving user, and the push function does not supply a record boundary marker.

2. *Reliability*. TCP must recover from data that are damaged, lost, duplicated, or delivered out of order by the Internet communication system. This is achieved by assigning a sequence number to each octet transmitted, and requiring a positive acknowledgment (ACK) from the receiving TCP. If the ACK is not received within a timeout interval, the data are retransmitted. At the receiver, the sequence numbers are used to correctly order segments that may be received out of order and to eliminate duplicates. Damage is handled by adding a checksum to each segment transmitted, checking it at the receiver, and discarding damaged segments. As long as the TCPs continue to function properly and the Internet system does not become completely partitioned, no trans mission errors will affect the correct delivery of data. TCP recovers from Internet communication system errors.

3. *Flow control*. TCP provides a means for the receiver to govern the amount of data sent by the sender. This is achieved by returning a “window” with every ACK indicating a range of acceptable sequence numbers beyond the last segment successfully received. The window indicates an allowed number of octets that the sender may transmit before receiving further permission.

4. *Multiplexing*. To allow for many processes within a single host to use TCP communication facilities simultaneously, the TCP provides a set of addresses or ports within each host. Concatenated with the network and host addresses from the Internet communication layer, this forms a socket. A pair of sockets uniquely identifies each connection. In other words, a socket may be simultaneously used in multiple connections. The binding of ports to processes is handled independently by each host. However, it proves useful to attach frequently used processes (a *logger* or timesharing service) to fixed sockets which are made known to the public. These services can then be accessed through the known addresses. Establishing and learning the port addresses of other processes may involve more dynamic mechanisms.

5. *Connections*. The reliability and flow-control mechanisms described above require that TCPs initialize and maintain certain status information for each data stream. The combination of this information, including sockets, sequence numbers, and window sizes, is called a *connection*. Each connection is uniquely specified by a pair of sockets identifying its two sides. When two processes wish to communicate, their TCPs must first establish a connection (initialize the status information on each side). When their communication is complete, the connection is terminated or closed to free the resources for other uses. Since connections must be established between unreliable hosts and over the unreliable Internet communication system, a handshake mechanism with clock-based sequence numbers is used to avoid erroneous initialization of connections.

6. *Precedence and security*. TCP users may indicate the security and precedence of their communication. Provision is made for default values to be used when these features are not needed.

11.3 TCP and the Host Environment

TCP is assumed to be a module in an operating system or a part of the protocol suite running on a given host. The users access the TCP much like they would access the file system. TCP may call on other operating system functions, for example, to manage data structures. The actual interface to the network is assumed to be controlled by a device driver module. The TCP does not call on the network device driver directly, but rather calls on the Internet datagram protocol module, which may, in turn, call on the device driver.

The mechanisms of TCP do not preclude implementation of the TCP in a front-end processor. However, in such an implementation, a host-to-front-end protocol must provide the functionality to support the type of TCP user interface described in this document.

Interfaces and TCP

The TCP/user interface provides for calls made by the user on the TCP to OPEN or CLOSE a connection, to SEND or RECEIVE data, or to obtain STATUS about a connection. These calls are like other calls from user programs on the operating system, for example, the calls to open, read from, and close a file.

TCP/Internet interface provides calls to send and receive datagrams addressed to TCP modules in hosts anywhere in the Internet system. These calls have parameters for passing the address, type of service, precedence, security, and other control information.

TCP Reliability

A stream of data sent on a TCP connection is delivered reliably and in order at the destination. Transmission is made reliable via the use of sequence numbers and acknowledgments. Conceptually, each octet of data is assigned a sequence number. The sequence number of the first octet of data in a segment is transmitted with that segment and is called the *segment sequence number*. Segments also carry an acknowledgment number which is the sequence number of the next expected data octet of transmissions in the reverse direction. When the TCP transmits a segment containing data, it puts a copy on a retransmission queue and starts a timer; when the acknowledgment for that data is received, the segment is deleted from the queue. If the acknowledgment is not received before the timer runs out, the segment is retransmitted.

An acknowledgment by TCP does not guarantee that the data have been delivered to the end user; It does mean that the receiving TCP has taken the responsibility to do so. To govern the flow of data between TCPs, a flow-control mechanism is employed. The receiving TCP reports a “window” to the sending TCP. This window specifies the number of octets, starting with the acknowledgment number, that the receiving TCP is currently prepared to receive.

TCP Connection Establishment and Clearing

To identify the separate data streams that a TCP may handle, the TCP provides a port identifier. Since port identifiers are selected independently by each TCP, they might not be unique. To provide for unique addresses within each TCP, we concatenate an Internet address identifying the TCP with a port identifier to create a *socket* which is unique throughout all networks connected together.

A connection is fully specified by the pair of sockets at the ends. A *local socket* may participate in many connections to different foreign sockets. A connection can be used to carry data in both directions; it is full-duplex.

TCPs are free to associate ports with processes in any way they choose. However, several basic concepts are necessary in any implementation. There must be well-known sockets which the TCP associates only with the appropriate processes by some means. We envision that processes may own ports, and that processes can initiate connections only on the ports they own. (Means for implementing ownership is a local issue, but we envision a REQUEST PORT user command, or a method of uniquely allocating a group of ports to a given process, by associating the high-order bits of a port name with a given process.)

A connection is specified in the OPEN call by the local port and foreign socket arguments. In return, the TCP supplies a (short) local connection name by which the user refers to the connection in subsequent calls. Several things must be remembered about a connection. To store this information, we imagine that there is a data structure called a *transmission control block* (TCB). One implementation strategy would have the local connection name be a pointer to the TCB for this connection. The OPEN call also specifies whether the connection establishment is to be actively pursued, or to be passively waited for.

A passive OPEN request means that the process wants to accept incoming connection requests rather than attempt to initiate a connection. Often the process requesting a passive OPEN will accept a connection request from any caller. In this case a foreign socket of all zeros is used to denote an unspecified socket. Unspecified foreign sockets are allowed only on passive OPENs. A service process that wished to provide services for unknown other processes would issue a passive OPEN request with an unspecified foreign socket. Then a connection could be made with any process that requested a connection to this local socket. It would help if this local socket were known to be associated with this service.

Well-known sockets are a convenient mechanism for a priori associating a socket address with a standard service. For instance, the TELNET server process is permanently assigned to a particular socket, and other sockets are reserved for file transfer, remote job entry, text generator, echoer, and sink processes. A socket address might be reserved for access to a lookup service which would return the specific socket at which a newly created service would be provided. The concept of a well-known socket is part of the TCP specification, but the assignment of sockets to services is outside this specification. Processes can issue passive OPENs and wait for matching active OPENs from other processes and be informed by the TCP when connections have been established. Two processes which issue active OPENs to each other at the same time will be correctly connected. This flexibility is critical for the support of distributed computing in which components act asynchronously with respect to each other.

There are two principal cases for matching the sockets in the local passive OPENs and an foreign active OPENs: (1) the local passive OPENs has fully specified the foreign socket, in which case the match must be exact; and (2) the local passive OPENs has left the foreign socket unspecified, in which case any foreign socket is acceptable as long as the local sockets match. Other possibilities include partially restricted matches.

If there are several pending passive OPENs (recorded in TCBs) with the same local socket, a foreign active OPEN will be matched to a TCB with the specific foreign socket in the foreign active OPEN, if such a TCB exists, before selecting a TCB with an unspecified foreign socket. The procedures to establish connections utilize the synchronize (SYN) control flag and involve an exchange of three messages. This exchange has been termed a *three-way handshake*.

A connection is initiated by the rendezvous of an arriving segment containing a SYN and a waiting TCB entry each created by a user OPEN command. The matching of local and foreign sockets determines when a connection has been initiated. The connection becomes established when sequence numbers have been synchronized in both directions. The clearing of a connection also involves the exchange of segments, in this case carrying the FIN control flag.

TCP and Data Communication

The data that flow on a connection may be thought of as a stream of octets. The sending user indicates in each SEND call whether the data in that call (and any preceding calls) should be immediately pushed through to the receiving user by the setting of the PUSH flag.

A sending TCP is allowed to collect data from the sending user and to send those data in segments at its own convenience, until the push function is signaled; then it must send all unsent data. When a receiving TCP sees the PUSH flag, it must not wait for more data from the sending TCP before passing the data to the receiving process. There is no necessary relationship between push functions and segment boundaries. The data in any particular segment may be the result of a single SEND call, in whole or part, or of multiple SEND calls.

The purpose of the push function and the PUSH flag is to push data through from the sending user to the receiving user. It does not provide a record service. There is a coupling between the push function and the use of buffers of data that cross the TCP/user interface. Each time a PUSH flag is associated with data placed into the receiving user's buffer, the buffer is returned to the user for processing even if the buffer is not filled. If data arrive that fills the user's buffer before a PUSH is seen, the data are passed to the user in buffer-size units. TCP also provides a means to communicate to the data receiver that at some point further along in the data stream than the receiver is currently reading there are urgent data. TCP does not attempt to define what the user specifically does on being notified of pending urgent data, but the general notion is that the receiving process will take action to process the urgent data quickly.

TCP Precedence and Security

The TCP makes use of the IP-type of service field and security option to provide precedence and security on a per-connection basis to TCP users. Not all TCP modules will necessarily function in a multilevel secure environment; some may be limited to unclassified use only, and others may operate at only one security level and compartment. Consequently, some TCP implementations and services to users may be limited to a subset of the multilevel secure case.

TCP modules which operate in a multilevel secure environment must properly mark outgoing segments with the security, compartment, and precedence. Such TCP modules must also provide to their users or higher-level protocols such as TELNET or THP an interface to allow them to specify the desired security level, compartment, and precedence of connections.

11.4 TCP Header Format

TCP segments are sent as Internet datagrams. The IP header carries several information fields, including the source and destination host addresses. A TCP header (Fig. 11-1) follows the IP header, supplying information specific to the TCP protocol. This division allows for the existence of host-level protocols other than TCP.

`Source Port`	16 bits. The source port number.
`Destination Port`	16 bits. The destination port number.
`Sequence Number`	32 bits. The sequence number of the first data octet in this segment (except when SYN is present). If SYN is present, the sequence number is the initial sequence number (ISN) and the first data octet is ISN + 1.
`Acknowledgment Number`	32 bits. If the ACK control bit is set, this field contains the value of the next sequence number that the sender of the segment is expecting to receive. Once a connection is established, this is always sent.

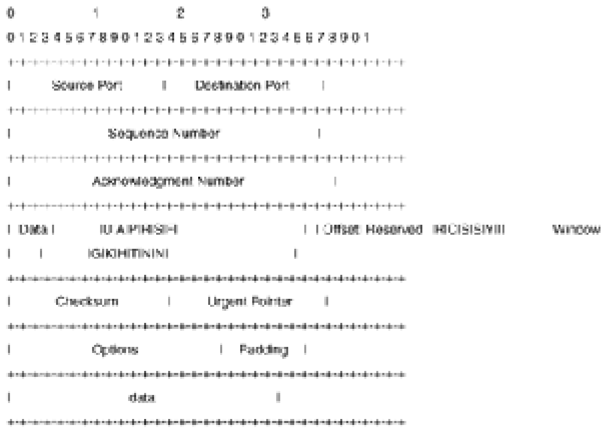


Figure 11-1
TCP header format.

`Data Offset`	4 bits. The number of 32-bit words in the TCP header. This indicates where the data begin. The TCP header (even one including options) is an integral number of 32 bits long.
`Reserved`	6 bits. Reserved for future use. Must be zero.
`Control Bits`	6 bits (from left to right): URG—urgent-pointer field significant, ACK—acknowledgment field significant, PSH—push function, RST—reset the connection, SYN—synchronize sequence numbers, FIN—no more data from sender.
`Window`	16 bits. The number of data octets beginning with the one indicated in the acknowledgment field which the sender of this segment is willing to accept.
`Checksum`	16 bits. The checksum field is the 16-bit one's complement of the one's-complement sum of all 16-bit words in the header and text.

If a segment contains an odd number of header and text octets to be checksummed, the last octet is padded on the right with zeros to form a 16-bit word for checksum purposes. The pad is not transmitted as part of the segment. While computing the checksum, the checksum field itself is replaced with zeros. The checksum also covers a 96-bit pseudoheader conceptually prefixed to the TCP header. This pseudoheader (Fig. 11-2) contains the `Source Address,` the `Destination Address,` the `Protocol,` and `TCP Length.` This gives the TCP protection against misrouted segments. This information is carried in the Internet Protocol and is transferred across the TCP/network interface in the arguments or results of calls by the TCP on the IP.

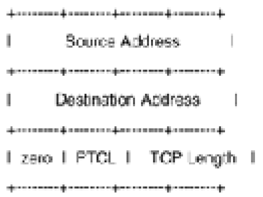


Figure 11-2
Checksum pseudoheader.

The `TCP Length` is the TCP header length plus the data length in octets (this is not an explicitly transmitted quantity, but is computed), and it does not count the 12 octets of the pseudoheader. The *urgent-pointer* field is 16 bits in length. This field communicates the current value of the urgent pointer as a positive offset from the sequence number in this segment. The urgent pointer points to the sequence number of the octet following the urgent data. This field is interpreted only in segments with the `URG` control bit set. The *options* field is variable. Options may occupy space at the end of the TCP header and are a multiple of 8 bits in length. All options are included in the checksum. An option may begin on any octet boundary. There are two cases for the format of an option:

Case 1: a single octet of option-kind

Case 2: an octet of option-kind, an octet of option-length, and the actual option-data octets.

The option-length counts the two octets of option-kind and option-length as well as the option-data octets.

The list of options may be shorter than the data offset field might imply. The content of the header beyond the end-of-option option must be header padding (i.e., zero). A TCP must implement all options. Currently defined options include (kind indicated in octals):

Kind	**Length**	**Meaning**
0	—	End-of-option list
1	—	No operation
2	4	Maximum segment size

Specific option definitions are shown in the `end-of-option` list code (Fig. 11-3) and no-operation option code (Fig. 11-4).

The end-of-option list code indicates the end of the option list. This might not coincide with the end of the TCP header according to the data-offset field. This is used at the end of all options, not the end of each option, and need be used only if the end of the options would not otherwise coincide with the end of the TCP header. The no-operation option code may be used between options, for example, to align the beginning of a subsequent option on a word boundary. There is no guarantee that senders will use this option, so receivers must be prepared to process options even if they do not begin on a word boundary.

The `maximum segment size` option (Fig. 11-5) data is 16 bits. If this option is present, then it communicates the maximum receive segment size at the TCP which sends this segment. This field must be sent only in the initial connection request (i.e., in segments with the `SYN` control bit set). If this option is not used, any segment size is allowed.

Padding is variable. The TCP header padding is used to ensure that the TCP header ends and data begin on a 32-bit boundary. The padding is composed of zeros.

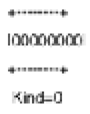


Figure 11-3
End-of-option list code.

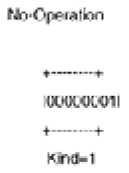


Figure 11-4
No-operation option code.

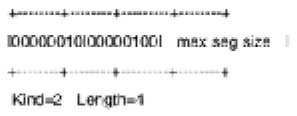


Figure 11-5
Maximum segment size.

11.5 TCP Terminology

It is important to understand some detailed terminology. Consider the following information. The maintenance of a TCP connection requires the user to remember several variables. We conceive of these variables as being stored in a connection record called a *transmission control block* (TCB). Among the variables stored in the TCB are the local and remote socket numbers, the security and precedence of the connection, pointers to the user's send and receive buffers, pointers to the retransmit queue, and pointers to the current segment. In addition, several variables relating to the send and receive sequence numbers are stored in the TCB.

1. *Send sequence variables:* `SND.UNA`—send unacknowledged; `SND.NXT`—send next; `SND.WND`—send window; `SND.UP`—send urgent pointer; `SND.WL1`—segment sequence number used for last window update; `SND.WL2`—segment acknowledgment number used for last window update; `ISS`—initial send sequence number.

2. *Receive sequence variables:* `RCV.NXT`—receive next; `RCV.WND`—receive window; `RCV.UP`—receive urgent pointer; `IRS`—initial receive sequence number.

Consider the following to relate some of these variables to sequence space.

1. *Send sequence space:*

```
1     2     3      4
--------|-----|-------|-----
   SND.UNA      SND.NXT  SND.UNA
                      +SND.WND
```

where 1 = old sequence numbers which have been acknowledged; 2 = sequence numbers of unacknowledged data; 3 = sequence numbers allowed for new data transmission; 4 = future sequence numbers which are not yet allowed.

2. *Receive sequence space:*

```
1    2     3
------|----|----
   RCV.NXT    RCV.NXT
          +RCV.WND
```

where 1 = old sequence numbers which have been acknowledged; 2 = sequence numbers allowed for new reception; 3 = future sequence numbers which are not yet allowed.

There are also some variables mentioned frequently in the discussion that take their values from the fields of the current segment:

`SEG.SEQ`	segment sequence number
`SEG.ACK`	segment acknowledgment number
`SEG.LEN`	segment length
`SEG.WND`	segment window
`SEG.UP`	segment urgent pointer
`SEG.PRC`	segment precedence value

A connection progresses through a series of states during its lifetime. The states are `LISTEN, SYN-SENT, SYN-RECEIVED, ESTABLISHED, FIN-WAIT-1, FIN-WAIT-2, CLOSE-WAIT, CLOSING, LAST-ACK, TIME-WAIT` and `CLOSED`.

The state `CLOSED` is fictional because it represents the state when there is no TCB, and therefore, no connection. Briefly the meanings of the states are as follows: `LISTEN`—represents waiting for a connection request from any remote TCP and port; `SYN-SENT` represents waiting for a matching connection request after having sent a connection request; `SYN-RECEIVED` represents waiting for a confirming connection request acknowledgment after having both received and sent a connection request; `ESTABLISHED` represents an open connection, indicating that data received can be delivered to the user.

The normal state for the data transfer phase of the connection is as follows: `FIN-WAIT-1` represents waiting for a connection termination request from the remote TCP, or an acknowledgment of the connection termination request previously sent; `FIN-WAIT-2` represents waiting for a connection termination request from the remote TCP; `CLOSE-WAIT` represents waiting for a connection termination request from the local user; `CLOSING` represents waiting for a connection termination request acknowledgment from the remote TCP; `LAST-ACK` represents waiting for an acknowledgment of the connection termination request previously sent to the remote TCP (which includes an acknowledgment of its connection termination request); `TIME-WAIT` represents waiting for enough time to pass to be sure the remote TCP received the acknowledgment of its connection termination request; and `CLOSED` represents no connection state at all.

A TCP connection progresses from one state to another in response to events. The events are the user calls, `OPEN, SEND, RECEIVE, CLOSE, ABORT,` and `STATUS`; the incoming segments, particularly those containing the `SYN, ACK, RST,` and `FIN` flags; and timeouts.

A state diagram would illustrate only state changes, together with the causing events and resulting actions, but addresses neither error conditions nor actions which are not connected with state changes. The reaction of the TCP to events is discussed in more detail later.

11.6 TCP Sequence Numbers

A fundamental notion in the design is that every octet of data sent over a TCP connection has a sequence number. Since every octet is sequenced, each of them can be acknowledged. The acknowledgment mechanism employed is cumulative; thus an acknowledgment of sequence number *X* indicates that all octets up to but not including *X* have been received. This mechanism allows for straightforward duplicate detection in the presence of retransmission. Numbering of octets within a segment is that the first data octet immediately following the header is the lowest-numbered, and the following octets are numbered consecutively.

It is essential to remember that the actual sequence number space is finite, although very large. This space ranges from 0 to 2**32 – 1. Since the space is finite, all arithmetic dealing with sequence numbers must be performed modulo 2**32. This unsigned arithmetic preserves the relationship of sequence numbers as they cycle from 2**32 – 1 to 0 again. There are some subtleties to computer modulo arithmetic, so great care should be taken in programming the comparison of such values. The symbol " = <" means "less than or equal to" (modulo 2**32).

The typical kinds of sequence number comparisons which the TCP must perform include determining that (1) an acknowledgment refers to some sequence number sent but not yet acknowledged, (2) all sequence numbers occupied by a segment have been acknowledged (e.g., to remove the segment from a retransmission queue), and (3) an incoming segment contains sequence numbers which are expected (i.e., that the segment "overlaps" the receive window).

In response to sending data, the TCP will receive acknowledgments. The following comparisons are needed to process the acknowledgments.

`SND.UNA`	Oldest unacknowledged sequence number
`SND.NXT`	Next sequence number to be sent
`SEG.ACK`	Acknowledgment from the receiving TCP (next sequence number expected by the receiving TCP)
`SEG.SEQ`	First sequence number of a segment
`SEG.LEN`	The number of octets occupied by the data in the segment (counting SYN and FIN)
`SEG.SEQ + SEG.LEN-1`	Last sequence number of a segment

A new acknowledgment (called an "acceptable ack"), is one for which the inequality below holds:

SND.UNA < SEG.ACK = < SND.NXT

A segment on the retransmission queue is fully acknowledged if the sum of its sequence number and length is less than or equal to the acknowledgment value in the incoming segment. When data are received, the following comparisons are needed:

`RCV.NXT`	Next sequence number expected on an incoming segment; the left or lower edge of the receive window
`RCV.NXT + RCV.WND-1`	Last sequence number expected on an incoming segment; the right or upper edge of the receive window
`SEG.SEQ`	First sequence number occupied by the incoming segment
`SEG.SEQ + SEG.LEN-1`	Last sequence number occupied by the incoming segment

A segment is judged to occupy a portion of valid receive sequence space if

RCV.NXT = < SEG.SEQ < RCV.NXT + RCV.WND

or

RCV.NXT = < SEG.SEQ + SEG.LEN-1 < RCV.NXT + RCV.WND

The first part of this test checks to see if the beginning of the segment falls in the window; the second part of the test checks to see if the end of the segment falls in the window. If the segment passes either part of the test, it contains data in the window. Actually, it is a little more complicated than this. Because there are zero windows and zero length segments, we have four cases for the acceptability of an incoming segment:

Segment Length	**Receive Window**	**Test**
0	0	`SEG.SEQ` = RCV.NXT
0	>0	`RCV.NXT` = < SEG.SEQ < RCV.NXT + RCV.WND
>0	0	Not acceptable
>0	>0	`RCV.NXT` = < SEG.SEQ < RCV.NXT + RCV.WND or RCV.NXT = < SEG.SEQ + SEG.LEN-1 < RCV.NXT + RCV.WND

When the receive window is zero, no segments should be acceptable except `ACK` segments. Thus, it is possible for a TCP to maintain a zero receive window while transmitting data and receiving `ACKs`. However, even when the receive window is zero, a TCP must process the `RST` and `URG` fields of all incoming segments.

We have taken advantage of the numbering scheme to protect certain control information as well. This is achieved by implicitly including some control flags in the sequence space so that they can be retransmitted and acknowledged without confusion (i.e., one and only one copy of the control will be acted on). Control information is not physically carried in the segment data space. Consequently, we must adopt rules for implicitly assigning sequence numbers to control. The `SYN` and `FIN` are the only controls requiring this protection, and these controls are used only at connection opening and closing. For sequence number purposes, the `SYN` is considered to occur before the first actual data octet of the segment in which it occurs, while the `FIN` is considered to occur after the last actual data octet in a segment in which it occurs. The segment length (`SEG.LEN`) includes both data and sequence space occupying controls. When a `SYN` is present, then `SEG.SEQ` is the sequence number of the `SYN`.

Initial Sequence Number Selection

The protocol places no restriction on a particular connection being used over and over again. A connection is defined by a pair of sockets. New instances of a connection will be referred to as incarnations of the connection. The problem that arises from this is how the TCP identifies duplicate segments from previous incarnations of the connection. This becomes apparent if the connection is being opened and closed in rapid succession, or if the connection breaks with loss of memory and is then reestablished.

To avoid confusion, we must prevent segments from one incarnation of a connection from being used while the same sequence numbers may still be present in the network from an earlier incarnation. We want to ensure this, even if a TCP crashes and loses all knowledge of the sequence numbers it has been using. When new connections are created, an initial sequence number (ISN) generator is employed which selects a new 32-bit ISN. The generator is bound to a (possibly fictitious) 32-bit clock whose low-order bit is incremented roughly every 4 μs. Thus, the ISN cycles approximately every 4.55 h. Since we assume that segments will stay in the network no more than the maximum segment lifetime (MSL) and that the MSL is less than 4.55 h we can reasonably assume that ISNs will be unique.

For each connection there is a send sequence number and a receive sequence number. The initial send sequence number (ISS) is chosen by the data sending TCP, and the initial receive sequence number (IRS) is learned during the connection establishing procedure. For a connection to be established or initialized, the two TCPs must synchronize on each other’s initial sequence numbers. This is done in an exchange of connection establishing segments carrying a control bit called `SYN` (for synchronize) and the initial sequence numbers. As a shorthand, segments carrying the `SYN` bit are also called `SYN`s. Hence, the solution requires a suitable mechanism for picking an initial sequence number and a slightly involved handshake to exchange the ISNs.

The synchronization requires each side to send its own initial sequence number and to receive a confirmation of it in acknowledgment from the other side. Each side must also receive the other side’s initial sequence number and send a confirming acknowledgment.

1. A —> B `SYN` my sequence number is *X*.

2. A <— B `ACK` your sequence number is *X*.

3. A <— B `SYN` my sequence number is *Y*.

4. A —> B `ACK` your sequence number is *Y*.

Steps 2 and 3 can be combined in a single message; this is called the *three-way* (or *three-message*) *handshake*. A three-way handshake is necessary because sequence numbers are not tied to a global clock in the network, and TCPs may have different mechanisms for picking the ISNs. The receiver of the first `SYN` has no way of knowing whether the segment was an old delayed one, unless it remembers the last sequence number used on the connection (which is not always possible), and so it must ask the sender to verify this `SYN`.

Knowing When to Keep Quiet

To ensure that a TCP does not create a segment that carries a sequence number which may be duplicated by an old segment remaining in the network, the TCP must keep quiet for a maximum segment lifetime (MSL) before assigning any sequence numbers on starting up or recovering from a crash in which memory of sequence numbers in use was lost. For this specification the MSL is taken to be 2 min. This is an engineering choice, and may be changed if experience indicates that it is desirable to do so. Note that if a TCP is reinitialized in some sense, yet retains its memory of sequence numbers in use, then it need not wait at all; it must only be sure to use sequence numbers larger than those recently used.

TCP Quiet-time Concept

This specification provides that hosts which “crash” without retaining any knowledge of the last sequence numbers transmitted on each active (i.e., not closed) connection shall delay emitting any TCP segments for at least the agreed maximum segment lifetime (MSL) in the Internet system of which the host is a part. This specification is explained in the following paragraphs.

TCP implementers may violate the “quiet time” restriction, but only at the risk of causing some old data to be accepted as new or new data rejected as old duplicated by some receivers in the Internet system. TCPs consume sequence number space each time a segment is formed and entered into the network output queue at a source host. The duplicate detection and sequencing algorithm in the TCP protocol relies on the unique binding of segment data to sequence space to the extent that sequence numbers will not cycle through all 2**32 values before the segment data bound to those sequence numbers has been delivered and acknowledged by the receiver and all duplicate copies of the segments have “drained” from the Internet. Without such an assumption, two distinct TCP segments could conceivably be assigned the same or overlapping sequence numbers, causing confusion at the receiver as to which data are new and which are old. Remember that each segment is bound to as many consecutive sequence numbers as there are octets of data in the segment.

Under normal conditions, TCPs keep track of the next sequence number to emit the oldest awaiting acknowledgment so as to avoid mistakenly using a sequence number over before its first use has been acknowledged. This alone does not guarantee that old duplicate data are drained from the net, so the sequence space has been made very large to reduce the probability that a wandering duplicate will cause trouble on arrival. At 2 Mbits/s it takes 4.5 h to use up 2**32 octets of sequence space. Since the maximum segment lifetime in the net is not likely to exceed a few tens of seconds, this is deemed ample protection for foreseeable nets, even if data rates escalate to tens of megabits per second. At 100 Mbits/s, the cycle time is 5.4 min, which may be a little short, but still within reason.

The basic duplicate detection and sequencing algorithm in TCP can be defeated, however, if a source TCP does not have any memory of the sequence numbers it last used on a given connection. For example, if the TCP were to start all connections with sequence number 0, then on crashing and restarting, a TCP might re-form an earlier connection (possibly after half-open connection resolution) and emit packets with sequence numbers identical to or overlapping with packets still in the network which were emitted on an earlier incarnation of the same connection. In the absence of knowledge about the sequence numbers used on a particular connection, the TCP specification recommends that the source delay for MSL seconds before emitting segments on the connection, to allow time for segments from the earlier connection incarnation to drain from the system.

Even hosts which can remember the time of day and use it to select initial sequence number values are not immune to this problem (i.e., even if time of day is used to select an initial sequence number for each new connection incarnation).

Suppose that a connection is opened starting with sequence number S. Suppose that this connection is not used much and that eventually the initial sequence number function ISN(t) takes on a value equal to the sequence number, say S_1, of the last segment sent by this TCP on a particular connection. Now suppose, at this instant, that the host crashes, recovers, and establishes a new incarnation of the connection. The initial sequence number chosen is S_1 = ISN(t)—the last used sequence number on old incarnation of the connection! If the recovery occurs quickly enough, any old duplicates in the net bearing sequence numbers in the neighborhood of S_1 may arrive and be treated as new packets by the receiver of the new incarnation of the connection. The problem is that the recovering host may not know for how long it crashed nor whether there are still old duplicates in the system from earlier connection incarnations. One way to deal with this problem is to deliberately delay emitting segments for one MSL after recovery from a crash—this is the quiet-time specification. Hosts which prefer to avoid waiting are willing to risk possible confusion of old and new packets at a given destination and may choose not to wait for the quiet time. Implementers may provide TCP users with the ability to decide on a connection-by-connection basis whether to wait after a crash, or may informally implement the quiet time for all connections.

Obviously, even where a user selects to "wait," this is not necessary after the host has been "up" for at least MSL seconds. To summarize: (1) every segment emitted occupies one or more sequence numbers in the sequence space; (2) the numbers occupied by a segment are *busy* or *in use* until MSL seconds have passed; (3) upon crashing, a space–time block is occupied by the octets of the last emitted segment; and (4) if a new connection is started too soon and uses any of the sequence numbers in the space–time footprint of the last segment of the previous connection incarnation, there is a potential sequence number overlap area which could cause confusion at the receiver.

11.7 Establishing a TCP Connection

The *three-way handshake* is the procedure used to establish a connection. This procedure normally is initiated by one TCP and responded to by another TCP. The procedure also works if two TCPs simultaneously initiate the procedure. When simultaneous attempts occur, each TCP receives a `SYN` segment which carries no acknowledgment after it has sent a `SYN`. Of course, the arrival of an old duplicate `SYN` segment can potentially make it appear, to the recipient, that a simultaneous connection initiation is in progress. Proper use of `RST` (reset) segments can disambiguate these cases. Although examples do not show connection synchronization using data-carrying segments, this is perfectly legitimate, as long as the receiving TCP doesn't deliver the data to the user until it is clear that the data are valid; that means the data must be buffered at the receiver until the connection reaches the `ESTABLISHED` state. The three-way handshake reduces the possibility of false connections. It is the implementation of a tradeoff between memory and messages to provide information for this checking.

The simplest three-way handshake is shown in the tabular format that follows this paragraph. This should be interpreted in the following way. Each line is numbered for reference purposes. Right arrows (—>) indicate departure of a TCP segment from TCP A to TCP B, or arrival of a segment at B from A. Left arrows (<—), indicate the reverse. Ellipses (...) indicate a segment which is still in the network (delayed). An "XXX" indicates a segment which is lost or rejected. Comments appear in parentheses. TCP states represent the state *after* the departure or arrival of the segment (whose contents are shown in the center of each line). Segment contents are shown in abbreviated form, with sequence number, control flags, and `ACK` field. Other fields such as window, addresses, lengths, and text have been left out for clarity.

TCP A	**TCP B**
1 CLOSED	LISTEN
2 SYN-SENT —> <SEQ = 100><CTL = SYN> —>	SYN-RECEIVED
3 ESTABLISHED <-<SEQ = 300><ACK = 101> <CTL = SYN, ACK>	SYN-RECEIVED
4 ESTABLISHED -><SEQ = 101><ACK = 301> <CTL = ACK>—>	ESTABLISHED
5 ESTABLISHED-><SEQ = 101><ACK = 301> <CTL = ACK><DATA>->	ESTABLISHED

In line 2 TCP A begins by sending a SYN segment indicating that it will use sequence numbers starting with sequence number 100. In line 3, TCP B sends a SYN and acknowledges the SYN it received from TCP A.

Note that the acknowledgment field indicates that TCP B is now expecting to hear sequence 101, acknowledging the SYN which occupied sequence 100. At line 4, TCP A responds with an empty segment containing an ACK for TCP B's SYN; and in line 5, TCP A sends some data. Note that the sequence number of the segment in line 5 is the same as in line 4 because the ACK does not occupy sequence number space (if it did, we would wind up ACKing ACKs!).

Simultaneous initiation is only slightly more complex. Each TCP cycles from CLOSED to SYN-SENT to SYN-RECEIVED to ESTABLISHED.

TCP A	**TCP B**
1 CLOSED	CLOSED
2 SYN-SENT—> <SEQ = 100><CTL = SYN>	...
3 SYN-RECEIVED <-<SEQ = 300><CTL = SYN>	<- SYN-SENT
4 ...<SEQ = 100><CTL = SYN>->	SYN-RECEIVED
5 SYN-RECEIVED -><SEQ = 100><ACK = 301>	<CTL = SYN,ACK>...
6 ESTABLISHED <-<SEQ = 300><ACK = 101>	<CTL = SYN,ACK><-SYN-RECEIVED
7 ...<SEQ = 101><ACK = 301><CTL = ACK> —>	ESTABLISHED

The principal reason for the three-way handshake is to prevent old duplicate connection initiations from causing confusion. To deal with this, a special control message, reset, has been devised. If the receiving TCP is in a nonsynchronized state (i.e., SYN-SENT, SYN-RECEIVED), it returns to LISTEN on receiving an acceptable reset. If the TCP is in one of the synchronized states (ESTABLISHED, FIN-WAIT-1, FIN-WAIT-2, CLOSE-WAIT, CLOSING, LAST-ACK, TIME-WAIT), it aborts the connection and informs its user.

Consider the following half-open connections:

	TCP A	TCP B
1	CLOSED	LISTEN
2	SYN-SENT-><SEQ = 100><CTL = SYN>	...
3	(duplicate)...<SEQ = 90><CTL = SYN>—>	SYN-RECEIVED>
4	SYN-SENT <-<SEQ = 300><ACK = 91>-<CTL = SYN,ACK><	SYN-RECEIVED
5	SYN-SENT-><SEQ = 91><CTL = RST>->	LISTEN
6	...<SEQ = 100><CTL = SYN>->	SYN-RECEIVED
7	SYN-SENT<-<SEQ = 400><ACK = 101>-<CTL = SYN,ACK>	SYN-RECEIVED
8	ESTABLISHED-><SEQ = 101><ACK = 401><CTL = ACK>->	ESTABLISHED

At line 3, an old duplicate SYN arrives at TCP B. TCP B cannot tell that this is an old duplicate, so it responds normally (line 4). TCP A detects that the ACK field is incorrect and returns a RST (reset) with its SEQ field selected to make the segment believable. TCP B, on receiving the RST, returns to the LISTEN state.

When the original SYN (pun intended) finally arrives at line 6, the synchronization proceeds normally. If the SYN at line 6 had arrived before the RST, a more complex exchange might have occurred with RSTs sent in both directions.

Half-open Connections and Other Anomalies

An established connection is said to be *half-open* if one of the TCPs has closed or aborted the connection at its end without the knowledge of the other, or if the two ends of the connection have become desynchronized because of a crash that resulted in loss of memory. Such connections will automatically become reset if an attempt is made to send data in either direction. However, half-open connections are expected to be unusual, and the recovery procedure is mildly involved.

If the connection no longer exists at site A, then an attempt by the user at site B to send any data on it will result in the site B TCP receiving a reset control message. Such a message indicates to the site B TCP that something is wrong, and it is expected to abort the connection.

Assume that two user processes—A and B—are communicating with one another when a crash occurs causing loss of memory to A's TCP. Depending on the operating system supporting A's TCP, it is likely that some error-recovery mechanism exists. When the TCP is up again, A is likely to start again from the beginning or from a recovery point. As a result, A will probably try to OPEN the connection again or try to SEND on the connection it believes open. In the latter case, it receives the error message "connection not open" from the local (A's) TCP. In an attempt to establish the connection, A's TCP will send a segment containing SYN.

After TCP A crashes, the user attempts to reopen the connection. TCP B, in the meantime, thinks the connection is open. Consider the following:

TCP A	TCP B
1 (CRASH)	(send 300<receive 100)
2 CLOSED	ESTABLISHED
3 SYN-SENT-><SEQ = 400><CTL = SYN>->	(??)
4 (!!)<-<SEQ = 300><ACK = 100><CTL = ACK>	<-ESTABLISHED
5 SYN-SENT-><SEQ = 100><CTL = RST>->	(Abort!!)
6 SYN-SENT	CLOSED
7 SYN-SENT-><SEQ = 400><CTL = SYN>	—>

When the SYN arrives at line 3, TCP B, which is in a synchronized state, and the incoming segment outside the window, responds with an acknowledgment indicating what sequence it next expects to hear (ACK = 100). TCP A sees that this segment does not acknowledge anything it sent and, being unsynchronized, sends a reset (RST) because it has detected a half-open connection. TCP B aborts at line 5. TCP A will continue to try to establish the connection; the problem is now reduced to the basic three-way handshake.

An interesting alternative case occurs when TCP A crashes and TCP B tries to send data on what it thinks is a synchronized connection. This is illustrated in the following example, where the data arriving at TCP A from TCP B (line 2) are unacceptable because no such connection exists, so TCP A sends a RST (the RST is acceptable, so TCP B processes it and aborts the connection):

TCP A	TCP B
1 (CRASH)	(end 300, receive 100)
2 (??) <-<SEQ = 300><ACK = 100><DATA = 10> <CTL = ACK>	<-ESTABLISHED
3 -><SEQ = 100><CTL = RST>->	(ABORT!!)

In the format following this paragraph two TCPs A and B with passive connections are waiting for SYN. An old duplicate arriving at TCP B (line 2) stirs B into action. A SYN-ACK is returned (line 3) and causes TCP A to generate a RST (the ACK in line 3 is not acceptable). TCP B accepts the reset and returns to its passive LISTEN state.

TCP A	TCP B
1 LISTEN	LISTEN
2 ...<SEQ = Z><CTL = SYN>->	SYN-RECEIVED
3 (??)<-SEQ = X><ACK = Z + 1> <CTL = SYN,ACK><-	SYN-RECEIVED
4 -><SEQ = Z + 1><CTL = RST>->	(return to LISTEN!)
5 LISTEN	LISTEN

The old duplicate SYN Initiates a reset on two passive sockets according to the rules for RST generation and processing described in the following paragraphs.

Reset Generation

As a general rule, reset (RST) must be sent whenever a segment arrives which apparently is not intended for the current connection. A reset must not be sent if it is not clear that this is the case. There are three groups of states:

1. If the connection does not exist (CLOSED), then a reset is sent in response to any incoming segment except another reset. In particular, SYNs addressed to a nonexistent connection are rejected by this means. If the incoming segment has an ACK field, the reset takes its sequence number from the ACK field of the segment, otherwise the reset has sequence number zero and the ACK field is set to the sum of the sequence number and segment length of the incoming segment. The connection remains in the CLOSED state.

2. If the connection is in any nonsynchronized state (LISTEN, SYN-SENT, SYN-RECEIVED), and the incoming segment acknowledges something not yet sent (the segment carries an unacceptable ACK), or if an incoming segment has a security level or compartment which does not exactly match the level and compartment requested for the connection, a reset is sent. If our SYN has not been acknowledged and the precedence level of the incoming segment is higher than the precedence level requested, then either raise the local precedence level (if allowed by the user and the system) or send a reset; or if the precedence level of the incoming segment is lower than the precedence level requested, then continue as if the precedence matched exactly (if the remote TCP cannot raise the precedence level to match ours, this will be detected in the next segment it sends, and the connection will be terminated then). If our SYN has been acknowledged (perhaps in this incoming segment), the precedence level of the incoming segment must match the local precedence level exactly, if it does not a reset must be sent. If the incoming segment has an ACK field, the reset takes its sequence number from the ACK field of the segment, otherwise the reset has sequence number zero and the ACK field is set to the sum of the sequence number and segment length of the incoming segment. The connection remains in the same state.

3. If the connection is in a synchronized state (ESTABLISHED, FIN-WAIT-1, FIN-WAIT-2, CLOSE-WAIT, CLOSING, LAST-ACK, TIME-WAIT), any unacceptable segment (out of window sequence number or unacceptable acknowledgment number) must elicit only an empty acknowledgment segment containing the current send-sequence number and an acknowledgment indicating the next sequence number expected to be received, and the connection remains in the same state. If an incoming segment has a security level, compartment, or precedence which does not exactly match the level, compartment, and precedence requested for the connection, a reset is sent and connection goes to the CLOSED state. The reset takes its sequence number from the ACK field of the incoming segment.

TCP Reset Processing

In all states except SYN-SENT, all reset (RST) segments are validated by checking their SEQ-fields. A reset is valid if its sequence number is in the window. In the SYN-SENT state (a RST received in response to an initial SYN), the RST is acceptable if the ACK field acknowledges the SYN.

The receiver of a RST first validates it, then changes state. If the receiver was in the LISTEN state, it ignores it. If the receiver was in SYN-RECEIVED state and had previously been in the LISTEN state, then the receiver returns to the LISTEN state, otherwise the receiver aborts the connection and goes to the CLOSED state. If the receiver was in any other state, it aborts the connection and advises the user and goes to the CLOSED state.

11.8 Closing a TCP Connection

CLOSE is an operation meaning “I have no more data to send.” The notion of closing a full-duplex connection is subject to ambiguous interpretation, of course, since it may not be obvious how to treat the receiving side of the connection. We have chosen to treat CLOSE in a simplex fashion. The user who CLOSEs may continue to RECEIVE until being told that the other side has CLOSED also. Thus, a program could initiate several SENDs followed by a CLOSE, and then continue to RECEIVE until signaled that a RECEIVE failed because the other side has CLOSED. We assume that the TCP will signal a user, even if no RECEIVEs are outstanding, that the other side has closed, so the user can terminate his side gracefully. A TCP will reliably deliver all buffers SENT before the connection was CLOSED so a user who expects no data in return need only wait to hear the connection was CLOSED successfully to know that all his data were received at the destination TCP. Users must keep reading connections they close for sending until the TCP says that there are no more data.

Essentially three cases exist: (1) the user initiates by telling the TCP to CLOSE the connection, (2) the remote TCP initiates by sending a FIN control signal, and (3) both users CLOSE simultaneously.

Case 1: Local user initiates the close. In this case, a FIN segment can be constructed and placed on the outgoing segment queue. No further SENDs from the user will be accepted by the TCP, and it enters the FIN-WAIT-1 state. RECEIVEs are allowed in this state. All segments preceding and including FIN will be retransmitted until acknowledged. When the other TCP has both acknowledged the FIN and sent a FIN of its own, the first TCP can ACK this FIN. Note that a TCP receiving a FIN will ACK but not send its own FIN until its user has CLOSED the connection also.

Case 2: TCP receives a FIN *from the network*. If an unsolicited FIN arrives from the network, the receiving TCP can ACK it and tell the user that the connection is closing. The user will respond with a CLOSE, on which the TCP can send a FIN to the other TCP after sending any remaining data. The TCP then waits until its own FIN is acknowledged, whereupon it deletes the connection. If an ACK is not forthcoming, after the user timeout the connection is aborted and the user is told.

Case 3: Both users CLOSE *simultaneously*. A simultaneous CLOSE by users at both ends of a connection causes FIN segments to be exchanged. When all segments preceding the FINs have been processed and acknowledged, each TCP can ACK the FIN it has received. Both will, upon receiving these ACKs, delete the connection.

	TCP A	**TCP B**
1	ESTABLISHED	ESTABLISHED
2	Close FIN-WAIT-1-><SEQ = 100><ACK = 300> < CTL = FIN,ACK>->CLOSE-WAIT	
3	FIN-WAIT-2<-<SEQ = 300><ACK = 101> <CTL = ACK><-CLOSE-WAIT	
4	Close TIME-WAIT<-<SEQ = 300><ACK = 101> <CTL = FIN,ACK><-LAST-ACK	
5	TIME-WAIT-><SEQ = 101><ACK = 301> <CTL = ACK>->	CLOSED
6	2 MSL	CLOSED

TCP A	TCP B
1 ESTABLISHED	ESTABLISHED
2 Close FIN-WAIT-1-><SEQ = 100><ACK = 300> <CTL = FIN,ACK...>FIN-WAIT-1<—<SEQ = 300> <ACK = 100><CTL = FIN,ACK>< . . . <SEQ = 100> <ACK = 300><CTL = FIN,ACK>->	Close
3 CLOSING-><SEQ = 101><ACK = 301><CTL = ACK> . . . <— <SEQ = 301><ACK = 101><CTL = ACK> <—...<SEQ = 101><ACK = 301><CTL = ACK>—>	CLOSING
4 TIME-WAIT 2 MSL CLOSED	TIME-WAIT 2 MSL CLOSED

Precedence and Security

The intent is that connection be allowed only between ports operating with exactly the same security and compartment values and at the higher precedence level requested by the two ports. The precedence and security parameters used in TCP are exactly those defined in the Internet Protocol (IP). Throughout this TCP specification the term *security/compartment* is intended to indicate the security parameters used in IP including security, compartment, user group, and handling restriction. A connection attempt with mismatched security/compartment values or a lower precedence value must be rejected by sending a reset. Rejecting a connection due to too low a precedence occurs only after an acknowledgment of the `SYN` has been received. TCP modules which operate only at the default value of precedence will still have to check the precedence of incoming segments and possibly raise the precedence level they use on the connection.

The security parameters may be used even in a nonsecure environment (the values would indicate unclassified data), thus hosts in nonsecure environments must be prepared to receive the security parameters, although they need not send them.

11.9 TCP and Data Communication

Once the connection is established, data are communicated by the exchange of segments. Because segments may be lost as a result of errors (checksum test failure), or network congestion, TCP uses retransmission (after a timeout) to ensure delivery of every segment. Duplicate segments may arrive following network or TCP retransmission. As discussed in the section on sequence numbers, the TCP performs certain tests on the sequence and acknowledgment numbers in the segments to verify their acceptability.

The data sender keeps track of the next sequence number to use in the variable `SND.NXT.` The receiver of data keeps track of the next sequence number to expect in the variable `RCV.NXT.` The sender of data keeps track of the oldest unacknowledged sequence number in the variable `SND.UNA.` If the data flow is momentarily idle and all data sent have been acknowledged, then the three variables will be equal. When the sender creates a segment and transmits it, the sender advances `SND.NXT.` When the receiver accepts a segment it advances `RCV.NXT` and sends an acknowledgment. When the data sender receives an acknowledgment, it advances `SND.UNA.` The extent to which the values of these variables differ is a measure of the delay in the communication.

The amount by which the variables are advanced is the length of the data in the segment. Note that once in the `ESTABLISHED` state all segments must carry current acknowledgment information. The CLOSE user call implies a push function, as does the `FIN` control flag in an incoming segment.

TCP Retransmission Timeout

Because of the variability of the networks that compose an internetwork system and the wide range of uses of TCP connections, the retransmission timeout must be dynamically determined. One procedure for determining a retransmission timeout is described here.

An example retransmission timeout procedure is to measure the elapsed time between sending a data octet with a particular sequence number and receiving an acknowledgment that covers that sequence number (segments sent do not have to match segments received). This measured elapsed time is the round-trip time (RTT). Next compute a smoothed round-trip time (SRTT) as

SRTT = (ALPHA * SRTT) 1 ((1-ALPHA) * RTT)

and, from this, compute the retransmission timeout (RTO) as

RTO = min[UBOUND,max[LBOUND,(BETA*SRTT)]]

where `UBOUND` is an upper bound on the timeout (e.g., 1 min), `LBOUND` is a lower bound on the timeout (e.g., 1 s), `ALPHA` is a smoothing factor (e.g., .8 to .9), and `BETA` is a delay variance factor (e.g., 1.3 to 2.0).

TCP Communication of Urgent Information

The objective of the TCP urgent mechanism is to allow the sending user to stimulate the receiving user to accept some urgent data and to permit the receiving TCP to indicate to the receiving user when all the currently known urgent data have been received by the user. This mechanism permits a point in the data stream to be designated as the end of urgent information. Whenever this point is in advance of the receive sequence number (`RCV.NXT`) at the receiving TCP, that TCP must tell the user to go into *urgent mode*; when the receive sequence number catches up to the urgent pointer, the TCP must tell the user to go into *normal mode*. If the urgent pointer is updated while the user is in urgent mode, the update will be invisible to the user.

This method employs an urgent field which is carried in all segments transmitted. The `URG` control flag indicates that the urgent field is meaningful and must be added to the segment sequence number to yield the urgent pointer. The absence of this flag indicates that there is no urgent data outstanding.

To send an urgent indication the user must also send at least one data octet. If the sending user also indicates a push, timely delivery of the urgent information to the destination process is enhanced.

Managing the Window

The window sent in each segment indicates the range of sequence numbers that the sender of the window (the data receiver) is currently prepared to accept. There is an assumption that this is related to the currently available data buffer space available for this connection.

Indicating a large window encourages transmissions. If more data arrive than can be accepted, they will be discarded. This will result in excessive retransmissions, adding unnecessarily to the load on the network and the TCPs. Indicating a small window may restrict the transmission of data to the point of introducing a round-trip delay between each new segment transmitted.

The mechanisms provided allow a TCP to advertise a large window and to subsequently advertise a much smaller window without having accepted that much data. This, so-called shrinking the window, is strongly discouraged. The robustness principle dictates that TCPs will not shrink the window themselves, but will be prepared for such behavior on the part of other TCPs.

The sending TCP must be prepared to accept from the user and send at least one octet of new data even if the send window is zero. The sending TCP must regularly retransmit to the receiving TCP even when the window is zero. Two minutes is recommended for the retransmission interval when the window is zero. This retransmission is essential to guarantee that when either TCP has a zero window, the reopening of the window will be reliably reported to the other.

When the receiving TCP has a zero window and a segment arrives, it must still send an acknowledgment showing its next expected sequence number and current window (zero). The sending TCP packages the data to be transmitted into segments which fit the current window, and may repackage segments on the retransmission queue. Such repackaging is not required, but may be helpful.

In a connection with a one-way data flow, the window information will be carried in acknowledgment segments that all have the same sequence number so there will be no way to reorder them if they arrive out of order. This is not a serious problem, but it will allow the window information to be on occasion temporarily based on old reports from the data receiver. A refinement to avoid this problem is to act on the window information from segments that carry the highest acknowledgment number (i.e., segments with acknowledgment number equal to or greater than the highest previously received).

Window management procedure has significant influence on communication performance. The following comments are suggestions:

1. Allocating a small window causes data to be transmitted in many small segments when better performance is achieved using fewer large segments.

2. Another suggestion for avoiding small windows is for the receiver to defer updating a window until the additional allocation is at least *X* percent of the maximum allocation possible for the connection (where *X* might be 20 to 40).

3. Another suggestion is for the sender to avoid sending small segments by waiting until the window is large enough before sending data. If the user signals a push function, then the data must be sent even if it is a small segment.

Acknowledgments should not be delayed, or unnecessary retransmis sions will result. One strategy would be to send an acknowledgment when a small segment arrives (without updating the window information), and then to send another acknowledgment with new window information when the window is larger. The segment sent to probe a zero window may also initiate a breakup of transmitted data into smaller and smaller segments. If a segment containing a single data octet sent to probe a zero window is accepted, it consumes one octet of the window now available. If the sending TCP simply sends as much as it can whenever the window is nonzero, the transmitted data will be broken into alternating big and small segments. As time passes, occasional pauses in the receiver making window allocation available will result in breaking the big segments into a small and not quite so big pair. And after a while the data transmission will be in mostly small segments.

TCP implementations need to actively attempt to combine small window allocations into larger windows, since the mechanisms for managing the window tend to lead to many small windows in the simplest minded implementations.

11.10 TCP Interfaces

There are, of course, two interfaces of concern: the user/TCP interface and the TCP/lower-level interface.

11.10.1 User/TCP Interface

The following functional description of user commands to the TCP is, at best, fictional, since every operating system will have different facilities. Consequently, we must warn readers that different TCP implementations may have different user interfaces. However, all TCPs must provide a certain minimum set of services to guarantee that all TCP implementations can support the same protocol hierarchy.

TCP User Commands

The following sections functionally characterize a user/TCP interface. The notation used is similar to most procedure or function calls in high-level languages, but this usage is not meant to rule out trap-type service calls (e.g., SVCs, UUOs, EMTs).

User commands described below specify the basic functions the TCP must perform to support interprocess communication. Individual implementations must define their own exact format, and may provide combinations or subsets of the basic functions in single calls. In particular, some implementations may wish to automatically OPEN a connection on the first SEND or RECEIVE issued by the user for a given connection.

In providing interprocess communication facilities, the TCP must not only accept commands but also return information to the processes it serves. The latter consists of (1) general information about a connection (e.g., interrupts, remote close, binding of unspecified foreign socket) and (2) replies to specific user commands indicating success or various types of failure.

Open

Format: OPEN (local port, foreign socket, active/passive [`,timeout`] [, `precedence`] [, `security/compartment`] [, `options`]) -. < `local connection name`.

We assume that the local TCP is aware of the identity of the processes it serves and will check the authority of the process to use the connection specified. Depending on the implementation of the TCP, the local network and TCP identifiers for the source address will be supplied by either the TCP or the lower-level protocol (e.g., IP). These considerations are the result of concern about security, to the extent that no TCP be able to masquerade as another one, and so on. Similarly, no process can masquerade as another without the collusion of the TCP.

If the active/passive flag is set to passive, then this is a call to LISTEN for an incoming connection. A passive open may have either a fully specified foreign socket to wait for a particular connection or an unspecified foreign socket to wait for any call. A fully specified passive call can be made active by the subsequent execution of a SEND.

A transmission control block (TCB) is created and partially filled in with data from the OPEN command parameters. On an active OPEN command, the TCP will begin the procedure to synchronize (i.e., establish) the connection at once. The timeout, if present, permits the caller to set up a timeout for all data submitted to TCP. If data are not successfully delivered to the destination within the timeout period, the TCP will abort the connection. The present global default is 5 min.

The TCP or some component of the operating system will verify the user’s authority to open a connection with the specified precedence or security/compartment. The absence of precedence or security/compartment specification in the OPEN call indicates that the default values must be used.

TCP will accept incoming requests as matching only if the security/compartment information is exactly the same and only if the precedence is equal to or higher than the precedence requested in the OPEN call.

The precedence for the connection is the higher of the values requested in the OPEN call and received from the incoming request, and fixed at that value for the life of the connection. Implementers may want to give the user control of this precedence negotiation. For example, the user might be allowed to specify that the precedence must be exactly matched, or that any attempt to raise the precedence be confirmed by the user.

A local connection name will be returned to the user by the TCP. The local connection name can then be used as a shorthand term for the connection defined by the <local socket, foreign socket= pair.

Send

Format: SEND (local connection name, buffer address, byte count, PUSH flag, URGENT flag [,`timeout`]).

This call causes the data contained in the indicated user buffer to be sent on the indicated connection. If the connection has not been opened, the SEND is considered an error. Some implementations may allow users to SEND first, in which case an automatic OPEN would be done. If the calling process is not authorized to use this connection, an error is returned.

If the PUSH flag is set, the data must be transmitted promptly to the receiver, and the PUSH bit will be set in the last TCP segment created from the buffer. If the PUSH flag is not set, the data may be combined with data from subsequent SENDs for transmission efficiency.

If the URGENT flag is set, segments sent to the destination TCP will have the urgent-pointer set. The receiving TCP will signal the urgent condition to the receiving process if the urgent pointer indicates that data preceding the urgent pointer have not been consumed by the receiving process. The purpose of signaling urgent is to stimulate the receiver to process the urgent data and to indicate to the receiver when all the currently known urgent data have been received. The number of times the sending user’s TCP signals urgent will not necessarily be equal to the number of times the receiving user will be notified of the presence of urgent data.

If no foreign socket was specified in the OPEN, but the connection is established (e.g., because a LISTENing connection has become specific due to the arrival of a foreign segment for the local socket), then the designated buffer is sent to the implied foreign socket. Users who make use of OPEN with an unspecified foreign socket can make use of SEND without ever explicitly knowing the foreign socket address.

However, if a SEND is attempted before the foreign socket becomes specified, an error will be returned. Users can use the STATUS call to determine the status of the connection. In some implementations the TCP may notify the user when an unspecified socket is bound.

If a timeout is specified, the current user timeout for this connection is changed to the new one. In the simplest implementation, SEND would not return control to the sending process until either the transmission was complete or the timeout had been exceeded. However, this simple method is both subject to deadlocks (e.g., both sides of the connection might try to do SENDs before doing any RECEIVEs) and offers poor performance, so it is not recommended. A more sophisticated implementation would return immediately to allow the process to run concurrently with network I/O, and, furthermore, to allow multiple SENDs to be in progress. Multiple SENDs are served in first-come, first-served order, so the TCP will queue those it cannot service immediately.

We have implicitly assumed an asynchronous user interface in which a SEND later elicits some kind of SIGNAL or pseudointerrupt from the serving TCP. An alternative is to return a response immediately. For instance, SENDs might return immediate local acknowledgment, even if the segment sent had not been acknowledged by the distant TCP. We could optimistically assume eventual success. If we are wrong, the connection will close anyway because of the timeout. In implementations of this kind (synchronous), there will still be some asynchronous signals, but these will deal with the connection itself, and not with specific segments or buffers.

In order for the process to distinguish among error or success indications for different SENDs, it might be appropriate for the buffer address to be returned along with the coded response to the SEND request. TCP-to-user signals are discussed below, indicating the information which should be returned to the calling process.

Receive

Format: RECEIVE (local connection name, buffer address, byte count) -< byte count, URGENT flag, PUSH flag.

This command allocates a receiving buffer associated with the specified connection. If no OPEN precedes this command or the calling process is not authorized to use this connection, an error is returned. In the simplest implementation, control would not return to the calling program until either the buffer was filled, or some error occurred, but this scheme is highly subject to deadlocks. A more sophisticated implementation would permit several RECEIVEs to be outstanding at once. These would be filled as segments arrive. This strategy permits increased throughput at the cost of a more elaborate scheme (possibly asynchronous) to notify the calling program that a PUSH has been seen or a buffer filled.

If enough data arrive to fill the buffer before a PUSH is seen, the PUSH flag will not be set in the response to the RECEIVE. The buffer will be filled with as much data as it can hold. If a PUSH is seen before the buffer is filled, the buffer will be returned partially filled and PUSH indicated. If there is urgent data, the user will have been informed as soon as it arrived via a TCP-to-user signal. The receiving user should thus be in *urgent mode*. If the URGENT flag is on, additional urgent data remains; if the URGENT flag is off, this call to RECEIVE has returned all the urgent data, and the user may now leave urgent mode. Note that data following the urgent pointer (nonurgent data) cannot be delivered to the user in the same buffer with preceding urgent data unless the boundary is clearly marked for the user.

To distinguish among several outstanding RECEIVEs and to accommodate a buffer that is not completely filled, the return code is accompanied by both a buffer pointer and a byte count indicating the actual length of the data received.

Alternative implementations of RECEIVE might have the TCP allocate buffer storage, or the TCP might share a ring buffer with the user.

Close

Format: CLOSE (local connection name). This command causes the connection specified to be closed. If the connection is not open or the calling process is not authorized to use this connection, an error is returned. Closing connections is intended to be a graceful operation in the sense that outstanding SENDs will be transmitted (and retransmitted), as flow control permits, until all have been serviced. Thus, it should be acceptable to make several SEND calls, followed by a PUSH, and expect all the data to be sent to the destination. It should also be clear that users should continue to RECEIVE on CLOSING connections, since the other side may be trying to transmit the last of its data. Thus, CLOSE means “I have no more to send” but does not mean “I will not receive any more.” It may happen (if the user level protocol is not well thought out) that the closing side is unable to get rid of all its data before timing out. In this event, CLOSE turns into ABORT, and the closing TCP gives up.

The user may CLOSE the connection at any time on his own initiative, or in response to various prompts from the TCP (e.g., remote close executed, transmission timeout exceeded, destination inaccessible). Because closing a connection requires communication with the foreign TCP, connections may remain in the closing state for a short time. Attempts to reopen the connection before the TCP replies to the CLOSE command will result in error responses. Close also implies a push function.

Status

Format: STATUS (local connection name) -> status data. This is an implementation-dependent user command and could be excluded without adverse effect. Information returned would typically come from the TCB associated with the connection.

This command returns a data block containing the following information: local socket, foreign socket, local connection name, receive window, send window, connection state, number of buffers awaiting acknowledgment, number of buffers pending receipt, urgent state, precedence, security/compartment, and transmission timeout.

Depending on the state of the connection, or on the implementation itself, some of this information may not be available or meaningful. If the calling process is not authorized to use this connection, an error is returned. This prevents unauthorized processes from gaining information about a connection.

Abort

Format: ABORT (local connection name). This command causes all pending SENDs and RECEIVEs to be aborted, the TCB to be removed, and a special RESET message to be sent to the TCP on the other side of the connection.

Depending on the implementation, users may receive abort indica tions for each outstanding SEND or RECEIVE, or may simply receive an ABORT-acknowledgment.

TCP-to-User Messages

It is assumed that the operating system environment provides a means for the TCP to asynchronously signal the user program. When the TCP does signal a user program, certain information is passed to the user. Often in the specification the information will be an error message. In other cases there will be information relating to the completion of processing a SEND or RECEIVE or other user call. The following information is provided:

Local connection name	Always
Response string	Always
Buffer address	Send and receive
Byte count (counts bytes received)	Receive
Push flag	Receive
Urgent flag	Receive

11.10.2 TCP/Lower-Level Interface

The TCP calls on a lower-level protocol module to actually send and receive information over a network. One case is that of the ARPA internetwork system where the lower-level module is the Internet Protocol (IP). If the lower-level protocol is IP it provides arguments for a type of service and for a time to live. TCP uses the following settings for these parameters: (1) *type of service* = precedence, routine; delay, normal; throughput, normal; Reliability, normal; or 00000000; (2) *time to live* = 1 min, or 00111100. The assumed maximum segment lifetime is 2 min.

Here we explicitly ask that a segment be destroyed if it cannot be delivered by the Internet system within 1 min.

If the lower level is IP (or other protocol that provides this feature) and source routing is used, the interface must allow the route information to be communicated. This is especially important to ensure that the source and destination addresses used in the TCP checksum are the originating source and ultimate destination. It is also important to preserve the return route to answer connection requests.

Any lower-level protocol will have to provide the source address, destination address, and protocol fields, and some way to determine the “TCP length,” both to provide the functional equivalent service of IP and to be used in the TCP checksum.

11.11 TCP Event Processing

The processing depicted in this section is an example of one possible implementation. Other implementations may have slightly different processing sequences. The activity of the TCP can be characterized as responding to events. The events that occur can be cast into three categories: user calls, arriving segments, and timeouts. The TCP does processing in response to each of the events. In many cases the processing required depends on the state of the connection.

Events that occur are *user calls*—OPEN, SEND, RECEIVE, CLOSE, ABORT, and STATUS; *arriving segments*—SEGMENT ARRIVES; *timeouts*—USER TIMEOUT, RETRANSMISSION TIMEOUT, and TIME-WAIT TIMEOUT.

The model of the TCP/user interface is that user commands receive an immediate return and possibly a delayed response via an event or pseudointerrupt. In the following descriptions, the term *signal* means cause a delayed response. Error responses are given as character strings. For example, user commands referencing connections that do not exist receive *error: connection not open*.

Please note in the following that all arithmetic on sequence numbers, acknowledgment numbers, windows, et cetera, is modulo 2**32, the size of the sequence number space. Also note that “ = <” means less than or equal to (modulo 2**32).

A natural way to think about processing incoming segments is to imagine that they are first tested for proper sequence number (i.e., that their contents lie in the range of the expected receive *window* in the sequence number space) and then that they are generally queued and processed in sequence number order.

When a segment overlaps other already received segments, we reconstruct the segment to contain just the new data, and adjust the header fields to be consistent.

Note that if no state change is mentioned, the TCP stays in the same state.

OPEN call

1. `CLOSED STATE` (i.e., TCB does not exist). Create a new transmission control block (TCB) to hold connection state information. Fill in local socket identifier, foreign socket, precedence, security/compartment, and user timeout information. Note that some parts of the foreign socket may be unspecified in a passive OPEN and are to be filled in by the parameters of the incoming `SYN` segment. Verify the security and precedence requested are allowed for this user, if not return “error: precedence not allowed” or “error: security/compartment not allowed.” If passive, enter the LISTEN state and return. If active and the foreign socket is unspecified, return “error: foreign socket unspecified”; if active and the foreign socket is specified, issue a `SYN` segment. An initial send sequence number (ISS) is selected. A `SYN` segment of the form <SEQ = ISS><CTL = SYN> is sent. Set `SND.UNA` to `ISS, SND.NXT` to `ISS` + 1, enter `SYN-SENT` state, and return. If the caller does not have access to the local socket specified, return “error: connection illegal for this process.” If there is no room to create a new connection, return “error: insufficient resources."

2. `LISTEN STATE.` If active and the foreign socket is specified, then change the connection from passive to active, select an `ISS.` Send a `SYN` segment, set `SND.UNA` to `ISS, SND.NXT` to `ISS` + 1. Enter `SYN-SENT` state. Data associated with SEND may be sent with `SYN` segment or queued for transmission after entering `ESTABLISHED` state. The urgent bit if requested in the command must be sent with the data segments sent as a result of this command. If there is no room to queue the request, respond with “error: insufficient resources.” If foreign socket was not specified, then return “error: foreign socket unspecified.” For `SYN-SENT STATE, SYN-RECEIVED STATE, ESTABLISHED STATE, FIN-WAIT-1 STATE, FIN-WAIT-2 STATE, CLOSE-WAIT STATE, CLOSING STATE, LAST-ACK STATE,` and `TIME-WAIT STATE,` return “error: connection already exists.”

SEND call

1. `CLOSED STATE` (i.e., TCB does not exist). If the user does not have access to such a connection, then return “error: connection illegal for this process”; otherwise, return “error: connection does not exist.”

2. `LISTEN STATE.` If the foreign socket is specified, then change the connection from passive to active, select an `ISS.` Send a `SYN` segment, set `SND.UNA` to `ISS, SND.NXT` to `ISS` + 1. Enter `SYN-SENT` state. Data associated with SEND may be sent with `SYN` segment or queued for transmission after entering `ESTABLISHED` state. The urgent bit if requested in the command must be sent with the data segments sent as a result of this command. If there is no room to queue the request, respond with “error: insufficient resources.” If foreign socket was not specified, then return “error: foreign socket unspecified.” For `SYN-SENT STATE` and `SYN-RECEIVED STATE` queue the data for transmission after entering `ESTABLISHED` state. If no space to queue, respond with “error: insufficient resources,” for `ESTABLISHED STATE` and `CLOSE-WAIT STATE,` segment the buffer and send it with a piggybacked acknowledgment (acknowledgment value = `RCV.NXT`). If there is insufficient space to remember this buffer, simply return “error: insufficient resources.” If the urgent flag is set, then `SND.UP <-` SND.NXT-1 and set the urgent pointer in the outgoing segments. For `FIN-WAIT-1 STATE, FIN-WAIT-2 STATE, CLOSING STATE, LAST-ACK STATE,` and `TIME-WAIT STATE,` return “error: connection closing” and do not service request.

RECEIVE call

1. `CLOSED STATE` (i.e., TCB does not exist). If the user does not have access to such a connection, return "error: connection illegal for this process." Otherwise return "error: connection does not exist." For `LISTEN STATE, SYN-SENT STATE,` and `SYN-RECEIVED STATE,` queue for processing after entering `ESTABLISHED` state. If there is no room to queue this request, respond with "error: insufficient resources." For `ESTABLISHED STATE, FIN-WAIT-1 STATE,` and `FIN-WAIT-2 STATE,` if insufficient incoming segments are queued to satisfy the request, queue the request. If there is no queue space to remember the RECEIVE, respond with "error: insufficient resources." Reassemble queued incoming segments into receive buffer and return to user. Mark "push seen" (PUSH) if this is the case. If `RCV.UP` is in advance of the data currently being passed to the user, notify the user of the presence of urgent data. When the TCP takes responsibility for delivering data to the user, that fact must be communicated to the sender via an acknowledgment. The formation of such an acknowledgment is described below in the discussion of processing an incoming segment.

2. `CLOSE-WAIT STATE.` Since the remote side has already sent `FIN,` RECEIVEs must be satisfied by text already on hand, but not yet delivered to the user. If no text is awaiting delivery, the RECEIVE will get a "error: connection closing" response. Otherwise, any remaining text can be used to satisfy the RECEIVE. For `CLOSING STATE, LAST-ACK STATE,` and `TIME-WAIT STATE,` return "error: connection closing."

CLOSE call

1. `CLOSED STATE` (i.e., TCB does not exist). If the user does not have access to such a connection, return "error: connection illegal for this process." Otherwise, return "error: connection does not exist."

2. `LISTEN STATE.` Any outstanding RECEIVEs are returned with "error: closing" responses. Delete TCB, enter `CLOSED` state, and return. For `SYN-SENT STATE,` delete the TCB and return "error: closing" responses to any queued SENDs, or RECEIVEs. For `SYN-RECEIVED STATE,` if no SENDs have been issued and there is no pending data to send, then form a `FIN` segment and send it, and enter `FIN-WAIT-1` state; otherwise queue for processing after entering `ESTABLISHED` state. Queue `ESTABLISHED STATE` until all preceding sends have been segmented, then form a `FIN` segment and send it. In any case, enter `FIN-WAIT-1` state. Then `FIN-WAIT-1 STATE` and `FIN-WAIT-2 STATE`—strictly speaking, this is an error and should receive an "error: connection closing" response. An "ok" response would be acceptable, too, as long as a second `FIN` is not emitted (the first `FIN` may be retransmitted, though). Queue the `CLOSE-WAIT STATE` request until all preceding SENDs have been segmented; then send a `FIN` segment, enter `CLOSING` state. Then for `CLOSING STATE, LAST-ACK STATE,` and `TIME-WAIT STATE,` respond with "error: connection closing."

ABORT call

1. `CLOSED STATE` (i.e., TCB does not exist). If the user should not have access to such a connection, return "error: connection illegal for this process." Otherwise return "error: connection does not exist."

2. `LISTEN STATE.` Any outstanding RECEIVEs should be returned with "error: connection reset" responses. Delete TCB, enter `CLOSED` state, and return.

3. `SYN-SENT STATE.` All queued SENDs and RECEIVEs should be given "connection reset" notification; delete the TCB, enter `CLOSED` state, and return.

4. `SYN-RECEIVED STATE, ESTABLISHED STATE, FIN-WAIT-1 STATE, FIN-WAIT-2 STATE,` and `CLOSE-WAIT STATE.` Send a reset segment: <SEQ = SND.NXT><CTL = RST>

All queued SENDs and RECEIVEs should be given “connection reset” notification; all segments queued for transmission (except for the RST formed above) or retransmission should be flushed, delete the TCB, enter CLOSED state, and return.

5. CLOSING STATE, LAST-ACK STATE, and TIME-WAIT STATE. Respond with OK and delete the TCB, enter closed state, and return.

STATUS call

1. CLOSED STATE (i.e., TCB does not exist). If the user should not have access to such a connection, return “error: connection illegal for this process.” Otherwise return “error: connection does not exist,” for LISTEN STATE, return “state = LISTEN” and the TCB pointer. For SYN-SENT STATE, return “state = SYN-SENT” and the TCB pointer. For SYN-RECEIVED STATE, return “state = SYN-RECEIVED” and the TCB pointer. For ESTABLISHED STATE, return “state = ESTABLISHED” and the TCB pointer. For FIN-WAIT-1 STATE, return “state = FIN-WAIT-1” and the TCB pointer. For FIN-WAIT-2 STATE, return “state = FIN-WAIT-2” and the TCB pointer. For CLOSE-WAIT STATE, return “state = CLOSE-WAIT” and the TCB pointer. For CLOSING STATE, return “state = CLOSING” and the TCB pointer. For LAST-ACK STATE, return “state = LAST-ACK” and the TCB pointer. For TIME-WAIT STATE, return “state = TIME-WAIT” and the TCB pointer. For SEGMENT ARRIVES, if the state is CLOSED (i.e., TCB does not exist), then all data in the incoming segment are discarded. An incoming segment containing a RST is discarded. An incoming segment not containing a RST causes a RST to be sent in response. The acknowledgment and sequence field values are selected to make the reset sequence acceptable to the TCP that sent the offending segment. If the ACK bit is off, sequence number zero is used; <SEQ = 0><ACK = SEG.SEQ + SEG.LEN><CTL = RST,ACK>. If the ACK bit is on, then <SEQ = SEG.ACK><CTL = RST> and return. If the state is LISTEN, then first check for a RST. An incoming RST should be ignored. Return. Then check for an ACK. Any acknowledgment is bad if it arrives on a connection still in the LISTEN state. An acceptable reset segment should be formed for any arriving ACK-bearing segment. The RST should be formatted as follows: <SEQ = SEG.ACK><CTL = RST>, then return, then check for a SYN. If the SYN bit is set, check the security. If the security/compartment on the incoming segment does not exactly match the security/compartment in the TCB, then send a reset and return.

2. ARRIVES: <SEQ = SEG.ACK><CTL = RST>. If the SEG.PRC is greater than the TCB.PRC, then, if allowed by the user and the system, set TCB.PRC<-SEG.PRC; if not allowed, send a reset and return. Then <SEQ = SEG.ACK><CTL = RST>. If the SEG.PRC is less than the TCB.PRC, then continue. Set RCV.NXT to SEG.SEQ + 1, IRS is set to SEG.SEQ and any other control or text should be queued for processing later. ISS should be selected and a SYN segment sent of the form <SEQ = ISS><ACK = RCV.NXT><CTL = SYN,ACK>. Then SND.NXT is set to ISS + 1 and SND.UNA to ISS. The connection state should be changed to SYN-RECEIVED. Note that any other incoming control or data (combined with SYN) will be processed in the SYN-RECEIVED state, but processing of SYN and ACK should not be repeated. If the listen was not fully specified (i.e., the foreign socket was not fully specified), then the unspecified fields should be filled in now. This is followed by *other text or control*. Any other control or text-bearing segment (not containing SYN) must have an ACK and thus would be discarded by the ACK processing. An incoming RST segment could not be valid, since it could not have been sent in response to anything sent by this incarnation of the connection. So you are unlikely to get here, but if you do, drop the segment, and return. If the state is SYN-SENT, then first check the ACK bit.

If the ACK bit is set If SEG.ACK = < ISS, or SEG.ACK < SND.NXT, send a reset (unless the RST bit is set, if so drop the segment and return) <SEQ = SEG.ACK><CTL = RST> and discard the segment. Return. If SND.UNA = < SEG.ACK = < SND.NXT, then the ACK is acceptable. Then check the RST bit.

3. SEGMENT ARRIVES. If the RST bit is set and if the ACK was acceptable, then signal the user “error: connection reset,” drop the segment, enter CLOSED state, delete TCB, and return. Otherwise (no ACK) drop the segment and return. Then check the security and precedence. If the security/compartment in the segment does not exactly match the security/compartment in the TCB, send a reset. If there is an ACK, then <SEQ = SEG.ACK><CTL = RST<. Otherwise <SEQ = 0><ACK = SEG.SEQ + SEG.LEN><CTL = RST,ACK>. If there is an ACK, the precedence in the segment must match the precedence in the TCB, if not, send a reset <SEQ = SEG.ACK><CTL = RST>.

If there is no ACK, and if the precedence in the segment is higher than the precedence in the TCB, then, if allowed by the user and the system, raise the precedence in the TCB to that in the segment, if not allowed to raise the precedence then send a reset.

<SEQ = 0><ACK = SEG.SEQ + SEG.LEN><CTL = RST,ACK>

If the precedence in the segment is lower than the precedence in the TCB, continue. If a reset was sent, discard the segment and return. Then check the SYN bit. This step should be reached only if the ACK is OK, or there is no ACK, and if the segment did not contain a RST. If the SYN bit is on and the security/compartment and precedence SEGMENT ARRIVES are acceptable, then RCV.NXT is set to SEG.SEQ + 1, IRS is set to SEG.SEQ. SND.UNA should be advanced to equal SEG.ACK (if there is an ACK), and any segments on the retransmission queue which are thereby acknowledged should be removed. If SND.UNA < ISS (our SYN has been ACKed), change the connection state to ESTABLISHED, form an ACK segment

<SEQ = SND.NXT><ACK = RCV.NXT><CTL = ACK>

and send it. Data or controls which were queued for transmission may be included. If there are other controls or text in the segment then continue processing at the sixth step below where the URG bit is checked, otherwise return. Otherwise enter SYN-RECEIVED, form a SYN,ACK segment

<SEQ = ISS><ACK = RCV.NXT><CTL = SYN,ACK>

and send it. If there are other controls or text in the segment, queue them for processing after the ESTABLISHED state has been reached, return. Then, if neither of the SYN or RST bits is set, drop the segment and return. Otherwise, first check sequence number

SYN-RECEIVED STATE
ESTABLISHED STATE
FIN-WAIT-1 STATE
FIN-WAIT-2 STATE
CLOSE-WAIT STATE
CLOSING STATE
LAST-ACK STATE
TIME-WAIT STATE

Segments are processed in sequence. Initial tests on arrival are used to discard old duplicates, but further processing is done in SEG.SEQ order. If a segment’s contents straddle the boundary between old and new, only the new parts should be processed.

There are four cases for the acceptability test for an incoming segment:

Segment Length	Receive Window	Test
0	0	SEG.SEQ = RCV.NXT
0	>0	RCV.NXT = < SEG.SEQ < RCV.NXT + RCV.WND
>0	0	Not acceptable
>0	>0	RCV.NXT = < SEG.SEQ < RCV.NXT + RCV.WND or RCV.NXT = < SEG.SEQ + SEG.LEN-1 < RCV.NXT + RCV.WND

If the RCV.WND is zero, no segments will be acceptable, but special allowance should be made to accept valid ACKs, URGs, and RSTs. If an incoming segment is not acceptable, an acknowledgment should be sent in reply (unless the RST bit is set, if so drop the segment and return):

<SEQ = SND.NXT><ACK = RCV.NXT><CTL = ACK>

After sending the acknowledgment, drop the unacceptable segment and return.

In the following it is assumed that the segment is the idealized segment that begins at RCV.NXT and does not exceed the window. One could tailor actual segments to fit this assumption by trimming off any portions that lie outside the window (including SYN and FIN), and only processing further if the segment then begins at RCV.NXT. Segments with higher beginning sequence numbers may be held for later processing. Then check the RST bit, SYN-RECEIVED STATE, if the RST bit is set. If this connection was initiated with a passive open (i.e., came from the LISTEN state), then return this connection to LISTEN state and return. The user need not be informed. If this connection was initiated with an active OPEN (i.e., came from SYN-SENT state), then the connection was refused; signal the user “connection refused.” In either case, all segments on the retransmission queue should be removed. And in the active OPEN case, enter the CLOSED state and delete the TCB, and return.

ESTABLISHED
FIN-WAIT-1
FIN-WAIT-2
CLOSE-WAIT

If the RST bit is set then, any outstanding RECEIVEs and SEND should receive “reset” responses. All segment queues should be flushed. Users should also receive an unsolicited general “connection reset” signal. Enter the CLOSED state, delete the TCB, and return.

CLOSING STATE
LAST-ACK STATE
TIME-WAIT

If the RST bit is set, then enter the CLOSED state, delete the TCB, and return.

1. SEGMENT ARRIVES. Check security and precedence.

2. SYN-RECEIVED. If the security/compartment and precedence in the segment do not exactly match the security/compartment and precedence in the TCB, then send a reset, and return.

3. ESTABLISHED STATE. If the security/compartment and precedence in the segment do not exactly match the security/compartment and precedence in the TCB, then send a reset; any outstanding RECEIVEs and SEND should receive "reset" responses. All segment queues should be flushed. Users should also receive an unsolicited general "connection reset" signal. Enter the CLOSED state, delete the TCB, and return. This check is placed following the sequence check to prevent a segment from an old connection between these ports with a different security or precedence from causing an abort of the current connection.

Then check the SYN bit:

SYN-RECEIVED
ESTABLISHED STATE
FIN-WAIT STATE-1
FIN-WAIT STATE-2
CLOSE-WAIT STATE
CLOSING STATE
LAST-ACK STATE
TIME-WAIT STATE

If the SYN is in the window, it is an error, then send a reset. Any outstanding RECEIVEs and SEND should receive "reset" responses, all segment queues should be flushed, the user should also receive an unsolicited general "connection reset" signal, enter the CLOSED state, delete the TCB, and return.

If the SYN is not in the window, this step would not be reached and an ACK would have been sent in the first step (sequence number check). Then check the ACK field; if the ACK bit is off, drop the segment and return if the ACK bit is on SYN-RECEIVED STATE. If SND.UNA = < SEG.ACK = < SND.NXT, then enter ESTABLISHED state and continue processing. If the segment acknowledgment is not acceptable, form a reset segment <SEQ = SEG.ACK><CTL = RST> and send it.

1. ESTABLISHED STATE. If SND.UNA < SEG.ACK = < SND.NXT, then set SND.UNA <- SEG.ACK. Any segments on the retransmission queue which are thereby entirely acknowledged are removed. Users should receive positive acknowledgments for buffers which have been SENT and fully acknowledged (i.e., SEND buffer should be returned with OK response). If the ACK is a duplicate (SEG.ACK < SND.UNA), it can be ignored. If the ACK acks something not yet sent (SEG.ACK > SND.NXT), then send an ACK, drop the segment, and return. If SND.UNA < SEG.ACK = < SND.NXT, the send window should be updated. If (SND.WL1 < SEG.SEQ or (SND.WL1 = SEG.SEQ and SND.WL2 = < SEG.ACK)), set SND.WND <- SEG.WND, set SND.WL1 <- SEG.SEQ, and set SND.WL2 <- SEG.ACK. SND.WND is an offset from SND.UNA; then SND.WL1 records the sequence number of the last segment used to update SND.WND, and then SND.WL2 records the acknowledgment number of the last segment used to update SND.WND. The check here prevents using old segments to update the window. For FIN-WAIT-1 STATE, in addition to the processing for the ESTABLISHED state, if our FIN is now acknowledged, then enter FIN-WAIT-2 and continue processing in that state. For FIN-WAIT-2 state in addition to the processing for the ESTABLISHED state, if the retransmission queue is empty, the user's close can be acknowledged (OK), but do not delete the TCB.

2. CLOSE-WAIT STATE. Do the same processing as for the ESTABLISHED state.

3. CLOSING STATE. In addition to the processing for the ESTABLISHED state, if the ACK acknowledges our FIN, then enter the TIME-WAIT state; otherwise ignore the segment.

4. LAST-ACK STATE. The only thing that can arrive in this state is an acknowledgment of our FIN. If our FIN is now acknowledged, delete the TCB, enter the CLOSED state, and return.

5. TIME-WAIT STATE. The only thing that can arrive in this state is a retransmission of the remote FIN. Acknowledge it, and restart the 2 MSL timeout. Then check the URG bit.

ESTABLISHED STATE
FIN-WAIT-1 STATE
FIN-WAIT-2 STATE

If the URG bit is set, RCV.UP <- max(RCV.UP,SEG.UP), and signal the user that the remote side has urgent data if the urgent pointer (RCV.UP) is in advance of the data consumed. If the user has already been signaled (or is still in the "urgent mode") for this continuous sequence of urgent data, do not signal the user again.

CLOSE-WAIT STATE
CLOSING STATE
LAST-ACK STATE
TIME-WAIT

This should not occur, since a FIN has been received from the remote side. Ignore the URG. Then, process the segment text,

ESTABLISHED STATE
FIN-WAIT-1 STATE
FIN-WAIT-2 STATE

Once in the ESTABLISHED state, it is possible to deliver segment text to user RECEIVE buffers. Text from segments can be moved into buffers until either the buffer is full or the segment is empty. If the segment empties and carries a push flag, then the user is informed, when the buffer is returned, that a PUSH has been received.

When the TCP takes responsibility for delivering the data to the user, it must also acknowledge the receipt of the data. Once the TCP takes responsibility for the data it advances RCV.NXT over the data accepted, it adjusts RCV.WND as appropriate to the current buffer availability. The total of RCV.NXT and RCV.WND should not be reduced.

Send an acknowledgment of the form

<SEQ = SND.NXT><ACK = RCV.NXT><CTL = ACK>

This acknowledgment should be piggybacked on a segment being transmitted if possible without incurring undue delay.

CLOSE-WAIT STATE
CLOSING STATE
LAST-ACK STATE
TIME-WAIT STATE

This should not occur, since a FIN has been received from the remote side. Ignore the segment text. Then, check the FIN bit. Do not process the FIN if the state is CLOSED, LISTEN, or SYN-SENT since the SEG.SEQ cannot be validated; drop the segment and return.

If the FIN bit is set, signal the user "connection closing" and return any pending RECEIVEs with the same message, advance RCV.NXT over the FIN, and send an acknowledgment for the FIN. Note that FIN implies PUSH for any segment text not yet delivered to the user.

SYN-RECEIVED STATE
ESTABLISHED STATE

Enter the CLOSE-WAIT state: FIN-WAIT-1 STATE.

If our FIN has been ACKed (perhaps in this segment), then enter TIME-WAIT, start the time-wait timer, turn off the other timers; otherwise enter the CLOSING state.

1. FIN-WAIT-2 STATE. Enter the TIME-WAIT state. Start the time-wait timer, and turn off the other timers.

2. CLOSE-WAIT STATE. Remain in the CLOSE-WAIT state.

3. CLOSING STATE. Remain in the CLOSING state.

4. LAST-ACK STATE. Remain in the LAST-ACK state.

5. TIME-WAIT STATE. Remain in the TIME-WAIT state. Restart the 2 MSL time-wait timeout and return.

6. USER TIMEOUT. For any state if the user timeout expires, flush all queues, signal the user error: connection aborted because of user timeout in general and for any outstanding calls, delete the TCB, enter the CLOSED state, and return.

7. RETRANSMISSION TIMEOUT. For any state, if the retransmission timeout expires on a segment in the retransmission queue, send the segment at the front of the retransmission queue again, reinitialize the retransmission timer, and return.

8. TIME-WAIT TIMEOUT. If the time-wait timeout expires on a connection delete the TCB, enter the CLOSED state and return.

11.12 TCP Glossary

ACK A control bit (acknowledge) occupying no sequence space, which indicates that the acknowledgment field of this segment specifies the next sequence number the sender of this segment is expecting to receive, hence acknowledging receipt of all previous sequence numbers.

ARPAnet message The unit of transmission between a host and an IMP in the ARPAnet. The maximum size is about 1012 octets (8096 bits).

ARPAnet packet A unit of transmission used internally in the ARPAnet between IMPs. The maximum size is about 126 octets (1008 bits).

connection A logical communication path identified by a pair of sockets.

datagram A message sent in a packet switched computer communications network.

destination address The destination address, usually the network and host identifiers.

FIN A control bit (finis) occupying one sequence number, which indicates that the sender will send no more data or control occupying sequence space.

fragment A portion of a logical unit of data, in particular an Internet fragment is a portion of an Internet datagram.

FTP A file-transfer protocol.

header Control information at the beginning of a message, segment, fragment, packet, or block of data.

host A computer; in particular, a source or destination of messages from the point of view of the communication network.

identification An Internet Protocol field. This identifying value assigned by the sender aids in assembling the fragments of a datagram.

IMP The Interface Message Processor, the packet switch of the ARPAnet.

Internet address A source or destination address specific to the host level.

Internet datagram The unit of data exchanged between an Internet module and the higher-level protocol together with the Internet header.

Internet fragment A portion of the data of an Internet datagram with an Internet header.

IP Internet Protocol.

IRS The initial receive sequence number. The first sequence number used by the sender on a connection.

ISN The initial sequence number. The first sequence number used on a connection, (either ISS or IRS). Selected on a clock-based procedure.

ISS The initial send sequence number. The first sequence number used by the sender on a connection.

leader Control information at the beginning of a message or block of data. In particular, in the ARPAnet, the control information on an ARPAnet message at the host-IMP interface.

left sequence This is the next sequence number to be acknowledged by the data receiving TCP (or the lowest currently unacknowledged sequence number) and is sometimes referred to as the left edge of the send window.

local packet The unit of transmission within a local network.

module An implementation, usually in software, of a protocol or other procedure.

MSL Maximum segment lifetime, the time a TCP segment can exist in the internetwork system. Arbitrarily defined to be 2 min.

octet An 8-bit byte.

options A field may contain several options, and each option may be several octets in length. The options are used primarily in testing situations; for example, to carry timestamps. Both the Internet Protocol and TCP provide for options fields.

packet A package of data with a header which may or may not be logically complete. More often a physical packaging than a logical packaging of data.

port The portion of a socket that specifies which logical input or output channel of a process is associated with the data.

process A program in execution. A source or destination of data from the point of view of the TCP or other host-to-host protocol.

`PUSH` A control bit occupying no sequence space, indicating that this segment contains data that must be pushed through to the receiving user.

`RCV.NXT` Receive next sequence number.

`RCV.UP` Receive urgent pointer.

`RCV.WND` Receive window.

Receive next sequence number This is the next sequence number the local TCP is expecting to receive.

Receive window This represents the sequence numbers the local (receiving) TCP is willing to receive. Thus, the local TCP considers that segments overlapping the range `RCV.NXT` to `RCV.NXT`+`RCV.WND` – 1 carry acceptable data or control. Segments containing sequence numbers entirely outside of this range are considered duplicates and discarded.

`RST` A control bit (reset), occupying no sequence space, indicating that the receiver should delete the connection without further interaction. The receiver can determine, on the basis of the sequence number and acknowledgment fields of the Incoming segment, whether it should honor the reset command or ignore it. In no case does receipt of a segment containing `RST` give rise to a `RST` in response.

RTP Real time protocol: a host-to-host protocol for communication of time-critical information.

`SEG.ACK` Segment acknowledgment.

`SEG.LEN` Segment length.

`SEG.PRC` Segment precedence value.

`SEG.SEQ` Segment sequence.

`SEG.UP` Segment urgent-pointer field.

`SEG.WND` Segment window field.

segment A logical unit of data, in particular a TCP segment is the unit of data transferred between a pair of TCP modules.

segment acknowledgment The sequence number in the acknowledgment field of the arriving segment.

segment length The amount of sequence number space occupied by a segment, including any controls which occupy sequence space.

segment sequence The number in the sequence field of the arriving segment.

send sequence This is the next sequence number the local (sending) TCP will use on the connection. It is initially selected from an initial sequence number curve (ISN) and is incremented for each octet of data or sequenced control transmitted.

send window This represents the sequence numbers which the remote (receiving) TCP is willing to receive. It is the value of the window field specified in segments from the remote (data receiving) TCP. The range of new sequence numbers which may be emitted by a TCP lies between `SND.NXT` and `SND.UNA`+`SND.WND` – 1. (Retransmissions of sequence numbers between `SND.UNA` and `SND.NXT` are expected, of course.)

`SND.NXT` Send sequence.

`SND.UNA` Left sequence.

`SND.UP` Send urgent pointer.

`SND.WL1` Segment sequence number at last window update.

`SND.WL2` Segment acknowledgment number at last window update.

`SND.WND` Send window.

socket An address which specifically includes a port identifier, that is, the concatenation of an Internet Address with a TCP port.

source address The source address, usually the network and host identifiers.

`SYN` A control bit in the incoming segment, occupying one sequence number, used at the initiation of a connection, to indicate where the sequence numbering will start.

TCB Transmission control block, the data structure that records the state of a connection.

`TCB.PRC` The precedence of the connection.

TCP Transmission Control Protocol: a host-to-host protocol for reliable communication in internetwork environments.

TOS Type of service, an Internet Protocol field.

type of service An Internet Protocol field which indicates the type of service for this Internet fragment.

URG A control bit (urgent), occupying no sequence space, used to indicate that the receiving user should be notified to do urgent processing as long as there is data to be consumed with sequence numbers less than the value indicated in the urgent pointer.

urgent pointer A control field meaningful only when the `URG` bit is on. This field communicates the value of the urgent pointer which indicates the data octet associated with the sending user’s urgent call.

11.13 Summary

TCP is a transport-level protocol that operates at layer five when compared to the OSI network layer model. TCP is connection-oriented, performs retransmissions, and provides flow-control capabilities.

TCP’s counterpart is UDP. TCP has a considerable number of popular programs that use it. Examples are FTP, TELNET, HyperText Transfer Protocol (HTTP), X-Windows, and SMTP. TCP is robust and reliable. In the TCP/IP network environment, TCP is the transport protocol of choice.

12
User Datagram Protocol

12.1 Perspective

The *User Datagram Protocol* (UDP) is defined to make available a datagram mode of packet-switched computer communication in the environment of an interconnected set of computer networks. This protocol assumes that the Internet Protocol (IP) is used as the underlying protocol.

This protocol provides a procedure for application programs to send messages to other programs with a minimum of protocol mechanism. The protocol is transaction-oriented, and delivery and duplicate protection are not guaranteed.

12.2 UDP Header Format

```
0       7 8     15 16      23 24    31
+--------+--------+--------+---------+
|     Source        |   Destination   |
|      Port        |       Port    |
+--------+--------+--------+---------+
|                |               |
|   Length        |     Checksum       |
+--------+--------+--------+--------+
|
|       data octets ...
+------------- ...
```

12.3 UDP Field Descriptions

Source port—an optional field. When meaningful, it indicates the port of the sending process, and may be assumed to be the port to which a reply should be addressed in the absence of any other information. If not used, a value of zero is inserted.

Destination port—has a meaning within the context of a particular Internet destination address.

Length—the length in octets of this user datagram including this header and the data. (This means that the minimum value of the length is 8.)

Checksum—the 16-bit one’s complement of the one’s-complement sum of a pseudoheader of information from the IP header, the UDP header, and the data, padded with zero octets at the end (if necessary) to make a multiple of two octets.

The pseudoheader conceptually prefixed to the UDP header contains the source address, the destination address, the protocol, and the UDP length (Fig. 12-1). This information gives protection against misrouted datagrams. This checksum procedure is the same as is used in TCP.

If the computed checksum is zero, it is transmitted as all ones (the equivalent in one’s-complement arithmetic). An all-zero transmitted checksum value means that the transmitter generated no checksum (for debugging or for higher-level protocols that don’t care).

A user interface should allow the creation of new receive ports; receive operations on the receive ports that return the data octets and an indication of source port and source address; and an operation that allows a datagram to be sent, specifying the data, source, and destination ports and addresses to be sent.

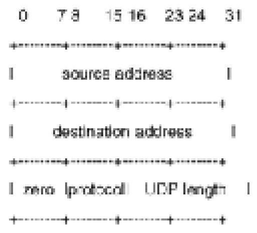


Figure 12-1
UDP pseudoheader.

12.4 IP Interface

The UDP module must be able to determine the source and destination Internet addresses and the protocol field from the Internet header. One possible UDP/IP interface would return the whole Internet datagram including all of the Internet header in response to a receive operation. Such an interface would also allow the UDP to pass a full Internet datagram complete with header to the IP to send. The IP would verify certain fields for consistency and compute the Internet header checksum.

12.5 Protocol Application

The major uses of this protocol is the Internet Name Server, and the Trivial File Transfer. The protocol number is 17 (21 octal) when used in the Internet Protocol.

12.6 Summary

UDP’s strength is that it provides the vehicle that an application can hook into, thus providing a transport protocol. Many custom applications use this protocol in the TCP/IP environment.

UDP applications do not have the support infrastructure in UDP that TCP applications have in TCP. Since this is the case, UDP applications must do much of *network work* that TCP applications leave to TCP to perform.

13
TELNET

The purpose of the TELNET protocol is to provide a fairly general, bidirectional, 8-bit byte-oriented communications facility. Its primary goal is to provide a standard method of interfacing terminal devices and terminal-oriented processes to each other. It is envisioned that the protocol may also be used for terminal-terminal communication (linking) and process-process communication (distributed computing).

13.1 Three Primary Functions

A TELNET connection is a Transmission Control Protocol (TCP) connection used to transmit data with interspersed TELNET control information.

The TELNET Protocol is built on three main ideas: (1) the concept of a network virtual terminal, (2) the principle of negotiated options, and (3) a symmetrical view of terminals and processes.

When a TELNET connection is first established, each end is assumed to originate and terminate at a *network virtual terminal* (NVT), an imaginary device which provides a standard, networkwide, intermediate representation of a canonical terminal, eliminating the need for “server” and “user” hosts to keep information about the characteristics of each other’s terminals and terminal-handling conventions. All hosts, both user and server, map their local device characteristics and conventions so as to appear to be dealing with an NVT over the network, and each can assume a similar mapping by the other party. The NVT is intended to strike a balance between being overly restricted (not providing hosts a sufficiently rich vocabulary for mapping into their local character sets) and too inclusive (penalizing users with modest terminals).

Note: *The “user” host is the host to which the physical terminal is normally attached, and the “server” host is the host which is normally providing some service. As an alternate point of view, applicable even in terminal-to-terminal or process-to-process communications, the user host is the host which initiated the communication.*

The principle of negotiated options recognizes that many hosts will wish to provide additional services over and above those available within an NVT, and many users will have sophisticated terminals and would like to have elegant, rather than minimal, services. Independent of, but structured within the TELNET Protocol are various options that will be sanctioned and may be used with the `DO, DON'T, WILL, WON'T` structure (discussed below) to allow a user and server to agree to use a more elaborate (or perhaps just different) set of conventions for their TELNET connection. Such options could include changing the character set, the echo mode, and other features.

The basic strategy for setting up the use of options is to have either party (or both) initiate a request that some option take effect. The other party may then either accept or reject the request. If the request is accepted, the option immediately takes effect; if it is rejected, the associated aspect of the connection remains as specified for an NVT. Clearly, a party may always refuse a request to enable, and must never refuse a request to disable some option since all parties must be prepared to support the NVT.

The syntax of option negotiation has been set up so that if both parties request an option simultaneously, each will see the other’s request as the positive acknowledgment of its own.

The symmetry of the negotiation syntax can potentially lead to nonterminating acknowledgment loops, in which each party sees the incoming commands not as acknowledgments but as new requests which must be acknowledged. To prevent such loops, the following rules prevail:

1. Parties may only request a change in option status; they may not send out a “request” merely to announce what mode it is in.

2. If a party receives what appears to be a request to enter some mode it is already in, the request should not be acknowledged. This nonresponse is essential to prevent endless loops in the negotiation. It is required that a response be sent to requests for a change of mode—even if the mode is not changed.

3. Whenever one party sends an option command to a second party, whether as a request or an acknowledgment, and use of the option will have any effect on the processing of the data being sent from the first party to the second, then the command must be inserted in the data stream at the point where it should take effect. (It should be noted that some time will elapse between the transmission of a request and the receipt of an acknowledgment, which may be negative. Thus, a host may wish to buffer data, after requesting an option, until it learns whether the request is accepted or rejected, in order to hide the “uncertainty period” from the user.)

Option requests are likely to flurry back and forth when a TELNET connection is first established, as each party attempts to get the best possible service from the other party. Beyond that, however, options can be used to dynamically modify the characteristics of the connection to suit changing local conditions. For example, the NVT, as will be explained later, uses a transmission discipline well suited to the many “line at a time” applications such as Beginner’s All-purpose Symbolic Instruction Code (BASIC), but poorly suited to the many “character at a time” applications such as natural-language software (NLS). A server might elect to devote the extra processor overhead required for a “character at a time” discipline when it was suitable for the local process and would negotiate an appropriate option. However, rather than being permanently burdened with the extra processing overhead, it could switch (i.e., negotiate) back to NVT when the detailed control was no longer necessary.

It is possible for requests initiated by processes to stimulate a nonterminating request loop if the process responds to a rejection by merely rerequesting the option. To prevent such loops from occurring, rejected requests should not be repeated until something changes. Operationally, this can mean that the process is running a different program, or the user has given another command, or whatever makes sense in the context of the given process and the given option. A good rule of thumb is that a repeated request should occur only as a result of subsequent information from the other end of the connection or when demanded by local human intervention.

Option designers should not feel constrained by the somewhat limited syntax available for option negotiation. The intent of the simple syntax is to make it easy to have options—since it is correspondingly easy to profess ignorance about them. If some particular option requires a richer negotiation structure than possible within `DO, DON'T, WILL, WON'T,` the proper tack is to use this structure to establish that both parties understand the option, and once this is accomplished, a more exotic syntax can be used freely. For example, a party might send a request to alter (establish) line length. If it is accepted, then a different syntax can be used for actually negotiating the line length—such a “subnegotiation” might include fields for minimum allowable, maxi mum allowable, and desired line lengths. The important concept is that such expanded negotiations should never begin until some prior (standard) negotiation has established that both parties are capable of parsing the expanded syntax.

In summary, `will` XXX is sent, by either party, to indicate that party's desire (offer) to begin performing option XXX, and `do` XXX and `dont` XXX are its positive and negative acknowledgments; similarly, `do` XXX is sent to indicate a desire (request) that the other party (i.e., the recipient of the `do`) begin performing option XXX, where `will` XXX and `wont` XXX are the positive and negative acknowledgments, respectively. Since the NVT is what is left when no options are enabled, the `dont` and `wont` responses are guaranteed to leave the connection in a state which both ends can handle. Thus, all hosts may implement their TELNET processes to be totally unaware of options that are not supported, simply returning a rejection to (i.e., refusing) any option request that cannot be understood.

As much as possible, the TELNET protocol has been made server-user symmetrical so that it easily and naturally covers the user-user (linking) and server-server (cooperating processes) cases. It is hoped, but not absolutely required, that options will further this intent. In any case, it is explicitly acknowledged that symmetry is an operating principle rather than an ironclad rule.

A companion document, *TELNET Option Specifications,* should be consulted for information about the procedure for establishing new options.

13.2 Network Virtual Terminal

The NVT is a bidirectional character device, with a printer and a keyboard. The printer responds to incoming data and the keyboard produces outgoing data which are sent over the TELNET connection and, if "echoes" are desired, to the NVT's printer as well. Echoes will not be expected to traverse the network (although options exist to enable a remote echoing mode of operation, no host is required to implement this option). The code set is 7-bit ASCII in an 8-bit field, except as modified herein. Any code conversion and timing considerations are local problems and do not affect the NVT.

Data Transmission

Although a TELNET connection through the network is intrinsically full-duplex, the NVT is to be viewed as a half-duplex device operating in a line-buffered mode. Thus, unless and until options are negotiated to the contrary, the following default conditions pertain to the transmission of data over the TELNET connection:

1. Insofar as the availability of local buffer space permits, data should be accumulated in the host where they are generated until a complete line of data is ready for transmission, or until some locally defined explicit signal to transmit occurs. This signal could be generated by either a process or a human user. The motivation for this rule is the high cost, to some hosts, of processing network input interrupts, coupled with the default NVT specification that echoes do not traverse the network. Thus, it is reasonable to buffer some amount of data at their source. Many systems take some processing action at the end of each input line (even line printers or card punches frequently tend to work this way), so the transmission should be triggered at the end of a line. On the other hand, a user or process may sometimes find it necessary or desirable to provide data which do not terminate at the end of a line; therefore implementers are cautioned to provide methods of locally signaling that all buffered data should be transmitted immediately.

2. When a process has completed sending data to an NVT printer and has no queued input from the NVT keyboard for further processing (i.e., when a process at one end of a TELNET connection cannot proceed without input from the other end), the process must transmit the TELNET GO AHEAD (GA) command. This rule is not intended to require that the TELNET GA command be sent from a terminal at the end of each line, since server hosts do not normally require a special signal (in addition to end-of-line or other locally defined characters) in order to commence processing. Rather, the TELNET GA is designed to help a user's local host operate a physically half-duplex terminal which has a "lockable" keyboard such as the IBM 2741. A description of this type of terminal may help to explain the proper use of the GA command.

The terminal-computer connection is always under control of either the user or the computer. Neither can unilaterally seize control from the other; rather, the controlling end must relinquish its control explicitly. At the terminal end, the hardware is constructed so as to relinquish control each time that a "line" is terminated [i.e., when the "new line" (ENTER) key is pressed by the user]. When this occurs, the attached (local) computer processes the input data, and decides if output should be generated and if not, returns control to the terminal. If output should be generated, control is retained by the computer until all output has been transmitted.

The difficulties of using this type of terminal through the network should be obvious. The "local" (user) computer is no longer able to decide whether to retain control after seeing an end-of-line signal; this decision can be made only by the "remote" (server) computer which is processing the data. Therefore, the TELNET GA command provides a mechanism whereby the remote computer can signal the "local" computer that it is time to pass control to the user of the terminal. It should be transmitted at those times, and only at those times, when the user should be given control of the terminal. Note that premature transmission of the GA command may result in the blocking of output, since the user is likely to assume that the transmitting system has paused, and therefore will fail to turn the line around manually.

The foregoing, of course, does not apply to the user-to-server direction of communication. In this direction, GAs may be sent at any time, but need not ever be sent. Also, if the TELNET connection is being used for process-to-process communication, GAs need not be sent in either direction. Finally, for terminal-to-terminal communication, GAs may be required in neither, one, or both directions. A host that plans to support terminal-to-terminal communication should provide the user with a means of manually signaling that it is time for a GA to be sent over the TELNET connection; this, however, is not a requirement on the implementers of a TELNET process.

Note: *The symmetry of the TELNET model requires that there be an NVT at each end of the TELNET connection, at least conceptually.*

13.3 Standard Presentation of Control Functions

As stated previously, the primary goal of the TELNET protocol is the provision of a standard interfacing of terminal devices and terminal-oriented processes through the network. Early experience with this type of interconnection showed certain functions to be implemented by most servers but wide variation in methods of invoking these functions. For a human user who interacts with several server systems, these differences are highly frustrating. TELNET, therefore, defines a standard representation for five of these functions, as described below. These standard representations have standard, but not required, meanings [with the exception that the interrupt process (IP) function may be required by other protocols which use TELNET]; that is, a system which does not provide the function to local users need not provide it to network users and may treat the standard representation for the function as a no-operation. On the other hand, a system which does provide the function to a local user is obliged to provide the same function to a network user who transmits the standard representation for the function.

Interrupt Process (IP)

Many systems provide a function which suspends, interrupts, aborts, or terminates the operation of a user process. This function is frequently used when one believes one's process is in an unending loop, or when an unwanted process has been inadvertently activated. IP is the standard representation for invoking this function. Implementers should note that IP may be required by other protocols which use TELNET, and therefore should be implemented if these other protocols are to be supported.

Abort Output (AO)

Many systems provide a function which allows an output-generating process to run to completion (or to reach the same stopping point it would reach if running to completion) but without sending the output to the user's terminal. Further, this function typically clears any output already produced but not yet actually printed (or displayed) on the user's terminal. AO is the standard representation for invoking this function. For example, some subsystem might normally accept a user's command, send a long text string to the user's terminal in response, and finally signal readiness to accept the next command by sending a prompt character (preceded by `<CR><LF>`) to the user's terminal. If the AO were received during transmission of the text string, a reasonable implementation would be to suppress the remainder of the text string, but transmit the prompt character and the preceding `<CR><LF>`. (This is possibly in distinction to the action which might be taken if an IP were received; the IP might cause suppression of the text string and an exit from the subsystem.)

Server systems which provide this function may need to clear buffers external to the system (in the network and the user's local host); the appropriate way to do this is to transmit the SYNC signal (described below) to the user system.

Are You There (AYT)?

Many systems provide a function which provides the user with some visible (e.g., printable) evidence that the system is still up and running. This function may be invoked by the user when the system is unexpectedly silent for a long time, because of the unanticipated (by the user) length of a computation or an unusually heavy system load. For instance, AYT is the standard representation for invoking this function.

Erase Character (EC)

Many systems provide a function which deletes the last preceding undeleted character or print position from the stream of data being supplied by the user. This function is typically used to edit keyboard input when typographic errors occur. EC is the standard representation for invoking this function.

Note: *A print position may contain several characters which are the result of overstrikes or of sequences such as,* `<char1> BS <char2>....`

Erase Line (EL)

Many systems provide a function which deletes all the data in the current "line" of input. This function is typically used to edit keyboard input. EL is the standard representation for invoking this function.

13.4 TELNET SYNC Signal

Most timesharing systems provide mechanisms which allow a terminal user to regain control of a runaway process; the IP and AO functions described above are examples of these mechanisms. Such systems, when used locally, have access to all the signals supplied by the user, whether these are normal characters or special out-of-band signals such as those supplied by the teletype BREAK key or the IBM 2741 ATTN key. This is not necessarily true when terminals are connected to the system through the network; the network's flow-control mechanisms may cause such a signal to be buffered elsewhere, such as in the user's host.

To counter this problem, the TELNET SYNC mechanism is introduced. A SYNC signal consists of a TCP URGENT notification, coupled with the TELNET command DATA MARK. The URGENT notification, which is not subject to the flow control pertaining to the TELNET connection, is used to invoke special handling of the data stream by the process which receives it. In this mode, the data stream is immediately scanned for "interesting" signals as defined below, discarding intervening data. The TELNET command DATA MARK (DM) is the synchronizing mark in the data stream which indicates that any special signal has already occurred and the recipient can return to normal processing of the data stream. The SYNC is sent via the TCP send operation with the URGENT flag set and the DM as the last (or only) data octet.

When several SYNCs are sent in rapid succession, the URGENT notifications may be merged. It is not possible to count URGENTs since the number received will be less than or equal to the number sent. When in normal mode, a DM is a no-operation; when in URGENT mode, it signals the end of the URGENT processing. If TCP indicates the end of URGENT data before the DM is found, TELNET should continue the special handling of the data stream until the DM is found.

If TCP indicates more URGENT data after the DM are found, it can only be because of a subsequent SYNC. TELNET should continue the special handling of the data stream until another DM is found.

"Interesting" signals are defined as the TELNET standard representations of IP, AO, and AYT (but not EC or EL); the local analogs of these standard representations (if any); all other TELNET commands; and other site-defined signals which can be acted on without delaying the scan of the data stream.

Since one effect of the SYNC mechanism is the discarding of essentially all characters (except TELNET commands) between SYNC sender and recipient, this mechanism is specified as the standard way to clear the data path if desired. For example, if a user at a terminal causes an AO to be transmitted, the server which receives the AO (if it provides that function at all) should return a SYNC to the user.

Finally, just as the TCP URGENT notification is needed at the TELNET level as an out-of-band signal, other protocols which make use of TELNET may also require a TELNET command which can be viewed as an out-of-band signal at a different level.

By convention the sequence [IP, SYNC] is to be used as such a signal. For example, suppose that some other protocol which uses TELNET defines the character string STOP analogously to the TELNET command AO. Imagine that a user of this protocol wishes a server to process the STOP string, but the connection is blocked because the server is processing other commands. Users should instruct their systems to send the (1) TELNET IP character, (2) TELNET SYNC sequence [i.e., send the DM as the only character in a TCP URGENT-mode send operation], (3) character string STOP, and (4) other protocol's analog of the TELNET DM, if any. The user (or process acting on the user's behalf) must transmit the TELNET SYNC sequence of step 2 to ensure that the TELNET IP gets through to the server's TELNET interpreter.

The URGENT notification should wake up the TELNET process; the IP should wake up the next-higher-level process.

13.5 The NVT Printer and Keyboard

The NVT printer has an unspecified carriage width and page length and can produce representations of all 95 ASCII graphics (codes 32 through 126). Of the 33 ASCII control codes (0 through 31 and 127), and the 128 uncovered codes (128 through 255), the following have specified meaning to the NVT printer:

Name	Code	Meaning
NULL (NUL)	0	No operation
Linefeed (LF)	10	Moves the printer to the next print line, keeping the same horizontal position.
Carriage return (CR)	13	Moves the printer to the left margin of the current line.

The following codes also have defined, but not required, effects on the NVT printer. Neither end of a TELNET connection may assume that the other party will take, or will have taken, any particular action on receipt or transmission of these:

Name	Code	Meaning
BELL (BEL)	7	Produces an audible or visible signal (which does NOT move the print head).
Backspace (BS)	8	Moves the print head one character position toward the left margin.
Horizontal tab (HT)	9	Moves the printer to the next horizontal tab stop. It remains unspecified how either party determines or establishes where such tab stops are located.
Vertical tab (VT)	11	Moves the printer to the next vertical tab stop. It remains unspecified how either party determines or establishes where such tab stops are located.
Formfeed (FF)	12	Moves the printer to the top of the next page, keeping the same horizontal position.

All remaining codes do not cause the NVT printer to take any action.

The sequence CR LF, as defined, will cause the NVT to be positioned at the left margin of the next print line (as would, e.g., the sequence LF CR). However, many systems and terminals do not treat CR and LF independently, and will have to go to some effort to simulate their effect (e.g., some terminals do not have a CR independent of the LF, but on such terminals it may be possible to simulate a CR by backspacing). Therefore, the sequence CR LF must be treated as a single new-line character and used whenever their combined action is intended, the sequence CR NUL must be used where a carriage return alone is actually desired, and the CR character must be avoided in other contexts. This rule gives assurance to systems which must decide whether to perform a new-line function or a multiple backspace that the TELNET stream contains a character following a CR that will allow a rational decision.

Note that CR LF or CR NUL is required in both directions (in the default ASCII mode), to preserve the symmetry of the NVT model. Even in situations (e.g., with remote echo and suppress-GA options in effect) in which characters are not being sent to an actual printer, for the sake of consistency, the protocol requires that a NUL be inserted following a CR not followed by an LF in the data stream. The converse of this is that a NUL received in the data stream after a CR (in the absence of options negotiations which explicitly specify otherwise) should be stripped out before the NVT is applied to local character-set mapping.

The NVT keyboard has keys, key combinations, or key sequences for generating all 128 ASCII codes. Note that although many have no effect on the NVT printer, the NVT keyboard is capable of generating them. In addition to these codes, the NVT keyboard shall be capable of generating the following additional codes which, except as noted, have defined—but not required—meanings. The actual code assignments for these “characters” are in the TELNET command section, because they are viewed as being, in some sense, generic and should be available even when the data stream is interpreted as being some other character set.

• *SYNC.* This key allows you to clear your data path to the other party. Activation of this key causes a DM (see command section) to be sent in the data stream and a TCP URGENT notification is associated with it. The pair DM-URGENT is to have the required meaning as defined previously.

• *Break (BRK).* This code is provided because it is a signal outside the ASCII character set which is currently given local meaning within many systems. It is intended to indicate that the BRK key or the ATTN key was hit. Note, however, that this is intended to provide a 129th code for systems which require it, not as a synonym for the IP standard representation.

• *Interrupt process (IP).* Suspend, interrupt, abort, or terminate the process to which the NVT is connected. Also, part of the out-of-band signal for other protocols which use TELNET.

• *Abort output (AO).* Allow the current process to (appear to) run to completion, but do not send its output to the user. Also, send a SYNC to the user.

• *Are you there (AYT)*? Send back to the NVT some visible (i.e., printable) evidence that the AYT was received.

• *Erase character (EC).* The recipient should delete the last preceding undeleted character or “print position” from the data stream.

• *Erase line (EL).* The recipient should delete characters from the data stream back to, but not including, the last CR LF sequence sent over the TELNET connection.

The spirit of these “extra” keys, and also the printer format effectors, is that they should represent a natural extension of the mapping that already must be done from NVT into the local terminal. Just as the NVT data byte 68 (104 octal) should be mapped into whatever the local code for uppercase D is, so the EC character should be mapped into whatever the local EC function is. Further, just as the mapping for 124 (174 octal) is somewhat arbitrary in an environment that has no vertical-bar character, the EL character may have a somewhat arbitrary mapping (or none at all) if there is no local erase-line facility. Similarly for format effectors; if the terminal actually does have a vertical tab, then the mapping for VT is obvious, and only when the terminal does not have a vertical tab should the effect of VT be unpredictable.

TELNET Command Structure

All TELNET commands consist of at least a 2-byte sequence: the *interpret as command* (IAC) escape character followed by the code for the command. The commands dealing with option negotiation are 3-byte sequences; the third byte is the code for the option referenced. This format was chosen so that as more comprehensive use of the data space is made—by negotiations from the basic NVT, of course—collisions of data bytes with reserved command values will be minimized, all such collisions requiring the inconvenience, and inefficiency, of “escaping” the data bytes into the stream. With the current setup, only the IAC need be doubled to be sent as data, and the other 255 codes may be passed transparently.

TELNET commands are defined in the following table. Note that these codes and code sequences have the indicated meaning only when immediately preceded by an IAC.

Name	Code	Meaning
SE	240	End of subnegotiation parameters
NOP	241	No operation
Data mark	242	The data stream portion of a SYNC; should always be accompanied by a TCP URGENT notification
Break	243	NVT character BRK
Interrupt process	244	The function IP
Abort output	245	The function AO
Are you there?	246	The function AYT
Erase character	247	The function EC
Erase line	248	The function EL
Go ahead	249	The GA signal
SB	250	Indicates that what follows is subnegotiation of the indicated option
WILL (option code)	251	Indicates the desire to begin performing, or confirmation that you are now performing, the indicated option
WON’T (option code)	252	Indicates the refusal to perform, or continue performing, the indicated option
DO (option code)	253	Indicates the request that the other party perform, or confirmation that you are expecting the other party to perform, the indicated option
DON’T (option code)	254	Indicates the demand that the other party stop performing, or confirmation that you are no longer expecting the other party to perform, the indicated option
IAC	255	Data byte 255

1. *Connection establishment.* The TELNET TCP connection is established between the user’s port U and the server’s port L. The server listens on its well-known port L for such connections. Since a TCP connection is full-duplex and identified by the pair of ports, the server can engage in many simultaneous connections involving its port L and different user ports U.

2. *Port assignment.* When used for remote user access to service hosts (i.e., remote terminal access), this protocol is assigned server port 23 (27 octal; i.e., L = 23).

13.6 TELNET Option Specifications

The intent of providing for options in the TELNET Protocol is to permit hosts to obtain more elegant solutions to the problems of communication between dissimilar devices than is possible within the framework provided by the network virtual terminal (NVT). It should be possible for hosts to invent, test, or discard options at will. Nevertheless, it is envisioned that options which prove to be generally useful will eventually be supported by many hosts; therefore it is desirable that options be carefully documented and well publicized. In addition, it is necessary to ensure that a single option code is not used for several different options.

The TELNET RFC specifies a method of option code assignment and standards for documentation of options. The individual responsible for assignment of option codes may waive the requirement for complete documentation for some cases of experimentation, but in general documentation will be required prior to code assignment. Options will be publicized by publishing their documentation as RFCs; inventors of options may, of course, publicize them in other ways as well.

TELNET Subnegotiation

Some options will require more information than a single option code to be passed between hosts; an example would be any option which requires a parameter. The strategy to be used consists of two steps: (1) both parties agree to “discuss” the parameter(s), and (2) the “discussion” takes place. Step 1, agreeing to discuss the parameters, takes place in the normal manner; one party proposes use of the option by sending a `DO` (or `WILL`) followed by the option code, and the other party accepts by returning a `WILL` (or `DO`) followed by the option code. Once both parties have agreed to use the option, subnegotiation takes place by using the command SB, followed by the option code, followed by the parameter(s), followed by the command SE. Each party is presumed to be able to parse the parameter(s), since each has indicated that the option is supported (via the initial exchange of `WILL` and `DO`). On the other hand, the receiver may locate the end of a parameter string by searching for the SE command (i.e., the string IAC SE), even if the receiver is unable to parse the parameters. Of course, either party may refuse to pursue further subnegotiation at any time by sending a `WONT` or `DONT` to the other party.

For example, for option ABC, which requires subnegotiation, the TELNET command formats are IAC WILL ABC, an offer to use option ABC (or favorable acknowledgment of the other party’s request); IAC DO ABC, a request for the other party to use option ABC (or favorable acknowledgment of the other party’s offer); and IAC SB ABC , <PARAMETERS> IAC SE, one step of subnegotiation, used by either party.

Options requiring subnegotiation must avoid unending loops in the subnegotiation process. For example, if each party can accept any value of a parameter, and both parties suggest parameters with different values, then one is likely to have an infinite oscillation of acknowledgments (where each receiver believes it is only acknowledging the new proposals of the other). Finally, if parameters in an option subnegotiation include a byte with a value of 255, it is necessary to double this byte in accordance with the general TELNET rules.

13.6.1 TELNET environment option

This section explains a way to pass environment information between a TELNET client and server. Use of this mechanism enables a TELNET user to propagate configuration information to a remote host when connecting. The TELNET command names and codes are

NEW-ENVIRON	39	VAR	0
IS	0	VALUE	1
SEND	1	ESC	2
INFO	2	USERVAR	3

Command Meanings

`IAC WILL NEW-ENVIRON`	The sender of this command is willing to send environment variables.
`IAC WONT NEW-ENVIRON`	The sender of this command refuses to send environment variables.
`IAC DO NEW-ENVIRON`	The sender of this command is willing to receive environment variables.
`IAC DONT NEW-ENVIRON`	The sender of this command refuses to accept environment variables.
`IAC SB NEW-ENVIRON SEND [type ... [type ... [...] ] ] IAC SE`	The sender of this command requests that the remote side send its environment variables. The type may be either `VAR` or `USERVAR,` to indicate either well-known or user-defined variable names. Only the side that is DO NEW-ENVIRON may initiate a SEND command. If a list of variables is specified, then only those variables should be sent. If no list is specified, then the default environment, of both well-known and user-defined variables, should be sent. If one of the variables has no name, then all the variables of that type (well-known or user-defined) in the default environment should be sent.
`IAC SB NEW-ENVIRON IS type ... [ VALUE ... ] [type ... [ VALUE ... ] [ ... ] ] IAC SE`	The sender of this command is sending environment variables. This command is sent in response to a SEND request. Only the side that is WILL NEW-ENVIRON may send an IS command. The type/VALUE pairs must be returned in the same order as the SEND request specified them, and there must be a response for each type. . . explicitly requested. The type will be VAR or *uservar*. Multiple environment variables may be sent. The characters following a type up to the next type or value specify the variable name. The characters following a VALUE up to the next type specify the value of the variable. If a type is not followed by a VALUE (e.g., by another VAR, USERVAR, or IAC SE), then that variable is undefined. If a VALUE is immediately followed by a type or IAC, then the variable is defined, but has no value. If an IAC is contained between the IS and the IAC SE, it must be sent as IAC IAC. If a variable or a value contains a VAR, it must be sent as ESC VAR. If a variable or a value contains a USERVAR, it must be sent as ESC USERVAR. If a variable a value contains a VALUE, it must be sent as ESC VALUE. If a variable or a value contains an ESC, it must be sent as ESC ESC.

`IAC SB NEW-ENVIRON INFO type... [VALUE...] [type... [VALUE...] [...] ] IAC SE`	The sender of this command is sending information about environment variables that have changed. It is identical to the IS command, except that the command is INFO instead of IS. Only the side that is WILL NEW-ENVIRON may send an INFO command. The INFO command is not to be used to send initial information; the SEND/IS sequence is to be used for that. The INFO command is to be used to propagate changes in environment variables, and may be spontaneously generated.

The *default specification* for this option is

WONT NEW-ENVIRON
DONT NEW-ENVIRON

This means that there will not be any exchange of environment information.

Motivation

Many operating systems have start-up information and environment variables that contain information that should be propagated to remote machines when TELNET connections are established. Rather than create a new TELNET option each time someone comes up with some new information that they need propagated through a TELNET session, but that the TELNET session itself doesn’t really need to know about, this generic information option can be used.

Well-known Variables

`USER`	This variable is used to transmit the user or account name that the client wishes to log into on the remote system. The format of the value of the USER variable is system dependent, as determined by the remote system.
`JOB`	This variable is used to transmit the job ID that the client wishes to use when logging into the remote system. The format of the value the JOB variable is system-dependent, as determined by the remote system.
`ACCT`	This variable is used to transmit the account ID that the client wishes to use when logging into the remote system. The format of the value the ACCT variable is system-dependent, as determined by the remote system.
`PRINTER`	This variable is used to identify the default location for printer output. Because there is as yet no standard way of naming a printer on a network, the format of this variable is currently undefined.

`SYSTEMTYPE`	This is used to transmit the type of operating system on the system that sends this variable. Its value is identical to the value of the SYSTEM (SYST) command in FTP (File Transfer Protocol). The format of the value shall have as its first word one of the system names listed in the current version of the *Assigned Numbers* document.
`DISPLAY`	This variable is used to transmit the X display location of the client. The format for the value of the DISPLAY variable is host: `<dispnum>` [.`<screennum`>].

This information is identical to the information passed using the TELNET `X-DISPLAY-LOCATION` option. If both this option and the `DISPLAY` environment variable are received, and both contain conflicting information, the most recently received information received should be used.

Because it is impossible to anticipate all variables that users may wish to exchange, the `USERVAR` type is provided to allow users to transmit arbitrary variable/value pairs. The use of an additional type allows implementations to distinguish between values derived by the remote host software and values supplied by the user. Paranoid implementations will most likely treat both types with an equal level of distrust. The results of a name-space collision between a well-known and a user-defined variable are implementation specific.

Implementation Rules

`WILL` and `DO` are used only at the beginning of the connection to obtain and grant permission for future negotiations. Once the two hosts have exchanged a `WILL` and a `DO,` the sender of the `DO NEW-ENVIRON` is free to request that environment variables be sent. Only the sender of the `DO` may send requests (`IAC SB NEW-ENVIRON SEND IAC SE`) and only the sender of the `WILL` may transmit actual environment information (via the `IAC SB NEW-ENVIRON IS...IAC SE` command). Although this option may be used at any time throughout the life of the TELNET connection, the exchange of environment information will usually happen at connection start-up. This is because many operating systems have mechanisms for propagating environment information only at process creation, so the information is needed before the user logs in.

The receiving host is not required to put all variables that it receives into the environment. For example, if the client should send across `USERVAR “TERM” VALUE “xterm”` as an environment variable, and the `TERMINAL-TYPE` option has already been used to determine the terminal type, the server may safely ignore the `TERM` variable. Also, some start-up information may be used in other ways; for example, `USER, ACCT,` and `PROJ` values might be used to decide which account to log into, and might never be put into the user’s environment. In general, if the server has already determined the value of an environment variable by some more accurate means, or if it does not understand a variable name, it may ignore the value sent in the `NEW-ENVIRON` option. The server may also prefer to just put all unknown information into the user’s environment. This is the suggested method of implementation, because it allows the user the most flexibility.

The following is an example of use of the option:

HOST1 **HOST2**

IAC DO NEW-ENVIRON IAC WILL NEW-ENVIRON
[Host1 is now free to request environment information]
IAC SB NEW-ENVIRON SEND VAR

“USER” VAR “ACCT” VAR USERVAR
IAC SE

The server has now explicitly asked for the `USER` and `ACCT` variables, the default set of well-known environment variables, and the default set of user-defined variables. Note the client includes the `USER` information twice: once because it was explicitly asked for, and once because it is part of the default environment.

IAC SB NEW-ENVIRON IS VAR “USER”
    VALUE “joe” VAR “ACCT” VALUE
    “kernel” VAR “USER” VALUE “joe”
    VAR “DISPLAY” VALUE “foo:0.0”
    USERVAR “SHELL” VALUE “/bin/csh”
    IAC SE

It is legal for a client to respond with an empty environment (no data between the `IAC SB` and `IAC SE`) when no well-defined or user-defined variables are currently defined. For example, `IAC SB NEW-ENVIRON IS IAC SE` is a valid response to any of the following:

IAC SB NEW-ENVIRON SEND IAC SE
IAC SB NEW-ENVIRON SEND VAR IAC SE
IAC SB NEW-ENVIRON SEND USERVAR IAC SE
IAC SB NEW-ENVIRON SEND VAR USERVAR IAC SE
(The last example is equivalent to the first. . .)

An earlier version of this specification incorrectly reversed the values for `VAR` and `VALUE,` which put the specification at odds with existing implementations. To resolve that problem, as well as other minor problems, a new option number has been assigned to the `NEW-ENVIRON` option. This allows implementations of this memo to interoperate with no ambiguity. Any implementation that supports the TELNET `NEW-ENVIRON` option is expected to support all of this specification.

Security Concerns

It is important for implementers of the `NEW-ENVIRON` option to understand the interaction of setting options and the login/authentication process. Specifically, careful analysis should be done to determine which variables are “safe” to set before the client logs in. An example of a poor choice would be permitting a variable to be changed that allows an intruder to circumvent or compromise the login/authentication program itself.

13.6.2 TELNET Linemode Option

This section explains an elective standard for the Internet community. Hosts on the Internet that support TELNET linemode within the TELNET protocol are expected to adopt and implement this standard.

Linemode TELNET is a way of doing terminal character processing on the client side of a TELNET connection. While in linemode with editing enabled for the local side, network traffic is reduced to a couple of packets per command line, rather than a couple of packets per character typed. This is very useful for long delay networks, because the user has local response time while typing the command line, and incurs network delays only after the command is typed. It is also useful to reduce costs on networks that charge on a per-packet basis.

Command Names and Codes

LINEMODE	34	SLC_EW	12
MODE	1	SLC_RP	13
EDIT	1	SLC_LNEXT	14
TRAPSIG	2	SLC_XON	15
MODE_ACK	4	SLC_XOFF	16
FORWARDMASK	2	SLC_FORW1	17
SLC	3	SLC_FORW2	18
SLC_SYNCH	1	SLC_DEFAULT	3
SLC_BRK	2	SLC_VALUE	2
SLC_IP	3	SLC_CANTCHANGE	1
SLC_AO	4	SLC_NOSUPPORT	0
SLC_AYT	5	SLC_LEVELBITS	3
SLC_EOR	6	SLC_ACK	128
SLC_ABORT	7	SLC_FLUSHIN	64
SLC_EOF	8	SLC_FLUSHOUT	32
SLC_SUSP	9	EOF	236
SLC_EC	10	SUSP	237
SLC_EL	11	ABORT	238

Command Meanings

The TELNET linemode function is broken down as follows:

`IAC WILL LINEMODE`	The sender of this command *requests* permission to begin subnegotiation of the editing/signaling status. This should be sent only by the client side of the connection.
`IAC WONT LINEMODE`	The sender of this command *demands* that subnegotiation of the editing/signaling status not be allowed.
`IAC DO LINEMODE`	The sender of this command *requests* that the remote side begin subnegotiation of the editing/signaling status. This should only be sent by the server side of the connection.
`IAC DONT LINEMODE`	The sender of this command *demands* that the remote side not begin subnegotiation of the editing/signaling status.
`LINEMODE suboption MODEIAC SB LINEMODE MODE mask IAC SE`	The sender of this command *confirms*, or *requests* or permission for, a switch to the mode defined by mask.

In the last command, the `mask` is a bit mask of various modes that the connection can be in. Under normal operation, the server side of the connection will initiate mode changes, and the client will confirm them. The currently defined modes are:

`EDIT`	When set, the client side of the connection should process all input lines, performing any editing functions, and only send completed lines to the remote side. When unset, the client side should not process any input from the user, and the server side should take care of all character processing that needs to be done.
`TRAPSIG`	When set, the client side should translate appropriate interrupts/signals to their TELNET equivalents (viz. `IP, BRK, AYT, ABORT, EOF,` and `SUSP`). When unset, the client should pass interrupts/signals as their normal ASCII values.
`FLOW`	Logically, this belongs in the mask. However, this would overlap the TELNET TOGGLE-FLOW-CONTROL option, so the TELNET TOGGLE-FLOW-CONTROL option is used instead. When DO/WILL LINEMODE is negotiated, DO/WILL TOGGLE-FLOW-CONTROL should also be negotiated.
`ECHO`	Logically, this belongs in the mask. However, this would overlap the TELNET ECHO option, so the TELNET ECHO option is used instead. The client side should never negotiate WILL ECHO. When the server has negotiated WILL ECHO, the client should not echo data typed by the user back to the user. When the server has negotiated WONT ECHO, the client is responsible for echoing data typed by the user back to the user.

When the client side of a connection receives a MODE command, it *must* agree with at least the state of the `EDIT` and `TRAPSIG` bits. If a MODE command is received with a mode mask that is currently in use (ignoring the `MODE_ACK` bit), the MODE command is ignored. If a MODE command is received that is different from the current mode mask, then a reply is sent with either the new mode mask and the `MODE_ACK` bit set, or a subset of the new mode mask. The only exception is that if the server receives a MODE with either the `EDIT` or `TRAPSIG` bits not set, it may set the `EDIT` and `TRAPSIG` bits in the response, and if the client receives a MODE with the `EDIT` or `TRAPSIG` bits set, it may not clear them in the response.

When a MODE command is received with the `MODE_ACK` bit set, and the mode differs from the current mode, the client will ignore the new mode and the server will switch to the new mode. This ensures that both sides of the connection will resolve to the same mode. In all cases, a response is never generated to a MODE command that has the `MODE_ACK` bit set.

LINEMODE suboption FORWARDMASK

`IAC SB LINEMODE DO FORWARDMASK mask0 mask1... mask31 IAC SE`	The sender of this command requests that the other side send any buffered data when any of the ASCII characters defined by the bit mask are received. Only the side of the connection that sent DO LINEMODE (the server side) may negotiate this. The mask is up to 32 octets long. Each octet represents 8 ASCII character codes. The high-order bit of mask0 corresponds to ASCII code 0; the low-order bit of mask0, to ASCII code 7; the high-order bit of mask1, to ASCII code 8; the low-order bit of mask1, corresponds to ASCII code 15; and so on. The mask list may be terminated before the end of the list, in which case all remaining mask octets are assumed to be reset (equal to zero). When the server side is in DONT TRANSMIT-BINARY mode, then only the first 16 octets of the mask (ASCII codes 0 through 127) are used. If any individual octet of the mask is equal to IAC, it must be sent as a double IAC.
`IAC SB LINEMODE DONT FORWARDMASK IAC SE`	The sender of this command requests that the other side stop using the forward mask to determine when to send buffered data.
`IAC SB LINEMODE WILL FORWARDMASK IAC SE`	This command is sent in response to a DO FORWARDMASK command. It indicates that the forward mask will be used to determine when to send buffered data.
`IAC SB LINEMODE WONT FORWARDMASK IAC SE`	This command is sent in response to a DO FORWARDMASK command. It indicates that the forward mask will not be used to determine when to send buffered data.
`LINEMODE suboption SLC`	The SLC (set local characters) suboption uses a list of octet triplets. The first octet specifies the function, the second octet specifies modifiers to the function, and the third octet specifies the ASCII character for the function.
`IAC SB LINEMODE SLC <list of octet triplets> IAC SE`	The sender of this command *requests* that the list of octet triplets be used to set the local character to be used to perform the specified function.

A function may be set to one of four levels: `SLC_NOSUPPORT` is the lowest level, `SLC_CANTCHANGE` is the next-higher level, `SLC_VALUE` is above that, and `SLC_DEFAULT` is the highest level.

• If the `SLC_LEVEL` bits in the second octet are equal to `SLC_DEFAULT,` then this particular function should use the system default on the other side of the connection.

• If the `SLC_LEVEL` bits in the second octet are equal to `SLC_VALUE,` then this function is supported, and the current value is specified by the third octet.

• If the SLC_LEVEL bits in the second octet are equal to SLC_CANTCHANGE, then this is a function that is supported, but the value for this function, specified in the third octet, cannot be changed.

• If the SLC_LEVEL bits in the second octet are equal to SLC_NOSUPPORT, then this particular function is not supported and should be disabled by the other side.

• If this is a response to a previous request to change a special character, and we are agreeing to the change, then the SLC_ACK bit must be set in the second octet.

• If the SLC_FLUSHIN bit is set in the second octet, then whenever this function is sent, a TELNET SYNC should be sent at the same time to flush the input stream.

• If the SLC_FLUSHOUT bit is set in the second octet, then whenever this function is sent, output data should be flushed.

Only the client may send an octet triplet with the first octet equal to zero. In this case, the SLC_LEVEL bits may only be set to SLC_DEFAULT or SLC_VALUE, and the third octet does not matter. When the server receives 0 SLC_DEFAULT 0, it should switch to its system default special character settings, and send all those special characters to the client. When the server receives 0 SLC_VALUE 0, it should just send its current special-character settings. Note that if the server does not support some of the editing functions, the functions should be sent as XXX SLC_DEFAULT 0, rather than as XXX SLC_NOSUPPORT 0, so that the client may choose to use its own values for those functions, rather than have to disable them even if it supports them. If any of the octets in the list of octet triplets is equal to IAC, it must be sent as a double IAC.

When a connection is established, the client must either request the remote default values for the special characters or send across what all the special characters should be set to.

The function values can be put into two groups: (1) functions that are to be translated to their TELNET equivalents before being sent across the TELNET connection and (2) functions that are to be recognized and processed locally.

First, we have those characters that are to be mapped into their TELNET equivalents:

SLC_SYNCH	Sync
SLC_BRK	Break
SLC_IP	Interrupt process
SLC_AO	Abort output
SLC_AYT	Are you there?
SLC_EOR	End of record
SLC_ABORT	Abort
SLC_EOF	End of file
SLC_SUSP	Suspend

Next, we have the locally interpreted functions:

`SLC_EC`	Erase character. This character is typed to erase one character from the input stream.
`SLC_EL`	Erase line. This character is typed to erase the entire contents of the current line of input.
`SLC_EW`	Erase word. This character is typed to erase one word from the input stream. When backing up in the input stream, a word is defined to be (optionally) whitespace (tab or space characters), and a string of characters up to, but not including, whitespace or line delimiters.
`SLC_RP`	Reprint line. This character is typed to cause the current line of input to be reprinted, leaving the cursor at the end of the line.
`SLC_LNEXT`	Literal next. This character is typed to indicate that the next character is to be taken literally, no character processing should be done with it, and if it is a special character that would normally get mapped into a TELNET option, that mapping should not be done.
`SLC_XON`	Start output. This character is sent to resume output to the user's terminal.
`SLC_XOFF`	Stop output. This character is sent to stop output to the user's terminal.
`SLC_FORW1`	Forwarding character. This character should cause all data currently being buffered, and this character, to be sent immediately.
`SLC_FORW2`	Forwarding character. This is another character that is to be treated in the same manner as `SLC_FORW1`.

Additional Control Characters

`IAC ABORT`	Abort. Similar to IAC IP, but means only to abort or terminate the process to which the NVT is connected. (The TELNET specification says that IP may "suspend, interrupt, abort or terminate" the process.) If a system does not have two methods of interrupting a process, then ABORT and IP should have the same effect.
`IAC SUSP`	Suspend the execution of the current process attached to the NVT in such a way that another process will take over control of the NVT and the suspended process can be resumed at a later time. If the receiving system does not support this functionality, it should be ignored.
`IAC EOF`	End of file. The recipient should notify the process connected to the NVT that an end of file has been reached. This is intended for systems that support the ability for the user to type in an EOF character at the keyboard.

The *default specification* for this option is

WONT LINEMODE
DONT LINEMODE

This means that there will not be any subnegotiation of the mode of the connection.

If `WILL LINEMODE` is negotiated, the defaults are

IAC SB LINEMODE MODE 0 IAC SE
IAC SB LINEMODE WONT FORWARDMASK IAC SE

If `DO LINEMODE` is negotiated, the defaults are

IAC SB LINEMODE MODE 0 IAC SE
IAC SB LINEMODE DONT FORWARDMASK IAC SE

Character values for SLC default to `SLC_NOSUPPORT.`

Motivations Behind Design

With increasing TELNET usage, it is apparent that the ability to do command-line processing on the local machine and send completed lines to the remote machine is necessary in several environments. First, in the case of a connection over long delay equipment, users become very frustrated when echoing of their data takes several seconds. Second, some supercomputers inherently do not handle and process single-character input well. For these machines, it is better to have the front-end computer do the character processing, and leave the supercomputer's cycles available to perform vectored number crunching.

There have been attempts to make local line editing work within the existing TELNET specs. Indeed, the 4.3-BSD (Berkeley Software Division) tape includes a version of TELNET that attempts to do this through recognition of the state of the `ECHO` and `SUPRESS-GO-AHEAD` options; other implementations do this recognition purely through the `ECHO` option. There are problems with both of these methods. Using just the `ECHO` provides no mechanism to have `ECHO` to the user turned off, and leave local character processing on, for example, when a user is typing a password.

Usage of the `SUPRESS-GO-AHEAD` comes from reading into RFC 858, where it states:

> In many TELNET implementations it will be desirable to couple the `SUPRESS-GO-AHEAD` option to the echo option so that when the echo option is in effect, the `SUPPRESS-GO-AHEAD` option is in effect simultaneously: both of these options will normally have to be in effect simultaneously to effect what it commonly understood to be character-at-a-time echoing by the remote computer.

The reverse reading of this is that without the `ECHO` option or the `SUPPRESS-GO-AHEAD` option, you are in line-at-a-time mode, implying local line editing. This has the obvious problem that that is not what the `SUPPRESS-GO-AHEAD` option is supposed to mean.

Other shortcomings are that the TELNET specification is not rich enough to handle all of the special characters that some of the current operating systems support. For example, the `ECHO/SGA` implementation supports two ways of interrupting a process, by borrowing the BRK option for the second interrupt. Some implementations have taken the EOR option to send an EOF. Obviously, this means using things for which they were not intended, and the correct solution would be to define new options.

Another problem is that some linemode implementations buffer up the input until the end of the line, and then send the whole line across, editing characters and all. No local editing of the line has been done.

After examining several implementations, it has become clear that the correct thing to do is to implement new options to enhance the current TELNET specification so that it can support local line editing in a reasonable, reliable, and consistent manner.

Three states are of interest: (1) local line editing and local signal trapping; (2) remote line editing, local signal trapping; and (3) remote line editing, remote signal trapping.

A fourth possible state, local line editing and remote signal trapping, would not be very interesting, because you wouldn't recognize the signals and could not send them to the remote side for it to recognize until the line has been completed. Also, special signals usually will have an effect on the line editing function, and if they are not being trapped locally, the desired action will not take place.

Local line editing means that all normal command-line character processing, such as ERASE CHARACTER and ERASE LINE, happen on the local system, and only when CR LF (or some other special character) is encountered are the edited data sent to the remote system.

Signal trapping means, for example, that if the user types the character associated with the IP function, then the IAC IP function instead of the character typed is sent to the remote side. Remote signal trapping means, for example, that if the user types the character associated with the IP function, then the IAC IP function is not sent to the remote side, but rather the actual character typed is sent to the remote side.

Implementation Rules and User Interface

Any implementation that supports the TELNET linemode option will theoretically support all of this specification.

Normally, the entire user interface is left up to the implementers. However, the user should be able to specify certain functionality on the client side of the connection. During a TELNET session, the client side should allow some mechanism for the user to give commands to the local TELNET process. These commands should at least allow the user to (1) change the mode of the connection—the user should be able to attempt to turn `EDIT, FLOW, TRAPSIG,` and `ECHO` on and off, while the server may refuse to change the state of the `EDIT` and `TRAPSIG` bits; (2) import or export SLC—the user should be able to tell the local TELNET process whether to use the local or the current or default remote definitions of the special characters; and (3) manual sending of options—the user should be able to tell the local TELNET process to explicitly send any of the TELNET options (IP, ABORT, AYT, etc.).

End-of-line Terminators

When `LINEMODE` is turned on, and when in `EDIT` mode, when any normal line terminator on the client-side operating system is typed, the line should be transmitted with CR LF as the line terminator. When `EDIT` mode is turned off, a carriage return should be sent as CR NUL, a linefeed should be sent as LF, and any other key that cannot be mapped into an ASCII character, but means that the line is complete (e.g., a `DOIT` or ENTER key), should be sent as CR LF.

Output Processing

Regardless of what mode has been negotiated, the server side is responsible for doing all output processing. Specifically, it should send CR LF when it wants the `newline` function, CR NUL when it wants just a carriage return, and LF when it wants just a linefeed.

A TELNET Terminal Driver

Conforming implementations need not do all the line editing themselves. There is nothing wrong with letting the system terminal driver handle the line editing, and have it hand to the TELNET application the completed and edited line, which is then sent to the remote system.

Setting of Local Characters

Originally the thought of setting local characters was for both sides of the connection to use their own defaults for the special characters, even if they were not the same on both sides of the connection. If this scheme is used, though, the user perceives that the local special characters are being used, and the remote character settings don't matter. It was decided that the client side of the connection should be in control of the character settings.

When `LINEMODE` is negotiated, the client must either export the local character settings to the server or send a request (`SLC 0 SLC_DEFAULT 0`) to import the server's special characters. The usual action would be that a client running on a full-fledged computer would export the special characters, and a client running where there are no local defaults (as on some terminal servers) would import the special characters.

When an SLC command is received, the action taken should be:

1. Ignore the command if it is the same as the current settings.

2. If the `SLC_LEVEL` bits are the same as the current level bits, but the value is different and the `SLC_ACK` bit is set, no reply is generated. On the server side, the command is ignored, and on the client side, a switch is made to the new value. This is to ensure that if a request to change the same character is generated by both the server and the client, they will both settle on the client's requested value.

3. If we agree with the new setting, we switch to it and reply with the same value, but also set the `SLC_ACK` bit.

4. If we don't agree, we send a response with what we think the value should be. The `SLC_ACK` bit is *not* set in this case. You may disagree with a value only by sending a different value at a lower level.

If the remote system doesn't support some of the line editing characters but the front end does, then the front end may use the local definitions for those characters when in LINEMODE. In this case, the server should send `SLC xxx SLC_DEFAULT 0` in response to an `SLC 0 SLC_DEFAULT 0` request, and just acknowledge whatever value the client requests to set the function to.

The `SLC_FORW2` character should be used only if `SLC_FORW1` is already in use.

FORWARDMASK and SLC_FORW1 and SLC_FORW2

To facilitate implementation of the client side, two methods of setting forwarding characters are provided. The `SLC_FORW1` and `SLC_FORW2` allow for the setting of two additional characters on which to forward buffered input data. Since many terminal drivers have the ability to set one or more line delimiters, it is fairly easy to support these without having to implement through the local terminal driver, rather than putting a terminal driver into TELNET. If the local terminal driver has functionality that maps easily into the `FORWARDMASK,` then it can also be easily supported. If the local terminal driver does not support that, then it would require more work to support `FORWARDMASK.`

Also note that the client side is required to forward data when it sees one of SLC_FORW1, SLC_FORW2, or FORWARDMASK characters, or when any normal line termination or special signal is encountered. The client side is also free to forward on other characters that it chooses. For example, if the server side sent a FORWARDMASK that asked for data to be forwarded on the first 20 control characters (ASCII codes 1 through 024), and the client side cannot have its local terminal driver forward on just the first 20 control characters but can have the local terminal driver forward on any control character (ASCII codes 1 through 039), then the client side could validly accept the FORWARDMASK, and forward on any control character. When in EDIT mode, care should be taken not to forward at random times, since once that data are forwarded, no more editing on the forwarded part of the line can be done. The only time (other than the normal times) that data should be forwarded when in EDIT mode would be if a single input line is too long to handle locally.

Valid and Invalid Modes and Values

At no time should DO LINEMODE be negotiated in both directions of the TELNET connection. The side that is the DO LINEMODE is considered to be the server side, and the side that is WILL LINEMODE is the client side.

At no time should SB LINEMODE DO/DONT FORWARDMASK, be sent unless DO LINEMODE was previously negotiated. At no time should SB LINEMODE WILL/WONT FORWARDMASK be sent unless WILL LINEMODE was previously negotiated.

If an ABORT, EOF, or SUSP request is received and the system does not support that functionality, it may simply be ignored.

Flushing Input and Output

When an IP, BRK, or ABORT is sent, it is usually desirable to be able to flush the input stream, and to flush output to the user until the IP, BRK, or ABORT is processed. The SLC_FLUSHIN and SLC_FLUSHOUT bits are used to indicate what action should be done. These bits are advisory only, but should be honored if possible. The standard method for processing the SLC_FLUSHIN is to use the TEL NET SYNC signal, and the SLC_FLUSHOUT is processed using the TIMING-MARK option. If both are to be sent, the IAC DM is sent before the DO TIMING-MARK. Thus, the sender would send IAC XXX IAC DM IAC DO TIMING-MARK, where XXX may be IP, BRK, or ABORT, or any other special character. The IAC DM is sent as TCP urgent data with the DM as the last (or only) data octet; this is used to flush the input stream. The IAC DO TIMING-MARK is used to determine when to stop flushing output; once it is sent, all data are discarded until an IAC WILL TIMING-MARK or an IAC WONT TIMING-MARK is received.

Since the SLC_FLUSHIN and SLC_FLUSHOUT bit are only advisory, the user interface should provide a method so that the user can override the sending (or not sending) of the SYNC and TIMING-MARK, but the default action should be to send them according to the SLC_FLUSHIN and SLC_FLUSHOUT bits.

Whenever an IAC AO is received, a SYNC must be returned. Whenever a SYNC is being processed (by the TCP connection going into urgent mode), all data must be discarded (but not TELNET commands!) until an IAC DM is found and the connection goes out of URGENT mode. (See Fig. 13.1.)

Receive	Response
f,SLC_DEFAULT,x	f,SLC_VALUE,v f,SLC_CANTCHANGE,v f,SLC_NOSUPPORT,x
f,SLC_VALUE,v	f,SLC_ACK|SLC_VALUE,v f,SLC_CANTCHANGE,w f,SLC_NOSUPPORT,x
f,SLC_CANTCHANGE,v	f,SLC_ACK|SLC_CANTCHANGE,v f,SLC_NOSUPPORT,x
f,SLC_NOSUPPORT,x	f,SLC_ACK|SLC_NOSUPPORT,x
x,SLC_ACK|x,x	No response

Examples of a Connection

In these examples, the symbolic names are used rather than the actual values, to make them readable. When two or more symbolic names are joined by a |, the actual value will be the logical OR of the values of the symbolic names. In the interest of clarity, for these examples the leading IAC and IAC SB sequences and the trailing IAC SE sequences have been omitted. Also, the `SLC_prefix` has been left off wherever it would normally occur.

Client	Server
WILL TOGGLE-FLOW-CONTROL	DO TOGGLE-FLOW-CONTROL
WILL LINEMODE	DO LINEMODE

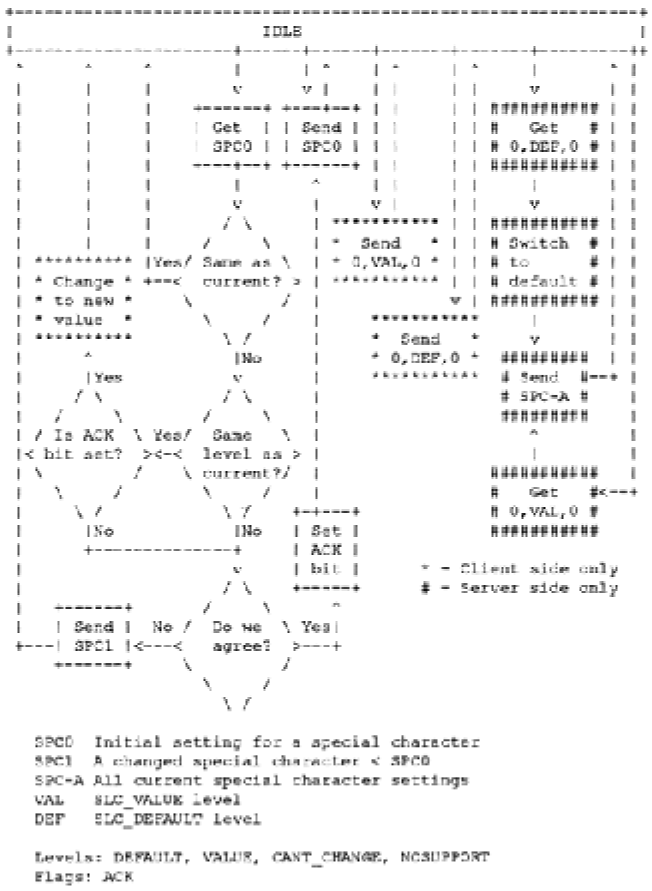


Figure 13.1
State diagram for SLC.

(Subnegotiation may now proceed in both directions. The client sends the list of special characters.)

LINEMODE SLC SYNCH DEFAULT 0
IP VALUE | FLUSHIN | FLUSHOUT 3 AO
VALUE 15 AYT DEFAULT 0 ABORT
VALUE | FLUSHIN | FLUSHOUT 28 EOF
VALUE 4 SUSP VALUE | FLUSHIN 26
EC VALUE 127 EL VALUE 21 EW
VALUE 23 RP VALUE 18 LNEXT
VALUE 22 XON VALUE 17 XOFF
VALUE 19

(Now that `LINEMODE` is enabled, the server sets the initial mode and acknowledges the special characters.)

LINEMODE MODE EDIT
LINEMODE SLC SYNCH NOSUPPORT 0 IP
VALUE | FLUSHIN | FLUSHOUT | ACK 3 AO
NOSUPPORT 0 AYT NOSUPPORT 0 ABORT
VALUE | FLUSHIN | FLUSHOUT | ACK 28 EOF
VALUE | ACK 4 SUSP NOSUPPORT 0 EC
VALUE | ACK 127 EL VALUE | ACK 21 EW
VALUE | ACK 23 RP VALUE | ACK 18 LNEXT
VALUE | ACK 22 XON VALUE | ACK 17 XOFF
VALUE | ACK 19

(The client gets the mode and ACK of the special characters, and acknowledges the mode and any special characters that the server changed.)

LINEMODE MODE EDIT | MODE_ACK
LINEMODE SLC SYNCH
NOSUPPORT | ACK 0 AO
NOSUPPORT | ACK 0 AYT | ACK NOSUP-PORT 0 SUSP OSUPPORT | ACK 0
“Login:”
“my_account”

(Turn off echo to the user.)

WILL ECHO
DO ECHO
“Password:”
“my_password”

(Turn back on echo to the user.)

WONT ECHO
DONT ECHO

(User does some stuff, and then runs an application that wants to use single character mode, doing its own echoing of characters, but keeping signal trapping on.)

WILL ECHO
DO ECHO
LINEMODE MODE TRAPSIG
LINEMODE MODE TRAPSIG | MODE_ACK

(Application finishes.)

WONT ECHO
DONT ECHO
LINEMODE MODE EDIT | TRAPSIG
LINEMODE MODE
EDIT | TRAPSIG | MODE_ACK

(Another application, that wants full control of everything.)

WILL ECHO
DO ECHO
LINEMODE MODE 0
LINEMODE MODE 0 | MODE_ACK

(Application finishes.)

WONT ECHO
DONT ECHO
LINEMODE MODE EDIT | TRAPSIG
LINEMODE MODE
EDIT | TRAPSIG | MODE_ACK

(The user changes his erase character to `^H.`)

LINEMODE SLC EC VALUE 8
LINEMODE SLC EC VALUE | ACK 8

(The user decides to revert to all the original client-side special characters.)

LINEMODE SLC SYNCH DEFAULT 0
IP VALUE | FLUSHIN | FLUSHOUT 3 AO
VALUE 15 AYT DEFAULT 0 ABORT
VALUE | FLUSHIN | FLUSHOUT 28 EOF
VALUE 4 SUSP VALUE | FLUSHIN 26
EC VALUE 127 EL VALUE 21 EW
VALUE 23 RP VALUE 18 LNEXT
VALUE 22 XON VALUE 17 XOFF
VALUE 19
LINEMODE SLC SYNCH NOSUPPORT 0 AO
NOSUPPORT 15 AYT NOSUPPORT 0 SUSP
NOSUPPORT | FLUSHIN 26 EC VALUE | ACK
127 EW VALUE | ACK 23 RP VALUE | ACK
18 LNEXT VALUE | ACK 22 XON
VALUE | ACK 17 XOFF VALUE | ACK 19
LINEMODE SLC SYNCH
NOSUPPORT | ACK 0 AO
NOSUPPORT | ACK 15 AYT
NOSUPPORT | ACK 0 SUSP
NOSUPPORT | ACK | FLUSHIN 26

(The user decides to import the remote-side default special characters.)

LINEMODE SLC 0 DEFAULT 0
LINEMODE SLC IP
VALUE | FLUSHIN | FLUSHOUT 3 ABORT
VALUE | FLUSHIN | FLUSHOUT 28 EOF
VALUE 4 EC VALUE 127 EL VALUE 21

(Since these are the same as the current local settings, no response is generated.)

This next example is what would happen if an editor were fired up and wanted to let the client side do the echoing and buffering of characters, but did not want it to do any line editing, and forward the data only when it got a control character. Note that we have preceded all the `0377`s in the FORWARD MASK with an `IAC.`

LINEMODE MODE 0
LINEMODE DO FORWARDMASK IAC 0377
IAC 0377 IAC 0377 IAC 0377 0 0 0 0 0 0 0 0 0 0 0 0 01
LINEMODE MODE 0
LINEMODE WILL FORWARDMASK

(Application runs to completion, and then things are to be set back to what they were before.)

LINEMODE MODE EDIT | TRAPSIG
LINEMODE DONT FORWARDMASK
LINEMODE MODE EDIT | TRAPSIG
LINEMODE WONT FORWARDMASK

13.6.3 TELNET Echo Option

This section reflects the standard for the Internet community using the TELNET echo option. The command name is `ECHO`; the command code is 1.

Command Meanings

`IAC WILL ECHO`	The sender of this command *requests* to begin, or confirms that it will now begin, echoing data characters it receives over the TELNET connection back to the sender of the data characters.
`IAC WONT ECHO`	The sender of this command *demands* to stop, or refuses to start, echoing the data characters it receives over the TELNET connection back to the sender of the data characters.
`IAC DO ECHO`	The sender of this command *requests* that the receiver of this command begin echoing, or confirms that the receiver of this command is expected to echo, data characters it receives over the TELNET connection back to the sender.
`IAC DONT ECHO`	The sender of this command *demands* that the receiver of this command stop, or not start, echoing data characters it receives over the TELNET connection.

TELNET Default

WONT ECHO
DONT ECHO

No echoing is done over the TELNET connection.

Motivation for the Option

The NVT has a printer and a keyboard which are nominally interconnected so that “echoes” need never traverse the network; in other words, the NVT nominally operates in a mode where characters typed on the keyboard are (by some means) locally turned around and printed on the printer. In highly interactive situations it is appropriate for the remote process (command language interpreter, etc.) to which the characters are being sent to control the way they are echoed on the printer. To support such interactive situations, a TELNET option is needed to allow the parties at both ends of the TELNET connection to agree that characters typed on an NVT keyboard are to be echoed by the party at the other end of the TELNET connection.

Description of the Option

When the echoing option is in effect, the party at the end performing the echoing is expected to transmit (echo) data characters it receives back to the sender of the data characters. The option does not require that the characters echoed be exactly the characters received (e.g., a number of systems echo the ASCII ESC character with something other than the ESC character). When the echoing option is not in effect, the receiver of data characters should not echo them back to the sender; this, of course, does not prevent the receiver from responding to data characters received.

The normal TELNET connection is two-way. That is, data flow in each direction on the connection independently; and neither, either, or both directions may be operating simultaneously in echo mode. There are five reasonable modes of operation for echoing on a connection pair:

```
              <-----------------------
Process 1                               Process 2
              ----------------------->
                    Neither end echoes
              <-----------------------
                                \
     Process 1            /  Process  2
          ----------------------->
   One end echoes for itself
             <-----------------------
                                \
     Process 1            /  Process  2
          ----------------------->
One end echoes for the other
             <-----------------------
                            \  /
   Process 1          /  \  Process 2
          ----------------------->
  Both ends echo for them-
                    selves
             <-----------------------
                           \  /
  Process 1          /  \  Process 2
          ----------------------->
One end echoes for both ends
```

This option provides the capability to decide whether either end will echo for the other. It does not, however, provide any control over whether an end echoes for itself; this decision must be left to the sole discretion of the systems at each end (although they may use information regarding the state of “remote” echoing negotiations in making this decision).

It should be noted that if *both* hosts enter the mode of echoing characters transmitted by the other host, then any character transmitted in either direction will be “echoed” back and forth indefinitely. Therefore, in each implementation care should be taken to ensure that if one site is echoing, echoing is not turned on at the other.

As discussed in the TELNET Protocol Specification, both parties to a full-duplex TELNET connection initially assume that each direction of the connection is being operated in the default mode which is nonecho (nonecho is not using this option, and the same as `DONT ECHO, WONT ECHO`).

If either party desires to echo characters to the other party or vice versa, that party gives the appropriate command (`WILL ECHO` or `DO ECHO`) and waits (and hopes) for acceptance of the option. If the request to operate the connection in echo mode is refused, then the connection continues to operate in nonecho mode. If the request to operate the connection in echo mode is accepted, the connection is operated in echo mode.

After a connection has been changed to echo mode, either party may demand that it revert to nonecho mode by giving the appropriate `DONT ECHO` or `WONT ECHO` command (which the other party must confirm, thereby allowing the connection to operate in nonecho mode). Just as each direction of the TELNET connection may be put in remote echoing mode independently, each direction of the TELNET connection must be removed from remote echoing mode separately.

Implementations of the echo option, as implementations of all other TELNET options, must follow the loop-preventing rules given in the General Considerations section of the TELNET Protocol Specification. Also, so that switches between echo and nonecho mode can be made with minimal confusion (momentary double echoing, etc.), switches in mode of operation should be made at times precisely coordinated with the reception and transmission of echo requests and demands. For instance, if one party responds to a `DO ECHO` with a `WILL ECHO,` all data characters received after the `DO ECHO` should be echoed and the `WILL ECHO` should immediately precede the first of the echoed characters.

The echoing option alone will normally not be sufficient to effect what is commonly understood to be remote computer echoing of characters typed on a terminal keyboard—the `SUPPRESS-GO AHEAD` option will normally have to be invoked in conjunction with the `ECHO` option to effect character-at-a-time remote echoing.

A Sample Implementation of the Option

A possible implementation for a simple user system called `UHOST` is described here. Suppose that for each user terminal, the `UHOST` would keep three state bits: (1) whether the terminal echoes for itself (`UHOST ECHO` always) or not (echo mode possible), (2) whether the (human) user prefers to operate in echo mode or in nonecho mode, and (3) whether the connection from this terminal to the server is in echo or nonecho mode. We will call these three bits P (physical), D (desired), and A (actual). When a terminal dials up the `UHOST` the P bit is set appropriately, the D bit is set equal to it, and the A bit is set to nonecho mode. The P and D bits may be manually reset by direct commands if the user so desires. For example, a user in Hawaii on a full-duplex terminal might choose not to operate in echo mode, regardless of the preference of a mainland server and thus should direct the `UHOST` to change the D bit from echo to nonecho mode.

When a connection is opened from the `UHOST` terminal to a server, the `UHOST` would send the server a `DO ECHO` command if the `MIN` (with `ECHO` less than `ECHO`) of the P and D bits is different from that of the A bit. If a `WONT ECHO` or `WILL ECHO` arrives from the server, the `UHOST` will set the A bit to the `MIN` of the received request, the P bit, and the D bit. If this changes the state of the A bit, the `UHOST` will send off the appropriate acknowledgment; if it does not, then the `UHOST` will send off the appropriate refusal if not changing meant that it had to deny the request (i.e., the `MIN` of the P and D bits was less than the received A request).

If, while a connection is open, the `UHOST` terminal user changes either the P or D bit, the `UHOST` will repeat the above tests and send off a `DO ECHO` or `DONT ECHO,` if necessary. When the connection is closed, the `UHOST` would reset the A bit to indicate `UHOST` echoing.

While the `UHOST`'s implementation would not involve `DO ECHO` or `DONT ECHO` commands being sent to the server except when the connection is opened or the user explicitly changes the echoing mode, bigger hosts might invoke such mode switches quite frequently. For instance, while a line-at-a-time system were running, the server might attempt to put the user in local echo mode by sending the `WONT ECHO` command to the user; but if a character-at-a-time system were running, the server might attempt to invoke remote echoing for the user by sending the `WILL ECHO` command to the user. Furthermore, while the `UHOST` will never send a `WILL ECHO` command and will send a `WONT ECHO` only to refuse a server sent a `DO ECHO` command, a server host might often send the `WILL` and `WONT ECHO` commands.

13.7 TELNET 5250 Interface and Additional Option Formats

This section explains the interface to the IBM 5250 TELNET implementation. The purpose here is to describe the details of the interface to enable a human user to implement a client TELNET which emulates an IBM 5250 workstation. It does not include all 5250 commands, aid codes, and other information specific to the 5250 data stream.

13.7.1 TELNET 5250 Options

No new TELNET options are defined for 5250 mode of operation. However, to enable 5250 mode, both client and server must agree to at least support the binary, end-of-record (EOR), and terminal-type TEL NET options. The complete list of 5250 terminal types is maintained in the Assigned Numbers RFC and includes the following:

IBM-5555-C01	24 × 80 double-byte character-set color display
IBM-5555-B01	24 × 80 double-byte character-Set (DBCS)
IBM-3477-FC	27 × 132 color display
IBM-3477-FG	27 × 132 monochrome display
IBM-3180-2	27 × 132 monochrome display
IBM-3179-2	24 × 80 color display
IBM-3196-A1	24 × 80 monochrome display
IBM-5292-2	24 × 80 color display
IBM-5291-1	24 × 80 monochrome display
IBM-5251-11	24 × 80 monochrome display

An example of a typical negotiation process to establish the 5250 mode of operation is shown below. In this example, the server initiates the negotiation by sending the `DO TERMINAL-TYPE` request.

```
SERVER: IAC DO TERMINAL-TYPE
CLIENT: IAC WILL TERMINAL-TYPE
SERVER: IAC SB TERMINAL-TYPE SEND IAC SE
CLIENT: IAC SB TERMINAL-TYPE IS IBM-5251-11 IAC SE
```

(The client has specified that its terminal type is an IBM-5251-11.)

SERVER: `IAC DO END-OF-RECORD`
CLIENT: `IAC WILL END-OF-RECORD`
SERVER: `IAC WILL END-OF-RECORD`
CLIENT: `IAC DO END-OF-RECORD`

(The server and the client have both agreed to transmit EORs.)

SERVER: `IAC DO TRANSMIT-BINARY`
CLIENT: `IAC WILL TRANSMIT-BINARY`
SERVER: `IAC WILL TRANSMIT-BINARY`
CLIENT: `IAC DO TRANSMIT-BINARY`

(The server and the client have both agreed to binary transmission.)

13.7.2 Data-stream Format

The actual data stream that is exchanged between client and server is composed of a header followed by the 5250 workstation data stream. [For information about this data stream refer to the *IBM 5250 Information Display System, Functions Reference Manual* (SA21-9247).] The header which prefixes the 5250 data stream was originally designed for the 5250 Display Station Pass-Through (DSPT) application. 5250 DSPT is an application similar to TELNET which runs on the IBM AS/400, System/36, and System/38 over an SNA network. This header is designed to be variable in length and is composed of two parts. The first, fixed part is always 6 octets long and has the following format:

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Logical Record Length      |          Record Type          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Reserved             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
Logical Record Length:   16 bits
```

This field indicates the length, in octets, of this logical record including the header length. The length is calculated *before* doubling any IAC characters in the data stream. The length does not include the <`IAC`><`EOR`> that is appended to the end of the data stream to mark the end of this logical record. The length is specified with the most significant octet first. For example, a length of 36 (decimals) would be specified as `'0024'`X.

Record Type: 16 bits

This field indicates the SNA record type. It should always be set to `'12A0'X` to indicate the General Data Stream (GDS) record type.

Reserved: 16 bits

This field is currently not used.

The second part of the header is designed to be variable in length. The length of this variable part is specified in the first octet. Currently this portion of the header will always be 4 octets long and has the following format:

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               |E|A| | | |S|T|H|               |               |
| Var Hdr Len   |R|T| | | |R|R|L|               |    Opcode     |
|               |R|H| | | |Q|Q|P|               |               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Var Hdr Len: 8 bits

The length, in octets, of the variable portion of the header. Currently this is always '04'X.

Flags: 16 bits

Bit 0: ERR This bit is set to indicate a data-stream output error. The negative response code is sent as data following the opcode field.

Bit 1: ATN This bit is set to indicate that the 5250 attention key was pressed.

Bits 2-4: * These bits are reserved (set to zero).

Bit 5: SRQ This bit is set to indicate that the 5250 SYSTEM REQUEST key was pressed.

Bit 6: TRQ This bit is set to indicate that the 5250 TEST REQUEST key was pressed.

Bit 7: HLP This bit is set to indicate the HELP in error-state function. The error code is sent as data following the header and is a four-digit packed-decimal number. For example, an error code of '0005'X indicates that the operator attempted to type in an area of the display that is not enabled for input.

Bits 8-15: * These bits are reserved (set to zero).

Opcode: 8 bits

This field contains the operation code. It is set to indicate the type of operation requested by the sender. The following are the valid values:

'00'X	No operation
'01'X	Invite operation
'02'X	Output only
'03'X	Put/get operation
'04'X	Save screen operation
'05'X	Restore screen operation
'06'X	Read immediate operation
'07'X	Reserved
'08'X	Read screen operation
'09'X	Reserved
'0A'X	Cancel invite operation
'0B'X	Turn on message light
'0C'X	Turn off message light

The actual 5250 workstation data stream will immediately follow the opcode field in the header and will be terminated by the `<IAC><EOR>` pair. For some operations the header will be immediately followed by an `<IAC><EOR>` without any 5250 workstation data stream in between. For example, the following request to turn on the message light could be sent by the server:

```
000A  12A0  0000  0400  00b  FFEF
|     |     |     |     |    |  |
|     |     |     |     |    |  End of Record Marker
|     |     |     |     |    |
|     |     |     |     |    Opcode = Turn On Message Light (‘OB’X)
|     |     |     |     |
|     |     |     |     Flags =‘0000’X
|     |     |     |
|     |     |     Variable Header Length =‘04’X
|     |     |
|     |     Reserved - Set to '0000'X
|     |
|     Record Type = General Data Stream (‘12A0’X)
|
Logical Record Length =‘000A’X for this record
```

In this example the requested operation is indicated by the opcode and there is no associated workstation data stream.

Examples of Data Flow

The following examples illustrate the flow of data between client and server for some of the more common operations. These examples are intended to show the order in which the logical records are exchanged between client and server, as well as the content and hexadecimal (hex) representation of these records. The way in which a client implements the various operations will differ between implementations; those details are not discussed here. In these examples, when the value of a field is dependent on the length of the screen data for a particular logical record, it will be represented as ‘LLLL’.

1. `Device query` *example.* A query command may be sent by the server system in order to determine the attributes of the device it is talking to. When a client receives a query command, it must send the query reply back to the server.

```
Server:Sends Write Structured     001112A0 00000400 000304F3 0005D970
       Field Query command        00FFEF
Client:Responds with a query 004712A0     00000400 00000000 88003AD9
       Reply, in this case, 70800600 01030000 00000000 00000000
       for a 3180-2       00000000 00000001 F3F1F8F0 F0F0F202
       00000061 50000100 00000018 11000000
       00000000 000000FF EF
```

2. `Cancel invite` *example.* The server will send a `cancel invite` when it needs to reverse the normal flow direction. When a client receives a `cancel invite,` it should reply with a `cancel invite` and not send any user data until the server has once again “invited” the workstation. A workstation is said to be “invited” when the server has sent a read command to the client. The `cancel invite` flow is as follows:

Server: Sends header with the 000A12A0 00000400 000AFFEF
Opcode = Cancel Invite
Client: Sends header with the 000A12A0 00000400 000AFFEF
Opcode = Cancel Invite to indicate that the workstation is no longer invited

3. `System Request` *example.* The 5250 `System Request` operation is invoked when a client wants to interrupt the server job to perform some function. In a typical scenario, the user presses the SYSTEM REQUEST key, or whatever key is mapped to such a key, which would cause the client TELNET to initiate the following flow:

Client: Sends header with the 000A12A0 00000404 0000FFEF
System Request bit set.

Note: *A client might include user data in this record following the header. The server would interpret these data as an option to be selected from the system request menu. If this were the case, the server would not send the system request menu and the flow would continue per the option selected. For this example, the client does not send any user data and the flow would continue as follows:*

```
Server:   Sends header with the 000A12A0 00000400 000AFFEF
          Opcode = Cancel Invite
Client:Sends header with the 000A12A0 00000400 000AFFEF
          Opcode = Cancel Invite
          to indicate that the workstation is no
          longer invited
Server:   Sends Save (Immediate)   000C12A0 00000400 00040402 FFEF
          command with
          Opcode = Save Screen
Client: Sends the screen image   LLLL12A0 00000400 00040412
          to be saved   <Screen Image> FFEF
Server:   Sends System Request   LLLL12A0 00000400 0003
          menu with   <System Request Menu> FFEF
          Opcode = Put/Get
Client:   Sends User Input to   LLLL12A0 00000400 0000
          the Sys Req menu   <User Input> FFEF
```

Note: *What happens next will depend on the system-request option selected by the user. After any system-request processing has completed, the server will continue with the following restore operation:*

```
Server:   Sends the saved          LLLL12A0 00000400 00050412
          screen to be restored,   <Saved Screen> FFEF
          Opcode = Restore Screen
          [No reply is necessary from the client]
Server:   Sends Read Modified   000E12A0 00000400 00010452
0000FFEF
          Data Tag (MDT) command,
          opcode = Invite
```

At this point the client would “invite” the workstation and enter the state it was in before the SYSTEM REQUEST key was pressed.

5250 Data-stream Enhancements

This section is intended as an addendum to the *IBM 5250 Information Display System, Functions Reference Manual* (SA21-9247-6). Enhancements to the 5250 data stream which are not yet documented in the current version of that manual (SA21-9247-6) are described in this section, as are corrections to erroneous information contained therein. The specific corrections and enhancements, with approximate page-number references to the SA21-9247-6 manual, are given in the following paragraph.

1. *Errors or inconsistencies in SA21-9247-6.* The Insert Cursor (IC) order on pages 2-136 and 2-137 is incorrectly listed with a value of Hex 03; the correct value is Hex 13.

On page 2-137, the listed “Restrictions” for the Insert Cursor, Repeat to Address, and Set Buffer Address orders should be updated to describe how Row and Column values must be valid for the current display screen size (either 24 × 80 or 27 × 132 pixels).

2. *Enhancements to existing 5250 data-stream commands and orders.* A new flag is added to the second byte of the Control Character on page 2-40. This flag is used to specify whether the cursor should be moved at the end of the Write to Display processing. Bit 1 of the second byte, which was previously reserved, will now be used for this flag. If bit 1 is a 0, the cursor continues to be moved to the system IC address on a Lock-to-Unlock keyboard transition. If bit 1 is a 1, the cursor is not moved. A new Field Control Word (FCW) will be added on page 2-65 to indicate that an entry field contains transparent data. This means that the entry-field contents are sent from the display screen directly to the host at read time with no formatting. Therefore, an entry field can contain any values (Hex 00 to Hex FF). A transparent field is indicated by a Hex 84xx FCW, where xx is any value. Unpredictable results will occur if a field is defined as both signed numeric and a transparent field. The Read Immediate, Read Input Fields, and Read MDT Fields commands have been enhanced to include support for transparent fields (page 2-5). If a transparent FCW is found for an input field, the field data are not formatted (e.g., nulls are not converted to blanks). The restriction listed for the Set Buffer Address (SBA) order (page 2-138) on the column address equal to zero is no longer always the case. A reference to Start of Field (SF) Row 1/Column 1 field support should be made. A note should be added in SF to describe Row 1/Column 1 field support. A Row 1/Column 1 field is defined by an SBA of Row 1/Column 0, followed by an SF. For a Row 1/Column 1 input field, the first input-capable position is Row 1/Column 1. If the SF defines an input field, the screen attribute is not allowed to be nondisplay. Writing of the screen attribute is suppressed for a Row 1/Column 1 field and the attribute discarded.

3. *New 5250 data-stream commands and orders.* The Read MDT Fields Alternate input command has been added. It is the same as the Read MDT Fields command except the command is indicated by an `X'82'.` Leading and embedded nulls within the field remain as nulls. The Read MDT Fields Immediate Alternate input command has been added. It is the same as the Read MDT Fields Alternate command except the command is indicated by an `X'83'.` The command is an immediate read command like Read Immediate; therefore, no control characters follow the command byte, field data are returned immediately, and the aid code is X’00’.

The Move Cursor order (MC) has been added. The MC order moves the cursor to the location specified by the two bytes following the order. Byte 1 gives the row address, and byte 2 gives the column address. The MC order is useful when the cursor is to be moved without affecting the system IC address. The MC order is unaffected by the Write to Display control character values including the “Leave Cursor” flag (CC1 bit 1). If more than one MC or IC are found in the data stream, the cursor will move to the address specified in the last MC or IC.

1. *Restrictions.* A parameter error will be posted when there are fewer than 2 bytes following the order, the row address is zero or greater than the number of rows on the display screen, or the column address is zero or greater than the number of columns on the display screen.

2. *Format*

Move Cursor Order	Byte 1	Byte 2
X'14'	Row Address	Column Address

3. *Results.* The address specified by the MC order is used to move the cursor when the Write to Display is completed.

The Transparent Data order (TD) has been added (page 2-137). The TD order is followed by two length bytes and transparent data. The transparent data are written to the display screen at the current display address; any values (Hex 00 to Hex FF) are allowed in the transparent data. All length values are valid as long as the end of the display screen is not overwritten.

1. *Restrictions.* A parameter error will be posted when there are fewer then 2 bytes following the order, there are fewer bytes in the data stream than specified in the length field, or one attempts to write beyond the end of the display screen.

2. *Format*

TD Order	Bytes 1 and 2	Bytes 3 to ?
X'10'	Length of transparent data (not counting length bytes)	Transparent data

3. *Results.* The transparent data are written to the display.

The Query command is a new input command (page 2-5) and is used by the server to obtain information on the functional capabilities of the client 5250 display. When the client receives a Query command, the client sends a Query Reply describing its capabilities back to the server.

The Query command must follow an Escape (`'04'X`) and Write Structured Field command (`'F3'X`). The format of the Query command is as follows:

Byte	**Value**	**Description**
0–1	X'0005'	Length of command
2	X'D9'	Command Class
3	X'70'	Command Type—Query
4	X'00'	Flag Byte
Bit 0:	B'0'	-Query Command
Bit 1–7:		-Reserved (set to zero)

The format of the Query Reply is as follows:

Byte	**Value**	**Description**
0–1	X'0000'	Cursor Row/Column (set to zero)

2	X’88’	Inbound Write Structured Field Aid
3–4	X’003A’	Length of Query Reply
5	X’D9’	Command Class
6	X’70’	Command Type - Query
7	X’80’	Flag Byte
	Bit 0:	B’1’ -Query Reply
	Bit 1-7:	-Reserved (set to zero)
8–9		Controller Hardware Class
	X’0001’	-Local Twinax Controller
	X’0061’	-Local ASCII Controller
	X’0101’	-SDLC/X.21/X.25 Twinax Controller
		(5394 emulating a 5294)
	X’0103’	-SDLC/X.21/X.25 Twinax Controller (5394)
	X’0200’	-PC DOS non-DBCS WSF
	X’0300’	-OS/2 non-DBCS WSF
	X’0400’	-PC DOS DBCS WSF
	X’0500’	-OS/2 DBCS WSF
	X’0600’	-Other WSF or any other 5250 Emulator
10–12		Controller Code Level
	X’010300’	-For example, Version 1 Rel 3.0
13–28	X’00’	Reserved (set to zero)
29		Device Type
	X’01’	-5250 Display or 5250 Emulation
30–33	C’cccc’	Device Type (e.g., 3180 for 3180 Mod 2)
34–36	C’ccc’	Device Model (e.g., 002 for 3180 Mod 2)
37		Keyboard ID
	X’02’	-Standard Keyboard
	X’82’	-G Keyboard
38	X’00’	Extended Keyboard ID
39	X’00’	Reserved
40–43	X’xxxxxxxx’	Display Serial Number
44–45		Maximum number of input fields
	X’0100’	-Typically = 256 input fields
46–48	X’00’	Reserved (set to zero)
49–53		Controller/Display Capability
	Bit 0–1:	B’00’ -No Row 1/Col 1 support

	B’01’	-Row 1/Col 1 support
	Bit 2:	B’0’ - No Read MDT Alternate Command support
	B’1’	-Read MDT Alternate Command support
	Bit 3:	B’0’ - Display does not have PA1/PA2 support
	B’1’	- Display does have PA1/PA2 support
	Bit 4:	B’0’ - Display does not have PA3 support
	B’1’	- Display does have PA3 support
	Bit 5:	B’0’ - Display does not have Cursor Select support
	B’1’	-Display does have Cursor Select support
	Bit 6:	B’0’ - Display does not have Move Cursor Order support
	B’1’	- Display does have Move Cursor Order support
	Bit 7:	B’0’ - No Read MDT Immediate Alt Command support
	B’1’	- Read MDT Immediate Alt Command support
50	Bit 0–3:	B’0001’ - 24 x 80 Screen Size
	B’0011’	- Capable of 24 x 80 and 27 x 132
	Bit 4:	B’0’ - No light pen support
	B’1’	- Light pen support
	Bit 5:	B’0’ - No Mag Stripe Reader support
	B’1’	- Mag Stripe Reader support
	Bit 6–7:	B’00’ - Mono display
	B’01’	- 5292/3179 style color, including color PCs
51	X’00’	- Reserved
52	Bit 0–2:	B’000’ - No Double Byte Character Set (DBCS) capability
	B’001’	- Presentation screen DBCS capability only
	Bit 3–7:	B’00000’ - Reserved
53	Bit 0–2:	B’000’ - No graphics capability
	B’001’	- 5292-2 style graphics
	Bit 3–7:	B’00000’ - Reserved
54–60	X’00’	Reserved (set to zero)

13.7.3 TELNET Terminal-type Option

The command name for this option is `TERMINAL-TYPE`; the code is 24.

Command Meanings

`IAC WILL TERMINAL-TYPE`	Sender is willing to send terminal-type information in a subsequent subnegotiation.
`IAC WONT TERMINAL-TYPE`	Sender refuses to send terminal-type information.
`IAC DO TERMINAL-TYPE`	Sender is willing to receive terminal-type information in a subsequent subnegotiation.
`IAC DONT TERMINAL-TYPE`	Sender refuses to accept terminal-type information.
`IAC SB TERMINAL-TYPE SEND IAC SE`	Server requests client to transmit its next terminal type, and switch emulation modes (if more than one terminal type is supported). The code for SEND is 1.
`IAC SB TERMINAL-TYPE IS...IAC SE`	Client is stating the name of its current (or only) terminal type. The code for IS is 0. (See text below.)

TELNET Default

`WONT TERMINAL-TYPE`	Terminal-type information will not be exchanged.
`DONT TERMINAL-TYPE`	Terminal-type information will not be exchanged.

Motivation for the Option

On most machines with bitmapped displays (e.g., PCs and graphics workstations), a client terminal emulation program is used to simulate a conventional ASCII terminal. Most of these programs have multiple emulation modes, frequently with widely varying characteristics. Likewise, modern host system software and applications can deal with a variety of terminal types. What is needed is a means for the client to present a list of available terminal emulation modes to the server, from which the server can select the one it prefers (for arbitrary reasons). There is also need for a mechanism to change emulation modes during the course of a session, perhaps according to the needs of applications programs.

Existing terminal-type passing mechanisms within TELNET were not designed with multiple emulation modes in mind. Although multiple names are allowed, they are assumed to be synonyms. Emulation mode changes are not defined, and the list of modes can be scanned only once.

This document defines a simple extension to the existing mechanisms, which meets both criteria described above. It makes one assumption about the behavior of implementations coded to the previous standard in order to obtain full backward-compatibility.

Description of the Option

Willingness to exchange terminal-type information is agreed on via conventional TELNET option negotiation. `WILL` and `DO` are used only to obtain and grant permission for future discussion. The actual exchange of status information occurs within option subcommands (`IAC SB TERMINAL-TYPE...`).

Once the two hosts have exchanged a `WILL` and a `DO,` the sender of the `DO TERMINAL-TYPE` (the server) is free to request type information. Only the server may send requests (`IAC SB TERMINAL-TYPE SEND IAC SE`), and only the client may transmit actual type information (within an `IAC SB TERMINAL-TYPE IS...AC SE` command). Terminal-type information may not be sent spontaneously, but only in response to a request. The terminal-type information is an NVT ASCII string; within this string, upper- and lowercase are considered equivalent.

The transmission of terminal-type information by the TELNET client in response to a query from the TELNET server implies that the client must simultaneously change emulation mode, unless the terminal type sent is a synonym of the preceding terminal type, or there are other prerequisites for entering the new regime (e.g., having agreed on the TELNET binary option). Receipt of such information by the TELNET server does not imply any immediate change of processing. However, the information may be passed to a process, which may alter the data it sends to suit the particular characteristics of the terminal. For example, some operating systems have a terminal driver that accepts a code indicating the type of terminal being driven. Using the `TERMINAL-TYPE` and `BINARY` options, a TELNET server program on such a system could arrange to have terminals driven as if they were directly connected, including special functions not available to a standard network virtual terminal.

This specification is deliberately asymmetrical. It is assumed that server operating systems and applications in general cannot change terminal types at arbitrary points in a session. Thus, the client may only send a new type (and potentially change emulation modes) when the server requests that it do so.

Implementation Considerations

The terminal-type information may be any NVT ASCII string meaningful to both ends of the negotiation. The list of terminal-type names in the Internet Assigned Number Authority (IANA) is intended to minimize confusion caused by alternative “spellings” of the terminal type. For example, confusion would arise if one party were to call a terminal “IBM3278-2” while the other called it “IBM-3278/2.” There is no negative acknowledgment for a terminal type that is not understood, but certain other options (such as switching to `BINARY` mode) may be refused if a valid terminal-type name has not been specified.

In some cases, either a particular terminal may be known by more than one name, for example, a specific type and a more generic type, or the client may be a workstation with integrated display capable of emulating more than one kind of terminal. In such cases, the sender of the `TERMINAL-TYPE IS` command should reply to successive `TERMINAL-TYPE` SEND commands with the various names. In this way, a TELNET server that does not understand the first response can prompt for alternatives. If different terminal emulations are supported by the client, the mode of the emulator must be changed to match the last type sent, unless the particular emulation has other TELNET options (e.g., `BINARY`) as prerequisites (in which case the emulation will switch to the last type sent when the prerequisite is fulfilled). When types are synonyms, they should be sent in order from most to least specific.

When the server (the receiver of the `TERMINAL-TYPE IS`) receives the same response two consecutive times, this indicates the end of the list of available types. Similarly, the client should indicate it has sent all available names by repeating the last one sent. If an additional request is received, this indicates that the server (the sender of the `IS`) wishes to return to the top of the list of available types (probably to select the least of *N* evils).

Server implementations conforming to the previous standard will cease sending `TERMINAL-TYPE` SEND commands after receiving the same response two consecutive times, which will work according to the old standard. It is assumed that client implementations conforming to the previous standard will send the last type on the list in response to a third query (as well as the second). New-style servers must recognize this and not send more queries.

The type `UNKNOWN` should be used if the type of the terminal is unknown or unlikely to be recognized by the other party. The maximum length of a terminal type name is 40 characters.

User Interfaces

TELNET clients and servers conforming to this specification should provide the following functions in their user interfaces:

1. Clients supporting multiple emulation modes should allow the user to specify which of the modes is preferred (which name is sent first), prior to connection establishment. The order of the names sent cannot be changed after negotiation has begun. This initial mode will also become the default with servers which do not support `TERMINAL-TYPE.`

2. Servers should store the current terminal-type name and the list of available names in a manner such that they are accessible to both the user (via a keyboard command) and any applications which need the information. In addition, there should be a corresponding mechanism to request a change of terminal types, by initiating a series of `SEND/IS` subnegotiations.

TELNET Implementation Examples

In the following example, the server finds the first type acceptable.

SERVER: `IAC DO TERMINAL-TYPE`
CLIENT: `IAC WILL TERMINAL-TYPE`

(Server may now request a terminal type at any time.)

SERVER: `IAC SB TERMINAL-TYPE SEND IAC SE`
CLIENT: `IAC SB TERMINAL-TYPE IS IBM-3278-2 IAC SE`

In the next example, the server requests additional terminal types, and accepts the second (and last on the client’s list) type sent:

SERVER: `IAC DO TERMINAL-TYPE`
CLIENT: `IAC WILL TERMINAL-TYPE`

(Server may now request a terminal type at any time.)

```
SERVER: IAC SB TERMINAL-TYPE SEND IAC SE
CLIENT: IAC SB TERMINAL-TYPE IS ZENITH-H19 IAC SE
SERVER: IAC SB TERMINAL-TYPE SEND IAC SE
CLIENT: IAC SB TERMINAL-TYPE IS UNKNOWN IAC SE
SERVER: IAC SB TERMINAL-TYPE SEND IAC SE
CLIENT: IAC SB TERMINAL-TYPE IS UNKNOWN IAC SE
```

In the following example, the server requests additional terminal types, and proceeds beyond the end of the list, to select the first type offered by the client (new-type client and server):

```
SERVER: IAC DO TERMINAL-TYPE
CLIENT: IAC WILL TERMINAL-TYPE
```

(Server may now request a terminal type at any time.)

```
SERVER: IAC SB TERMINAL-TYPE SEND IAC SE
CLIENT: IAC SB TERMINAL-TYPE IS DEC-VT220 IAC SE
SERVER: IAC SB TERMINAL-TYPE SEND IAC SE
CLIENT: IAC SB TERMINAL-TYPE IS DEC-VT100 IAC SE
SERVER: IAC SB TERMINAL-TYPE SEND IAC SE
CLIENT: IAC SB TERMINAL-TYPE IS DEC-VT52 IAC SE
SERVER: IAC SB TERMINAL-TYPE SEND IAC SE
CLIENT: IAC SB TERMINAL-TYPE IS DEC-VT52 IAC SE
SERVER: IAC SB TERMINAL-TYPE SEND IAC SE
CLIENT: IAC SB TERMINAL-TYPE IS DEC-VT220 IAC SE
```

13.7.4 TELNET CHARSET Option

This section explains a mechanism for passing character set and translation information between a TELNET client and server. Use of this mechanism enables an application used by a TELNET user to send and receive data in the correct character set. Either side can (subject to option negotiation) at any time request that a (new) character set be used.

Command Names and Codes

```
CHARSET       42
REQUEST       01
ACCEPTED       02
REJECTED       03
TTABLE-IS       04
TTABLE-REJECTED       05
TTABLE-ACK       06
TTABLE-NAK       07
```

For convenience, standard TELNET text and codes for commands used are presented here. All TELNET commands consist of at least a 2-byte sequence: the “interpret as command” (IAC) escape character followed by the code for the command. The commands dealing with option negotiation are 3-byte sequences; the third byte being the code for the option referenced. Only the `iac` need be doubled to be sent as data, and the other 255 codes may be passed transparently. The following are some defined TELNET commands. Note that these codes and code sequences have the indicated meaning only when immediately preceded by an IAC.

Name	Code	Meaning
SE	240	End of subnegotiation parameters
SB	250	Indicates that what follows is subnegotiation of the indicated option
WILL	251	Indicates the desire to begin performing, or confirmation that you are now performing, the indicated option
WONT	252	Indicates the refusal to perform, or continue performing, the indicated option
DO	253	Indicates the request that the other party perform, or confirmation that you are expecting the other party to perform, the indicated option
DONT	254	Indicates the demand that the other party stop performing, or confirmation that you are no longer expecting the other party to perform, the indicated option
IAC	255	Data Byte 255

Command Meanings

A very simple metasyntax is used, where most tokens represent previously defined items (such as IAC); angle brackets (<>) are used for items to be further defined; curly braces ({}) are used around optional items; ellipses represent repeated sequences of items; and quotes are used for literal strings.

`IAC WILL CHARSET`	The sender *requests* permission to, or *agrees* to, use CHARSET option subnegotiation to choose a character set.
`IAC WONT CHARSET`	The sender *refuses* to use CHARSET option subnegotiation to choose a character set.
`IAC DO CHARSET`	The sender *requests* that, or *agrees* to have, the other side use CHARSET option subnegotiation to choose a character set.
`IAC DONT CHARSET`	The sender *demands* that the other side not use the CHARSET option subnegotiation.

IAC SB CHARSET REQUEST { "[TTABLE]" <Version>} <char set list> IAC SE
Char set list: <sep> <character set> {. . .<sep> <character set> }

This message initiates a new `CHARSET` subnegotiation. It can be sent only by a side that has received a `DO CHARSET` message and sent a `WILL CHARSET` message (in either order). The sender requests that all text sent to and by it be encoded in one of the specified character sets.

If the string `[TTABLE]` appears, the sender is willing to accept a mapping (translation table) between any character set listed in <charset list> and any character set desired by the receiver.

<Version> is an octet whose binary value is the highest-version level of the `TTABLE-IS` message which can be sent in response. This field must not be zero. See the `TTABLE-IS` message for the permitted version values.

<Charset list> is a sequence of 7-bit ASCII printable characters. The first octet defines the separator character (which must not appear within any character set). It is terminated by the `IAC SE` sequence. Case is not significant. It consists of one or more character sets. The character sets should appear in order of preference (most preferred first).

<Sep> is a separator octet, the value of which is chosen by the sender. Examples include a space or a semicolon. Any value other than `IAC` is allowed. The obvious choice is a space or any other punctuation symbol which does not appear in any of the character set names.

<Character set> is a sequence of 7-bit ASCII printable characters. Case is not significant.

If a requested character set name does not start with `X-` or `x-`, it *must* be registered with the Internet Assigned Number Authority (IANA).

The receiver responds in one of four ways:

1. If the receiver is already sending text to and expecting text from the sender to be encoded in one of the specified character sets, it sends a positive acknowledgment (`CHARSET ACCEPTED`); it *must not* ignore the message.

2. If the receiver is capable of handling at least one of the specified character sets, it can respond with a positive acknowledgment for one of the requested character sets. Normally, it should pick the first set it is capable of handling but may choose one on the basis of its own preferences. After doing so, each side *must* encode subsequent text in the specified character set.

3. If the string `[TTABLE]` is present, and the receiver prefers to use a character set not included in <char set list>, and is capable of doing so, it can send a translate-table (`TTABLE-IS`) response.

4. If the receiver is not capable of handling any of the specified character sets, it sends a negative acknowledgment (`CHARSET REJECTED`).

Because it is not valid to reply to a `CHARSET REQUEST` message with another `CHARSET REQUEST` message, if a `CHARSET REQUEST` message is received after the sending side has just sent one, then both sides have sent them simultaneously. In this case, the server side *must* issue a negative acknowledgment and the client side *must* respond to the one from the server.

IAC SB CHARSET ACCEPTED <Charset> IAC SE

This is a positive acknowledgment response to a `CHARSET REQUEST` message; the receiver of this message acknowledges its receipt and accepts the indicated character set. <Charset> is a character sequence identical to one of the character sets in the `CHARSET REQUEST` message. It is terminated by the `IAC SE` sequence.

Text messages which follow this response must now be coded in the indicated character set. This message terminates the current `CHARSET` subnegotiation.

IAC SB CHARSET REJECTED IAC SE

This is a negative acknowledgment response to a `CHARSET REQUEST` message; the receiver of the `CHARSET REQUEST` message acknowledges its receipt but refuses to use any of the requested character sets. Messages cannot be sent in any of the indicated character sets. This message can also be sent by the sender of a `TTABLE-IS` message, if multiple `TTABLE-NAK` messages were sent in response. This message terminates the current `CHARSET` subnegotiation.

IAC SB CHARSET TTABLE-IS <version> <syntax for version> IAC SE

In response to a `CHARSET REQUEST` message in which `[TTABLE]` was specified, the receiver of the `CHARSET REQUEST` message acknowledges its receipt and is transmitting a pair of tables which define the mapping between specified character sets. <Version> is an octet whose binary value is the version level of this `TTABLE-IS` message. Different versions have different syntax. The lowest version level is one (zero is not valid). The current highest version level is also one. This field is provided so that future versions of the `TTABLE-SEND` message can be specified, for example, to handle character sets for which there is no simple one-to-one character-for-character translation. This might include some forms of multi-octet character sets for which translation algorithms or subsets need to be sent.

Syntax for version 1 is as follows:

<sep> <char set name 1> <sep> < char size 1> < char count 1> <char set name 2> <sep> <char size 2> <char count 2> <map 1> <map 2>

<sep> is a separator octet, the value of which is chosen by the sender. Examples include a space or a semicolon. Any value other than `IAC` is allowed. The obvious choice is a space or any other punctuation symbol which does not appear in either of the character set names.

<Char set name 1> and <Char set name 2>

These are sequences of 7-bit ASCII printable characters which identify the two character sets for which a mapping is being specified. Each is terminated by <sep>. Case is not significant. If a character set name does not start with `X-` or `x-`, it *must* be registered with IANA. <Charset name 1> *must* be chosen from the <char set list> in the `CHARSET REQUEST` message. <Char set name 2> can be arbitrarily chosen. Text on the wire *must* be encoded using <char set name 2>.

<Char size 1> and <char size 2>

These are single octets each. The binary value of the octet is the number of bits nominally required for each character in the corresponding table. It *should* be a multiple of eight.

<Char count 1> and <char count 2>

These are each 3-octet binary fields in network byte order. Each specifies how many characters (of the maximum `2**`<char size>) are being transmitted in the corresponding map.

<Map1> and <Map 2>

These consist of the corresponding <char count> number of characters. These characters form a mapping from all or part of the characters in one of the specified character sets to the correct characters in the other character, set. If the indicated <char count> is less than `2**`<char size>, the first <char count> characters are being mapped, and the remaining characters are assumed to not be changed (and thus map to themselves). In other words, each map contains characters 0 through <char count> -1. <Map 1> maps from <char set name 1> to <char set name 2>. <Map 2> maps from <char set name 2> to <char set name 1>. Translation between the character sets is thus an obvious process of using the binary value of a character as an index into the appropriate map. The character at that index replaces the original character. If the index exceeds the <char count> for the map, no translation is performed for the character.

Since TELNET works in octets, it is possible for octets of value 255 to appear “spontaneously” when using multioctet or non-8-bit characters. All octets of value 255 (other than `IAC`) *must* be quoted to conform with TELNET requirements. This applies even to octets within a table, or text in a multioctet character set.

IAC SB CHARSET TTABLE-ACK IAC SE

The sender acknowledges the successful receipt of the translate table. Text messages which follow this response must now be coded in the character set specified as <char set name 2> of the `TTABLE-IS` message. This message terminates the current `CHARSET` subnegotiation.

IAC SB CHARSET TTABLE-NAK IAC SE

The sender reports the unsuccessful receipt of the translate table and requests that it be resent. If subsequent transmission attempts also fail, a `TTABLE-REJECTED` or `CHARSET REJECTED` message (depending on which side sends it) should be sent instead of additional futile `TTABLE-IS` and `TTABLE-NAK` messages.

IAC SB CHARSET TTABLE-REJECTED IAC SE

In response to a `TTABLE-IS` message, the receiver of the `TTABLE-IS` message acknowledges its receipt and indicates it is unable to handle it. This message terminates the current `CHARSET` subnegotiation.

Any system which supports the `CHARSET` option *must* fully support the `CHARSET REQUEST, ACCEPTED, REJECTED,` and `TTABLE-REJECTED` subnegotiation messages. It may optionally fully support the `TTABLE-IS, TTABLE-ACK,` and `TTABLE-NAK` messages. If it does fully support the `TTABLE-IS` message, it *must* also fully support the `TTABLE-ACK` and `TTABLE-NAK` messages. The *default* is

WONT CHARSET
WONT CHARSET

Motivation for the Option

Many TELNET sessions need to transmit data which is not in 7-bit ASCII. This is usually done by negotiating `BINA RY,` and using local conventions (or terminal-type kludges) to determine the character set of the data. However, such methods seldom interoperate well, and have difficulties when multiple character sets need to be supported by different sessions.

Many computer systems now utilize a variety of character sets. Increasingly, a server computer needs to document character sets or translate transmissions and receptions using different pairs of character sets on a per-application or per-connection basis. This is becoming more common as client and server computers become more geographically disperse (and as servers are consolidated into large hubs, serving increasingly wide areas). For files, databases, and so on to contain correct data, the server must determine the character set in which the user is sending and the character set in which the application expects to receive.

In some cases, it is sufficient to determine the character set of the end user (because every application on the server expects to use the same character set, or because applications can handle the user's character set), but in other cases different server applications expect to use different character sets. In the former case, an initial `CHARSET` subnegotiation suffices. In the latter case, the server may need to initiate additional `CHARSET` subnegotiations as the user switches between applications.

At a minimum, the option described in this memo allows both sides to be clear as to which character set is being used. A minimal implementation would have the server send `DO CHARSET,` and the client send `WILL CHARSET` and `CHARSET REQUEST.` The server could then communicate the client's character set to applications using whatever means are appropriate. Such a server might refuse subsequent `CHARSET REQUEST` messages from the client (e.g., if it lacked the ability to communicate changed character set information to applications). Another system might have a method in which various applications could communicate to the TELNET server their character set needs and abilities, which the server would handle by initiating new `CHARSET REQUEST` negotiations as appropriate.

In some cases, servers may have a large set of clients which tend to connect often (such as daily) and over a long period of time (e.g., years). The server administrators may strongly prefer that the servers not do character-set translation (to save CPU cycles when serving very large numbers of users). To avoid manually configuring each copy of the user TELNET software, the administrators might prefer that the software supports translate tables. (If the client software received a translate table from the server and stored it, the table would need to be sent only once.)

Description of the Option

When the client TELNET program is able to determine the user's character set, it should offer to specify the character set by sending `IAC WILL CHARSET.`

If the server system is able to make use of this information, it replies with `IAC DO CHARSET.` The client TELNET is then free to request a character set in a subnegotiation at any time.

Likewise, when the server is able to determine the expected character set(s) of the user's application(s), it should send `IAC DO CHARSET` to request that the client system specify the character set it is using. Or the server could send `IAC WILL CHARSET` to offer to specify the character sets.

Once a character set has been determined, the server can either perform the translation between the user and application character sets itself, or request by additional `CHARSET` subnegotiations that the client system do so.

Once it has been established that both sides are capable of character-set negotiation (i.e., each side has received either a `WILL CHARSET` or a `DO CHARSET` message, and has also sent either a `DO CHARSET` or a `WILL CHARSET` message), subnegotiations can be requested at any time by whichever side has sent a `WILL CHARSET` message and also received a `DO CHARSET` message (this may be either or both sides). Once a `CHARSET` subnegotiation has started, it must be completed before additional `CHARSET` subnegotiations can be started (there must never be more than one `CHARSET` subnegotiation active at any given time). When a subnegotiation has completed, additional subnegotiations can be started at any time.

If either side violates this rule and attempts to start a `CHARSET` subnegotiation while one is already active, the other side *must* reject the new subnegotiation by sending a `CHARSET REJECTED` message. Receipt of a `CHARSET REJECTED` or `TTABLE-REJECTED` message terminates the subnegotiation, leaving the character set unchanged. Receipt of a `CHARSET ACCEPTED` or `TTABLE-ACK` message terminates the subnegotiation, with the new character set in force.

In some cases, both the server and the client systems are able to perform translations and to send and receive in the character set(s) expected by the other side. In such cases, either side can request that the other use the character set it prefers. When both sides simultaneously make such a request (send `CHARSET REQUEST` messages), the server *must* reject the client's request by sending a `CHARSET REJECTED` message. The client system *must* respond to the server's request. (See the `CHARSET REQUEST` described above.)

When the client system makes the request first, and the server is able to handle the requested character set(s) but prefers that the client system instead use the server's (user application) character set, it may reject the request, and issue a `CHARSET REQUEST` of its own. If the client system is unable to comply with the server's preference and issues a `CHARSET REJECTED` message, the server can issue a new `CHARSET REQUEST` message for one of the previous character sets (one of those which the client system originally requested). The client system would obviously accept this character set.

While a `CHARSET` subnegotiation is in progress, data *should* be queued. Once the `CHARSET` subnegotiation has terminated, the data can be sent (in the correct character set).

Note that regardless of `CHARSET` negotiation, translation applies only to text (not commands), and occurs only when in `BINARY` mode. If not in `BINARY` mode, all data are assumed to be in NVT ASCII.

Also note that the `CHARSET` option should be used with the `END OF RECORD` option for block-mode terminals in order to be clear on what character represents the end of each record.

As an example of character-set negotiation, consider a user on a workstation using TELNET to communicate with a server. In this example, the workstation normally uses the Cyrillic (ASCII) character set but is capable of using `EBCDIC-Cyrillic`, and the server normally uses `EBCDIC-Cyrillic.` The server could handle the (ASCII) Cyrillic character set, but prefers that instead the client system use the `EBCDIC-Cyrillic` character set. (This and the following examples do not show the full syntax of the subnegotiation messages.)

Client	**Server**
WILL CHARSET	WILL CHARSET
DO CHARSET	DO CHARSET
CHARSET REQUEST Cyrillic EBCDIC-Cyrillic	CHARSET ACCEPTED EBCDIC-Cyrillic

Now consider a case in which the workstation can't handle `EBCDIC-Cyrillic`, but can accept a translate table:

Client	**Server**
WILL CHARSET	WILL CHARSET
DO CHARSET	DO CHARSET
CHARSET REQUEST	CHARSET TTABLE-IS [TTABLE] 1 Cyrillic 1 Cyrillic
CHARSET TTABLE-ACK	EBCDIC-Cyrillic

For another example, consider a case similar to the previous case, but now the user switches server applications in the middle of the ses sion (denoted by ellipses), and the new application requires a different character set:

Client	**Server**
WILL CHARSET	WILL CHARSET
DO CHARSET	DO CHARSET
CHARSET REQUEST	CHARSET TTABLE-IS [TTABLE] 1 1 Cyrillic Cyrillic EBCDIC-INT CHARSET TTABLE-IS 1 Cyrillic EBCDIC-Cyrillic
CHARSET TTABLE-ACK ...	...
	CHARSET REQUEST EBCDIC-INT
CHARSET ACCEPTED EBCDIC-INT	

13.7.5 TELNET Authentication Option

Command Names and Codes

AUTHENTICATION	37
IS	0
SEND	1
REPLY	2
NAME	3

Authentication Types

NULL	0
KERBEROS_V4	1
KERBEROS_V5	2
SPX	3
RSA	6
LOKI	10

Modifiers

AUTH_WHO_MASK	1
AUTH_CLIENT_TO_SERVER	0
AUTH_SERVER_TO_CLIENT	1
AUTH_HOW_MASK	2
AUTH_HOW_ONE_WAY	0
AUTH_HOW_MUTUAL	2

Command Meanings

Reference to *client* and *server* is used with TELNET. The *server* is the side of the connection that did the passive TCP open (TCP `LISTEN` state), and the *client* is the side of the connection that did the active open.

`IAC WILL AUTHENTICATION`	The client side of the connection sends this command to indicate that it is willing to send and receive authentication information.
`IAC DO AUTHENTICATION`	The server side of the connection sends this command to indicate that it is willing to send and receive authentication information.
`IAC WONT AUTHENTICATION`	The client side of the connection sends this command to indicate that it refuses to send or receive authentication information; the server side sends this command if it receives a DO AUTHENTICATION command.

`IAC DONT AUTHENTICATION`	The server side of the connection sends this command to indicate that it refuses to send or receive authentication information; the client side sends this command if it receives a WILL AUTHENTICATION command.
`IAC SB AUTHENTICATION SEND authentication-type-pair-list IAC SE`	The sender of this command (the server) requests that the remote side send authentication information for one of the authentication types listed in authentication-type-pair-list, which is an ordered list of authentication-type pairs. Only the server side (DO AUTHENTICATION) is allowed to send this.
`IAC SB AUTHENTICATION IS authentication-type-pair <auth data> IAC SE`	The sender of this command (the client) is sending the authentication information for authentication type authentication-type-pair. Only the client side (WILL AUTHENTICATION) is allowed to send this.
`IAC SB AUTHENTICATION REPLY authentication-type-pair <auth data> IAC SE`	The sender of this command (the server) is sending a reply to the authentication information received in a previous IS command. Only the server side (DO AUTHENTICATION) is allowed to send this.
`IAC SB AUTHENTICATION NAME remote-user IAC SE`	This optional command is sent to specify the account name on the remote host that the user wishes to be authorized to use. In order for authentication to succeed the authorization to use a particular account may still fail. Some authentication mechanisms may ignore this command.

The `authentication-type-pair` is 2 octets: an authentication type and a modifier to the type. There are currently two 1-bit fields defined in the modifier: the `AUTH_WHO_MASK` bit and the `AUTH_HOW_MASK` bit, so there are four possible combinations:

`AUTH_CLIENT_TO_SERVER AUTH_HOW_ONE_WAY`	The client will send authentication information about the local user to the server. If the negotiation is successful, the server will have authenticated the user on the client side of the connection.
`AUTH_SERVER_TO_CLIENT AUTH_HOW_ONE_WAY`	The server will authenticate itself to the client. If the negotiation is successful, the client will know that it is connected to the right (desired) server.
`AUTH_CLIENT_TO_SERVER AUTH_HOW_MUTUAL`	The client will send authentication information about the local user to the server, and then the server will authenticate itself to the client. If the negotiation is successful, the server will have authenticated the user on the client side of the connection, and the client will know that it is connected to the right server.
`AUTH_SERVER_TO_CLIENT AUTH_HOW_MUTUAL`	The server will authenticate itself to the client, and then the client will authenticate itself to the server. If the negotiation is successful, the client will know that it is connected to the right server that it wants to be connected to, and the server will know that the client is who it claims to be.

Default Specification

The default specification for this option is

WONT AUTHENTICATION
DONT AUTHENTICATION

This means that there will be no exchange of authentication information.

Motivation for the Option

One deficiency of the TELNET protocol is that in order to log into remote systems, users have to type their passwords, which are passed in clear text through the network. If the connections do not go through trusted networks, passwords may be compromised by someone watching the packets as they go by.

The purpose of the `AUTHENTICATION` option is to provide a framework for passing authentication information through the TELNET session. This means that (1) the user’s password will not be sent in clear text across the network; and (2) if the front-end TELNET process has the appropriate authentication information, it can auto matically send the information, and the user will not have to type any password.

The `AUTHENTICATION` option is intended to be sufficiently general to be used to pass information for any authentication system.

Security Implications

The ability to negotiate a common authentication mechanism between client and server is a feature of the authentication option that should be used with caution. When the negotiation is performed, no authentication has yet occurred. Therefore, each system has no way of knowing whether it is talking to the right (intended) system. An intruder could attempt to negotiate the use of an authentication system which is either weak or already compromised by the intruder.

Implementation Rules

`WILL` and `DO` are used only at the beginning of the connection to obtain and grant permission for future negotiations.

The authentication is negotiated in only one direction; the server must send the `DO,` and the client must send the `WILL.` This restriction is due to the nature of authentication; there are three possible cases: server authenticates client, client authenticates server, and server and client authenticate each other. By only negotiating the option in one direction, and then determining which of the three cases is being used via the suboption, potential ambiguity is removed. If the server receives a `DO,` it must respond with a `WONT.` If the client receives a `WILL,` it must respond with a `DONT.`

Once the two hosts have exchanged a `DO` and a `WILL,` the server is free to request authentication information. In the request, a list of supported authentication types is sent. Only the server may send requests (`IAC SB AUTHENTICATION SEND authentication-type-pair-list IAC SE`). Only the client may transmit authentication information via the `IAC SB AUTHENTICATION IS authentication-type. . .IAC SE` command. Only the server may send replies (`IAC SB AUTHENTICATION REPLY authentication-type. . .IAC SE`). As many `IS` and `REPLY` suboptions may be exchanged as are needed for the particular authentication scheme chosen.

If the client does not support any of the authentication types listed in the `authentication-type-pair-list,` a type of NULL should be used to indicate this in the `IS` reply. Note that in this case, the server may choose to close the connection.

The order of the authentication types *must* be ordered to indicate a preference for different authentication types: the first type the most preferred, and the last type the least preferred.

The following is an example of use of the option:

CLIENT: `IAC DO AUTHENTICATION`
SERVER: `IAC WILL AUTHENTICATION`

(The server is now free to request authentication information.)

```
SERVER: IAC SB AUTHENTICATION SEND
    KERBEROS_V4 CLIENT|MUTUAL
        KERBEROS_V4 CLIENT|ONE_WAY IAC SE
```

(The server has requested mutual Kerberos authentication, but is willing to do just one-way Kerberos authentication. The client will now respond with the name of the user that it wants to log in as, and the Kerberos ticket.)

```
CLIENT: IAC SB AUTHENTICATION NAME "joe"
        IAC SE
        IAC SB AUTHENTICATION IS
        KERBEROS_V4 CLIENT|MUTUAL AUTH 4
        7 1 67 82 65 89 46 67 7 9 77 0
        48 24 49 244 109 240 50 208 43
        35 25 116 104 44 167 21 201 224
        229 145 20 2 244 213 220 33 134
        148 4 251 249 233 229 152 77 2
        109 130 231 33 146 190 248 1 9
        31 95 94 15 120 224 0 225 76 205
        70 136 245 190 199 147 155 13
        AC SE
```

(The server responds with an `ACCEPT` command to state that the authentication was successful.)

```
SERVER: IAC SB AUTHENTICATION REPLY
        KERBEROS_V4 CLIENT|MUTUAL ACCEPT
        IAC SE
```

(Next, the client sends across a `CHALLENGE` to verify that it is really talking to the right server.)

```
CLIENT: IAC SB AUTHENTICATION IS
        KERBEROS_V4 CLIENT|MUTUAL
        CHALLENGE xx xx xx xx xx xx xx xx IAC SE
```

(Finally, the server sends across a `RESPONSE` to prove that it really is the right server.)

```
SERVER: IAC SB AUTHENTICATION REPLY
        KERBEROS_V4 CLIENT|MUTUAL
        RESPONSE yy yy yy yy yy yy yy yy
        IAC SE
```

It is expected that any implementation that supports the TELNET `AUTHENTICATION` option will support all of this specification.

13.7.6 TELNET Remote-Flow-Control Option

Command Names and Codes

```
TOGGLE-FLOW-CONTROL  33
OFF   0
ON     1
RESTART-ANY   2
RESTART-XON   3
```

Command Meanings

`IAC WILL TOGGLE-FLOW-CONTROL`	Sender is willing to enable and disable flow control on command.
`IAC WONT TOGGLE-FLOW-CONTROL`	Sender refuses to enable and disable flow control. Nothing is implied about whether sender uses flow control. It is simply unwilling to enable and disable it using this protocol.
`IAC DO TOGGLE-FLOW-CONTROL`	Sender is willing to send commands to enable and disable flow control.
`IAC DONT TOGGLE-FLOW-CONTROL`	Sender refuses to send command to enable and disable flow control.
`IAC SB TOGGLE-FLOW-CONTROL OFF IAC SE`	Sender requests receiver to disable flow control.
`IAC SB TOGGLE-FLOW-CONTROL ON IAC SE`	Sender requests receiver to enable flow control.
`IAC SB TOGGLE-FLOW-CONTROL RESTART-ANY IAC SE`	Sender requests that when flow control is enabled, the receiver allow any character (except another XOFF) to restart output.
`IAC SB TOGGLE-FLOW-CONTROL RESTART-XON IAC SE`	Sender requests that when flow control is enabled, the receiver allows only the XON character to restart output.

TELNET Default Specification

The default specification for this option is `WONT TOGGLE-FLOW-CONTROL DONT TOGGLE-FLOW-CONTROL,` meaning flow-control information will not be exchanged in either direction.

Motivation for this Option

This section describes a method of remotely toggling flow control between a user TELNET process and the attached terminal. Only flow control of data being transmitted from the TELNET process to the terminal is considered. Many systems will also allow flow control of data from the terminal to the TELNET process; however, there is seldom any need to change this behavior repeatedly during the session. There are two common ways of doing flow control: hardware and software. *Hardware* flow control uses signals on wires dedicated for this purpose. *Software* flow control uses one or two specific characters sent along the same path as normal input data. Most commonly, `XOFF` (`control-S`) and `XON` (`control-Q`) are used to stop and start output, respectively. The option described herein is useful primarily where software flow control is being used. (Since hardware flow control does not preempt any characters, there is normally no need to disable it.) It should also be noted that flow control can be generated either automatically by the terminal when its buffers are close to overflowing, or manually by a user who cannot read the information as fast as it is being displayed, and unread information will start scrolling off the screen.

The primary difficulty with software flow control is that it preempts one or two characters. Host software often requires the user to be able to input every possible ASCII character. (Certain editors are notorious for having XOFF and XON as commonly used commands.) For this reason, operating systems often allow programs to disable flow control. While it is disabled, the characters that normally signal flow control may be read as normal input. In a TELNET environment, flow control is normally done by the user TELNET process, not by the host computer. In addition, in many operating systems, when flow control is enabled, the user may specify whether the XOFF character is the only character that is allowed to reenable the output of data, or whether any typed character should reenable the flow of data. Thus this RFC defines a way to propagate flow control status from the host computer to the user TELNET process.

Description of the Option

Use of the option requires two phases. In the first phase, the TELNET processes agree that one of them will TOGGLE-FLOW-CONTROL. WILL and DO are used only in this first phase. In general there will be only one exchange of WILL and DO for a session. Subnegotiations must not be issued until DO and WILL have been exchanged. It is permissible for either side to turn off the option by sending a WONT or DONT. Should this happen, no more subnegotiations may be sent, unless the option is reenabled by another exchange of DO and WILL.

Once the hosts have exchanged a WILL and a DO, the sender of the DO TOGGLE-FLOW-CONTROL is free to send subnegotiations to enable and disable flow control in the other process, and to send subnegotiations for recommendations on when to restart output. Normally, the sender of the DO will be a host, and the other end will be a user TEL NET process, which is connected to a terminal. Thus the protocol is normally asymmetrical; however, it may be used in both directions without confusion should need for this arise.

As soon as the DO and WILL have been exchanged, the sender of the WILL must enable flow control, allowing flow control to begin in a known state. The decision of whether to restart output only when the XON character is received, or when any character received, starts in a system-dependent state. (This is to make it consistent with older implementations of the TOGGLE-FLOW-CONTROL option that do not understand the RESTART-ANY and RESTART-XON suboptions.) The sender of the DO should send either a RESTART-ANY or RESTART-XON suboption to put the restart characteristics to a known state. Some clients might not be able to support both of the RESTART-ANY and RESTART-XON modes because of system limitations, and would then choose to ignore these commands. There is no way for the server to be notified of this condition, but a client should make every attempt to honor the state requested by the RESTART-ANY and RESTART-XON modes. Should the option be disabled by exchange of DONT and WONT, flow control may revert to an implementation-defined default state. It is not safe to assume that flow control will remain in the state requested by the most recent subnegotiation.

In most implementations of software flow control, when enabled, the XOFF and XON characters are never propagated to the server; they are typically eaten by the terminal driver between the TELNET client and the attached terminal. In most implementations that support the RESTART-ANY functionality, the typed character that reenables the output is not eaten by the terminal driver, unless it is the XON character.

Currently, only four command codes are defined for the subnegotiations: flow control off (code 0), flow control on (code 1), restart output on any character (code 2), and restart output only on XON (code 3). None of these codes requires any additional data; however, it is possible that additional commands may be added. Thus subnegotiations having command codes other than those defined in this document should be silently ignored.

This option does not deal with the issue of allowing the `DO` side of the connection to inform the `WILL` side which characters should be used for `XON` and `XOFF.` That functionality is already supplied by the `LINEMODE` option.

Example Following is an example of the use of this option:

```
SERVER: IAC DO TOGGLE-FLOW-CONTROL
CLIENT: IAC WILL TOGGLE-FLOW-CONTROL
```

(The server is now free to send commands to change flow control. The client must now have enabled flow control, but that whether it is restart on `XON` only or on any character is client-specific.)

```
SERVER: IAC SB TOGGLE-FLOW CONTROL
        RESTART-ANY IAC SE
```

(The client should now switch to allowing output to restart when the user types any character, if the client is able to support that functionality.)

```
SERVER: IAC SB TOGGLE-FLOW-CONTROL OFF
         IAC SE
         IAC SB TOGGLE-FLOW-CONTROL ON
         IAC SE
```

13.7.7 TELNET Authentication: SPX

Command Names and Codes. Authentication Types:

SPX	3

Suboption comma nds:

AUTH	0
REJECT	1
ACCEPT	2

Command Meanings

`IAC SB AUTHENTICATION IS <authentication-type-pair> AUTH <SPX authentication token> IAC SE`	This is used to pass the SPX authentication token to the remote side of the connection. (A document which describes the authentication token syntax is forthcoming.) The first octet of the <`authentication-type-pair`> value is SPX. The second octet is a modifier to the SPX authentication type.
`IAC SB AUTHENTICATION REPLY <authentication-type-pair> ACCEPT <mutual response> IAC SE>`	This command indicates that the authentication was successful. After an SPX authentication exchange, both sides have securely established a random 8-byte key to be used as the default key for the ENCRYPTION option. If the `AUTH_HOW_MUTUAL` bit is set in the second octet of the authentication-type-pair, the sender includes the mutual response bytes. The receiver of the ACCEPT command compares the "mutual response" with its expected mutual response. If the `AUTH_HOW_ ONE_WAY` bit is set in the second octet of the `authentication-type-pair`, the sender includes zero bytes of mutual response.
`IAC SB AUTHENTICATION REPLY <authentication-type-pair< REJECT <optional reason for rejection> IAC SE`	This command indicates that the authentication was not successful, and if there are any more data in the suboption, it is an ASCII text message of the reason for the rejection.

Implementation Rules

Every command after the first `AUTHENTICATION IS` must carry the same set of modifiers (e.g., `CLIENT|MUTUAL`) for subsequent `AUTHENTICATION IS` and `AUTHENTICATION REPLY` commands.

If the second octet of the `authentication-type-pair` has the `AUTH_WHO` bit set to `AUTH_WHO_CLIENT,` then the client sends the initial `auth` command, and the server responds with either ACCEPT or REJECT.

If the second octet of the `authentication-type-pair` has the `AUTH_WHO` bit set to `AUTH_WHO_SERVER,` then the server sends the initial AUTH command, and the client responds with either ACCEPT or REJECT.

Examples

User `joe` may wish to log in as user `pete` on machine `foo`. If `pete` has set things up on `foo` to allow `joe` access to his account, then the client would send `IAC SB AUTHENTICATION NAME "pete" IAC SE IAC SB AUTHENTICATION IS SPX AUTH` <joe's spx authentication token> `IAC SE`. The server would then authenticate the user as `joe` from the token information, and the server would send back either ACCEPT or `reject`. If mutual authentication is being used, the server would include in the ACCEPT message, a mutual response. The authorization check to see if `pete` is allowing `joe` to use his account is made after the authentication exchange is complete. Therefore, it is possible for the client to receive an ACCEPT response (based on the authentication token), but for `joe` to be denied access to login to `pete`'s account.

```
SERVER: IAC DO AUTHENTICATION
CLIENT: IAC WILL AUTHENTICATION
```

(The server is now free to request authentication information.)

```
SERVER IAC SB AUTHENTICATION SEND SPX
        CLIENT|MUTUAL SPX CLIENT|ONE_WAY
        IAC SE
```

(The server has requested mutual SPX authentication. If mutual authentication is not supported, then the server is willing to do one-way SPX authentication.)

The client will now respond with the name of the user that it wants to log in as, and the SPX authentication token:

```
CLIENT IAC SB AUTHENTICATION NAME
        "pete" IAC SE
        IAC SB AUTHENTICATION IS SPX
        CLIENT|MUTUAL AUTH <spx
        authentication token
        information> IAC SE
```

(The server responds with an ACCEPT command to state that the authentication was successful.)

If `AUTH_HOW_MUTUAL,` the server responds with the mutual response so the client can verify that it is really talking to the right server. If `AUTH_HOW_ONE_WAY,` the server responds with a NULL mutual response, since the client is willing to trust the server already:

```
SERVER: IAC SB AUTHENTICATION REPLY SPX
         CLIENT|MUTUAL ACCEPT <mutual
         response> IAC SE
```

13.7.8 TELNET Binary Transmission

Command Name and Code

`TRANSMIT-BINARY,` 0.

Command Meanings

`IAC WILL TRANSMIT-BINARY`	The sender of this command *requests* permission to begin transmitting, or confirms that it will now begin transmitting characters which are to be interpreted as 8 bits of binary data by the receiver of the data.
`IAC WONT TRANSMIT-BINARY`	If the connection is already being operated in binary transmission mode, the sender of this command *demands* to begin transmitting data characters which are to be interpreted as standard NVT ASCII characters by the receiver of the data. If the connection is not already being operated in binary transmission mode, the sender of this command *refuses* to begin transmitting characters which are to be interpreted as binary characters by the receiver of the data (i.e., the sender of the data demands to continue transmitting characters in its present mode). A connection is being operated in binary transmission mode only when one party has requested it and the other has acknowledged it.
`IAC DO TRANSMIT-BINARY`	The sender of this command *requests* that the sender of the data start transmitting, or confirms that the sender of data is expected to transmit, characters which are to be interpreted as 8 bits of binary data (i.e., by the party sending this command).
`IAC DONT TRANSMIT-BINARY`	If the connection is already being operated in binary transmission mode, the sender of this command *demands* that the sender of the data start transmitting characters which are to be interpreted as standard NVT ASCII characters by the receiver of the data (i.e., the party sending this command). If the connection is not already being operated in binary transmission mode, the sender of this command *demands* that the sender of data continue transmitting characters which are to be interpreted in the present mode.

A connection is being operated in binary transmission mode only when one party has requested it and the other has acknowledged it.

Default

WONT TRANSMIT-BINARY
DONT TRANSMIT-BINARY

The connection is not operated in binary mode.

Motivation for the Option

It is sometimes useful to have available a binary transmission path within TELNET without having to utilize one of the more efficient, higher-level protocols providing binary transmission (such as the File Transfer Protocol). The use of the `IAC` prefix within the basic TELNET protocol provides the option of binary transmission in a natural way, requiring only the addition of a mechanism by which the parties involved can agree to INTERPRET the characters transmitted over a TELNET connection as binary data.

Description of the Option

With the binary transmission option in effect, the receiver should interpret characters received from the transmitter which are not preceded by `IAC` as 8-bit binary data, with the exception of `IAC` followed by `IAC` which stands for the 8-bit binary data with the decimal value 255. `IAC` followed by an effective TELNET command (plus any additional characters required to complete the command) is still the command even with the binary transmission option in effect. `IAC` followed by a character which is not a defined TELNET command has the same meaning as `IAC` followed by `NOP,` although an `IAC` followed by an undefined command should not normally be sent in this mode.

Implementation Suggestions

Implementations of the binary transmission option will choose to refuse some other options (such as the EBCDIC transmission option) while the binary transmission option is in effect. However, if a pair of hosts can understand being in binary transmission mode simultaneous with being in, for example, echo mode, then it is all right if they negotiate that combination.

It should be mentioned that the meanings of `WONT` and `DONT` are dependent on whether the connection is presently being operated in binary mode. Consider a connection operating in, say, EBCDIC mode which involves a system that has decided not to implement any knowledge of the binary command. If this system were to receive a `DO TRANSMIT-BINARY,` it would not recognize the `TRANSMIT-BINARY` option and therefore would return a `WONT TRANSMIT-BINARY.` If the default for the `WONT TRANSMIT-BINARY` were always NVT ASCII, the sender of the `DO TRANSMIT-BINARY` would expect the recipient to have switched to NVT ASCII, whereas the receiver of the `DO TRANSMIT-BINARY` would not make this interpretation.

Thus, we have the rule that when a connection is not presently operating in binary mode, the default (i.e., the interpretation of `WONT` and `DONT`) is to continue operating in the current mode, whether that is NVT ASCII, EBCDIC, or some other mode. This rule, however, is not applied once a connection is operating in a binary mode (as agreed to by both ends); this would require each end of the connection to maintain a stack, containing all the encoding-method transitions which had previously occurred on the connection, in order to properly interpret a `WONT` or `DONT.` Thus, a `WONT` or `DONT` received after the connection is operating in binary mode causes the encoding method to revert to NVT ASCII.

It should be remembered that a TELNET connection is a two-way communication channel. The binary transmission mode must be negotiated separately for each direction of data flow, if that is desired.

Implementation of the binary transmission option, as is the case with implementations of all other TELNET options, must follow the loop-preventing rules given in the General Considerations section of the TELNET Protocol Specification.

Consider some issues of binary transmission both to and from both a process and a terminal:

1. *Binary transmission from a terminal.* The implementer of the binary transmission option should consider how (or whether) a terminal transmitting over a TELNET connection with binary transmission in effect is allowed to generate all 8-bit characters, ignoring parity and other considerations on input from the terminal.

2. *Binary transmission to a process.* The implementer of the binary transmission option should consider how (or whether) all characters are passed to a process receiving over a connection with binary transmission in effect. As an example of the possible problem, TOPS-20 intercepts certain characters (e.g., ETX, the terminal control-C) at monitor level and does not pass them to the process.

3. *Binary transmission from a process.* The implementer of the binary transmission option should consider how (or whether) a process transmitting over a connection with binary transmission in effect is allowed to send all 8-bit characters with no characters intercepted by the monitor and changed to other characters. An example of such a conversion may be found in the TOPS-20 system, where certain nonprinting characters are normally converted to a circumflex (up arrow) followed by a printing character.

4. *Binary transmission to a terminal.* The implementer of the binary transmission option should consider how (or whether) all characters received over a connection with binary transmission in effect are sent to a local terminal. At issue may be the addition of timing characters normally inserted locally, parity calculations, and any normal code conversion.

14
TN3270 Protocol and Use

The number TN3270 refers to an implementation of the TCP/IP application TELNET that displays the characteristics of the 3270 data stream typically found in traditional SNA networks and reflects the hybrid nature of the application. TELNET, by definition, is both an application and a protocol. Consequently, it is versatile and can be implemented in numerous ways. In its original form, TELNET used an ASCII character set and was categorically couched in asynchronous environments.

Integration of TCP/IP and SNA in the 1980s began to radically transform the appearance of TCP/IP and SNA on integration. TN3270 is a prime example of this. TN3270 presents a formatted data stream to the display in the same way as the 3270 data stream does in native form in the SNA environment.

Information is provided to aid in the implementation of TN3270 servers as well as client terminal emulators. The following areas pertaining to TN3270 implementations are explained in this chapter: (1) the TELNET options negotiated to transition from an NVT ASCII state to a TN3270 state ready to process incoming 3270 data-stream commands, (2) a method for sending and receiving 3270 data, (3) a method of handling some special keys known as SYSREQ and ATTN using current available TELNET commands, and (4) the events that will transition a TN3270 session back to an NVT session.

14.1 A Perspective on 3270 Data and TELNET

3270 display terminal data differ from those found in non-3270 display terminals. Data represented via a 3270 terminal (display) are repre sented in block mode and use EBCDIC rather than the non-3270 system (which is typically ASCII-oriented). Non-3270 terminals (ASCII) display information in ASCII format and utilize what is referred to as *character mode.* The differences between character-set and block modes constitute the primary reason for the differentiation between TN3270 and standard TELNET in this chapter. Standard TELNET uses an ASCII character set and character mode for display; in contrast, most IBM terminals are 3270; that is, they use EBCDIC and are formatted in block mode.

Existing complex IBM 3270 display terminal networks are not easily integrated with the increasing number of multiplatform networking environments, specifically TCP/IP. These complex networks include terminals attached to a 3270 host using SNA (Systems Network Architecture) and non-SNA connections. To address the issue of easily connecting display terminals to 3270 hosts using IP networks, several vendors have introduced TELNET servers that provide TCP/IP users a connection to existing IBM mainframes by supporting display terminal emulation using a subset of the existing TELNET protocol. TELNET servers may be directly connected to the host, or indirectly connected using SNA or non-SNA methods. IBM terminals are generically referred to as 3270s, a classification which includes a broad range of terminals and devices, not all of which actually begin with the numbers 327X.

All 3270 terminals in the IBM SNA network environment have two sessions with the host computer application. One is used for communicating with the host application; the other, with the SSCP (system services control point) that links the terminal with the appropriate host computer. For the purposes of TN3270, this distinction is not apparent or relevant since there is actually only a single TELNET session with the host computer or server. On an IBM SNA network, the 3270 terminal has a special key that toggles between the two sessions (SYSREQ). A brief discussion on how some TELNET servers deal with this is included. In an SNA environment, a client session is identified by a logical unit (LU) name. In a non-SNA environment, no LU name is associated with a client session. The closest thing to an LU name in the TN3270 environment is the client’s IP address. Although some TELNET servers are connected to the host using SNA, TN3270 clients using these servers have no defined way to determine the LU name associated with the session.

TELNET servers that exist in non-SNA environments do not have to be concerned about providing TN3270 clients with support for the SNA functions described in this document. TN3270 does not support typical SNA responses and is classified as a non-SNA protocol. A TN3270 emulator is not aware of or concerned about how the TELNET server is connected to a 3270 host application.

Some typical SNA functions such as the SYSREQ and ATTN (attention) keys have been mapped to existing TELNET commands and are supported by some TELNET server implementations. Support for 3270 terminal emulation over TELNET is accomplished by the de facto standard of negotiating three separate TELNET options: TERMINAL-TYPE, BINARY TRANSMISSION, and END OF RECORD (EDR). This negotiation and the resulting data flow are explained in this chapter.

The Internet RFC 1041 attempted to standardize the method of negotiating 3270 terminal support by defining the TELNET 3270 Regime option. Historically, very few developers and vendors have implemented RFC 1041.

14.2 TELNET Options and Commands Used

TN3270 makes use of existing TELNET options and does not define any additional options or commands.

TELNET option	Value (decimal)
BINARY	0
TERMINAL-TYPE	24
EOR	25

Four additional options may be used during a TN3270 session and are interpreted according to their respective RFCs: `3270-REGIME, SUPPRESS-GO-AHEAD, ECHO`, and `TIMING-MARK` (TM). Other options should be rejected unless they are specifically handled by the client for NVT mode. Commands that may be encountered during a TN3270 session and are described in RFC 854 include NOP, BREAK, and INTERRUPT PROCESS (IP).

14.3 Connection Negotiation

The following example describes a TN3270-capable server and a TN3270 client establishing a connection. The TCP/IP port used for the connection is 23 (TELNET). At any place before and during the TN3270 connection negotiation process, other TELNET commands and data may be transferred and will be interpreted under the existing TELNET state. Some existing TN3270 servers initiate a client connection using an NVT TELNET dialog to establish parameters needed to complete the TN3270 connection to the desired host. The order of negotiating TERMINAL-TYPE, EOR, and BINARY, is not significant; this example shows a typical TN3270 connection.

```
SERVER: IAC DO TERMINAL-TYPE
CLIENT: IAC WILL TERMINAL-TYPE
SERVER: IAC SB TERMINAL-TYPE SEND IAC SE
CLIENT: IAC SB TERMINAL-TYPE IS <terminal type>IAC SE
```

The <terminal type>. listed here is a string consisting of terminal model, type, and support of enhanced attribute bytes; an example is IBM-3278-2. Some of the acceptable values are listed in RFC 1340, *Assigned Numbers.*

The -2 following 3278 designates the alternate screen size. 3270 terminals have the ability to switch between the standard (24 × 80) screen size and an alternate screen size. Model -2 is 24 × 80, which is the standard size; model -3 is 32 × 80, model -4 is 43 × 80, and model -5 is 27 × 132.

Appending the two-character string “-E” to the end of the terminal type signifies that the terminal is capable of handling 3270 extended data stream. This is interpreted to mean that the terminal is able to handle structured fields, which are described below. Some TELNET server implementations also interpret this to mean that the terminal is capable of handling extended attributes (highlighting, field validation, character set, outlining, etc.). The 3279 series of terminals is capable of extended attributes; while the 3278 series is not.

SERVER: `IAC DO EOR IAC WILL EOR`
CLIENT: `IAC WILL EOR IAC DO EOR`
SERVER: `IAC DO BINARY IAC WILL BINARY`
CLIENT: `IAC WILL BINARY IAC DO BINARY`
SERVER: <3270 data stream> `IAC EOR`
CLIENT: <3270 data stream> `IAC EOR`

And this client/server interchange continues. To terminate the connection, the socket is closed by one of the session partners. Typically, when the user logs off of the host, the TELNET server closes the connection.

If the TELNET server wishes to go back to NVT mode, it may issue the following TELNET options:

SERVER: `IAC WONT BINARY`
CLIENT: `IAC DONT BINARY`

or

SERVER: `IAC WONT EOR`
CLIENT: `IAC DONT EOR`

Either one of these two scenarios causes the connection to fail to satisfy the requirements for a valid TN3270 session. The TELNET client would then process data from the server as though they were NVT ASCII data.

14.4 TN3270 Options

The following examples show how a TN3270 client handles the `3270-REGIME, SUPPRESS-GO-AHEAD, ECHO,` and `TM` options.

14.4.1 3270 Regime Option

Very few servers support the TELNET 3270 Regime option. If the client does not support this option and responds negatively as shown in the following example, the server will proceed to the more typical example shown below.

SERVER: `IAC DO 3270-REGIME`
CLIENT: `IAC WONT 3270-REGIME`

Normal negotiation:

SERVER: `IAC DO TERMINAL-TYPE`
CLIENT: (See above)

14.4.2 Suppress-Go-Ahead Option

The Suppress-Go-Ahead option is requested by some servers. The RFC for this option lists the default as go-aheads being transmitted to signal the receiver to begin transmitting. Since TN3270 negotiates binary and end-of-record and is a block-mode protocol, the TELNET go-ahead character is not sent. Most servers do not negotiate this option even though they do not use the TELNET go-ahead character.

SERVER: `IAC DO SUPPRESS-GO-AHEAD`
CLIENT: `IAC WILL SUPPRESS-GO-AHEAD`

14.4.3 Echo Option

The echo option is negotiated by those servers that make use of the TELNET NVT mode to allow the user to enter information prior to negotiating the options necessary for TN3270. This information includes but is not limited to user identification, password, and destination 3270 host. Some servers accept the default for this option in which the client does not do a local echo of the characters the user enters at the keyboard. This allows the server to decide if it should echo characters back to the client (or should not, in the case of password). Echoing characters back to the client causes slow response time since every character is typically echoed individually. Therefore, some servers negotiate for the client to do its own local echoing (except for passwords). The following example illustrates this case:

SERVER: `IAC DO ECHO`
CLIENT: `IAC WILL ECHO`

(Client does local display of all characters.)

SERVER: `IAC WONT ECHO`
CLIENT: `IAC DONT ECHO`

(Client enters password—not locally displayed or remotely echoed.)

SERVER: `IAC DO ECHO`
CLIENT: `IAC WILL ECHO`

(Client resumes local display of all characters.)

14.4.4 Timing-Mark Option

The timing-mark option is used by some servers to test for the continued presence of a TN3270 client. The following example will assure the server that the client is still alive.

SERVER: `IAC DO TIMING-MARK`
CLIENT: `IAC WONT TIMING-MARK`

14.5 Testing for Session Presence and Handling 3270 Data

Testing for Session Presence

Some servers use the NOP command (hexadecimal F1) to test for the continued presence of a TN3270 client. If a client has terminated abnormally, TCP/IP SEND errors will occur. The timing-mark option, described above, is also used to test for presence.

SERVER: `IAC NOP`
CLIENT: <ignore / no response>

Handling 3270 Data

The 3270 data stream consists of a command and its associated data. Commands include but are not limited to ERASE SCREEN, ERASE AND WRITE TO SCREEN, and READ CURRENT SCREEN.

The reason for negotiating the EOR TELNET option is to provide a method for separating these commands since no length information is specified. 3270 commands are interpreted by the TELNET client in their entirety. Each 3270 command and possible data is terminated with the IAC EOR sequence.

The binary option is also required since 3270 data may contain the FF (hexadecimal) or IAC character. When this character is encountered during a TN3270 connection, it is handled as per the binary RFC.

3270 Structured Fields

The 3270 structured fields provide a much wider range of features than "old-style" 3270 data, such as support for graphics, partitions, and IPDS printer data streams. A structured field is a 3270 data type that allows non-3270 data to be embedded within 3270 data. Briefly, a structured field consists of the structured field command followed by one or more data blocks. Each data block has a length and a structured field identifier, followed optionally by additional data.

Not every TN3270 client can be expected to support all structured field functions. There must be a mechanism by which those clients that are capable of supporting some or all structured field functions can indicate their wishes. This is typically done by adding "-E" to the end of the terminal type string. Thus, when the terminal identifies itself as being able to handle extended attributes, it also is capable of sending and receiving structured fields.

The design of 3270 structured fields provides a convenient means to convey the level of support (including no support) for the various structured field functions. This mechanism is the READ PARTITION QUERY (RPQ) command, which is sent from the host application to the client. The client responds with a QUERY REPLY, listing which, if any, structured field functions it supports.

A TN3270 client that supports structured fields will respond to an RPQ command with the appropriate reply. The sequence of events when a client receives an RPQ and does not support structured fields is left up to the client implementation. Typically clients can identify at least this structured field and reply with a null set.

14.6 TN3270 Keys

14.6.1 The 3270 ATTN Key

The 3270 ATTN (attention) key is interpreted by many host applications in an SNA environment as an indication that the user wishes to interrupt the execution of the current process. A majority of the TELNET servers currently accept the TELNET IAC BREAK (code 243) sequence to signal this event. Originally the TELNET INTERRUPT PROCESS (IP) command was defined expressly for this purpose and was used to implement support for the ATTN key. Use of this key requires two things:

1. TN3270 clients provide as part of their keyboard mapping a single key or a combination of keys that map to the 3270 ATTN key. When the user presses this key(s), the client transmits a TELNET IP or BREAK command to the server.

2. TN3270 servers translate the BREAK command received from a TN3270 client into the appropriate form and pass it along to the host application as an ATTN key. In other words, the server representing a secondary logical unit (SLU) in an SNA session would send a SIGNAL RU (request or response unit) to the host application.

The ATTN key is not supported in a non-SNA environment; therefore, a TN3270 server representing non-SNA 3270 devices should ignore any TELNET IP (BREAK) commands it receives from a client.

14.6.2 The 3270 SYSREQ Key

The 3270 SYSREQ key is useful in an environment where the TELNET server is attached to the host using SNA. The SYSREQ key is useful in this environment when the host application becomes locked or the user wishes to terminate the session without closing the TELNET connection.

The TELNET INTERRUPT PROCESS (IP) command is interpreted by some TELNET servers as a SYSREQ key. Other servers recognize the 3270 TEST REQUEST key as a SYSREQ key. In an SNA environment, pressing this key toggles the terminal between the host application session and the SSCP session. Usually the user will enter LOGOFF once this key has been pressed to terminate the application session and then select a new host to connect to. Sometimes, if SYSREQ is pressed again, the host application will become unlocked and normal activities may then proceed.

It is entirely up to the TELNET server to interpret this command and send the appropriate commands to the host as well as format the resulting host data for display on the TELNET client. The data format during the SSCP session is in a format slightly different from that of normal 3270 data. Since the TELNET server has no way to pass this data directly to the TELNET client, it must either handle it entirely and ignore SYSREQ events or convert it to 3270 data to present to the client.

To implement SYSREQ key support, TN3270 clients provide a key (or combination of keys) that is identified as mapping to the 3270 SYSREQ key. When the user presses this key(s), the client would either transmit a TELNET IP command or TEST REQUEST key to the server, depending on the server implementation.

TN3270 servers representing non-SNA 3270 terminals may ignore any TELNET IP commands or TEST REQUEST keys they receive from a client.

14.7 Items Not Addressed by TN3270

Several items are not supported by current TN3270 implementations; for instance

• TN3270 provides no capability for clients to emulate the 328x class of printers.

• There is no mechanism enabling a TELNET client to request that a connection be associated with a given 3270 device name. This can be important when a terminal session is being established, since many host applications behave differently depending on the network name of the terminal. In the case of printer emulation, this capability is an absolute necessity because a large number of host applications have some method of predefining printer destinations.

• The 3270 ATTN and SYSREQ keys are not universally supported.

• There is no support for the SNA positive/negative response process. All data sent are assumed to be either handled or ignored. The lack of SNA response processing in TN3270 is part of what makes TN3270 efficient. A *negative* response indicates a client-side error while processing the previously received data; this could be caused by the host application building a 3270 data stream that contains an invalid command, or by a mechanical error at the client side. *Positive* responses indicate that processing of the previously received data has completed.

• There is no mechanism enabling the client to access the SNA BIND information. The BIND image in an SNA environment contains a detailed description of the session between the TELNET server and the host application.

• The connection negotiation does not make it clear whether clients should support 3270 structured fields.

14.8 A Perspective on TN3270 Enhancements

Currently, support for 3270 terminal emulation over TELNET is accomplished by the de facto standard of negotiating three separate TELNET options: `terminal-type, binary transmission`, and `end of record`. Note that there is no RFC that specifies this negotiation as a standard. RFC 1041 attempted to standardize the method of negotiating 3270 terminal support by defining the TELNET 3270 Regime option. Very few developers and vendors have implemented RFC 1041.

This document will refer to the existing practice of negotiating these three TELNET options before exchanging the 3270 data stream as "traditional TN3270."

Note: *For the purposes of the remaining portion of this chapter there is no differentiation between TELNET servers that represent SNA devices and those that represent non-SNA 3270 devices.*

The shortcomings of traditional TN3270 include all those listed in Sec. 14.7.

In order to address the concerns presented earlier, the TELNET option TN3270E (TN3270 enhanced) is presented here for consideration. In native mode, TELNET clients and servers would be free to decide whether to negotiate support of the TN3270E option. If either side does not support TN3270E, traditional TN3270 can be used; otherwise, a subnegotiation will occur to determine what subset of TN3270E will be used on the session. This presupposes that a TELNET client or server is capable of both types of 3270 emulation and one of them would attempt to negotiate TN3270E first, and negotiate traditional TN3270 only if the other side refuses TN3270E.

Once a TELNET client and server have agreed to use TN3270E, negotiation of the TN3270E suboptions can begin. The two major elements of TN3270E subnegotiation are:

• A device-type negotiation that is similar to, but more complicated than, the existing TELNET terminal-type option.

• The negotiation of a set of supported 3270 functions, such as printer data-stream type [3270 data stream or SNA Character Stream SCS)], positive/negative response exchanges, device-status information, and the passing of BIND information from server to client.

Successful negotiation of these two suboptions signals the beginning of 3270 data-stream transmission. In order to support several of the new functions in TN3270E, each data message must be prefixed by a header. This header will contain flags and indicators that convey such things as positive and negative responses and what type of data follow the header (e.g., 3270 data stream, SCS, or device-status information).

14.9 TN3270E Commands; Functions; and Default, Negotiation, and Device Specifications

Names	Codes
`TN3270E`	40
`ASSOCIATE`	00
`CONNECT`	01
`DEVICE-TYPE`	02
`FUNCTIONS`	03
`IS`	04
`REASON`	05
`REJECT`	06
`REQUEST`	07
`SEND`	08

Reason	Codes
`CONN-PARTNER`	00
`DEVICE-IN-USE`	01
`INV-ASSOCIATE`	02
`INV-DEVICE-NAME`	03
`INV-DEVICE-TYPE`	04
`TYPE-NAME-ERROR`	05
`UNKNOWN-ERROR`	06
`UNSUPPORTED-REQ`	07

Function Names	Codes
`BIND-IMAGE`	00
`DATA-STREAM-CTL`	01
`RESPONSES`	02
`SCS-CTL-CODES`	03
`SYSREQ`	04

TN3270E Command Meanings

`IAC WILL TN3270E`	The sender of this command is willing to send TN3270E information in subsequent subnegotiations.
`IAC WONT TN3270E`	The sender of this command refuses to send TN3270E information.
`IAC DO TN3270E`	The sender of this command is willing to receive TN3270E information in subsequent subnegotiations.
`IAC DONT TN3270E`	The sender of this command refuses to receive TN3270E information.

While they are not explicitly negotiated, the equivalent of the TELNET binary transmission option and the TELNET end-of-record option are implied in the negotiation of the TN3270E option. The part of the negotiation that agrees to support TN3270E is automatically required to support bidirectional binary and EOR transmissions.

`IAC SB DEVICE-TYPE IAC SE TN3270E SEND`	Only the server may send this command. This command is used to request that the client transmit device-type and, optionally, device-name information.
`IAC SB TN3270E DEVICE-TYPE REQUEST <device-type> [CONNECT  ASSOCIATE <device-name>] IAC SE`	Only the client may send this command. It is used in response to the server's SEND DEVICE-TYPE command, as well as to suggest another device-type command after the server has sent a DEVICE-TYPE REJECT command (see below). This command requests emulation of a specific 3270 device type and model. The REQUEST command may optionally include either the CONNECT or the ASSOCIATE command (but not both). If present, CONNECT and ASSOCIATE must both be followed by `<device-name>`. (See the section entitled "Device-type negotiation" for more detailed information.)
`IAC SB TN3270E DEVICE-TYPE IS <device-type> CONNECT <device-name> IAC SE`	Only the server may send this command. This command is used to accept a client's DEVICE-TYPE REQUEST command and to return the `server-defined device-name` command.
`IAC SB TN3270E DEVICE-TYPE REJECT REASON <reason-code> IAC SE`	Only the server may send this comman. This command is used to reject a client's DEVICE-TYPE REQUEST command.

`IAC SB TN3270E FUNCTIONS REQUEST <function-list> IAC SE`	Either side may send this command. This command is used to suggest a set of 3270 functions that will be supported on this session. It is also sent as an implicit rejection of a previous FUNCTIONS REQUEST command sent by the other side (see the section on “Functions negotiation” for more information). When used to reject a FUNCTIONS REQUEST command, the function-list command must not be identical to that received in the previous REQUEST command.
`IAC SB TN3270E FUNCTIONS IS <function-list> IAC SE`	Either side may send this command. This command is sent as a response to a FUNCTIONS REQUEST command and implies acceptance of the set of functions sent to it in the REQUEST command. Note that the list of functions in the FUNCTIONS IS command must match the list that was received in the previous FUNCTIONS REQUEST command.

TN3270E Default Specification

WONT TN3270E
DONT TN3270E

The result is that TN3270E will not be used.

TN3270E Subnegotiation Rules

All TN3270E commands and parameters are NVT ASCII strings in which upper- and lowercase are considered equivalent. Once it has been agreed that TN3270E will be supported, the first subnegotiation must concern the device-type (and possibly device-name) information. Only after that has been successfully negotiated can the client and server exchange functions information. Only after both device-type and functions have been successfully negotiated can 3270 data-stream transmission occur.

Device-Type Negotiation

Device-type (and device-name) negotiation begins when the server transmits the `DEVICE-TYPE SEND` command to the client. The client responds with the `DEVICE-TYPE REQUEST` command, which must include a device-type and may include a device-name request.

Valid device-types are

Terminals	IBM-3278-2 IBM-3278-2-E (24-row × 80-column display) IBM-3278-3 IBM-3278-3-E (32-row × 80-column display) IBM-3278-4 IBM-3278-4-E (43-row × 80-column display) IBM-3278-5 IBM-3278-5-E (27-row × 132-column display) IBM-DYNAMIC (no pre-defined display size)
Printers	IBM-3287-1

Use of 3278 and 3287 is *not* intended to exclude any particular device capabilities; they are used here only because they are commonly known designations for a terminal and a printer member of the 3270 family of devices. The intention is to simplify the device-type negotiation (in comparison to traditional TN3270) by minimizing the number of possible device types, and by breaking the association of a specific piece of IBM hardware with a related set of data-stream capabilities. For example, negotiation of device-type IBM-3278-2-E alone does *not* preclude the use of any of the functions associated with a physical 3279 model S2B. A client's ability to support the more advanced functions of the 3270 data stream will be indicated not by negotiation of an IBM device-type and model number, but rather by the combination of READ PARTITION QUERY (RPQ), and QUERY REPLY.

All of the terminal device types support a "primary" display size of 24 rows by 80 columns. The -3, -4, and -5 types each support an "alternate" display size as noted in the preceding list. The IBM-Dynamic device type implies no predefined alternate display size; this value will be passed from the client to host applications as part of the QUERY REPLY structured field, and it can represent any display size the client and the host application can support.

Terminal device-types with the -E suffix should be negotiated only by clients that are willing to support some subset of the 3270 extended data stream. This usually includes at a minimum support for extended colors and highlighting, but may also include a number of other functions, such as graphics capability, alternate character sets, and partitions.

Clients that negotiate a terminal device-type with the -E suffix or the IBM-Dynamic type, as well as those that negotiate a printer device type, must be able to accept and respond to an RPQ command (see the section on "3270 structured fields"). This allows the client to indicate to host applications which subsets of the 3270 extended data stream the client is willing to support.

In a VTAM/SNA environment, negotiation of IBM-Dynamic as the device type should result in a BIND in which the presentation services (layer 6) usage screen field (the 11th byte in the logmode's `PSERVIC` field) is set to 0 × 03, indicating that the alternate screen size will be determined by the QUERY REPLY (usable area).

Device Pools

An explanation of the CONNECT and ASSOCIATE commands first requires a discussion of the organization of terminal and printer device pools that the server maintains and from which it selects device names to assign to session requests. (The terms *device name, LU name,* and *network name* can be considered interchangeable in this document.) Also, for the purposes of this discussion, the term *generic session request* will be used to describe a request for a session by a TELNET client (either traditional or TN3270E) that does not include a request for a specific device name. The term *specific session request* will be used to describe a request for a session by a TN3270E client that includes a request for a specific device name (via either CONNECT or ASSOCIATE). As is the case with traditional TN3270, the TN3270E server must maintain a set of terminal device names. A generic request for a terminal session would result in the server selecting any available device name from this pool. The server, however, may also maintain a separate pool of terminal device names which can be used only to satisfy specific terminal session requests. This is to ensure that a terminal device that has some significance to host applications (and is therefore likely to be the target of a specific session request) is not “accidentally” assigned to a generic request and winds up associated with a client that has no use for it. Note that the reverse situation is allowed; that is, a specific terminal session request could ask for a device name that happens to be in the *generic terminal pool.*

For each terminal device (in both the *generic* and the *specific* pools), the TN3270E server could also have defined a “partner” or “paired” printer device. There should be a unique, one-to-one mapping between a terminal and its associated printer. The reasoning behind such a configuration is to allow for those host applications that produce printed output bound for a printer whose device name is determined by the device name of the terminal that initiated the print request. These printer devices can only be assigned to specific printer session requests that use the ASSOCIATE command.

In addition, the TN3270E server may also maintain a pool of printer device names that are not associated with any terminal. These printer devices can be assigned only to specific printer session requests that use the CONNECT command. This allows for those host applications that generate printed output bound for a printer whose device name is determined by something other than the device name of the terminal that initiated the print request (e.g., when the user ID of the person signed on to a terminal determines the print destination). Furthermore, it is possible that a pool of printer device names could be maintained and used only to satisfy generic requests for printers.

TN3270E CONNECT Command

CONNECT is used by the client to request that the server assign a specific device name to the TELNET session in question; it may be used when requesting either a terminal or a printer session. The specified device name must not conflict with the device type. For example, if the client requests `DEVICE-TYPE IBM-3287-1` (a printer) and specifies `CONNECT T1000001,` but T1000001 is defined at the host as a terminal, then the server should deny the request. Further, if the requested device name is already associated with some other TELNET session, or if it is not defined to the server, the server should deny the request.

TN3270E ASSOCIATE Command

ASSOCIATE can be used by the client only when requesting a device type that represents a printer. The ASSOCIATE command requests that this session be assigned the device name of the printer that is paired with the terminal named in the request. If the device type does not represent a printer, or if the device name is not that of a terminal, then the server should deny the request. It is anticipated that the device name specified in this request would be one returned by the server when accepting a previous terminal session request (see the is command below). Since no means of authentication has been provided for, it is possible that the printer paired with the terminal specified in the ASSOCIATE command has already been assigned to some other TELNET session; in this case, the server should deny the request.

Device Selection Rules

Assume a TN3270E server has the following device pools defined to it (device names that begin with a T are terminal devices; those that begin with a P are printers):

Generic terminal pool	**Specific terminal pool**	**Generic printer pool**	**Specific printer pool**
TG000001 <--> PTG00001	TS000001 <--> PTS00001	PG000001	PS000001
TG000002 <--> PTG00002	TS000002 <--> PTS00002	PG000002	PS000002
TG000003 <--> PTG00003	TS000003 <--> PTS00003	PG000003	PS000003

The only pool that must be defined to the server is the generic terminal pool. The absence of other pools (or of partner printers for a terminal pool) means that the server is unable to satisfy as wide a variety of requests as would be possible if all pools were defined to it.

Given the configuration shown above, the following rules apply:

• A generic terminal request can be satisfied only from the generic terminal pool (device names TG000001 to TG000003).

• A specific terminal request (allowable only via the CONNECT command) can be satisfied from either the generic or the specific terminal pool, although it is anticipated that the majority of such requests would ask for terminals in the specific terminal pool (TS000001 to TS000003).

• A generic printer request can be satisfied only from the generic printer pool (device names PG000001 to PG000003).

• Specific printer requests may come in one of two forms: (1) via ASSOCIATE, where the request can only be satisfied using the partner of the specified terminal, which may be in the generic or the specific terminal pool (thus, devices PTG00001 to PTG00003 and PTS00001 to PTS00003 can be used to satisfy the request) or (2) via CONNECT, where the request can be satisfied either from the generic or the specific printer pools (although, as with specific terminal requests, most such requests will probably name printers in the specific printer pool)—this request cannot be satisfied with the partner printer of a terminal in either the specific or the generic terminal pools.

Accepting a Request

The server must accept the client’s request or deny it as a whole—it cannot, for example, accept the device-type request but deny the CONNECT portion.

If the server wishes to accept the request, it sends back the `DEVICE-TYPE IS` command confirming the requested device-type and the CONNECT command specifying the device name of the terminal or printer assigned to this TELNET session. This device name may be the one directly requested (via CONNECT) by the client, the one indirectly requested (via ASSOCIATE) by the client, or one chosen by the server if the client specified neither CONNECT nor ASSOCIATE.

REJECT Command

If the server wishes to deny the request, it sends back the `DEVICE-TYPE REJECT` command with one of the following reason code names and explanations:

Reason code name	**Explanation**
INV-DEVICE-TYPE	The server does not support the requested device type.
INV-DEVICE-NAME	The device name specified in the CONNECT or ASSOCIATE command is not known to the server.
DEVICE-IN-USE	The requested device name is already associated with another TELNET session.
TYPE-NAME-ERROR	The requested device name is incompatible with the requested device type (e.g., terminal/printer mismatch).
UNSUPPORTED-REQ	The server is unable to satisfy the type of request sent by the client; e.g., a specific terminal or printer was requested but the server does not have such a pool of device names defined to it, or the ASSOCIATE command was used but no partner printers are defined to the server.
INV-ASSOCIATE	The client used the ASSOCIATE command and either the device type is not a printer or the device-name is not a terminal.
CONN-PARTNER	The client used the CONNECT command to request a specific printer but the device name requested is the partner to some terminal.
UNKNOWN-ERROR	Any other error in device type or name processing has occurred.

The process of negotiating a device type and device name that are acceptable to both client and server may entail several iterations of `DEVICE-TYPE REQUEST` and `DEVICE-TYPE REJECT` commands. The client should make use of the reason code specified by the server in any `DEVICE-TYPE REJECT` command(s) to minimize the amount of negotiation necessary. For example, if the client initially requests that it be assigned a specific terminal device name via the CONNECT command, and the server rejects the request with a reason code of `UNSUPPORTED-REQ,` the client should make no further specific terminal requests in the negotiations. If at any point in the process either side wishes to bail out, it can simply send a `WONT` (or `DONT`) TN3270E command to the other side. At this point both sides are free to negotiate other TELNET options (including traditional TN3270).

TN3270 FUNCTIONS Negotiation

Once the `DEVICE-TYPE` negotiation has been successfully completed (i.e., when the client receives the `DEVICE-TYPE IS` command), the client should initiate the `FUNCTIONS` negotiation by sending the `FUNCTIONS REQUEST` command to the server. After this initial REQUEST command, both sides are free to transmit `FUNCTIONS REQUEST` and `FUNCTIONS IS` commands as needed.

Commands are as follows. The `FUNCTIONS REQUEST` command contains a list of the 3270 functions that the sender would like to see supported on this session. All unlisted functions are to be considered unsupported. The function list consists of a string of 2-byte entries separated from one another by a single-spaced character.

The list is terminated by the IAC (interpret as command) code that precedes the SE command. Functions may appear in any order in the list.

On receipt of a `FUNCTIONS REQUEST` command, the recipient may respond either (1) *positively* (indicating that it agrees to support all functions in the list, and not to transmit any data related to functions not in the list)—to do this, it sends the `FUNCTIONS IS` command with the function list exactly as it was received, at which point, `FUNCTIONS` negotiation will have been successfully completed; or (2) *negatively* by sending a `FUNCTIONS REQUEST` command in which the function list differs from the one it received (and not simply in the order of appearance of functions in the list—at least one function must have been added to, or removed from, the list). To avoid endlessly looping, neither party should add to the function list it receives any function that it has previously added and that the other side has removed.

The process of sending `FUNCTIONS REQUEST` commands back and forth continues until one side receives a function list it is willing to live with. It uses the `FUNCTIONS IS` command to accept the list, and, once this command is received by the other side, all necessary negotiation has been completed. At this point, 3270 data-stream transmission can begin. It is possible that the function list agreed to is null; this is referred to as "basic TN3270E."

The following list briefly describes the TN3270E functions that may be negotiated in the function list:

Function name	**Description**
`SCS-CTL-CODES`	(Printer sessions only.) Allows the use of the SNA Character Stream (SCS) and SCS control codes on the session. SCS is used with LU type 1 SNA sessions.
`DATA-STREAM-CTL`	(Printer sessions only.) Allows the use of the standard 3270 data stream. This corresponds to LU type 3 SNA sessions.
`RESPONSES`	Provides support for positive and negative response handling. Allows the server to reflect to the client any and all definite, exception, and no response requests sent by the host application.
`BIND-IMAGE`	Allows the server to send the SNA BIND image and UNBIND notification to the client.
`SYSREQ`	Allows the client and server to emulate some (or all, depending on the server) of the functions of the SYSREQ key in an SNA environment.

TN3270E Data Messages

3270 device communications are generally understood to be block oriented in nature; that is, each partner buffers data until an entire "message" has been built, at which point the data are sent to the other side. The outbound message (from host to device) consists of a 3270 command and a series of buffer orders, buffer addresses, and data, while the inbound message contains only buffer orders, addresses, and data. The end of a message is understood to be the last byte transmitted (note that this discussion disregards SNA chaining). The TELNET EOR command is used to delimit these natural blocks of 3270 data within the TELNET data stream.

In TN3270E, each 3270 message must be prefixed with a TN3270E header, which consists of 5 bytes. A *data message* in TN3270E therefore has the construction <TN3270E Header><data><IAC EOR>.

It should be noted that it is possible that, for certain message types, there is no data portion present. In this case, the TN3270E data message consists of <TN3270E Header><IAC EOR>.

If either side wishes to transmit the decimal value 255 and have it interpreted as data, it must double this byte. In other words, a single occurrence of decimal 255 will be interpreted by the other side as an IAC, while two successive bytes containing decimal 255 will be treated as one data byte with a value of decimal 255.

It is strongly recommended that TELNET commands (other than IAC IAC) should be sent between TN3270E data messages, with no header and no trailing IAC EOR. If a TN3270E data message containing either IAC IP (to be interpreted as 3270 ATTN) or IAC AO (to be interpreted as SYSREQ) is received, the receiver should defer processing the command until the 3270 data have been processed. If a TN3270E data message containing any other IAC-command sequence (other than IAC IAC) is received, it is implementation-dependent when the IAC-command sequence will be processed, but it must be processed. The receiver may process it immediately, which in effect causes it to be processed as if it had been received before the current TN3270E data message, or the processing may be deferred until after the current TN3270E data message has been processed. It is because of this ambiguity that the presence of TELNET commands within a TN3270E data message (i.e., between the header and the trailing IAC EOR) is not recommended; neither clients nor servers should send such data.

TN3270E Message Header

As stated earlier, each data message in TN3270E must be prefixed by a header, which consists of 5 bytes and is formatted as follows:

```
________________________________
| DATA-TYPE | REQUEST-FLAG | RESPONSE-FLAG | SEQ-NUMBER |
________________________________
1 byte 1 byte 1 byte 2 bytes
```

DATA-TYPE Field

The `DATA-TYPE` field indicates how the data portion of the message are to be interpreted by the receiver. Possible values for the `DATA-TYPE` field are

Data-type name	**Code**	**Meaning**
`3270-DATA`	0 × 00	The data portion of the message contains only the 3270 data stream.
`SCS-DATA`	0 × 01	The data portion of the message contains SNA Character Stream data.
`RESPONSE`	0 × 02	The data portion of the message constitutes device-status information, and the `RESPONSE-FLAG` field indicates whether this is a positive or negative response.
`BIND-IMAGE`	0 × 03	The data portion of the message is the SNA BIND image from the session established between the server and the host application.
`UNBIND`	0 × 04	The data portion of the message is an UNBIND reason code.
`NVT-DATA`	0 × 05	The data portion of the message is to be interpreted as NVT data.

REQUEST	0 × 06	There is no data portion present in the message; only the REQUEST-FLAG field has any meaning.
SSCP-LU DATA	0 × 07	The data portion of the message is data from the SSCP-LU session.

REQUEST-FLAG Field

The REQUEST-FLAG field has meaning only when the DATA-TYPE field has a value of REQUEST; otherwise, the REQUEST-FLAG field must be ignored by the receiver and should be set to 0 × 00 by the sender. Possible values for the REQUEST-FLAG field are

Request-flag name	**Code**	**Meaning**
ERR-COND-CLEARED	0 × 00	The client sends this to the server when some previously encountered printer error condition has been cleared.

RESPONSE-FLAG Field

The RESPONSE-FLAG field only has meaning for certain values of the DATA-TYPE field. For DATA-TYPE field values of 3270-DATA and SCS-DATA, the RESPONSE-FLAG indicates whether the data sender expects to receive a response. In this case the possible values of RESPONSE-FLAG are

Response-flag name	**Code**	**Meaning**
NO-RESPONSE	0 × 00	The sender does not expect the receiver to respond either positively or negatively to this message; therefore the receiver must not send any response to this data message.
ERROR-RESPONSE	0 × 01	The sender expects the receiver to respond to this message only if some type of error occurred, in which case a negative response must be sent by the receiver.
ALWAYS-RESPONSE	0 × 02	The sender expects the receiver to respond negatively if an error occurs, or positively if no errors occur. One or the other response must always be sent by the receiver.

For a DATA-TYPE field value of RESPONSE, the RESPONSE-FLAG is an actual response to a previous data message (which must by definition have had a DATA-TYPE of either 3270-DATA or SCS-DATA and a RESPONSE-FLAG value of either ERROR-RESPONSE or ALWAYS-RESPONSE). In this case the possible values of RESPONSE-FLAG are:

Response-flag name	Code	Meaning
POSITIVE-RESPONSE	0 × 00	The previous message was received and executed successfully with no errors.
NEGATIVE-RESPONSE	0 × 01	The previous message was received but an error(s) occurred while processing it.

Accompanying status information will be found in the data portion of the message. For any other values of the DATA-TYPE field, the RESPONSE-FLAG field must be ignored by the receiver and should be set to 0 × 00 by the sender.

SEQ-NUMBER Field

The SEQ-NUMBER field is used only when the RESPONSES function has been agreed to. It contains a 2-byte binary number, and is used to correlate positive and negative responses to the data messages for which they were intended. See the section on the RESPONSES function for further information. When the RESPONSES function is not agreed to, this field should always be set to 0 × 0000 by the sender and ignored by the receiver.

Basic TN3270E

Whether the use of each of the TN3270E functions is allowed on a session is negotiated when the connection is established. It is possible that none of the functions are agreed to (in this case, the function-list in the FUNCTIONS REQUEST and FUNCTIONS IS commands is null). This mode of operation is referred to as “basic TN3270E.” Note that, since neither the SCS-CTL-CODES function nor the DATA-STREAM-CTL function is agreed to, “basic TN3270E” refers to terminal sessions only.

Basic TN3270E requires the support of only the following TN3270E header values:

Header field	Value
DATA-TYPE	3270-DATA
DATA-TYPE	NVT-DATA

The REQUEST-FLAG, RESPONSE-FLAG, and SEQ-NUMBER fields are not used in basic TN3270E.

3270 Mode and NVT Mode

At any given time, a TN3270E connection can be considered to be operating in either “3270 mode” or “NVT mode.” In 3270 mode, each party may send data messages with the DATA-TYPE flag set to 3270-DATA; sending a DATA-TYPE flag set to NVT-DATA constitutes a request to switch modes. In NVT mode, each party may send data messages with the DATA-TYPE flag set to NVT-DATA; sending 3270-DATA is a request to switch modes. The connection is initially in 3270 mode when TN3270E operation is successfully negotiated. When a party receives a message with a DATA-TYPE different from the mode it is operating in, the mode of operation for the connection is switched. Switching modes results in the client performing the equivalent of a 3270 ERASE/RESET operation, using the default partition (screen) size. The server cannot assume the client preserves any attributes of the previous environment across a mode switch.

When sending `NVT-DATA,` each side should buffer data until an entire message is built (for the client, this would normally mean until the user presses ENTER). At that point, a complete TN3270E data message should be built to transmit the NVT data.

Typically, NVT data are used by a server to interact with the client's user. It allows the server to do this using a simple NVT data stream, instead of requiring a 3270 data stream. An example would be a server which displays a list of 3270 applications to which it can connect the client. The server would use NVT data to display the list and read the user's choice. Then the server would connect to the application, and begin the exchange of 3270 data between the application and the client.

Processing TN3270E Functions

Agreement by both parties to a specific function in the `FUNCTIONS REQUEST` function list implies agreement by each party to support a related set of values in the TN3270E header. It also implies a willingness to adhere to the rules governing the processing of data messages with regard to the agreed-on function. Either party that fails to accept header values associated either with agreed-on functions or with basic TN3270E, or attempts to use header values associated with a function that is not a part of basic TN3270E and was not agreed on, will be considered nonconforming and in violation of the protocol. The following sections detail for each TN3270E function the associated header values and processing rules.

SCS-CTL-CODES Function

This function can be supported only on a 3270 printer session. Agreement to support this function requires that the party support the following TN3270E header values:

Header field	**Value**
`DATA-TYPE`	`SCS-DATA`

A client representing a printer device uses this function to indicate its willingness to accept a data stream that includes SCS control codes. For the purposes of NVT mode versus 3270 mode, `SCS-DATA` should be treated exactly like `3270-DATA` (i.e., it can cause a switch from NVT mode to 3270 mode).

When a printer device type has been negotiated, either the `SCS-CTL-CODES` function or the `DATA-STREAM-CTL` function, or both, must be negotiated. This enables the server to know when it should and should not accept a session with a host application on behalf of the client. If only the `SCS-CTL-CODES` function is agreed to, then the serv er will not establish sessions with host applications that would send 3270 data-stream control. If both `SCS-CTL-CODES` and `DATA-STREAM-CTL` are agreed to, then the server will establish sessions both with host applications that would send SCS control codes and with those that would send 3270 orders.

DATA-STREAM-CTL Function

This function can only be supported on a 3270 printer session. Agreement to support this function requires that the party support the following TN3270E header values:

Header Field	**Value**
`DATA-TYPE`	**`3270-DATA`**

A client representing a printer device uses this function to indicate its willingness to accept a data stream that includes 3270 orders and attributes. When a printer device type has been negotiated, either the `SCS-CTL-CODES` function or the `DATA-STREAM-CTL` function, or both, must be negotiated, enabling the server to know when it should and should not accept a session with a host application on behalf of the client. If only the `DATA-STREAM-CTL` function is agreed to, then the server will not establish sessions with host applications that would send SCS control codes in a data stream. If both `SCS-CTL-CODES` and `DATA-STREAM-CTL` are agreed to, then the server will establish sessions with host applications that would send SCS control codes as well as with those that would send 3270 orders.

BIND-IMAGE Function

This function can be supported only when the TN3270E server represents SNA terminals and printers. Agreement to support this function requires that the party support the following TN3270E header values:

Header	**Field value**
DATA-TYPE	`BIND-IMAGE`
DATA-TYPE	`UNBIND`
DATA-TYPE	`SSCP-LU-DATA`

When `BIND-IMAGE` is in effect, the server must inform the client when an SNA session has been established with a host application, and when such a session has been terminated. It uses `DATA-TYPE` values of `BIND-IMAGE` and `UNBIND` to convey this information.

When establishing an SNA session on behalf of a client, the server will receive a `BIND RU` from the host application. It will also receive a `Start Data Traffic` RU. Once both of these have been responded to positively by the server, it must then inform the client of the presence of this session by sending it a data message with the `DATA-TYPE` flag set to `BIND-IMAGE`. The data portion of this message must contain the BIND-IMAGE exactly as it was received in the BIND RU that the server accepted on behalf of the client.

When an SNA session between the server and a host application is terminated, the server should send a data message to the client with the `DATA-TYPE` flag set to `UNBIND`. If the server was notified of the session termination via an SNA UNBIND RU, it should include the `UNBIND` reason code in the data portion of the message it sends to the client. If the server itself requested the SNA session termination (e.g., as part of SYSREQ key processing), it should set the data portion of the UNBIND message to 0 × 01, indicating “normal end of session.”

Another aspect of the `BIND-IMAGE` function alters the allowable `DATA-TYPE` flag values. When BIND-IMAGE is in effect, data messages with `DATA-TYPE` set to `3270-DATA` or `SCS-DATA` are not allowed before the first `BIND-IMAGE` is received by the client; only `SSCP-LU-DATA` or `NVT-DATA` can be used to transmit user-oriented data. The same applies to data messages exchanged after an `UNBIND` is sent and before another BIND-IMAGE is received by the client. Once the client receives a `BIND-IMAGE` data message, the allowable DATA-TYPE values include `3270-DATA` and/or `SCS-DATA,` depending on whether a terminal or printer device-type was negotiated, and whether a printer client agreed to `DATA-STREAM-CTL, SCS-CTL-CODES,` or both.

The RESPONSES Function

This function can be supported for both terminal and printer sessions connected to both SNA and non-SNA servers. Agreement to support this function requires that the party support the following TN3270E header values:

Header field	**Value**
`DATA-TYPE`	`RESPONSE`
`DATA-TYPE`	`REQUEST`
`RESPONSE-FLAG`	All values
`REQUEST-FLAG`	`ERR-COND-CLEARED`
`SEQ-NUMBER`	Binary values from 0 to 32767

Whenever a data message is sent with a `DATA-TYPE` of either `SCS-DATA` or `3270-DATA,` the sender must set the `RESPONSE-FLAG` field to either `NO-RESPONSE, ERROR-RESPONSE,` or `ALWAYS-RESPONSE.` It is anticipated that the client side will normally set `RESPONSE-FLAG` to `NO-RESPONSE.` The server, if it represents an SNA device, should set `RESPONSE-FLAG` to reflect the response value set in the RH of the RU that generated this data message: *definite response* resulting in a `RESPONSE-FLAG` value of `ALWAYS-RESPONSE;` *exception response* resulting in `ERROR-RESPONSE` being set, and *no response* causing a setting of `NO-RESPONSE.` A non-SNA server should set `RESPONSE-FLAG` to `ERROR-RESPONSE.`

In addition, the sender must keep a count of the messages with a `DATA-TYPE` of `3270-DATA` or `SCS-DATA` that it sends on a given session. This counter should start at zero for the first such message, and be incremented by one for each subsequent message. If the counter reaches the maximum of 32767, it should be restarted (reset) at zero. The sender should place this value in the `SEQ-NUMBER` field of the TN3270E header before it sends the message. Note that the `SEQ-NUMBER` field must be set regardless of the value of the `RESPONSE-FLAG` field.

Response Messages

Whenever a data message with a `DATA-TYPE` of either `SCS-DATA` or `3270-DATA` is received, the receiver must attempt to process the data in the data portion of the message, then determine whether it should send a data message with a `DATA-TYPE` of `RESPONSE`. If the data message it has just processed had a `RESPONSE-FLAG` value of `NO-RESPONSE,` or if it had a value of `ERROR-RESPONSE` and there were no errors encountered while processing the data, then no `RESPONSE`-type message should be sent. Otherwise, a data message should be sent in which the header `DATA-TYPE` field is set to `RESPONSE,` and in which the `SEQ-NUMBER` field is a copy of the `SEQ-NUMBER` field from the message to which this response corresponds. The `RESPONSE-FLAG` field in this header must have a value of either `POSITIVE-RESPONSE` or `NEGATIVE-RESPONSE.` A `POSITIVE-RESPONSE` should be sent if the previously processed message's header specified `ALWAYS-RESPONSE` and no errors were encountered in processing the data. A `NEGATIVE-RESPONSE` should be sent if either (1) the previously processed message specified `ERROR-RESPONSE` or `ALWAYS-RESPONSE` or (2) some kind of error occurred while processing the data.

Normally only the client will be constructing and sending these `RESPONSE` messages. A negative response sent by the client to the server is the equivalent of a unit check status. The data portion of a `RESPONSE` message must consist of one byte of binary data. The value of this byte gives a more detailed account of the results of having processed the previously received data message. The possible values for this byte are

1. `RESPONSE-FLAG value of POSITIVE-RESPONSE:` value 0 × 00, meaning successful completion (when sent by the client, this is equivalent to “device end”).

2. `RESPONSE-FLAG value of NEGATIVE-RESPONSE:` value 0 × 00, meaning an invalid 3270 command was received (equivalent to “command reject”); value 0 × 01, meaning printer is not ready (equivalent to “intervention required”); value 0 × 02, meaning an illegal 3270 buffer address or order sequence was received (equivalent to “operation check”); and value 0 × 03, meaning printer is powered OFF or not connected (equivalent to “component disconnected”).

When the server receives any of these responses, it should pass along the appropriate information to the host application. The appropriate information is determined by whether the server represents an SNA or a non-SNA device.

An SNA server should pass along a `POSITIVE-RESPONSE` from the client as an SNA positive response unit to the host application. It should translate a `NEGATIVE-RESPONSE` from the client into an SNA negative response unit in which the sense data indicator bit is powered ON and contains one of the following sense codes:

RESPONSE-FLAG	**Equivalent**	**SNA sense code**
0 × 00	Command reject	0 × 10030000
0 × 01	Intervention required	0 × 08020000
0 × 02	Operation check	0 × 10050000
0 × 03	Component	
	Disconnected	0 × 08310000

A non-SNA server should pass along a `POSITIVE-RESPONSE` from the client by setting the `Device End Status` bit on. It should reflect a `NEGATIVE-RESPONSE` from the client by setting the `Unit Check Status Bit` on, and setting either the `Command Reject, Intervention Required,` or `Operation Check Sense` bit on when responding to the SENSE command.

In the case of intervention required or component disconnected being passed by the server to the host application, the host would normally refrain from sending any further data to the printer. If and when the error condition at the client has been resolved, the client must send to the server a data message whose header `DATA-TYPE` field is set to `REQUEST,` and whose `REQUEST-FLAG` is set to `ERR-COND-CLEARED.` Note that this message has no data portion. On receipt of this message, the server should pass along the appropriate information to the host application so that it may resume sending printer output. Again, the form of this information depends on whether the server represents an SNA or a non-SNA device.

An SNA server should reflect an `ERR-COND-CLEARED` to the host application by sending an SNA `LUSTAT RU` with one of the following sense codes: (1) if the previous error condition was an *intervention required,* the server should send sense code 0 × 00010000; (2) if the previous error condition was *component disconnected,* the server should send sense code 0 × 082B0000.

A non-SNA server should set the corresponding bits in the *ending status* and *sense condition* bytes.

SYSREQ Function

This function can only be supported when the TN3270E server represents SNA devices. Agreement to support this function requires that the party support the following TN3270E header values: header field `DATA-TYPE;` value `SSCP-LU-DATA.`

The 3270 SYSREQ key can be useful in an SNA environment when the ATTN key is not sufficient to terminate a process.

Background of the 3270 SYSREQ Key

In SNA, there is a session between the host application [the PLU (primary logical unit)] and the TN3270E server representing the client [the SLU (secondary logical unit)]. This is referred to as the *PLU-SLU* session, and it is the one on which normal communications flow. There is also a session between the host telecommunications access method [the SSCP (system services control point)] and the SLU, and it is referred to as the *SSCP-LU* session. This session is used to carry various control information and is normally transparent to the user; normal 3270 data stream orders are not allowed in these data.

The terminal display and keyboard are usually “owned” by the PLU-SLU session, meaning any data the user types is sent to the host application. The SYSREQ key is used to toggle ownership of the keyboard and display between the PLU-SLU session and the SSCP-LU session. In other words, the user is able to press SYSREQ and then communicate directly with the host SSCP. The user may then enter any valid unformatted systems services (USS) commands, which are defined in the USS table associated with the SLU. The most common USS command users employ is LOGOFF, which requests that the SSCP immediately terminate the PLU-SLU session. The usual reason for requesting such an action is that the host application (the PLU) has stopped responding altogether.

Whenever the keyboard and display are owned by the SSCP-LU session, no data are allowed to flow in either direction on the PLU-SLU session. Once “in” the SSCP-LU session, the user may decide to switch back to the PLU-SLU session by again pressing the SYSREQ key.

How TN3270E Implements SYSREQ

The design of some TN3270E servers allows them to fully support the SYSREQ key because they are allowed to send USS commands on the SSCP-LU session. Other TN3270E servers operate in an environment which does not allow them to send USS commands to the SSCP; this makes full support of the SYSREQ key impossible. For such servers, TN3270E provides for emulation of a minimal subset of functions, namely, for the sequence of pressing SYSREQ and typing LOGOFF that many users employ to immediately terminate the PLU-SLU session.

The TELNET ABORT OUTPUT (AO) command is the mechanism used to implement SYSREQ key support in TN3270E because, in a real SNA session, once the user presses the SYSREQ key, the host application is prevented from sending any more output to the terminal (unless the user presses SYSREQ a second time), but the user’s process continues to execute.

In order to implement SYSREQ key support, TN3270E clients that have agreed to the SYSREQ function should provide a key (or combination of keys) that is identified as mapping to the 3270 SYSREQ key. When the user presses this key(s), the client should transmit a TELNET AO command to the server.

On receipt of the AO command, a TN3270E server that has agreed to the SYSREQ function should enter what will be loosely termed “suspended mode” for the connection. If a server that has not agreed to the SYSREQ function receives an AO command, it should simply ignore it. Any attempt by the host application to send data to the client while the connection is “suspended” should be responded to by the server with a negative response, sense code 0 × 082D, indicating an “LU Busy” condition. The server should not transmit anything to the client on behalf of the host application. While the connection is “suspended,” any data messages (except TN3270E responses) exchanged between the client and server should have the `DATA-TYPE` flag set to `SSCP-LU-DATA.`

At this point, the behavior of the server depends upon whether it is allowed to send USS commands on the SSCP-LU session. Servers that have this ability should simply act as a vehicle for passing USS commands and responses between the client and the SSCP.

Servers that are not allowed to send USS commands on the SSCP-LU session should behave as follows:

1. If the user transmits the string LOGOFF (upper- or lowercase), the server should send an UNBIND SNA RU to the host application. This will result in termination of the PLU-SLU session. If the `BIND-IMAGE` function was agreed on, then the server should also send a data message to the client with the `DATA-TYPE` flag set to `UNBIND` and the data portion set to 0 × 01.

2. If the user transmits anything other than LOGOFF, the server should respond with the string `COMMAND UNRECOGNIZED` to the client. The server should not send anything to the host application on behalf of the client.

Regardless of which kind of server is present (i.e., whether it may send USS commands on the SSCP-LU session), while the connection is suspended, the user may press the SYSREQ key again. This will result in the transmission of another AO to the server. The server should then send to the host application an `LUSTAT RU` with a value of 0 × 082B indicating “presentation space integrity lost.” The server will then “unsuspend” the TELNET connection to the client, meaning that it will allow the host application to once again send data to the client.

3270 Structured Fields

3270 structured fields provide a much wider range of features than “old-style” 3270 data, such as support for graphics, partitions, and IPDS printer data streams. It would be unreasonable to expect all TN3270E clients to support all possible structured field functions, yet there must be a mechanism that enables those clients capable of supporting some or all structured field functions to indicate their wishes.

The design of 3270 structured fields provides a convenient means to convey the level of support (including no support) for the various structured field functions. This mechanism is the READ PARTITION QUERY (RPQ) command, which is sent from the host application to the device. The device responds with a QUERY REPLY structured field(s) listing which, if any, structured field functions it supports.

The QUERY REPLY is also used to indicate some device capabilities which do not require the use of structured fields, such as extended color support and extended highlighting capability. Most host applications will use RPQ to precisely determine a device’s capabilities when there has been some indication that the device supports the “extended data stream.”

Therefore, all TN3270E clients that negotiate a terminal device type that contains an -E suffix, the Dynamic terminal type, or a printer device type, must be able to respond to an RPQ command. Note that these clients must support both the RPQ (type 02), and all forms of the RPQ list (type 03).

3270 Data-Stream Notes

Implementers of TN3270E clients should note that the command codes for the various 3270 READ and WRITE commands have different values depending on how the server is connected to the host (local vs. remote, SNA vs. non-SNA). Clients should be coded to check for the various possible values if they wish to be compatible with the widest range of servers.

Negotiation of the TN3270E TELNET Option

Since TN3270E is a TELNET option, both client and server are free to attempt to initiate negotiation of TN3270E by sending a `DO TN3270E` command. However, just as is usually the case with the TELNET `DO TERMINAL-TYPE,` it is anticipated that the server will normally be the one sending the `DO TN3270E,` and the client will be responding with a `WILL` or a `WONT TN3270E.`

Keep-Alive Mechanism

In many environments, it is very helpful to have in place a mechanism that allows timely notification of the loss of a 3270 session. TN3270E does not require that any form of keep-alive mechanism be employed by either clients or servers, but implementers wishing to support such a mechanism should consider the following guidelines.

There are at least two possible means of providing a keep-alive mechanism in TN3270E: the TELNET `IAC NOP` command and the TELNET `DO TIMING-MARK` option. Both methods have their advantages and disadvantages. It is recommended that TN3270E clients and servers that support keep-alives should accept both NOPs and `TIMING-MARKs`, and that both sides should always respond to `TIMING-MARKs`.

Both clients and servers could be configured to “actively” implement keep-alives; that is, both sides could send a `TIMING-MARK` or an NOP in order to determine whether the partner is still alive. Alternatively, network administrators may wish to configure only one side to send `TIMING-MARKs` or NOPS; in his case, the other side would be a “passive” participant which simply responds to the keep-alives it receives.

Implementers who want their code to be capable of being an “active” keep-alive participant should make their client or server configurable so that administrators can set which, if any, keep-alive mechanism should be employed, and how often the NOP or `TIMING-MARK` should be sent on each session.

On failure of a session on which keep-alives are used, both parties should make the proper notifications. A client should give the user some indication of the failure, such as an error code in the operator information area of the screen. A server should notify the host application that the session has been terminated, for example, by sending an UNBIND with type CLEANUP in an SNA environment.

14.10 TN3270 Communication Examples

The following example shows a TN3270E-capable server and a traditional TN3270 client establishing a connection:

```
SERVER: IAC DO TN3270E
CLIENT: IAC WONT TN3270E
SERVER: IAC DO TERMINAL-TYPE
CLIENT: IAC WILL TERMINAL-TYPE
SERVER: IAC SB TERMINAL-TYPE SEND IAC SE
CLIENT: IAC SB TERMINAL-TYPE IS IBM-3278-2 IAC SE
SERVER: IAC DO EOR IAC WILL EOR
CLIENT: IAC WILL EOR IAC DO EOR
SERVER: IAC DO BINARY IAC WILL BINARY
CLIENT: IAC WILL BINARY IAC DO BINARY
```

<3270 data stream is exchanged>

The following example shows a TN3270E-capable server and a TN3270E-capable client establishing a generic pool (non-specific) terminal session:

```
SERVER: IAC DO TN3270E
CLIENT: IAC WILL TN3270E
SERVER: IAC SB TN3270E SEND DEVICE-TYPE IAC SE
CLIENT: IAC SB TN3270E DEVICE-TYPE REQUEST IBM-3278-2 IAC SE
SERVER: IAC SB TN3270E DEVICE-TYPE IS IBM-3278-2 CONNECT anyterm IAC SE
CLIENT: IAC SB TN3270E FUNCTIONS REQUEST RESPONSES IAC SE
SERVER: IAC SB TN3270E FUNCTIONS IS RESPONSES IAC SE
```

<270 data stream is exchanged>

The following example shows a TN3270E-capable server and a TN3270E-capable client establishing a terminal session where the client requests a specific device name:

```
SERVER: IAC DO TN3270E
CLIENT: IAC WILL TN3270E
SERVER: IAC SB TN3270E SEND DEVICE-TYPE IAC SE
CLIENT: IAC SB TN3270E DEVICE-TYPE REQUEST IBM-3278-5-E CONNECT myterm IAC
SE
SERVER: IAC SB TN3270E DEVICE-TYPE IS IBM-3278-5-E CONNECT myterm IAC SE
CLIENT: IAC SB TN3270E FUNCTIONS REQUEST RESPONSES BIND-IMAGE IAC SE
SERVER: IAC SB TN3270E FUNCTIONS IS RESPONSES BIND-IMAGE IAC SE
```

<3270 data stream is exchanged>

The following example shows a TN3270E-capable server and a TN3270E-capable client attempting to establish a terminal session; multiple attempts are necessary because the device name initially requested by the client is already in use:

```
SERVER: IAC DO TN3270E
CLIENT: IAC WILL TN3270E
SERVER: IAC SB TN3270E SEND DEVICE-TYPE IAC SE
CLIENT: IAC SB TN3270E DEVICE-TYPE REQUEST IBM-3278-5 CONNECT myterm IAC SE
SERVER: IAC SB TN3270E DEVICE-TYPE REJECT REASON DEVICE-IN-USE IAC SE
CLIENT: IAC SB TN3270E DEVICE-TYPE REQUEST IBM-3278-2 CONNECT herterm IAC SE
SERVER: IAC SB TN3270E DEVICE-TYPE IS IBM-3278-2 CONNECT herterm IAC SE
CLIENT: IAC SB TN3270E FUNCTIONS REQUEST RESPONSES IAC SE
SERVER: IAC SB TN3270E FUNCTIONS IS RESPONSES IAC SE
```

<3270 data stream is exchanged>

The following example shows a TN3270E-capable server and a TN3270E-capable client establishing a printer session where the client requests a specific device name, and where some amount of 3270 function negotiation is required before an agreement is reached:

```
SERVER: IAC DO TN3270E
CLIENT: IAC WILL TN3270E
SERVER: IAC SB TN3270E SEND DEVICE-TYPE IAC SE
CLIENT: IAC SB TN3270E DEVICE-TYPE REQUEST IBM-3287-1 CONNECT myprt IAC SE
SERVER: IAC SB TN3270E DEVICE-TYPE IS IBM-3287-1 CONNECT myprt IAC SE
CLIENT: IAC SB TN3270E FUNCTIONS REQUEST DATA-STREAM-CTL IAC
SERVER: IAC SB TN3270E FUNCTIONS REQUEST DATA-STREAM-CTL RESPONSES IAC SE
CLIENT: IAC SB TN3270E FUNCTIONS REQUEST DATA-STREAM-CTL IAC
SERVER: IAC SB TN3270E FUNCTIONS IS DATA-STREAM-CTL IAC SE
```

<3270 data stream is exchanged>

The following example shows a TN3270E-capable server and a TN3270E-capable client establishing first a generic terminal session, then a printer session in which the “partner” printer for the assigned terminal is requested:

```
SERVER: IAC DO TN3270E
CLIENT: IAC WILL TN3270E
SERVER: IAC SB TN3270E SEND DEVICE-TYPE IAC SE
CLIENT: IAC SB TN3270E DEVICE-TYPE REQUEST IBM-3278-2 IAC SE
SERVER: IAC SB TN3270E DEVICE-TYPE IS IBM-3278-2 CONNECT termXYZ IAC SE
CLIENT: IAC SB TN3270E FUNCTIONS REQUEST RESPONSES IAC SE
SERVER: IAC SB TN3270E FUNCTIONS IS RESPONSES IAC SE
```

<3270 data stream is exchanged>

```
.    .
.    .
.    .
```

<User decides to request a printer session, so client again connects to TELNET port on server>

SERVER: `IAC DO TN3270E`
CLIENT: `IAC WILL TN3270E`
SERVER: `IAC SB TN3270E SEND DEVICE-TYPE IAC SE`
CLIENT: `IAC SB TN3270E DEVICE-TYPE REQUEST IBM-3287-1 ASSOCIATE termXYZ IAC SE`
SERVER: `IAC SB TN3270E DEVICE-TYPE IS IBM-3287-1 CONNECT termXYZ’s-prt IAC SE`
CLIENT: `IAC SB TN3270E FUNCTIONS REQUEST SCS-CTL-CODES RESPONSES IAC SE`
SERVER: `IAC SB TN3270E FUNCTIONS IS SCS-CTL-CODES RESPONSES IAC SE`

<3270 data stream is exchanged>

14.11 Security Considerations

Security issues are not addressed in this document. It is anticipated that once authentication mechanisms have become well established, they can be utilized by TN3270E. One of the important uses of authentication would be to answer the question of whether a given user should be allowed to “use” a specific terminal or printer device name.

15
File Transfer Protocol

15.1 Perspective

Objectives of the File Transfer Protocol (FTP) are to (1) promote sharing of files (computer programs and/or data), (2) encourage indirect or implicit (via programs) use of remote computers, (3) shield a user from variations in file storage systems among hosts, and (4) transfer data reliably and efficiently. FTP, although usable directly by a user at a terminal, is designed mainly for use by programs.

This chapter explains the FTP specification. It also assumes a significant understanding of the Transmission Control Protocol (TCP) and the TELNET protocol.

15.2 Brief History of FTP

FTP has had a long evolution over the years. The first proposed file-transfer mechanisms were presented in 1971 and were developed for implementation on hosts at MIT.

RFC 172 provided a user-level protocol for file transfer between host computers (including terminal IMPs). A revision of this as RFC 265, restated FTP for additional review, while RFC 281 suggested further changes. The use of a “set data type” transaction was proposed in RFC 294 in January 1982.

RFC 354 obsolete RFCs 264 and 265. The File Transfer Protocol was now defined as a protocol for file transfer between hosts on the ARPAnet, with the primary function of FTP defined as transferring files efficiently and reliably among hosts and allowing the convenient use of remote file storage capabilities. RFC 385 further commented on errors, emphasis points, and additions to the protocol, while RFC 414 provided a status report on the working server and user FTPs. RFC 430, issued in 1973 (among other RFCs too numerous to mention) presented further comments on FTP. Finally, an “official” FTP document was published as RFC 454.

By July 1973, considerable changes from the previous versions of FTP were made, but the general structure remained the same. RFC 542 was published as a new “official” specification to reflect these changes. However, many implementations based on the older specification were not updated.

In 1974, RFCs 607 and 614 continued comments on FTP. RFC 624 proposed further design changes and minor modifications. In 1975, RFC 686, entitled *Leaving Well Enough Alone,* discussed the differences between all the early and later versions of FTP. RFC 691 presented a minor revision of RFC 686, regarding the subject of print files.

Motivated by the transition from the NCP to the TCP as the underlying protocol, a phoenix was born out of all of the above efforts in RFC 765 as the specification of FTP for use on TCP.

The following new optional commands are included in this edition of the specification: CDUP—change to parent directory, SMNT—structure mount, STOU—store unique, RMD—remove directory, MKD—make directory, PWD—print directory, and SYST—system.

15.3 Terminology

access controls Access controls define users’ access privileges to the use of a system and to the files in that system. Access controls are necessary to prevent unauthorized or accidental use of files. It is the prerogative of a server-FTP process to invoke access controls.

ASCII The ASCII character set is as defined in the *ARPA-Internet Protocol Handbook.* In FTP, ASCII characters are defined to be the lower half of an 8-bit code set (i.e., the most significant bit is zero).

byte size There are two byte sizes of interest in FTP: the logical byte size of the file and the transfer byte size used to transmit the data. The transfer byte size is always 8 bits but is not necessarily the byte size in which data are to be stored in a system, nor is it the logical byte size for interpretation of the structure of the data.

control connection The communication path between the user-PI and server-PI for the exchange of commands and replies. This connection follows the TELNET protocol.

data connection A full-duplex connection over which data are transferred, in a specified mode and type. The data transferred may be a part of a file, an entire file, or a number of files. The path may be between a server-DTP and a user-DTP, or between two server-DTPs.

data port The passive data-transfer process “listens” on the data port for a connection from the active transfer process in order to open the data connection.

data-transfer process (DTP) The data-transfer process establishes and manages the data connection. The DTP can be passive or active.

end of file (EOF) The end-of-file condition that defines the end of a file being transferred.

end of line (EOL) The end-of-line sequence defines the separation of printing lines. The sequence is *carriage return,* followed by *linefeed.*

end of record (EOR) The end-of-record condition that defines the end of a record being transferred.

error recovery A procedure that allows a user to recover from certain errors such as failure of either host system or transfer process. In FTP, error recovery may involve restarting a file transfer at a given checkpoint.

file An ordered set of computer data (including programs), of arbitrary length, uniquely identified by a pathname.

FTP commands A set of commands that constitute the control information flowing from the user-FTP to the server-FTP process.

mode The mode in which data are to be transferred via the data connection. The mode defines the data format during transfer, including EOR and EOF. The transfer modes defined in FTP are described in the section on transmission modes.

NVFS The network virtual file system. A concept which defines a standard network file system with standard commands and pathname conventions.

NVT The network virtual terminal as defined in the TELNET protocol.

page A file may be structured as a set of independent parts called *pages.* FTP supports the transmission of discontinuous files as independent indexed pages.

pathname Pathname is defined to be the character string which must be input to a file system by a user in order to identify a file. Pathname normally contains device and/or directory names, and filename specification. FTP does not yet specify a standard pathname convention. Each user must follow the filenaming conventions of the file systems involved in the transfer.

PI The protocol interpreter. The user and server sides of the protocol have distinct roles implemented in a user-PI and a server-PI.

record A sequential file may be structured as a number of contiguous parts called *records.* Record structures are supported by FTP, but a file need not have record structure.

reply A reply is an acknowledgment (positive or negative) sent from server to user via the control connection in response to FTP commands. The general form of a reply is a completion code (including error codes) followed by a text string. The codes are for use by programs, and the text is usually intended for human users.

server-DTP The data-transfer process, in its normal “active” state, establishes the data connection with the “listening” data port. It sets up parameters for transfer and storage, and transfers data on command from its PI. The DTP can be placed in a “passive” state to listen for, rather than initiate, a connection on the data port.

server-FTP process A process or set of processes which perform the function of file transfer in cooperation with a user-FTP process and, possibly, another server. The functions consist of a protocol interpreter (PI) and a data-transfer process (DTP).

server-PI The server protocol interpreter “listens” on port L for a connection from a user-PI and establishes a control communication connection. It receives standard FTP commands from the user-PI, sends replies, and governs the server-DTP.

type The data representation type used for data transfer and storage. *Type* implies certain transformations between the time of data storage and data transfer. The representation types defined in FTP are described in the section on establishing data connections.

user A person or a process on behalf of a person wishing to obtain file-transfer service. The human user may interact directly with a server-FTP process, but use of a user-FTP process is preferred since the protocol design is weighted toward automata.

user-DTP The data-transfer process “listens” on the data port for a connection from a server-FTP process. If two servers are transferring data between them, the user-DTP is inactive.

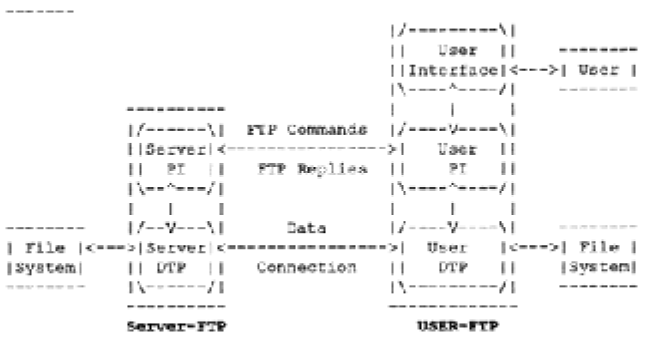
user-FTP process A set of functions including a protocol interpreter, a data-transfer process, and a user interface which together perform the function of file transfer in cooperation with one or more server-FTP processes. The user interface allows a local language to be used in the command-reply dialogue with the user.

user-PI The user protocol interpreter initiates the control connection from its port U to the server-FTP process, initiates FTP commands, and governs the user-DTP if that process is part of the file transfer.

15.4 The FTP Model

With these definitions in mind, let us study the FTP system shown in Fig. 15-1.

In Fig. 15-1 the user-protocol interpreter initiates the control connection. The control connection follows the TELNET protocol. At the initiation of the user, standard FTP commands are generated by the user-PI and transmitted to the server process via the control connec tion. (The user may establish a direct control connection to the server-FTP, from a terminal access control (TAC) terminal, for example, and generate standard FTP commands independently, bypassing the user-FTP process.) Standard replies are sent from the server-PI to the user-PI over the control connection in response to the commands.



NOTES:
1. The data connection may be used in either direction.
2. The data connection need not exist all of the time.

Figure 15-1
Model for FTP use.

The FTP commands specify the parameters for the data connection (data port, transfer mode, representation type, and structure) and the nature of file system operation (store, retrieve, append, delete, etc.). The user-DTP or its designate should “listen” on the specified data port, and the server should initiate the data connection and data transfer in accordance with the specified parameters. The data port need not be in the same host that initiates the FTP commands via the control connection, but the user or the user-FTP process must ensure a “listen” on the specified data port. It also should be noted that the data connection may be used for simultaneous sending and receiving.

In another situation a user might wish to transfer files between two hosts, neither of which is a local host. The user sets up control connections to the two servers and then arranges for a data connection between them. In this manner, control information is passed to the user-PI but data are transferred between the server data-transfer processes. Figure 15-2 shows a model of this server-server interaction.

The protocol requires that the control connections be open while data transfer is in progress. The user is responsible for requesting the clos ing of control connections on completion of the FTP service; however, it is the server who takes the action. The server may abort data transfer if the control connections are closed without command.

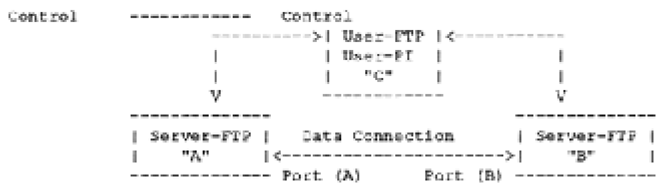


Figure 15-2
Server-server interaction.

The Relationship Between FTP and TELNET

The FTP uses the TELNET protocol on the control connection. This can be achieved in one of two ways: (1) the user-PI or the server-PI may implement the rules of the TELNET protocol directly in their own procedures or (2) the user-PI or the server-PI may make use of the existing TELNET module in the system. Ease of implementation, sharing code, and modular programming argue for the second approach. Efficiency and independence argue for the first approach. In practice, FTP relies on very little of the TELNET protocol, so the first approach does not necessarily involve a large amount of code.

15.5 Data-Transfer Functions

Files are transferred only via the data connection. The control connection is used for the transfer of commands, which describe the functions to be performed, and the replies to these commands. Several commands are concerned with the transfer of data between hosts. These data-transfer commands include the MODE command, which specifies how the data bits are to be transmitted, and the structure and TYPE commands, which are used to define how the data are to be represented. The transmission and representation are basically independent, but the *stream* transmission mode is dependent on the file structure attribute, and if *compressed* transmission mode is used, the nature of the filler byte depends on the representation type.

15.6 Data Representation and Storage

Data are transferred from a storage device in the sending host to a storage device in the receiving host. Often it is necessary to perform certain transformations on the data because data storage representations in the two systems are different. For example, NVT-ASCII has different data storage representations in different systems. Digital Equipment Corp. (DEC) TOPS-20s generally store NVT-ASCII as five 7-bit ASCII characters, left-justified in a 36-bit word. IBM mainframes store NVT-ASCII as 8-bit EBCDIC codes. Multics stores NVT-ASCII as four 9-bit characters in a 36-bit word. It is desirable to convert characters into the standard NVT-ASCII representation when transmitting text between dissimilar systems. The sending and receiving sites would have to perform the necessary transformations between the standard representation and their internal representations.

A different problem in representation arises when transmitting binary data (not character codes) between host systems with different word lengths. It is not always clear how the sender should send data and how the receiver should store it. For example, when transmitting 32-bit bytes from a 32-bit word-length system to a 36-bit word-length system, it may be desirable (for reasons of efficiency and usefulness) to store the 32-bit bytes right-justified in a 36-bit word in the latter system. In any case, the user should have the option of specifying data representation and transformation functions. It should be noted that FTP provides for very limited data-type representations. Transformations desired beyond this limited capability should be performed by the user directly.

15.7 Data Types

Data representations are handled in FTP by a user specifying a representation type. This type may implicitly (as in ASCII or EBCDIC) or explicitly (as in local byte) define a byte size for interpretation which is referred to as the *logical byte size.* This has nothing to do with the byte size used for transmission over the data connection, called the *transfer byte size,* and the two should not be confused. For example, NVT-ASCII has a logical byte size of 8 bits. If the type is local byte, then the `TYPE` command has an obligatory second parameter specifying the logical byte size. The transfer byte size is always 8 bits.

15.7.1 ASCII Type

This is the default type and must be accepted by all FTP implementations. It is intended primarily for the transfer of text files, except when both hosts would find the EBCDIC type more convenient.

The sender converts the data from an internal character representation to the standard 8-bit NVT-ASCII representation (see the TELNET specification). The receiver will convert the data from the standard form to its own internal form.

In accordance with the NVT standard, the <CRLF> sequence should be used where necessary to denote the end of a line of text. Using the standard NVT-ASCII representation means that data must be interpreted as 8-bit bytes. The format parameters for ASCII and EBCDIC types is discussed below.

15.7.2 EBCDIC Type

This type is intended for efficient transfer between hosts which use EBCDIC for their internal character representation.

For transmission, the data are represented as 8-bit EBCDIC characters. The character code is the only difference between the functional specifications of EBCDIC and ASCII types. The end-of-line character (as opposed to end-of-record character—see the discussion of structure) will probably be rarely used with EBCDIC type for purposes of denoting structure, but where it is necessary the <NL> character should be used.

15.7.3 Image Type

The data are sent as contiguous bits which, for transfer, are packed into the 8-bit transfer bytes. The receiving site must store the data as contiguous bits. The structure of the storage system might necessitate padding the file (or each record, for a record-structured file) to some convenient boundary (byte, word, or block). This padding, which must be all zeros, may occur only at the end of the file (or at the end of each record), and there must be a way of identifying the padding bits so that they may be stripped off if the file is retrieved. The padding transformation should be well publicized to enable a user to process a file at the storage site.

Image type is intended for the efficient storage and retrieval of files and for the transfer of binary data. It is recommended that this type be accepted by all FTP implementations.

15.7.4 Local Type

The data is transferred in logical bytes of the size specified by the obligatory second parameter: byte size. The value of byte size must be a decimal integer; there is no default value. The logical byte size is not necessarily the same as the transfer byte size. If there is a difference in byte sizes, then the logical bytes should be packed contiguously, disregarding transfer byte boundaries and with any necessary padding at the end. On reaching the receiving host, the data will be transformed in a manner dependent on the logical byte size and the particular host. This transformation must be invertible (i.e., an identical file can be retrieved if the same parameters are used) and should be well publicized by the FTP implementers.

For example, a user sending 36-bit floating-point numbers to a host with a 32-bit word could send that data as local byte with a logical byte size of 36. The receiving host would then be expected to store the logical bytes so that they could be easily manipulated; in this example putting the 36-bit logical bytes into 64-bit double words should suffice.

In another example, a pair of hosts with a 36-bit word size may send data to one another in words by using type L36. The data would be sent in the 8-bit transmission bytes packed so that 9 transmission bytes would carry two host words.

15.8 Format Control

Data types ASCII and EBCDIC also take a second (optional) parameter; this is to indicate what kind of vertical format control, if any, is associated with a file. The following data representation types are defined in FTP.

A character file may be transferred to a host for one of three purposes: for printing, for storage and later retrieval, or for processing. If a file is sent for printing, the receiving host must know how the vertical format control is represented. In the second case, it must be possible to store a file at a host and then retrieve it later in exactly the same form. Finally, it should be possible to move a file from one host to another and process the file at the second host without undue trouble. A single ASCII or EBCDIC format does not satisfy all these conditions. Therefore, these types have a second parameter specifying one of the following three formats:

1. *Nonprint.* This default format should be used if the second (format) parameter is omitted. Nonprint format must be accepted by all FTP implementations. The file need contain no vertical format information. If it is passed to a printer process, this process may assume standard values for spacing and margins. Normally, this format will be used with files destined for processing or just storage.

2. *TELNET format controls.* The file contains ASCII/EBCDIC vertical format controls (i.e., <CR>, <LF>, <NL>, <VT>, <FF>) which the printer process will interpret appropriately. <CRLF>, in exactly this sequence, also denotes end of line.

3. *Carriage Control* (ASA). The file contains ASA (FORTRAN) vertical format control characters. In a line or a record formatted according to the ASA Standard, the first character is not to be printed. Instead, it should be used to determine the vertical movement of the paper which should take place before the rest of the record is printed.

The ASA Standard specifies the following control characters:

Character	**Vertical Spacing**
Blank	Move paper up one line
0	Move paper up two lines
1	Move paper to top of next page
+	No movement, i.e., overprint

Clearly there must be some way for a printer process to distinguish the end of the structural entity. If a file has record structure (see Sec. 15.9), this is no problem; records will be explicitly marked during transfer and storage. If the file has no record structure, the <CRLF> end-of-line sequence is used to separate printing lines, but these format effects are overridden by the ASA controls.

15.9 Data Structure

In addition to different representation types, FTP allows the structure of a file to be specified. Three file structures are defined in FTP: (1) file structure, in which there is no internal structure and the file is considered to be a continuous sequence of data bytes; (2) record structure, where the file is made up of sequential records; and (3) page structure, where the file is made up of independent indexed pages.

1. *File structure.* This is the default to be assumed if the STRUCTURE command has not been used but both file and record structures must be accepted for “text” files (i.e., files with type ASCII or EBCDIC) by all FTP implementations. The structure of a file will affect the transfer mode, interpretation, and storage of the file. The “natural” structure of a file will depend on which host stores the file. A source-code file will usually be stored on an IBM mainframe in fixed-length records but on a DEC TOPS-20 as a stream of characters partitioned into lines, such as by <CRLF>. If file transfer between such disparate sites is to be useful, there must be some way for one site to recognize the other’s assumptions about the file. Since some sites are naturally file-oriented and others naturally record-oriented, problems may arise if a file with one structure is sent to a host oriented to the other. If a text file is sent with record structure to a host which is file-oriented, then that host should apply an internal transformation to the file based on the record structure. Obviously, this transformation should be useful, but it must also be invertible so that an identical file may be retrieved using record structure. For a file sent with file structure to a record-oriented host, the criteria the host should use to divide the file into records which can be processed locally must be determined. If this division is necessary, the FTP implementation should use the end-of-line sequence, <CRLF> for ASCII, or <NL> for EBCDIC text files, as the delimiter. If an FTP implementation adopts this technique, it must be prepared to reverse the transformation if the file is retrieved with file structure. In file structure there is no internal structure and the file is considered to be a continuous sequence of data bytes.

2. *Record structure.* Record structures must be accepted for “text” files (i.e., files with type ASCII or EBCDIC) by all FTP implementations. In record structure the file is made up of sequential records.

3. *Page structure.* To transmit files that are discontinuous, FTP defines a page structure. Files of this type are sometimes known as *random access files* or even as “holey files.” These files sometimes contain other information associated with the file as a whole (e.g., a file descriptor), or with a section of the file (e.g., page access controls), or both. In FTP, the sections of the file are called *pages.*

To provide for various page sizes and associated information, each page is sent with a page header. The page header has the following defined fields: (1) *header length*—the number of logical bytes in the page header including this byte (the minimum header length is 4); (2) *page index*—the logical page number of this section of the file (this is not the transmission sequence number of this page, but the index used to identify this page of the file); (3) *data length*—the number of logical bytes in the page data (the minimum data length is 0); (4) *page type*—indicates the type of page (the following page types are defined: 0 = *last page*—used to indicate the end of a paged structured transmission—header length must be 4 and data length, 0; 1 = *simple page*—normal type for simple paged files with no page level-associated control information—header length must be 4; 2 = *descriptor page*—used to transmit the descriptive information for the file as a whole; 3 = *access-controlled page*—includes an additional header field for paged files with page level access control information—header length must be 5; and (5) *optional fields*—further header fields may be used to supply per-page control information, for example, per-page access control).

All fields are one logical byte in length. The logical byte size is specified by the TYPE command. A file must be stored and retrieved with the same parameters if the retrieved version is to be identical to the version originally transmitted. Conversely, FTP implementations must return a file identical to the original if the parameters used to store and retrieve a file are the same.

15.10 Establishing Data Connections

The mechanics of transferring data consists of setting up the data connection to the appropriate ports and choosing the parameters for trans fer. Both the user and the server-DTPs have a default data port. The user-process default data port is the same as the control connection port (i.e., U). The server-process default data port is the port adjacent to the control connection port (i.e., L-1).

The transfer byte size is 8-bit bytes. This byte size is relevant only for the actual transfer of the data; it has no bearing on representation of the data within a host's file system.

The passive data-transfer process (this may be a user-DTP or a second server-DTP) "listens" on the data port prior to sending a transfer request command. The FTP request command determines the direction of the data transfer. The server, on receiving the transfer request, will initiate the data connection to the port. When the connection is established, the data transfer begins between DTPs, and the server-PI sends a confirming reply to the user-PI. Every FTP implementation must support the use of the default data ports, and only the user-PI can initiate a change to nondefault ports.

It is possible for the user to specify an alternate data port by use of the PORT command. The user may want a file dumped on a TAC line printer or retrieved from a third-party host. In the latter case, the user-PI sets up control connections with both server-PIs. One server is then told (by an FTP command) to "listen" for a connection which the other will initiate. The user-PI sends one server-PI a PORT command indicating the data port of the other. Finally, both are sent the appropriate transfer commands.

In general, it is the server's responsibility to maintain the data connection—to initiate it and to close it. The exception to this is when the user-DTP is sending the data in a transfer mode that requires the connection to be closed to indicate end of file (EOF). The server *must* close the data connection under the following conditions: (1) The server has completed sending data in a transfer mode that requires a close to indicate EOF, (2) the server receives an ABORT command from the user, (3) the port specification is changed by a command from the user, (4) the control connection is closed legally or otherwise, or (5) an irrecoverable error condition occurs. Otherwise the close is a server option, the exercise of which the server must indicate to the user-process by either a 250 or 226 reply only.

15.11 Data Connection Management

1. *Default data connection ports.* All FTP implementations must support use of the default data connection ports, and only the user-PI may initiate the use of nondefault ports.

2. *Negotiating nondefault data ports.* The user-PI may specify a nondefault user-side data port with the PORT command. The user-PI may request the server side to identify a nondefault server-side data port with the PASV command. Since a connection is defined by the pair of addresses, either of these actions will suffice to get a different data connection, but issuing both commands to use new ports on both ends of the data connection is permitted.

3. *Reuse of the data connection.* When using the stream mode of data transfer, the end of the file must be indicated by closing the connection. This causes a problem if multiple files are to be transferred in the session because TCP must hold the connection record for a timeout period to guarantee reliable communication. Thus the connection cannot be reopened at once. There are two solutions to this problem: negotiate a nondefault port or use another transfer mode.

A comment on transfer modes. The stream transfer mode is inherently unreliable, since one cannot determine whether the connection closed prematurely. The other transfer modes (block, compressed) do not close the connection to indicate the end of file. They have enough FTP encoding that the data connection can be parsed to determine the end of the file. Thus using these modes one can leave the data connection open for multiple file transfers.

15.12 Transmission Modes

The next consideration in transferring data is choosing the appropriate transmission mode. There are three modes: one which formats the data and allows for restart procedures, one which also compresses the data for efficient transfer, and one which passes the data with little or no processing. In this last case the mode interacts with the structure attribute to determine the type of processing. In the compressed mode, the representation type determines the filler byte.

All data transfers must be completed with an end-of-file (EOF) marker which may be explicitly stated or implied by closing the data connection. For files with record structure, all the end-of-record (EOR) markers are explicit, including the final one. For files transmitted in page structure a “last-page” page type is used.

Note: *In the rest of this section, byte means “transfer byte” except where explicitly stated otherwise.*

For the purpose of standardized transfer, the sending host will translate its internal end-of-line or end-of-record denotation into the representation prescribed by the transfer mode and file structure, and the receiving host will perform the inverse translation to its internal denotation. An IBM mainframe record count field may not be recognized at another host, so the end-of-record information may be transferred as a 2-byte control code in stream mode or as a flagged bit in a block or compressed mode descriptor. End-of-line in an ASCII or EBCDIC file with no record structure should be indicated by <CRLF> or <NL>, respectively. Since these transformations imply extra work for some systems, identical systems transferring non-record-structured text files might wish to use a binary representation and stream mode for the transfer.

The following transmission modes are defined in FTP.

15.12.1 Stream Mode

The data are transmitted as a stream of bytes. There is no restriction on the representation type used; record structures are allowed.

In a record-structured file EOR and EOF will each be indicated by a 2-byte control code. The first byte of the control code will be all ones, the escape character. The second byte will have the low-order bit on and zeros elsewhere for EOR and the second low-order bit on for EOF; that is, the byte will have value 1 for EOR and value 2 for EOF. EOR and EOF may be indicated together on the last byte transmitted by turning both low-order bits on (i.e., the value 3). If a byte of all ones was intended to be sent as data, it should be repeated in the second byte of the control code. If the structure is a file structure, the EOF is indicated by the sending host closing the data connection and all bytes are data bytes.

15.12.2 Block Mode

The file is transmitted as a series of data blocks preceded by one or more header bytes. The header bytes contain a count field and descriptor code. The count field indicates the total length of the data block in bytes, thus marking the beginning of the next data block (there are no filler bits). The descriptor code defines the last block in the file (EOF), the last block in the record (EOR), and the restart marker.

This last code is *not* intended for error control within FTP. It is motivated by the desire of sites to exchange certain types of data (e.g., seismic or weather data) to send and receive all the data despite local errors (such as magnetic tape-read errors), but to indicate in the transmission that certain portions are suspect). Record structures are allowed in this mode, and any representation type may be used.

The block header consists of 3 bytes. Of the 24 bits of header information, the 16 low-order bits represent byte count, and the 8 high-order bits represent descriptor codes as shown in Fig. 15-3.

The descriptor codes are indicated by bit flags in the descriptor byte. Four codes have been assigned, where each code number is the decimal value of the corresponding bit in the byte.

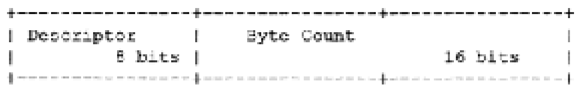


Figure 15-3
Block header.

Code	Meaning
128	End of data block is EOR
64	End of data block is EOF
32	Suspected errors in data block
16	Data block is a restart marker

With this encoding, more than one descriptor-coded condition may exist for a particular block. As many bits as necessary may be flagged.

The restart marker is embedded in the data stream as an integral number of 8-bit bytes representing printable characters in the language being used over the control connection (e.g., default—NVT-ASCII). <SP> (space, in the appropriate language) must *not* be used *within* a restart marker. For example, to transmit a six-character marker, the information shown in Fig. 15-4 would be sent.

15.12.3 Compressed Mode

Three kinds of information are sent: *regular data,* sent in a byte string; *compressed data,* consisting of replications or filler; and *control information,* sent in a 2-byte escape sequence. If *n*<0 bytes (up to 127) of regular data are sent, these *n* bytes are preceded by a byte with the left- most bit set to 0 and the rightmost 7 bits containing the number *n.* (See Fig. 15-5.)

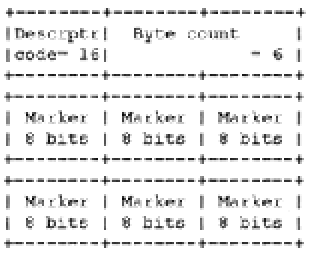


Figure 15-4
Block-mode transmission of a six-character marker.

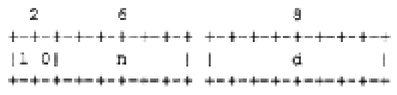


Figure 15-5
Byte string.

To compress a string of *n* replications of the data byte *d*, 2 bytes are sent (Fig. 15-6).

A string of *n* filler bytes can be compressed into a single byte, where the filler byte varies with the representation type. If the type is ASCII or EBCDIC, the filler byte is <SP> (space, ASCII code 32, EBCDIC code 64). If the type is Image or Local byte, the filler is a zero byte. (See Fig. 15-7.)

The escape sequence is a double byte, consisting of the escape byte (all zeros) and the byte containing descriptor codes as defined in block mode. The descriptor codes have the same meaning as in block mode and apply to the succeeding string of bytes.

Compressed mode is useful for increasing bandwidth on very large network transmissions at a little extra CPU cost. It can be most effectively used to reduce the size of printer files such as those generated by RJE hosts.

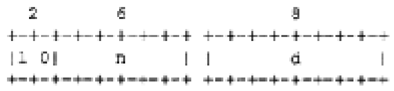


Figure 15-6
Replicated byte.

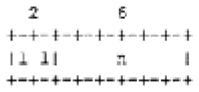


Figure 15-7
Filler string.

15.13 Error Recovery and Restart

There is no provision for detecting bits lost or scrambled in data transfer; this level of error control is handled by the TCP. However, a restart procedure is provided to protect users from gross system failures (including failures of a host, an FTP-process, or the underlying network).

The restart procedure is defined only for the block and compressed modes of data transfer. It requires the sender of data to insert a special marker code in the data stream with some marker information. The marker information has meaning only to the sender, but must consist of printable characters in the default or negotiated language of the control connection (ASCII or EBCDIC). The marker could represent a bit-count, a record-count, or any other information by which a system may identify a data checkpoint. The data receiver, if it implements the restart procedure, would then mark the corresponding position of this marker in the receiving system, and return this information to the user.

In the event of a system failure, the user can restart the data transfer by identifying the marker point with the FTP restart procedure. The following example illustrates the use of the restart procedure. The data sender inserts an appropriate marker block in the data stream at a convenient point. The receiving host marks the corresponding data point in its file system and conveys the last known sender and receiver marker information to the user, either directly or over the control connection in a 110 reply (depending on who is the sender). In the event of a system failure, the user or controller process restarts the server at the last server marker by sending a restart command with server’s marker code as its argument. The restart command is transmitted over the control connection and is immediately followed by the command (e.g., RETR, STOR, or LIST) which was being executed when the system failure occurred.

15.14 File-Transfer Functions

The communication channel from the user-PI to the server-PI is established as a TCP connection from the user to the standard server port. The user protocol interpreter is responsible for sending FTP commands and interpreting the replies received; the server-PI interprets commands, sends replies, and directs its DTP to set up the data connection and transfer the data. If the second party to the data transfer (the passive transfer process) is the user-DTP, it is governed through the internal protocol of the user-FTP host; if it is a second server-DTP, it is governed by its PI on command from the user-PI. The FTP replies are discussed in the next section. In the description of a few of the commands in this section, it is helpful to be explicit about the possible replies.

15.15 FTP Commands

Access-Control Commands

The following commands specify access control identifiers (command codes are shown in parentheses).

1. *User name (USER).* The argument field is a TELNET string identifying the user. The user identification is that which is required by the server for access to its file system. This command will normally be the first command transmitted by the user after the control connections are made (some servers may require this). Additional identification information in the form of a password and/or an account command may also be required by some servers. Servers may allow a new USER command to be entered at any point in order to change the access control and/or accounting information. This has the effect of flushing any user, password, and account information already supplied and beginning the login sequence again. All transfer parameters are unchanged and any file transfer in progress is completed under the old access-control parameters.

2. *Password (PASS).* The argument field is a TELNET string specifying the user's password. This command must be immediately preceded by the user name command, and, for some sites, completes the user's identification for access control. Since password information is quite sensitive, it is desirable in general to "mask" it or suppress typeout. It appears that the server has no foolproof way to achieve this. It is therefore the responsibility of the user-FTP process to hide the sensitive password information.

3. *Account (ACCT).* The argument field is a TELNET string identifying the user's account. The command is not necessarily related to the USER command, as some sites may require an account for login and others only for specific access, such as storing files. In the latter case the command may arrive at any time. Reply codes are used to differentiate these cases for the automation: when account information is required for login, the response to a successful PASSWORD command is reply code 332. On the other hand, if account information is *not* required for login, the reply to a successful PASSWORD command is 230; and if the account information is needed for a command issued later in the dialog, the server should return a 332 or 532 reply depending on whether it stores (pending receipt of the ACCOUNT command) or discards the command, respectively.

4. *Change working directory (CWD).* This command allows the user to work with a different directory or dataset for file storage or retrieval without altering his login or accounting information. Transfer parameters are similarly unchanged. The argument is a pathname specifying a directory or other system-dependent file group designator.

5. *Change to parent directory (CDUP).* This command is a special case of CWD, and is included to simplify the implementation of pro grams for transferring directory trees between operating systems having different syntaxes for naming the parent directory. The reply codes shall be identical to the reply codes of CWD.

6. *Structure mount (SMNT).* This command allows one to mount a different file system data structure without altering one's login or accounting information. Transfer parameters are similarly unchanged. The argument is a pathname specifying a directory or other system-dependent file group designator.

7. *Reinitialize (REIN).* This command terminates a USER, flushing all I/O and account information, except to allow any transfer in progress to be completed. All parameters are reset to the default settings and the control connection is left open. This is identical to the state in which a user may be immediately after the control connection is opened. A USER command may be expected to follow.

8. *Logout (QUIT).* This command terminates a USER, and if file transfer is not in progress, the server closes the control connection. If file transfer is in progress, the connection will remain open for result response and the server will then close it. If the user-process is transferring files for several USERs but does not wish to close and then reopen connections for each, the REIN command should be used instead of QUIT. An unexpected close on the control connection will cause the server to take the effective action of an abort (ABOR) and a logout (QUIT).

15.15.1 Transfer Parameter Commands

All data-transfer parameters have default values, and the commands specifying these parameters are required only if the default parameter values are to be changed. The default value is the last specified value, or if no value has been specified, the standard default value is as stated here. This implies that the server must “remember” the applicable default values. The commands may be in any order except that they must precede the FTP service request. The following commands specify data-transfer parameters:

1. *Data port (PORT).* The argument is a `HOST-PORT` specification for the data port to be used in data connection. There are defaults for both the user and server data ports, and under normal circumstances this command and its reply are not needed. If this command is used, the argument is the concatenation of a 32-bit Internet host address and a 16-bit TCP port address. This address information is broken into 8-bit fields, and the value of each field is transmitted as a decimal number (in character string representation). The fields are separated by commas. A port command would be `PORT h1,h2,h3,h4,p1,p2,` where `h1` is the high-order 8 bits of the Internet host address

2. *Passive (PASV).* This command requests the server-DTP to “listen” on a data port (which is not its default data port) and to wait for a connection rather than initiate one on receipt of a transfer command. The response to this command includes the host and port address this server is listening on.

3. *Representation type (TYPE).* The argument specifies the representation type. Several types take a second parameter. The first parameter is denoted by a single TELNET character, as is the second format parameter for ASCII and EBCDIC; the second parameter for local byte is a decimal integer to indicate byte size. The parameters are separated by a <SP> (space, ASCII code 32). The following codes are assigned for type:

```
           \    /
A - ASCII  |    | N - Nonprint
           |-><-| T - TELNET format effecters
E - EBCDIC |    | C - Carriage control (ASA)
           /    \
I - Image
L <byte size> - Local byte Byte size
```

The default representation type is ASCII nonprint. If the format parameter is changed, and later just the first argument is changed, the format parameter then returns to the nonprint default.

4. *File structure (STRU).* The argument is a single TELNET character code specifying file structure described in the section on data representation and storage. The codes assigned for structure are F—file (no record structure), R—record structure, and P—page structure. The default structure is *file.*

5. *Transfer mode (MODE).* The argument is a single TELNET character code specifying the data-transfer modes described in the section on transmission modes. The codes assigned for transfer modes are S—stream, B—block, and C—compressed. The default transfer mode is *stream.*

15.15.2 FTP Service Commands

The FTP service commands define the file-transfer or the file system function requested by the user. The argument of an FTP service command will normally be a pathname. The syntax of pathnames must conform to server site conventions (with standard defaults applicable) and the language conventions of the control connection. The suggested default handling is to use the last specified device, directory or file name, or the standard default defined for local users. The commands may be in any order except that a “rename from” command must be followed by a “rename to” command and the restart command must be followed by the interrupted service command (e.g., STOR or ETR). The data, when transferred in response to FTP service commands, shall always be sent over the data connection, except for certain informative replies. The following commands specify FTP service requests:

1. *Retrieve (RETR).* This command causes the server-DTP to transfer a copy of the file, specified in the pathname, to the server- or user-DTP at the other end of the data connection. The status and contents of the file at the server site shall be unaffected.

2. *Store (STOR).* This command causes the server-DTP to accept the data transferred via the data connection and to store the data as a file at the server site. If the file specified in the pathname exists at the server site, then its contents shall be replaced by the data being transferred. A new file is created at the server site if the file specified in the pathname does not already exist.

3. *Store unique (STOU).* This command behaves like STOR except that the resultant file is to be created in the current directory under a name unique to that directory. The 250 *transfer started* response must include the name generated.

4. *Append (with CREATE*) (*APPE).* This command causes the server-DTP to accept the data transferred via the data connection and to store the data in a file at the server site. If the file specified in the pathname exists at the server site, then the data shall be appended to that file; otherwise the file specified in the pathname shall be created at the server site.

5. *Allocate (ALLO).* This command may be required by some servers to reserve sufficient storage to accommodate the new file to be transferred. The argument shall be a decimal integer representing the number of bytes (using the logical byte size) of storage to be reserved for the file. For files sent with record or page structure, a maximum record or page size (in logical bytes) might also be necessary; this is indicated by a decimal integer in a second argument field of the command. This second argument is optional, but when present should be separated from the first by the three TELNET characters <SP>, R, and <SP>. This command shall be followed by a STORe or APPEnd command. The ALLO command should be treated as a NOOP (no operation) by servers which do not require that the maximum size of the file be declared beforehand, and servers interested in only the maximum record or page size should accept a dummy value in the first argument and ignore it.

6. *Restart (REST).* The argument field represents the server marker at which file transfer is to be restarted. This command does not cause file transfer but skips over the file to the specified data checkpoint. This command shall be immediately followed by the appropriate FTP service command which shall cause file transfer to resume.

7. *Rename from (RNFR).* This command specifies the old pathname of the file which is to be renamed. This command must be immediately followed by a “rename to” command specifying the new file pathname.

8. *Rename to (RNTO).* This command specifies the new pathname of the file specified in the immediately preceding “rename from” command. Together the two commands cause a file to be renamed.

9. *Abort (ABOR).* This command tells the server to abort the previous FTP service command and any associated transfer of data. No action is to be taken if the previous command has been completed (including data transfer). The control connection is not to be closed by the server, but the data connection must be closed. There are two cases for the server on receipt of this command: (1) the FTP service command was already completed, or (2) the FTP service command is still in progress. In case 1, the server closes the data connection (if it is open) and responds with a 226 reply, indicating that the abort command was successfully processed. In case 2, the server aborts the FTP service in progress and closes the data connection, returning a 426 reply to indicate that the service request terminated abnormally. The server then sends a 226 reply, indicating that the abort command was successfully processed.

10. *Delete (DELE).* This command causes the file specified in the pathname to be deleted at the server site. If an extra level of protection is desired (such as the query “Do you really wish to delete?”), it should be provided by the user-FTP process.

11. *Remove directory (RMD).* This command causes the directory specified in the pathname to be removed as a directory (if the pathname is absolute) or as a subdirectory of the current working directory (if the pathname is relative).

12. *Make directory (MKD).* This command causes the directory specified in the pathname to be created as a directory (if the pathname is absolute) or as a subdirectory of the current working directory (if the pathname is relative).

13. *Print working directory (PWD).* This command causes the name of the current working directory to be returned in the reply.

14. *List (LIST).* This command causes a list to be sent from the server to the passive DTP. If the pathname specifies a directory or other group of files, the server should transfer a list of files in the specified directory. If the pathname specifies a file, then the server should send current information on the file. A null argument implies the user’s current working or default directory. The data transfer is over the data connection in type ASCII or EBCDIC (the user must ensure that the type is appropriately ASCII or EBCDIC). Since the information on a file may vary widely from system to system, this information may be hard to use automatically in a program, but may be quite useful to a human user.

15. *Name list (NLST).* This command causes a directory listing to be sent from the server to the user site. The pathname should specify a directory or other system-specific file group descriptor; a null argument implies the current directory. The server will return a stream of names of files and no other information. The data will be transferred in ASCII or EBCDIC type over the data connection as valid pathname strings separated by <CRLF> or <NL>. (Again the user must ensure that the *type* is correct.) This command is intended to return information that can be used by a program to further process the files automatically, such as in the implementation of a “multiple get” function.

16. *Site parameters (SITE).* This command is used by the server to provide services specific to the system that are essential to file transfer but not sufficiently universal to be included as commands in the protocol. The nature of these services and the specification of their syntax can be stated in a reply to the HELP SITE command.

17. *System (SYST).* This command is used to find out the type of operating system at the server. The reply shall have as its first word one of the system names listed in the current version of the *Assigned Numbers* document.

18. *Status (STAT).* This command shall cause a status response to be sent over the control connection in the form of a reply. The command may be sent during a file transfer, along with the TELNET IP and SYNC signals, in which case the server will respond with the status of the operation in progress, or it may be sent between file transfers. In the latter case, the command may have an argument field. If the argument is a pathname, the command is analogous to the “list” command except that data shall be transferred over the control connection. If a partial pathname is given, the server may respond with a list of filenames or attributes associated with that specification. If no argument is given, the server should return general status information about the server FTP process. This should include current values of all transfer parameters and the status of connections.

19. *Help (HELP).* This command shall cause the server to send helpful information regarding its implementation status over the control connection to the user. The command may take an argument (e.g., any command name) and return more specific information as a response. The reply is type 211 or 214. It is suggested that HELP be allowed before entering a USER command. The server may use this reply to specify site-dependent parameters, such as in response to HELP SITE.

20. *No operation (NOOP).* This command does not affect any parameters or previously entered commands. It specifies no action other than that the server send an OK reply.

The File Transfer Protocol follows the specifications of the TELNET protocol for all communications over the control connection. Since the language used for TELNET communication may be a negotiated option, all references in the next two sections will be to the “TELNET language” and the corresponding “TELNET end-of-line code.” Currently, one may take these to mean NVT-ASCII and <CRLF>. No other specifications of the TELNET protocol will be cited.

FTP commands are TELNET strings terminated by the TELNET end-of-line code. The command codes themselves are alphabetic characters terminated by the character <SP> (space) if parameters follow and TELNET-EOL otherwise.

FTP commands may be partitioned as those specifying access-control identifiers, data-transfer parameters, or FTP service requests. Certain commands (e.g., ABOR, STAT, QUIT) may be sent over the control connection while a data transfer is in progress. Some servers may not be able to monitor the control and data connections simultaneously, in which case some special action will be necessary to get the server’s attention. The following ordered format is tentatively recommended: (1) user system inserts the TELNET *interrupt process* (IP) signal in the TELNET stream; (2) user system sends the TELNET SYNC signal; (3) user system inserts the command (e.g., ABOR) in the TELNET stream; and (4) server PI, after receiving IP, scans the TELNET stream for *exactly one* FTP command. (For other servers this may not be necessary, but these actions should have no unusual effect.)

15.16 FTP Replies

Replies to File Transfer Protocol commands are devised to ensure the synchronization of requests and actions in the process of file transfer, and to guarantee that the user process always knows the state of the server. Every command must generate at least one reply; if more than one reply is generated, the multiple replies must be easily distinguished. In addition, some commands occur in sequential groups, such as USER, PASS and ACCT, or RNFR and RNTO. The replies show the existence of an intermediate state if all preceding commands have been successful. A failure at any point in the sequence necessitates the repetition of the entire sequence from the beginning. The details of the command-reply sequence are made explicit in a set of state diagrams later in this chapter (Figs. 15.8 to 15.13).

An FTP reply consists of a three-digit number (transmitted as three alphanumeric characters) followed by some text. The number is intend ed for use by automata to determine what state to enter next; the text is intended for the human user. The three digits should contain enough encoded information so that the user-process (the user-PI) will not need to examine the text and may either discard it or pass it on to the user, as appropriate. In particular, the text may be server-dependent, so texts may vary for each reply code.

A reply is defined to contain the three-digit code, followed by a space <SP>, followed by one line of text (where some maximum line length has been specified), and terminated by the TELNET EOL code. In some cases, however, the text will be longer than a single line; then the complete text must be bracketed to inform the user-process when it may stop reading the reply (i.e., stop processing input on the control connection) and do other things. This requires a special format on the first line to indicate that more than one line is coming, and another on the last line to designate it as the last. At least one of these must contain the appropriate reply code to indicate the state of the transaction. To satisfy all factions, it was decided that both the first and last line codes should be the same. Thus the format for multiline replies is that the first line will begin with the exact required reply code, followed immediately by a hyphen (-) (also used as minus), followed by text. The last line will begin with the same code, followed immediately by a space <SP>, optionally some text, and the TELNET EOL code. For example, 123—first line; 234—second line, a line beginning with numbers; and 123—the last line. The user-process then simply needs to search for the second occurrence of the same reply code, followed by <SP> (space), at the beginning of a line, and ignore all intermediary lines. If an intermediary line begins with a three-digit number, the server must pad the front to avoid confusion.

This scheme allows standard system routines to be used for reply information (such as for the STAT reply), with “artificial” first and last lines tacked on. In rare cases where these routines are able to generate three digits and a space at the beginning of any line, the beginning of each text line should be offset by some neutral text, such as a space. This scheme assumes that multiline replies may not be nested.

The three digits of the reply each have a special significance. This is intended to allow a range of very simple to very sophisticated responses by the user-process. The first digit denotes whether the response is good, bad, or incomplete. Referring to the state diagram, an unsophisticated user-process will be able to determine its next action (proceed as planned, redo, retrench, etc.) by simply examining this first digit. A user-process that wants to know approximately what kind of error occurred (e.g., file system error, command syntax error) may examine the second digit, reserving the third digit for the finest gradation of information (e.g., RNTO command without a preceding RNFR).

There are five values for the first digit of the reply code:

`1yz Positive Preliminary reply`	The requested action is being initiated; expect another reply before proceeding with a new command. (The user-process sending another command before the completion reply would be in violation of protocol, but server-FTP processes should queue any commands that arrive while a preceding command is in progress.) This type of reply can be used to indicate that the command was accepted and the user-process may now pay attention to the data connections, for implementations where simultaneous monitoring is difficult. The server-FTP process may send at most one `1yz` reply per command.
`2yz Positive Completion reply`	The requested action has been successfully completed. A new request may be initiated.
`3yz Positive Intermediate reply`	The command has been accepted, but the requested action is being held in abeyance, pending receipt of further information. The user should send another command specifying this information. This reply is used in command sequence groups.
`4yz Transient Negative Completion reply`	The command was not accepted and the requested action did not take place, but the error condition is temporary and the action may be requested again. The user should return to the beginning of the command sequence, if any. It is difficult to assign a meaning to *transient*, particularly when two distinct sites (server- and user-processes) have to agree on the interpretation. Each reply in the `4yz` category might have a slightly different time value, but the intent is that the user-process is encouraged to try again. A rule of thumb in determining if a reply fits into the `4yz` or the `5yz` (Permanent Negative) category is that replies are `4yz` if the commands can be repeated without any change in command form or in properties of the user or server (e.g., the command is spelled the same with the same arguments used; user does not change user file access or user name; server does not put up a new implementation).
`5yz Permanent Negative Completion reply`	The command was not accepted and the requested action did not take place. The user-process is discouraged from repeating the exact request (in the same sequence). Even some “permanent” error conditions can be corrected, so the human user may want to direct the user-process to reinitiate the command sequence by direct action at some point in the future (e.g., after spelling has been changed or user has altered user directory status).

The following function groupings are encoded in the second digit:

`x0z`	Syntax—replies referring to syntax errors, syntactically correct commands that don’ t fit any functional category, unimplemented or superfluous commands
`x1z`	Information—replies to requests for information, such as status or help
`x2z`	Connections—replies referring to the control and data connections
`x3z`	Authentication and accounting—replies for the login process and accounting procedures
`x4z`	Unspecified as yet
`x5z`	File system—replies indicating the status of the server file system vis-à -vis the requested transfer or other file system action

The third digit gives a finer gradation of meaning in each function category, specified by the second digit. The list of replies in Sec. 15.16.1 illustrates this. Note that the text associated with each reply is recommended, rather than mandatory, and may even change according to the command with which it is associated. The reply codes, on the other hand, must strictly follow the defined specification; that is, server implementations should not invent new codes for situations that are only slightly different from the ones described here, but rather should adapt codes already defined.

A command such as TYPE or ALLO whose successful execution does not offer the user-process any new information will cause a 200 reply to be returned. If the command is not implemented by a particular server-FTP process because it has no relevance to that computer system, for example, ALLO at a TOPS-20 site, a *positive completion* reply is still desired so that the simple user-process knows it can proceed with its course of action. A 202 reply is used in this case with, for example, the reply text: “No storage allocation necessary.” If, on the other hand, the command requests a non-site-specific action and is unimplemented, the response is 502. A refinement of that is the 504 reply for a command that is implemented, but that requests an unimplemented parameter.

15.16.1 FTP Reply Codes by Function Groups

200	Command okay.
500	Syntax error, command unrecognized (may include errors such as command line too long).
501	Syntax error in parameters or arguments.
202	Command not implemented, superfluous at this site.
502	Command not implemented.
503	Bad sequence of commands.
504	Command not implemented for that parameter.

110	Restart marker reply. In this case, the text is exact and not left to the particular implementation; it must read `MARK yyyy = mmmm`, where `yyyy` is the user-process data-stream marker and `mmmm` is the server's equivalent marker (note the spaces between markers and "`=` ")
211	System status, or system help reply.
212	Directory status.
213	File status.
214	Help message on how to use the server or the meaning of a particular nonstandard command. This reply is useful only to the human user.
215	`NAME` system type, where `NAME` is an official system name from the list in the IANA document.
120	Service ready in *nnn* minutes.
220	Service ready for new user.
221	Service closing control connection; logged out if appropriate.
421	Service not available, closing control connection. This may be a reply to any command if the service knows that it must shut down.
125	Data connection already open; transfer starting.
225	Data connection open; no transfer in progress.
425	Can't open data connection.
226	Closing data connection. Requested file action successful (e.g., file transfer or file abort).
426	Connection closed; transfer aborted.
227	Entering passive mode (`h1,h2,h3,h4,p1,p2`).
230	User logged in, proceed.
530	Not logged in.
331	User name okay, need password.
332	Need account for login.
532	Need account for storing files.
150	File status okay; about to open data connection.
250	Requested file action okay, completed.
257	`PATHNAME` created.
350	Requested file action pending further information.
450	Requested file action not taken; file unavailable (e.g., file busy).
550	Requested action not taken; file unavailable (e.g., file not found, no access).
451	Requested action aborted; local error in processing.
551	Requested action aborted; page type unknown.
452	Requested action not taken; insufficient storage space in system.
552	Requested file action aborted; exceeded storage allocation (for current directory or data set).
553	Requested action not taken; filename not allowed.

15.16.2 FTP Numeric Order List of Reply Codes

110	Restart marker reply. In this case, the text is exact and not left to the particular implementation; it must read `MARK yyyy = mmmm`, where `yyyy` is the user-process data-stream marker and `mmmm` represents the server's equivalent marker (note the spaces between markers and " = ").
120	Service ready in *nnn* minutes.
125	Data connection already open; transfer starting.
150	File status okay; about to open data connection.
200	Command okay.
202	Command not implemented, superfluous at this site.
211	System status, or system help reply.
212	Directory status.
213	File status.
214	Help message on how to use the server or the meaning of a particular nonstandard command. This reply is useful only to the human user.
215	`NAME` system type, where `NAME` is an official system name from the list in the IANA document.
220	Service ready for new user.
221	Service closing control connection; logged out if appropriate.
225	Data connection open; no transfer in progress.
226	Closing data connection; requested file action successful (e.g., file transfer or file abort).
227	Entering passive mode (`h1,h2,h3,h4,p1,p2`).
230	User logged in, proceed.
250	Requested file action okay, completed.
257	`PATHNAME` created.
331	User name okay, need password.
332	Need account for login.
350	Requested file action pending further information.
421	Service not available, closing control connection. This may be a reply to any command if the service knows that it must shut down.
425	Can't open data connection.
426	Connection closed; transfer aborted.

450	Requested file action not taken; file unavailable (e.g., file busy).
451	Requested action aborted; local error in processing.
452	Requested action not taken; insufficient storage space in system.
500	Syntax error, command unrecognized. This may include errors such as command line too long.
501	Syntax error in parameters or arguments.
502	Command not implemented.
503	Bad sequence of commands.
504	Command not implemented for that parameter.
530	Not logged in.
532	Need account for storing files.
550	Requested action not taken; file unavailable (e.g., file not found, no access).
551	Requested action aborted: page type unknown.
552	Requested file action aborted; exceeded storage allocation (for current directory or dataset).
553	Requested action not taken; filename not allowed.

15.17 Declarative Specifications

Minimum Implementation

To make FTP workable without needless error messages, the following minimum implementation is required for all servers: `TYPE`—ASCII nonprint; `MODE`—stream; `STRUCTURE`—file, record; `COMMANDS`—`USER`, `QUIT`, `PORT`, `TYPE`, `MODE`, `STRU`, for the default values `RETR`, `STOR`, `NOOP`. The default values for transfer parameters are `TYPE`—ASCII nonprint, `MODE`—stream, `STRU`—file. All hosts must accept these as the standard defaults.

Connections

The server protocol interpreter shall “listen” on port L. The user or user protocol interpreter shall initiate the full-duplex control connection. Server- and user-processes should follow the conventions of the TELNET protocol as specified in the *ARPA-Internet Protocol Handbook.* Servers are under no obligation to provide for editing of command lines and may require that it be done in the user host. The control connection shall be closed by the server at the user’s request after all transfers and replies are completed.

The user-DTP must “listen” on the specified data port; this may be the default user port (U) or a port specified in the PORT command. The server shall initiate the data connection from the server’s own default data port (L-1) using the specified user data port. The direction of the transfer and the port used will be determined by the FTP service command.

Remember that all FTP implementation must support data transfer using the default port, and that only the user-PI may initiate the use of nondefault ports.

When data is to be transferred between two servers, A and B, the user-PI (C) sets up control connections with both server-PIs. One of the servers, say, A, is then sent a PASV command to “listen” on the server’s own data port rather than initiate a connection on receipt of a transfer service command. After receiving an acknowledgment to the PASV command, which includes the identity of the host and port being listened on, the user-PI then sends A’s port, a, to B in a PORT command; a reply is returned. The user-PI may then send the corresponding service commands to A and B. Server B initiates the connection and the transfer proceeds. The command-reply sequence is listed below, where the messages are vertically synchronous but horizontally asynchronous:

User-PI: server A	**User-PI: server B**
C → A: connect	C → B: connect
C → A: PASV	
A → C: 227 entering passive mode `A1,A2,A3,A4,a1,a2`	
	C → B: PORT `A1,A2,A3,A4,a1,a2`
	B → C: 200 okay
C → A: STOR	C → B: RETR
B → A: connect to `HOST-A, PORT-a`	

If the data connection is to be closed following a data transfer where closing the connection is not required to indicate the end of file, the server must do so immediately. Waiting until after a new transfer command is not permitted because the user-process will have already tested the data connection to see if it needs to do a “listen” (remember that the user must “listen” on a closed data port *before* sending the transfer request). To prevent a race condition here, the server sends a reply (226) after closing the data connection [or if the connection is left open, a “file transfer completed” reply (250), and the user-PI should wait for one of these replies before issuing a new transfer command].

If either the user or server sees that the connection is being closed by the other side, it should promptly read any remaining data queued on the connection and issue the close on its own side.

15.18 Other Commands and Replies

The commands are TELNET character strings transmitted over the control connections. The commands begin with a command code followed by an argument field. The command codes are four or fewer alphabetic characters. Upper- and lower-case alphabetic characters are to be treated identically. Thus, any of the following may represent the retrieve command:

RETR Retr retr ReTr rETr

This also applies to any symbols representing parameter values, such as A or a for ASCII type. The command codes and the argument fields are separated by one or more spaces.

The argument field consists of a variable-length character string ending with the character sequence <CRLF> (carriage return, linefeed) for NVT-ASCII representation; for other negotiated languages a different end-of-line character might be used. It should be noted that the server is to take no action until the end-of-line code is received.

The syntax is specified below in NVT-ASCII. All characters in the argument field are ASCII characters, including any ASCII-represented decimal integers. Square brackets denote an optional argument field. If the option is not taken, the appropriate default is implied.

FTP Commands

The following are the FTP commands:

```
USER <SP> <username><CRLF>
PASS <SP> <password> <CRLF>
ACCT <SP> <account-information> <CRLF>
CWD <SP> <pathname> <CRLF>
CDUP <CRLF>
SMNT <SP> <pathname> <CRLF>
QUIT <CRLF>
REIN <CRLF>
PORT <SP> <host-port> <CRLF>
PASV <CRLF>
TYPE <SP> <type-code> <CRLF>
STRU <SP> <structure-code> <CRLF>
MODE <SP> <mode-code> <CRLF>
RETR <SP> <pathname> <CRLF>
STOR <SP> <pathname> <CRLF>
STOU <CRLF>
APPE <SP> <pathname> <CRLF>
ALLO <SP> <decimal-integer>
  [<SP> R <SP> <decimal-integer>] <CRLF>
REST <SP> <marker> <CRLF>
RNFR <SP> <pathname> <CRLF>
RNTO <SP> <pathname> <CRLF>
ABOR <CRLF>
DELE <SP> <pathname> <CRLF>
RMD <SP> <pathname> <CRLF>
MKD <SP> <pathname> <CRLF>
PWD <CRLF>
LIST [<SP> <pathname>] <CRLF>
NLST [<SP> <pathname>] <CRLF>
SITE <SP> <string> <CRLF>
SYST <CRLF>
STAT [<SP> <pathname>] <CRLF>
HELP [<SP> <string>] <CRLF>
NOOP <CRLF>
```

FTP Command Arguments

The syntax of the argument fields listed above [using BNF (Backus-Naur form) notation where applicable] is

```
<username>::= <string>
<password> :: = <string>
<account-information> :: = <string>
<string> :: = <char> < | <char><string>
<char> :: = any of the 128 ASCII characters except <CR> and
    <LF>
<marker> :: = <pr-string>
<pr-string> :: = <pr-char> < | > <pr-char><pr-string>
<pr-char> :: = printable characters, any
  ASCII code 33 through 126
<byte-size> :: = <number>
<host-port> :: = <host-number>,<port-number>
<host-number> :: = <number>,<number>,<number>,<number>
<port-number> :: = <number>,<number>
<number> :: = any decimal integer 1 through 255
<form-code> :: = N < | > T < | > C
<type-code> -:: = A [<sp> <form-code>]
            | E [<sp> <form-code>]
            | I
            | L <sp> <byte-size>
<structure-code> :: = F | R | P
<mode-code> :: = S | B | C
<pathname> :: = <string>
<decimal-integer> :: = any decimal integer
```

Sequencing of Commands and Replies

Communication between the user and server is intended to be an alternating dialog. As such, the user issues an FTP command and the server responds with a prompt primary reply. The user should wait for this initial primary success or failure response before sending further commands.

Certain commands require a second reply for which the user should also wait. These replies may, for example, report on the progress or completion of file transfer or the closing of the data connection. They are secondary replies to file-transfer commands.

One important group of informational replies is the connection greetings. Under normal circumstances, a server will send a 220 reply, “awaiting input,” when the connection is completed. The user should wait for this greeting message before sending any commands. If the server is unable to accept input right away, a 120 “expected delay” reply should be sent immediately and a 220 reply when ready. The user will then know not to hang up if there is a delay.

Spontaneous Replies

Sometimes “the system” spontaneously has a message to be sent to a user (usually all users), for example, “system going down in 15 min.” There is no provision in FTP for such spontaneous information to be sent from the server to the user. It is recommended that such information be queued in the server-PI and delivered to the user-PI in the next reply (possibly making it a multiline reply).

Section 15.21 on connection establishment contains a list of alternative success and failure replies for each command. These must be strictly adhered to; a server may substitute text in the replies, but the meaning and action implied by the code numbers and by the specific command-reply sequence cannot be altered.

Command-Reply Sequences

The command-reply sequence is presented. Each command is listed with its possible replies; command groups are listed together. Preliminary replies are listed first (with their succeeding replies indented and under them), then positive and negative completion, and finally intermediary replies with the remaining commands from the sequence following. This listing forms the basis for the state diagrams, which will be presented separately.

Connection establishment is as follows:

120	500, 501, 503, 421
220	CWD
220	250
421	500, 501, 502, 421, 530, 550
Login	CDUP
USER	200
230	500, 501, 502, 421, 530, 550
530	SMNT
500, 501, 421	202, 250
331, 332	500, 501, 502, 421, 530, 550
PASS	Logout
230	REIN
202	120
530	220
500, 501, 503, 421	220
332	421
ACCT	500, 502
230	QUIT
202	221
530	500

Transfer parameters are

PORT	LIST
200	125, 150
500, 500, 421, 530	226, 250
PASV	425, 426, 451

227

500, 501, 502, 421, 530

MODE

200

500, 501, 504, 421, 530

TYPE

200

500, 501, 504, 421, 530

STRU

200

500, 501, 504, 421, 530

File action commands

ALLO

200

202

500, 501, 504, 421, 530

REST

500, 501, 502, 421, 530

350

STOR

125, 150

(110)

226, 250

425, 426, 451, 551, 552

532, 450, 452, 553

500, 501, 421, 530

STOU

125, 150

(110)

226, 250

425, 426, 451, 551, 552

532, 450, 452, 553

450

500, 501, 502, 421, 530

NLST

125, 150

226, 250

425, 426, 451

450

500, 501, 502, 421, 530

APPE

125, 150

(110)

226, 250

425, 426, 451, 551, 552

532, 450, 550, 452, 553

500, 501, 502, 421, 530

RNFR

450, 550

500, 501, 502, 421, 530

350

RNTO

250

532, 553

500, 501, 502, 503, 421, 530

DELE

250

450, 550

500, 501, 502, 421, 530

RMD

250

500, 501. 502, 421, 530, 550

MKD

257

500, 501, 421, 530	500, 501, 502, 421, 530, 550
RETR	PWD
125, 150	257
(110)	500, 501, 502, 421, 530
226, 250	ABOR
425, 426, 451	225, 226
450, 550	500, 501, 502, 421
500, 501, 421, 530	

Informational commands are

SYST	450
215	500, 501, 502, 421, 530
500, 501, 502, 421	HELP
STAT	211, 214
211, 212, 213	500, 501, 502, 421

Miscellaneous commands are

SITE	NOOP
200	200
202	500, 421
500, 501, 530	

15.19 State Diagrams

Here we present state diagrams for a very simple FTP implementation. Only the first digit of the reply codes is used. There is one state diagram for each group of FTP commands or command sequences.

The command groupings were determined by constructing a model for each command and then collecting together the commands with structurally identical models.

For each command or command sequence there are three possible outcomes: success (S), failure (F), and error (E). In the state diagrams below we use the symbol B for “begin” and the symbol W for “wait for reply.” We first present the diagram that represents the largest group of FTP commands (see Fig. 15-8).

Figure 15-8 models the commands ABOR, ALLO, DELE, CWD, CDUP, SMNT, HELP, MODE, NOOP, PASV, QUIT, SITE, PORT, SYST, STAT, RMD, MKD, PWD, STRU, and TYPE. The other large group of commands is represented by a very similar diagram (Fig. 15-9).

Figure 15-9 models the commands APPE, LIST, NLST, REIN, RETR, STOR, and STOU. This second model could also be used to represent the first group of commands; the only difference is that in the first group the 100 series replies are unexpected and therefore treated as errors, while the second group expects (and some may require) 100 series replies. Remember that at most, one 100 series reply is allowed per command.

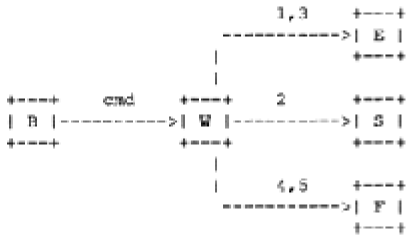


Figure 15-8
FTP commands state diagram.

The remaining diagrams model command sequences; perhaps the simplest of these is the rename sequence (Fig. 15-10).

Figure 15-11 is a simple model of the restart command.

We note that the models shown in Figs. 15.9 to 15.11 are similar. The restart differs from the rename only in the treatment of 100 series replies at the second stage, while the second group expects (and some may require) 100 series replies. Remember that at most, one 100 series reply is allowed per command.

The most complicated diagram is that for the login sequence (see Fig. 15-12).

Finally, we present a generalized diagram that could be used to model the command-reply interchange (see Fig. 15-13).

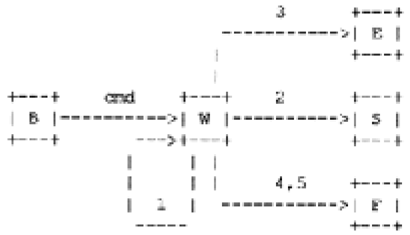


Figure 15-9
FTP commands state diagram.

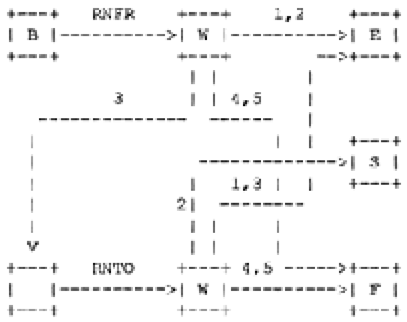
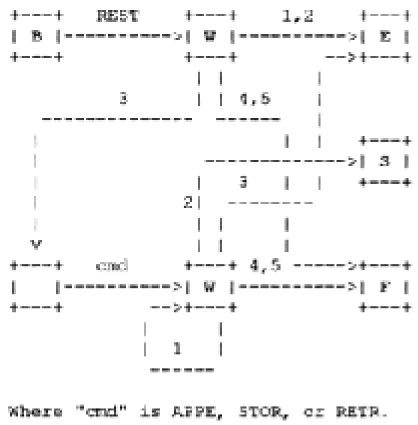


Figure 15-10
Renaming sequence state diagram.



Where "cmd" is APPE, STOR, or RETR.

Figure 15-11
Simple model of the restart command, where `cmd` is APPE, STOR, or RETR.

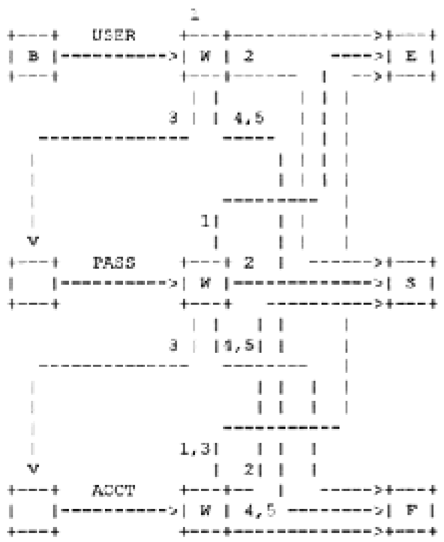


Figure 15-12
The login sequence.

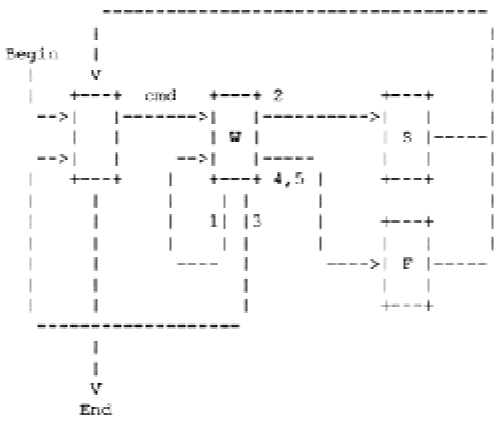


Figure 15-13
Generalized diagram that could be used to model the command-reply interchange.

15.20 Typical FTP Scenario

The user at host U wants to transfer files to and from host S. In general, the user will communicate to the server via a mediating user-FTP process. The following may be a typical scenario. The user-FTP prompts are shown in parentheses, ‘ ——>’ represents commands from host U to host S, and ‘<——’ represents replies from host S to host U. Local commands by user action involved are as follows:

```
ftp (host) multics<CR>      Connect to host S, port L,
      establishing control connections.
      <—— 220 Service ready <CRLF>.
username Doe <CR>      USER Doe<CRLF>——.
      <—— 331 User name ok,
      need password<CRLF>.
password mumble <CR>      PASS mumble<CRLF>——.
      <—— 230 User logged in<CRLF>.
      retrieve (local type) ASCII<CR>
(local pathname) test 1 <CR>      User-FTP opens local file in ASCII.
(for. pathname) test.pl1<CR> RETR test.pl1<CRLF> ——.
      <—— 150 File status okay;
      about to open data
      connection<CRLF>.
      Server makes data connection
      to port U.
      <—— 226 Closing data connection,
      file transfer successful<CRLF>.
type Image<CR>      TYPE I<CRLF> ——.
      <—— 200 Command OK<CRLF>
store (local type) image<CR>
(local pathname) file dump<CR>      User-FTP opens local file in Image.
(for.pathname) >udd>cn>fd<CR>      STOR >udd>cn>fd<CRLF> ——>
      <—— 550 Access denied<CRLF>
terminate      QUIT <CRLF> ——>
      Server closes all
      connections.
```

15.21 Connection Establishment, Page Structure, and Directory Commands

Connection Establishment

The FTP control connection is established via TCP between the user process port U and the server process port L. This protocol is assigned the service port 21 (25 octal): L = 21.

FTP and Page Structure

The need for FTP to support page structure derives principally from the need to support efficient transmission of files between TOPS-20 systems, particularly the files used by NLS.

The file system of TOPS-20 is based on the concept of pages. The operating system is most efficient at manipulating files as pages. The operating system provides an interface to the file system so that many applications view files as sequential streams of characters. However, a few applications use the underlying page structures directly, and some of these create "holey" files.

A TOPS-20 disk file consists of four things: a pathname, a page table, a (possibly empty) set of pages, and a set of attributes. The *pathname* is specified in the RETR or STOR command. It includes the directory name, filename, filename extension, and generation number. The *page table* contains up to 2**18 entries. Each entry may be empty, or may point to a page. If it is not empty, there are also some page-specific access bits; not all pages of a file need have the same access protection. A page is a contiguous set of 512 words of 36 bits each. The *attributes* of the file, in the file descriptor block (FDB), contain such items as creation time, write time, read time, writer's byte size, end-of-file pointer, count of reads and writes, and backup system tape numbers.

There is *no* requirement that entries in the page table be contiguous. There may be empty page table slots between occupied ones. Also, the EOF pointer is simply a number. There is no requirement that it in fact point at the "last" datum in the file. Ordinary sequential I/O calls in TOPS-20 will cause the EOF pointer to be left after the last datum written, but other operations may cause it not to be so, if a particular programming system so requires. In fact, in both of these special cases, "holey" files and EOF pointers *not* at the end of the file occur with NLS data files.

The TOPS-20-paged files can be sent with the FTP transfer parameters: TYPE L 36, STRU P, and MODE S (in fact, any mode could be used). Each page of information has a header. Each header field, which is a logical byte, is a TOPS-20 word, since the TYPE is L 36. The header fields are

Word 0: header length. The header length is 5.

Word 1: page index. If the data are in a disk file page, this is the number of that page in the file's page map. Empty pages (holes) in the file are simply not sent. Note that a hole is *not* the same as a page of zeros.

Word 2: data length. The number of data words in this page, following the header. Thus, the total length of the transmission unit is the header length plus the data length.

Word 3: page type. A code for what type of chunk this is. A data page is type 3; the FDB page is type 2.

Word 4: page access control. The access bits associated with the page in the file's page map. (This full word quantity is put into `AC2` of an `SPACS` by the program reading from net to disk.)

After the header are data-length data words. Data length is cur rently either 512 for a data page or 31 for an FDB. Trailing zeros in a disk file page may be discarded, making data length less than 512 in that case.

FTP Directory Commands

Since UNIX has a tree-like directory structure in which directories are as easy to manipulate as ordinary files, it is useful to expand the FTP servers on these machines to include commands which deal with the creation of directories. Since there are other hosts on the ARPA-Internet which have tree-like directories (including TOPS-20 and Multics), these commands are as general as possible.

Four directory commands for consideration here include (1) MKD pathname—make a directory with the name *pathname,* (2) RMD pathname—remove the directory with the name *pathname,* (3) PWD—print the current working directory name, and (4) CDUP—change to the parent of the current working directory.

The *pathname* argument should be created (removed) as a subdirectory of the current working directory, unless the pathname string contains sufficient information to specify otherwise to the server, for instance, *pathname* is an absolute pathname (in UNIX and Multics) or is something like <abso.lute.path> to TOPS-20.

Reply codes are described in the following paragraphs. The CDUP command is a special case of CWD, and is included to simplify the implementation of programs for transferring directory trees between operating systems having different syntaxes for naming the parent directory. The reply codes for CDUP be identical to the reply codes of CWD. The reply codes for RMD are identical to those for its file analogue, DELE. The reply codes for MKD, however, are a bit more complicated. A freshly created directory will probably be the object of a future CWD command. Unfortunately, the argument to MKD may not always be a suitable argument for CWD. This is the case, for example, when a TOPS-20 subdirectory is created by giving only the subdirectory name. Thus, with a TOPS-20 server FTP, the command sequence

```
MKD MYDIR
CWD MYDIR
```

will fail. The new directory may be referred to only by its “absolute” name; for instance, if the MKD command above were issued while connected to the directory <DFRANKLIN>, the new subdirectory could be referred to only by the name <DFRANKLIN.MYDIR>.

Even on UNIX and Multics, however, the argument given to MKD may not be suitable. If it is a “relative” pathname (i.e., a pathname which is interpreted relative to the current directory), the user would need to be in the same current directory in order to reach the subdirectory. Depending on the application, this may be inconvenient. It is not very robust in any case.

To solve these problems, on successful completion of an MKD command, the server should return a line of the form

```
257<space>“<directory-name>”<space><commentary>
```

Thus, the server will tell the user what string to use when referring to the created directory. The directory name can contain any character; embedded double-quotes should be escaped by double-quotes (the “quote-doubling” convention).

For example, a user connects to the directory `/usr/dm,` and creates a subdirectory, named pathname:

```
CWD /usr/dm
200 directory changed to /usr/dm
MKD pathname
257 “/usr/dm/pathname”directory created
```

An example with an embedded double quote:

```
MKD “foo”bar
257 “/usr/dm/foo”“bar”directory created
CWD “/usr/dm/foo”bar
200 directory changed to /usr/dm/foo”bar
```

The prior existence of a subdirectory with the same name is an error, and the server must return an “access denied”error reply in that case.

```
CWD /usr/dm
200 directory changed to /usr/dm
MKD pathname
521-"/usr/dm/pathname" directory already exists;
521 taking no action
```

The failure replies for MKD are analogous to its file-creating cousin, STOR. Also, an "access denied" return is given if a filename with the same name as the subdirectory will conflict with the creation of the subdirectory (this is a problem on UNIX, but shouldn't be one on TOPS-20).

Essentially because the PWD command returns the same type of information as the successful MKD command, the successful PWD command uses the 257 reply code as well.

There are some subtleties here. Because these commands will be most useful in transferring subtrees from one machine to another, carefully observe that the argument to MKD is to be interpreted as a subdirectory of the current working directory, unless it contains enough information for the destination host to tell otherwise. A hypothetical example of its use in the TOPS-20 world would be

```
CWD <some.where>
200 Working directory changed
MKD overrainbow
257 ",some.where.overrainbow>" directory created
CWD overrainbow
431 No such directory
CWD <some.where.overrainbow>
200 Working directory changed
```

```
CWD <some.where>
200 Working directory changed to some.where
MKD <unambiguous>
257 "<unambiguous>" directory created
CWD <unambiguous>
```

The first example results in a subdirectory of the connected directory. In contrast, the argument in the second example contains enough information for TOPS-20 to discern that the <unambiguous> directory is a top-level directory. Note also that in the first example the user "violated" the protocol by attempting to access the freshly created directory with a name other than the one returned by TOPS-20. Problems could have resulted in this case had there been an <overrainbow> directory; this is an ambiguity inherent in some TOPS-20 implementations. Similar considerations apply to the RMD command. The point is that except where doing so would violate a host's conventions for denoting relative versus absolute pathnames, the host should treat the operands of the MKD and RMD commands as subdirectories. The 257 reply to the MKD command must always contain the absolute pathname of the created directory.

15.22 RFCs on FTP

Additional information on FTP can be obtained in any of the following RFCs.

Bhushan, Abhay, *A File Transfer Protocol,* RFC 114 (NIC 5823), MIT-Project MAC, April 16, 1971.

Bhushan, Abhay, et al. *The File Transfer Protocol,* RFC 172 (NIC 6794), MIT-Project MAC, June 23, 1971.

Bhushan, Abhay, et al. *The File Transfer Protocol,* RFC 265 (NIC 7813), MIT-Project MAC, Nov. 17, 1971.

Bhushan, Abhay, *The Use of "Set Data Type" Transaction in File Transfer Protocol,* RFC 294 (NIC 8304), MIT-Project MAC, Jan. 25, 1972.

Bhushan, Abhay, *The File Transfer Protocol,* RFC 354 (NIC 10596), MIT-Project MAC, July 8, 1972.

Bhushan, Abhay, *Comments on the File Transfer Protocol* (RFC 354), RFC 385 (NIC 11357), MIT-Project MAC, Aug. 18, 1972.

Bhushan, Abhay, *File Transfer Protocol (FTP) Status and Further Comments,* RFC 414 (NIC 12406), MIT-Project MAC, Nov. 20, 1972.

Bhushan, Abhay, *FTP Comments and Response to RFC 430,* RFC 463 (NIC 14573), MIT-DMCG, Feb. 21, 1973.

Bhushan, Abhay, *FTP and Network Mail System,* RFC 475 (NIC 14919), MIT-DMCG, March 6, 1973.

Braden, Bob, *Comments on DTP and FTP Proposals,* RFC 238 (NIC 7663), UCLA/CCN, Sept. 29, 1971.

Braden, Bob, *Comments on File Transfer Protocol,* RFC 430 (NIC 13299), UCLA/CCN, Feb. 7, 1973.

Braden, Bob, *Print Files in FTP,* RFC 448 (NIC 13299), UCLA/CCN, password Feb. 27, 1973.

Braden, Bob, *FTP Data Compression,* RFC 468 (NIC 14742), UCLA/CCN, March 8, 1973.

Braden, Bob, *TENEX FTP Problem,* RFC 571 (NIC 18974), UCLA/CCN, Nov. 15, 1973.

Bressler, Bob, and Bob Thomas, *Mail Retrieval via FTP,* RFC 458 (NIC 14378), BBN-NET and BBN-TENEX, Feb. 20, 1973.

Bressler, Bob, and Bob Thomas, *FTP Server-Server Interaction—II,* RFC 478 (NIC 14947), BBN-NET and BBN-TENEX, March 26, 1973.

Day, John, *Memo to FTP Group (Proposal for File Access Protocol),* RFC 520 (NIC 16819), Illinois, June 25, 1973.

Harrenstien, Ken, *FTP Extension: XSEN,* RFC 737 (NIC 42217), SRI-KL, Oct. 31, 1977.

Harrenstien, Ken, *FTP Extension: XRSQ/XRCP,* RFC 743 (NIC 42758), SRI-KL, Dec. 30, 1977.

Harslem, Eric, and John Heafner, *Comments on RFC 114 (a File Transfer Protocol),* RFC 141 (NIC 6726), RAND, April 29, 1971.

Harvey, Brian, *Leaving Well Enough Alone,* RFC 686 (NIC 32481), SU-AI, May 10, 1975.

Harvey, Brian, *One More Try on the FTP,* RFC 691 (NIC 32700), SU-AI, May 28, 1975.

Hicks, Greg, *User FTP Documentation,* RFC 412 (NIC 12404), Utah, Nov. 27, 1972.

Krilanovich, Mark, and George Gregg, *Comments on the File Transfer Protocol,* RFC 607 (NIC 21255), UCSB, Jan. 7, 1974.

Krilanovich, Mark, George Gregg, Wayne Hathaway, and Jim White, *Comments on the File Transfer Protocol*, RFC 624 (NIC 22054), UCSB, Ames Research Center, SRI-ARC, Feb. 28, 1974.

Lebling, P. David, *Survey of FTP Mail and MLFL,* RFC 751, MIT, Dec. 10, 1978.

Lieb, J., *CWD Command of FTP,* RFC 697 (NIC 32963), July 14, 1975.

Mankins, David, Dan Franklin, and Buzz Owen, *Directory Oriented FTP Commands,* RFC 776, BBN, Dec. 1980.

McKenzie, Alex, *A Suggested Addition to File Transfer Protocol,* RFC 281 (NIC 8163), BBN, Dec. 8, 1971.

McKenzie, Alex, *File Transfer Protocol,* RFC 454 (NIC 14333), BBN, Feb. 16, 1973.

McKenzie, Alex, and Jon Postel, *Telnet and FTP Implementation—Schedule Change,* RFC 593 (NIC 20615), BBN and MITRE, Nov. 29, 1973.

Merryman, Robert, *The UCSD-CC Server-FTP Facility,* RFC 532 (NIC 17451), UCSD-CC, June 22, 1973.

Neigus, Nancy, *File Transfer Protocol,* RFC 542 (NIC 17759), BBN, July 12, 1973.

Padlipsky, Mike, *An FTP Command-Naming Problem,* RFC 506 (NIC 16157), MIT-Multics, June 26, 1973.

Padlipsky, Michael, *FTP Unique-Named Store Command,* RFC 949, MITRE, July 1985.

Pogran, Ken, and Nancy Neigus, *Response to RFC 607—Comments on the File Transfer Protocol,* RFC 614 (NIC 21530), BBN, Jan. 28, 1974.

Postel, Jon, *Revised FTP Reply Codes,* RFC 640 (NIC 30843), UCLA/NMC, June 5, 1974.

Postel, Jon, *File Transfer Protocol Specification,* RFC 765, ISI, June 1980.

Sussman, Julie, *FTP Error Code Usage for More Reliable Mail Service,* RFC 630 (NIC 30237), BBN, April 10, 1974.

Thomas, Bob, and Bob Clements, *FTP Server-Server Interaction,* RFC 438 (NIC 13770), BBN, Jan. 15, 1973.

White, Jim, *Use of FTP by the NIC Journal,* RFC 479 (NIC 14948), SRI-ARC, March 8, 1973.

White, Jim, *Host-Dependent FTP Parameters,* RFC 480 (NIC 14949), SRI-ARC, March 8, 1973.

16
Electrical Considerations and Power Conditioning

Network components require electrical power. Power is fundamental. I know of few, if any, devices that operate with networks that do not require electricity to operate. During my years working with network professionals, I have realized many of them do not understand some fundamental principles of electricity. I decided to cover basic topics in this chapter. First, I will present some electrical principles and facts, and then help the reader understand how they are meaningful to everyday work with networks. Second, I decided to cover power conditioning in this chapter, prior to what I call the core of the book, because if there is a single focal point in all networks it surely is power. Let me put this in perspective.

Look around you. If you are in an office environment, all the better. Now consider that I have just discovered the circuit breaker box supplying electricity to your office. Now suppose I turned all of the switches off. Got any idea what is going on now in your office? Not much. You can’t print, compute, or use anything that relies on electrical power. Power is fundamental. Yes, I am aware of solar-powered calculators and other such devices as well as battery-driven notebook computers and cellular phones that run on batteries, but without doubt the core of your business relies on electricity for daily operations.

Electricity is taken for granted for the most part, and this is unfortunate. Electricity is an artificially created force in the way it is harnessed and used today in our modern world. However, electricity is not a source available worldwide. Granted in most places where business is conducted in some form or fashion, electricity is available. Electricity is artificially produced in the way it is used in the business community, but technically it does exist in nature as a fundamental characteristic of the universe.

I am going to a little extreme to make a point. Those who understand the critical role of power in networks, computers, and modern offices realize all operations revolve around *power;* likewise the converse is true. Without power, or proper power conditioning and backup, operations come to a halt in a hurry. This chapter presents the basic electrical principles for those unfamiliar with electricity, sample electrical readings and how to interpret them and—most important—information about power conditioning and its aspects.

16.1 Terminology

If you are new to the study of electricity, you could be concerned or a little scared. My first comment is don’t loose your respect for electricity because if you don’t give it respect and become careless, it will take respect. This is one chapter of information not to be played with. Yes, the rest of the book is serious, too, but this chapter is very serious. I recommend caution at every turn when you work with any electrical device, and particularly with topics presented here.

To understand and work with electricity, one needs to understand some terminology and concepts. Once some terms and concepts are understood, then application of knowledge can be realized. Consider the terminology presented here to begin with.

alternating current (ac) An electric current that is continually varying in value and reversing its direction of flow at regular intervals, usually in a sinusoidal manner. Each repetition from zero to a maximum in one direction and then to a maximum in the other direction and back to zero, is called a *cycle.*

alternating-current frequency The speed at which an ac voltage or current waveform repeats itself. Frequency is measured in hertz.

amperage The amount of current in amperes.

ampere (A) A practical unit of electrical current. One ampere of current is 6.24×10^{-18} electrons passing one point in 1 s and equals 1 C/s (1 coulomb per second). This is the result of 1 V across a resistance of 1 ohm.

apparent power The product of current and voltage, expressed as kilovoltamperes (kVA). It is the real power (kW) divided by the power factor (PF).

atom The smallest particle into which an element can be divided and still retain its chemical properties.

balanced current A current flowing in the two conductors of a balanced line so that, at every point along the line, they are equal in magnitude and opposite in direction.

balanced line A transmission line consisting of two conductors which are capable of being operated so that the voltages of the two conductors on any transverse plane are equal in magnitude and opposite in polarity with respect to the ground. The currents in the two conductors are then equal in magnitude and opposite in direction.

balanced voltages Voltages that are equal in magnitude and opposite in polarity with respect to ground. They are also called *push-pull voltages.*

brownout Normally a voltage reduction initiated by the utility to counter excessive demand on its electric power generation and distribution system.

conductor A material that serves as a conduit for electric current easily because it offers little electrical resistance. Some examples of conductors are tin, metals, salt water, aluminum, copper, gold, nickel, and platinum.

current In electricity, the flow of electrons, or the movement thereof, through a conductive material; it is the flow of electrons or holes measured in amperes, (A) or in fractions of an ampere such as milliamperes (mA), microamperes (μA), nanoamperes (nA), or picoamperes (pA). Current can be induced by the application of an electric field through a conductor or by changing the electric field through a conductor or across a capacitor (which is known as *displacement current).*

direct current (dc) An electric current that flows in one direction.

electric field The region around an electrically charged body in which other charged bodies are acted on by an attracting or repelling force.

electricity A fundamental quantity realized in nature that consists of electrons and protons either at rest or in motion. Electricity at rest (not in motion) has an electric field which possesses potential energy and can exert force. Electricity in motion (not at rest) has the characteristics of an electric and magnetic field that possess potential energy and can exert force.

electron An elementary atomic particle that carries the smallest negative electric charge. It is highly mobile and orbits the nucleus of an atom.

frequency The number of complete cycles per unit of time for a periodic quantity such as alternating current, sound waves, or radio waves. A frequency of 1 cycle per second is 1 hertz; represented by the letter *f.*

harmonic A sinusoidal component of a periodic wave, with a frequency that is an integral multiple of the fundamental frequency. The frequency of the second harmonic is twice that of the fundamental frequency or first harmonic. Also called *harmonic component* and *harmonic frequency*.

harmonic analysis Any method for identifying and evaluating the harmonics that make up a complex form of voltage, current, or some other varying quantity.

hertz (Hz) The SI (International System of Units) unit of frequency, equal to 1 cycle per second.

impedance The total opposition offered by a component or circuit to the flow of an alternating or varying current; represented by the letter *Z*. Impedance in expressed in ohms (Ω), and is the combination of resistance *R* and reactance.

inductance This is the property of a circuit or circuit element that opposes a change in current flow. Inductance causes current changes to lag behind voltage changes. Inductance is measured in henrys, millihenrys, and microhenrys. It is represented by the letter *L*.

kilo Prefix meaning one thousand.

kilovoltampere (kVA) One thousand voltamperes. This term is used for rating devices. The number is derived by multiplying the output in amperes by its rate operating voltage.

linear A term referring to a relationship in which one function is directly proportional to another function, providing a straight sloped line when plotted on a graph.

magnetic field Any space or region that a magnetic force exerts on moving electric charges. This field can be produced by a current-carrying coil or conductor, by a permanent magnet, or by the earth itself.

nonlinear Anything with this characteristic is not directly proportional to its complement.

nonlinear element An element in which an increase in applied voltage does not produce a proportional increase in current.

nonlinear load A load for which the relationship between voltage and current is not a linear function. An example of nonlinear loads could be fluorescent lighting and uninterruptible power supply (UPS) systems, loads that cause abnormal heating and voltage distortion.

ohm (Ω) The SI unit of resistance and impedance. The electric resistance between two points of a conductor when a constant of potential difference of 1 V applied to these points produces in the conductor a current of 1 A; meaning that the conductor is not the seat of any electromotive force.

period The time required for one complete cycle of a regularly repeated series of events.

photon A quantum of electromagnetic radiation equal to Planck's constant multiplied by the frequency in hertz (*hv)*. Electromagetic radiation can be considered as photons of light, x rays, gamma rays, or radio rays.

power The time rate of doing work or the speed in which work is done; represented by the letter *P* and measured in watts (W).

power factor The extent to which the voltage zero differs from the current zero. In ac circuits, inductance and capacitance cause the point at which the voltage wave passes through zero to differ from the point at which the current wave passes through zero. One complete cycle is 360°; consequently, the difference between zero points (e.g., voltage and current) can be expressed in angles. The power factor is the cosine of the angle between zero points, is expressed as a decimal fraction of 0.8 or 80 percent, and is the kW/kVA ratio, where kW equals kVA times the power factor.

proton An elementary particle that has a positive charge equal in magnitude to the negative charge of the electron. The atomic number of an element indicates the number of protons in the nucleus of each atom of that element. The mass of a proton at rest is 1.67×10^{-24} g or 1836.13 times that of an electron.

reactance The opposition to ac flow by pure inductance or capacitance in a circuit expressed in ohms. It is the component of impedance that is not due to resistance.

resistance The opposition that a device or material offers to the flow of direct current, measured in ohms, kilohms, or even megohms. In ac circuits, resistance is the real component of impedance.

root-mean-squared (rms) The square root of the average of the squares of a series of related values. The effective value of an alternating current, corresponding to the dc value that will produce the same heating effect. The rms value is computed as the square root of the average of the squares of the instantaneous amplitude for one complete cycle. For a sine wave the rms value is 0.707 times the peak value. Unless otherwise specified, alternating quantities are assumed to be the rms values. In simpler terms this is the effective value of voltage, current, and power when they are expressed as such.

sag An ac power-line undervoltage condition that lasts more than 1/60th of a second. A condition lasting longer than this is referred to as a *brownout.*

sinusoidal The variation in proportion to the sine of an angle or time function. An ordinary alternating current is considered sinusoidal.

spike A short-duration transient whose amplitude considerably exceeds the average amplitude of the associated pulse or signal.

surge A long-duration overvoltage or overcurrent condition.

time The measure of a duration of an event. The fundamental unit of time is the second.

transient A sudden, very brief spike of high voltage on a power line caused by lightning, electrostatic discharge, or power-line switching.

unbalanced line A transmission line that conducts voltage on its two conductors that are not equal with respect to the ground.

volt (V) The SI unit of voltage or potential difference. The difference in electrical potential between two points of a conducting wire carrying a constant current of 1 A when the power dissipated between these points is equal to 1 W.

voltampere (VA) The unit of apparent power in an ac circuit that contains reactance. Apparent power is equal to the voltage in volts multiplied by the current in amperes, without considering phase.

watt (W) The unit of electric power. It gives rise in one second to the ener gy of one joule (J). In an ac circuit, the true power is effective volts multiplied by effective amperes, then multiplied by the circuit power factor.

wave A propagated disturbance whose intensity at any point in the medium is a function of time and any given instant is a function of the position of the point. A wave can be electric, electromagnetic, acoustic, or mechanical.

16.2 Practical Information

If you read many books you probably realize some of them have little practical value. I am attempting to provide you with practical information in this book. I realize many of you do not have a background in electricity. However, the information provided in this chapter and the remaining chapters in this book will help you in your network planning.

16.2.1 Wire

There are many types of wire. The focus of this section is on electrical wire, specifically wire sizes and types. Understanding electrical wire is easy. The smaller the number, the thicker the wire, and hence the more voltage and current it can accommodate. For example, many types of common table lamps have a 14- to 18-gauge wire.

Wire is generally referred to by the *American Wire Gauge* (AWG) number. The smaller the number, the larger the wire. Consider:

26	16	6	1
24	14	4	0
22	12	3	00
20	10	2	000
18	8		

A 26-gauge wire is very thin; in fact it is smaller than the wire in most telephone cords that connect a handset to the base. Much of the 10BaseT wire is between 20- and 24-gauge. As mentioned previously, many ordinary table lamps are wired with 14- to 18-gauge wire. Some extension cords that can be purchased in 99¢-type stores are 14- to 18-gauge. These are considered *general-purpose* cords.

Most houses are wired with 12-gauge wire; some use 10-gauge. A 12-gauge wire for an extension cord would be considered adequate for many shop-type purposes. Many outdoor-use extension cords are 12-gauge.

Ten (10)-gauge wire is fairly heavy-duty. Going up in capability and down in numeric order are 8-, 6-, 4-, 2-, and 0-gauge. Wire sizes below 10 become increasingly large and can carry significant voltage and current. Remember, voltage and current are two different things. Let me illustrate this. Consider Fig. 16-1.

As I write this, I am listening to an ordinary AM/FM radio/cassette. It operates on 120-V ac or batteries. Figure 16-1 shows the voltage, fre quency, and amperage the radio uses. Technically, the voltage and frequency is what is available at the wall outlet. The amperage reading is what the radio is using to operate. Note that the amperage draw as indicated from channel 2 is 191.7 mA. This is not very much, and rightfully so—it is simply a desktop radio. You may be thinking this is odd. But wait! Now consider Fig. 16-2.

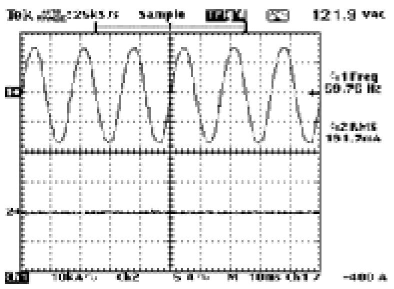


Figure 16-1
AM/FM radio voltage and current readings.

Figure 16-2 shows the readings I took from a microwave oven! Yes, I took the microwave from the kitchen counter and brought it into the lab. I connected it to the test outlet. No other device was connected except the Tektronix THS720P power analyzer. Notice the voltage, frequency, and amperage readings. The microwave is drawing 4.724 A as shown by channel 2, indicated on the lower portion of the figure. This illustration is the snapshot on initial power-on of the microwave. Now consider Fig. 16-3.

Figure 16-3 shows the reading of the microwave oven, and this time the current draw is 7.764 A. This reading was taken with the microwave in operation, not just after initial power-on. It sustains a significant current draw.

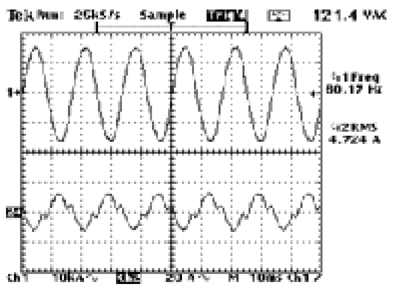


Figure 16-2
Microwave oven voltage and current
readings upon initial power-on.

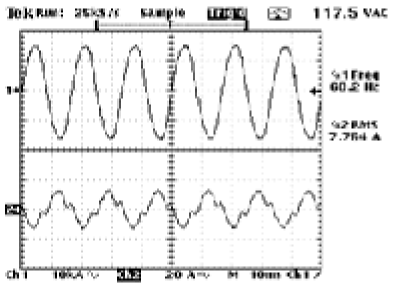


Figure 16-3
Microwave oven voltage and current
readings during operation.

These illustrations are important because they illustrate the difference in device current requirements. Remember, amperage is how much current a device uses. Yes, this is a loose definition, but stay with me. Current is the flow of electrons, noted by amperes (A) or in fractions of an ampere such as milliamperes (mA), microamperes (μA), nanoamperes (nA), or picoamperes (pA). Current can be induced by the application of an electric field through a conductor or by changing the electric field through a conductor or by changing the electric field across a capacitor (which is known as displacement current).

The measure of current (the amperage draw) is technically "how much" electricity a given device uses. For example, the AM/FM radio obviously uses less electricity than does the microwave oven as shown in Figs. 16.1 and 16.2. How can this be? Don't they both use 120 V ac? The measure of how much electricity is used is generally referred to as *current.* Technically the current draw is referred to as *amperage,* expressed in amperes (A). Now, the more current or the higher the amperage rating, the more electrons are flowing through a wire. The more electrons flowing through a wire, the hotter the wire gets and consequently, the more amperes required for a given device to function and the larger the wire size required to deliver that amount of amperage.

Most wire sizes have an amperage rating. *Ampacity* refers to the amount of amperage a given wire can accommodate. Generally, the lower the wire AWG number (e.g., 6 or 2), the more amperes the wire can accommodate. Different tables apply when measuring wire size and ampacity. Whether the wire (also called *conductor*) is single or multiple, operating in free air, and other variables factor into the capacity of the wire. See Table 16-1, which is presented here as a general reference.

One factor not presented in Table 16-1 is distance. The length of wire always, let me repeat, always, factors into the equation of current draw. The longer the wire run, the greater (larger) wire size needed to deliver *X* number of amperes. Think about this; it makes sense. Amperage is the movement of electrons. Anytime the length of something is extended, more resistance enters into the equation. Consider the following real-world example I encountered in 1986:

Table 16-1 Current-Carrying Capacity of Two or Three Conductors, Reflecting AWG Values in a Neoprene Jacket

AWG	Current-carrying capacity, A
22	6
20	8
18	14
16	18
14	25
12	30
10	40
8	55
6	75
4	95
3	110
2	130
1	150
0	170
00	195
000	225
0000	260

In December 1986 the religious group I was affiliated with at the time was planning to present a Christmas special. During preparation for the musicians, speakers, singers, and others, we moved a lot of electrical equipment to a new location. During setup of the public address (PA) equipment, the person directing the show realized we needed an extension cord to provide electricity to the PA equipment. This person proceeded to obtain extension cords and began connecting everything together. The second day of setup I realized this person had used three 100-ft, 14-gauge extension cords to supply electricity to the power amplifier for the PA equipment. I knew from previous education and experience that this was not going to work; the voltage drop was too great at that distance. I tried to inform the person in charge of the show of the problem; he wouldn't do anything after I told him (I think he thought I was too young to understand). I finally told him: "Go ahead, the power amps [amplifiers] will roll over and go belly-up." He looked at me askance and went on with the final touches. This person was also notorious for doing things on the fly (meaning no trial runs). So, the presentation began and about two minutes into the beginning, guess what happened? That's right, the power amps went belly-up. The voltage drop was too great.

Why the personal story? A couple of reasons. First, sensitive equipment such as computers and power amplifiers, are very sensitive to voltage drops, whether this drop is the result of a wire that is too long or some other factor. Second, use of an extension cord does not guarantee that electrical equipment, whatever it is, will operate correctly. Table 16-1, showing wire size and ampacity, indicated that a 20-gauge wire could accommodate 8 A. I doubt this would be the case if the 20-gauge wire were 400 ft long! See what I mean?

Another factor that also affects wire size and ampacity is the number of conductors. For example, some wire has only one conductor, some two, some three, some four, and so forth; this factors into the equation as well.

Still another consideration is temperature. Different types of wire shielding work differently according to varying temperature. Consider the information listed in Table 16-2.

The information presented in Table 16-2 is from one particular wire vendor; other vendors refer to wire by type as well and the values may differ.

Wire used in houses and commercial buildings is typically *Romex*, a stiff outside jacket used to wrap the wire. Inside, each conductor is insulated and the earth ground is typically wrapped in a form of heavy-gauge paper. In contrast, many ordinary extension cords are of SJ cable, which is not as stiff as Romex but is relatively flexible. Jackets of SO and SO-W cable are more pliable, like rubber. Most of them are oil- and water-resistant. This type of jacketing is used in many commercial-grade extension cords.

In summary, the important considerations to remember with wire are (1) wire size and length determine current capacity; (2) wire has a voltage rating; (3) the smaller the wire number, the larger the wire; (4) the number of conductors affects the ampacity; and (5) the longer the cable length, the greater the resistance and the voltage drop.

Table 16-2 Temperature as a Factor in Wire Shielding

Material type	**Operating temperature range, °C**[*]
Crosslinked polyethylene	–55 to 150
Hermetic	–60 to 125
Hypalon	–30 to 105
EPDM[†]	–60 to 150
Neoprene	–30 to 90
PVC	–35 to 105
Silicone, braided	–75 to 200
Silicone, braidless	–75 to 200
Teflon	–100 to 260

[*]To convert degrees Celsius into degrees Fahrenheit, multiply the Celsius reading by 9, divide by 5, and subtract 32.
[†]Ethylenepropylene terpolymer (EPT).

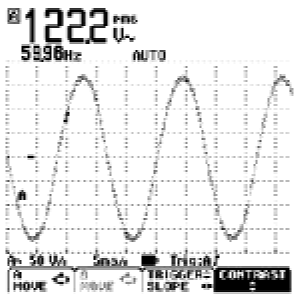


Figure 16-4
Voltage and cycly reading at breaker 1.

Site Wiring

Houses, commercial buildings, and the majority of building facilities in modern cities today are already wired for electricity. Many of them are similar. If you want to accuse me of overgeneralization, consider my explanation here as a highlighted overview of my house as an example (see Fig. 16-4).

Figure 16-4 shows a voltage reading of 122.2 V ac. I took this reading inside the load center where the wire connects to a circuit breaker supplying the outlets in my dining room. This is simply a typical reading; most likely the reading of this line taken at a dining-room outlet fed by it would yield the same results. Now consider Fig. 16-5.

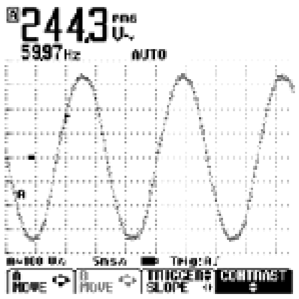


Figure 16-5
Input voltage readiing from the
local electric company.

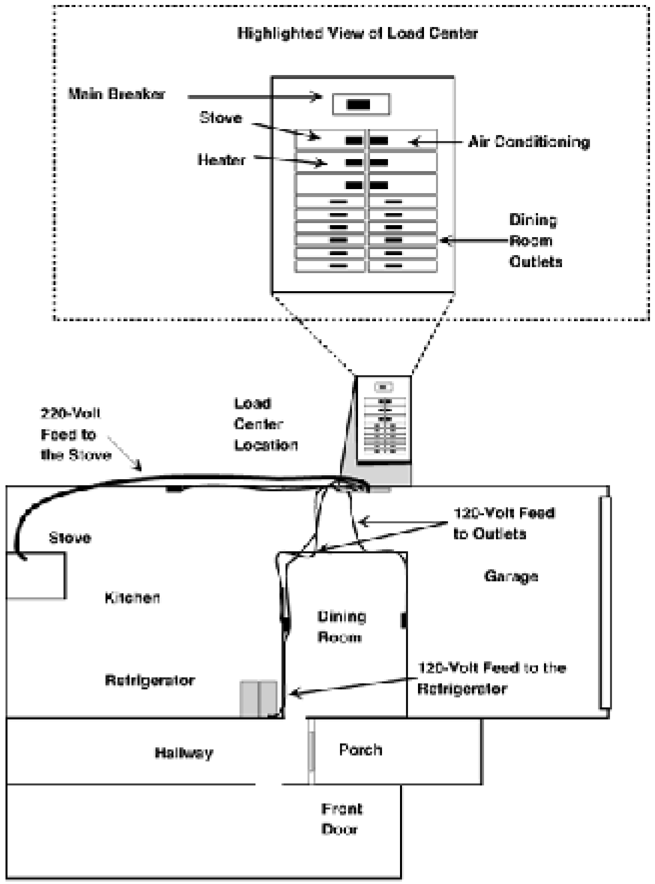


Figure 16-6
Overview of house wiring.

Figure 16-5 shows a 244.3-V, 59.97-cycle reading. I took this voltage measurement at the lines feeding my house load center. Now consider Fig. 16-6.

Figure 16-6 shows a highlighted view of the load center, the garage, kitchen, dining room, hallway, porch, front door, and other noted items. Note that the load center is in the garage. I have illustrated two ac outlets in the dining room, and this is a fairly accurate representation of their locations. I show a stove and refrigerator in the kitchen. Note that the stove is wired to the load center, and it appears to be 220 V ac. Note that the wire size (AWG) for the stove (4-gauge) is greater than that for the outlets (12-gauge). The refrigerator is wired directly to the load center as well and is 120 V ac, and 6-gauge wire is used. The ac outlets are 120 V ac. I have shown the wiring as it actually appears in the attic. A significant point here is that many of the outlets in the house are wired in parallel, meaning that multiple outlets run from a single breaker. This is important because parallel wiring is also frequently used for economic reasons in large commercial buildings, which generally have numerous dedicated electric outlets. Thus the primary reason for this parallel wiring is to save money. Why are you not surprised? Unless you custom-build, most builders will install the outlets and other components pretty much to the same standards as in the industry and to electrical code. Notice also in Fig. 16-6 that the stove has a dedicated outlet, along with the heater and air-conditioning unit.

Now consider Fig. 16-7.

Figure 16-7 shows a detailed view of the load center. Notice three leads coming into the load center from the electric meter: two 120-V ac feeds and a neutral feed. Also note the cable going to the dining-room outlet, and the hot lead connecting to a breaker but the neutral and ground connecting to the neutral busbar as shown. Beneath the breakers, two busbars carry the voltage. Also note a single main circuit breaker at the top of the load center. This breaker is the single *switch* used to turn on or off the feed to the entire house. Most load centers are similar to this, except those operating at much higher voltage. Now consider Fig. 16-8.

Figure 16-8 shows the voltage reading at the top of the busbar where the hot feeds come in from the electric meter. I obtained a reading of 243.7 V at this location. The cable size coming into the load center from the electric meter is 00.

So what is the purpose in this? To illustrate some fundamental concepts and facts that those who are not directly experienced in or familiar with the practical applications of electricity may not be aware of. Electricity is serious business, and should be handled only by those who are trained or educated in it. These illustrations (Figs. 16.4 to 16.8) also show that not all outlets or even lighting, have dedicated circuits. Most wiring in my house is of 12-gauge Romex wire, which is an unusually heavy load for a residential house, but considering the various loads that network equipment can generate, one might need a larger wire. Why does this matter? Well, consider what would happen if you plugged two microwave ovens into the same circuit, which was only 20 A. At 7.764 A per microwave, as Fig. 16-3 shows, you would be far beyond a 50 percent utilization of that breaker. Odds are that one more device on that breaker would overload it. Just remember this when you begin working with laser or thermal printers. They tend to draw significant current, both at startup and during the cycles of heating the drum.

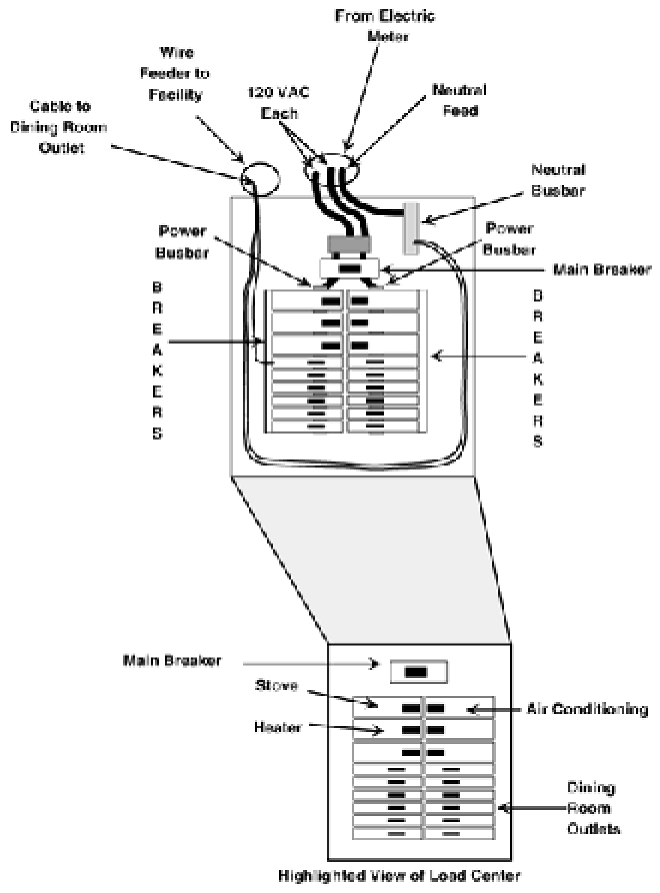


Figure 16-7
Detailed view inside load center.

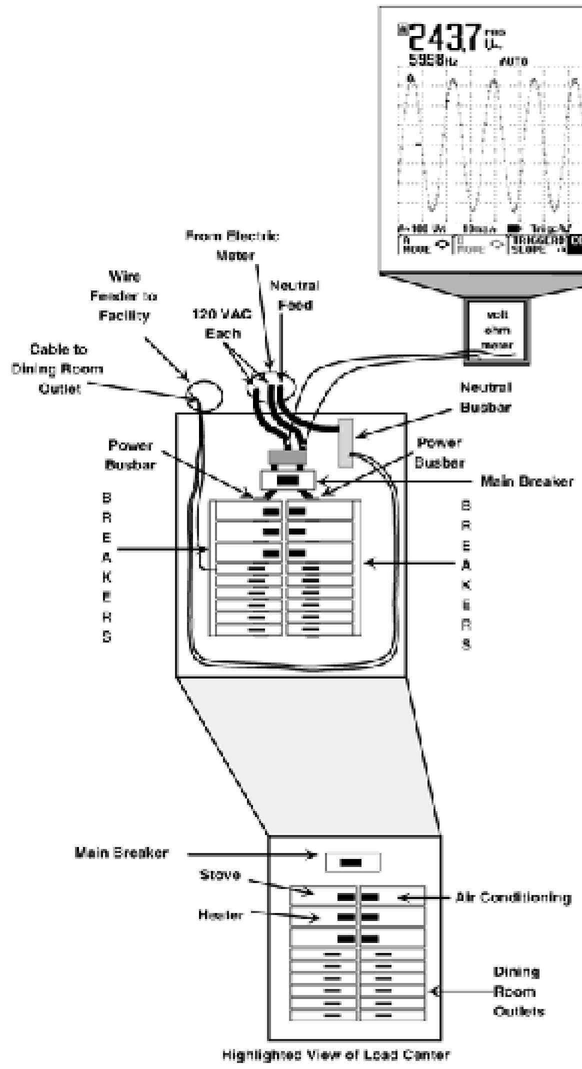


Figure 16-8
Voltage reading from electric meter.

I presented this discussion and the wiring illustrations for you to consider in light of your own work premises. If you think of them from the perspective I have shown here, you might have a healthier view of your own work environment.

16.2.2 Harmonics

Why harmonics? Well, harmonic evaluation reveals the quality of power. Depending on your site, this could be very significant, because the more sensitive the equipment, the more susceptible it is to harmonic distortion. Remember the definition of harmonics: a sinusoidal component of a periodic wave, with a frequency that is an integral multiple of the fundamental frequency. The frequency of the second harmonic is twice that of the fundamental frequency or first harmonic. This is also called the *harmonic component* and *harmonic frequency.* Remember also that *harmonic analysis* refers to any method used to identify and evaluate the harmonics (frequencies) that make up a complex form of voltage, current, or some other varying quantity. Consider Fig. 16-9.

Figure 16-9 shows channels 1 and 2 monitored on the Tekscope. This is a voltage harmonic reading from the fundamental harmonic to the 11th. The rms harmonic is 117.9 while the rms voltage is 118 V ac. Now consider Fig. 16-10.

Figure 16-10 shows the amperage harmonics while the network printer is at idle after a number of pages have been printed. The rms amperage reading is 1.038 A as channel 2 indicates, while the rms harmonic amperage reading is 835.4 mA. Note that the scope has also calculated the rms voltage from channel 1 and it reads 118.2 V ac. Now consider Fig. 16-11.

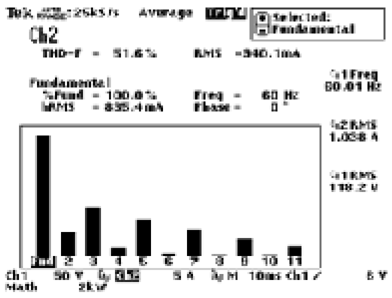


Figure 16-9
Voltage harmonics on the feed to the printer.

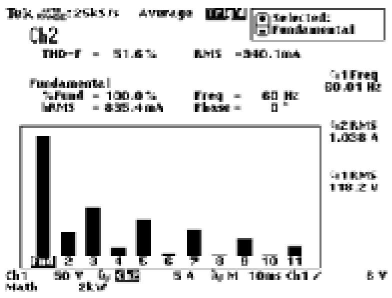


Figure 16-10
Amperage harmonics of the printer.

Figure 16-11 shows the odd harmonics of the voltage reading from the fundamental harmonic to the 21st. This level of detail is quite helpful for those who need to analyze their power quality. Clean power is important, and ascertaining the harmonic values of various readings is important in evaluating power.

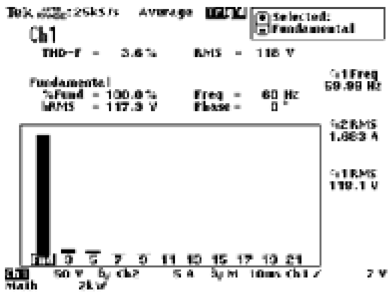


Figure 16-11
Odd harmonics.

16.2.3 *Ground Loops*

Ground loops occur when multiple grounds exist and there is a potential difference between them. The significance of this is that you can incur grounding from network cabling and through your electrical wiring. Ground loops can eat away at your bandwidth because they will inject distortion into a line. Likewise, should the ground loop be sufficient, it can cause power quality degradation. So, what do you do? Well, just be aware that it is possible to pick up grounding in equipment wherever it may reside. It is best to check your telephone wiring and network cabling, and check for stray wires that could create a ground loop.

I encourage you to know early on in your project information such as the electrical requirements you have (what to consider and how to calculate these are given in the following section); wiring of the existing facility; estimates, if possible, for power quality; and information for planning purposes on the grounding of the facility your equipment will operate within.

16.3 Calculating Your Power Requirements

I am presenting fundamental information here. I recommend that you contact a Liebert (Columbus, Ohio) representative to assist you in determining exactly what your power requirements are.

Equipment Category and List

First, it is a good idea to categorize. The equipment and categories I derived and worked from were printers; computers; peripherals; equipment required 90 percent of the time, 50 percent, or less than 50 percent of the time; equipment in multiple locations; equipment in one location; number of hours you can afford to lose work or for downtime; and amount of money you can afford to lose.

This list is where I began categorizing my plan for power requirements. If you do not have an inventory list yet, start one. If you are starting in the middle of a network already established, then begin first by inventorying the equipment you currently have, such as how many computers and monitors you have. Then move onto the printers; how many are networked, and how many are standalone? Is it feasible for all or some of them to use the same power source? List your peripherals as well. It would be a very good idea to access the original vendor documentation and/or reference the information on the equipment label and document the electrical and environmental requirements.

Now, list the equipment required 50 and 90 percent of the time. In other words, if a PC is used all day everyday in your environment, then list it under the 90 percent category. If you have a system that is used only occasionally, then list it as such.

It is also wise to list the equipment you have, or will have, at the location in which it will be installed. This is important because networks generally span multiple facilities and sometimes multiple locations. The result is you may be required to purchase multiple types of equipment in order to get the coverage you need.

Another important issue is to determine how much money you can afford to lose if your equipment is *cooked* by a lightning hit. Don’t laugh. I know some people who lost equipment who did not plan ahead and prepare for this scenario.

Similar to losing equipment is the number of hours you can lose in lack of productive work or downtime. How many workers-hours can you afford to lose? If you have 25 people, all using computers and you lose an entire day, this equates to 25 worker-days.

Calculations

Liebert has a very helpful tool when it comes to protection calculation. The equation here reflects their efforts. First, determine whether you have one- or three-phase equipment. Most of you will have one phase. Be sure to obtain what is considered steady-state operating amperage. This is not the circuit-breaker rating, surge current, or inrush current. Use this number in the amperage portion of the equation. If you do not use steady-state current, erroneous conclusions may result. To determine the amount of UPS coverage, you can use the following equation:

$$\text{kVA} = \frac{\text{VA}}{1000}$$

The number you arrive at should be about 20 to 50 percent greater than this; this number will be the approximate size you need. Now, reflect back to the Liebert UPS rating as shown previously as kVA ratings. This is very simplified, and you need to remember this. As the Liebert documentation correctly indicates, you would be better served by load balancing your equipment through the use of three-phase UPS protection and then determining the amount of UPS on the basis of the largest load. This topic goes beyond the scope of the purpose here; I recommend contacting Liebert directly and letting them help you with three-phase power.

16.4 Types of Power Protection

Some of the basic types of power protection and their purpose are discussed here.

Surge Protectors

Surge protectors do exactly that; they protect against surges. Spikes (sudden voltage increases) are typically inhibited or prevented by surge-protection devices. This is probably the mini mum level of protection you need. Even computer use at home should have this minimum level of protection.

Voltage Regulators

Voltage regulators have the ability to maintain a certain voltage and current level for the device drawing on the electrical source. A voltage regulator is a good device to use with printers, especially laser printers.

Uninterruptible Power Supply (UPS)

A UPS device is designed to provide interim power. UPS devices are not a solution to power outages; they are an interim or stop-gap solution to bridge the time when power ceases and is restored. Large UPS units may be able to provide a significant number of minutes or even hours of uptime. However, UPS use presupposes that power from the electrical company will be restored or alternative power supply will be provided from a source such as a backup generator.

Generators

Generators create or *generate* electricity. They are typically used as alternative power sources in case the main source of power fails. A sufficiently large generator can keep generating electric power indefinitely—assuming you can provide the diesel or gasoline to keep it operating. Generators are typically reserved for use in large commercial settings.

Transfer Switch Gear

Transfer switch gear is used to change the source of electric power to a generator. Switch gear does what its name implies: it switches.

Parallel Switch Gear

Parallel switch gear is used in settings where multiple generators are used in parallel with the other. This is advanced technology and not for the faint-hearted. Generators, transfer switches, and paralleling UPSs are complex. Such implementations are usually found in complex commercial settings, such as hospitals.

16.5 The UPS in My Network

Because of the amount of equipment in my network, Liebert's reputation, and the quality of their products, I selected a 15-kVA UPS unit from Liebert. Consider Fig. 16-12.

Figure 16-12 shows the network equipment used. It also shows the Liebert UPS as the power-protection device used to accommodate all the equipment in the network. Now consider Fig. 16-13.

Figure 16-13 shows the rear view of the UPS I used in my network. A great advantage of the UPS is the accessibility it provides for the components that need it.

After sizing up the UPS, I determined I needed a 15-kVA unit. It arrived weighing in at over 800 lb. With the person who made the delivery helping, it took only an hour to get the well-packed UPS in the place I had prepared for it. About 2 weeks before UPS arrived, I located the place where I wanted to receive it from the shipper. I marked off the locations so enough space would be unobstructed. Doing this level of preplanning saved a lot of time.

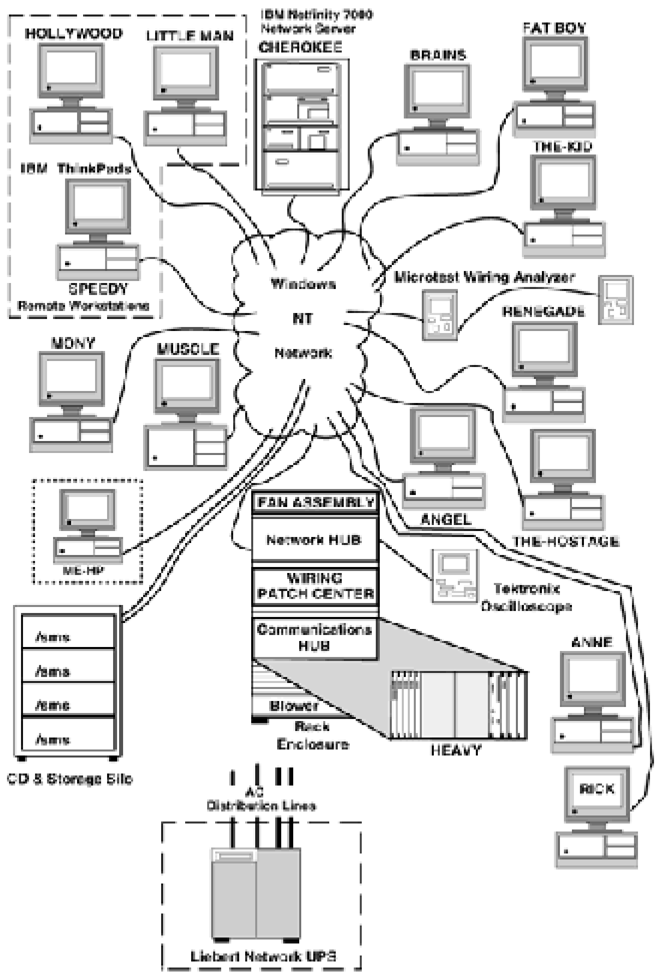


Figure 16-12
Liebert UPS for the network.

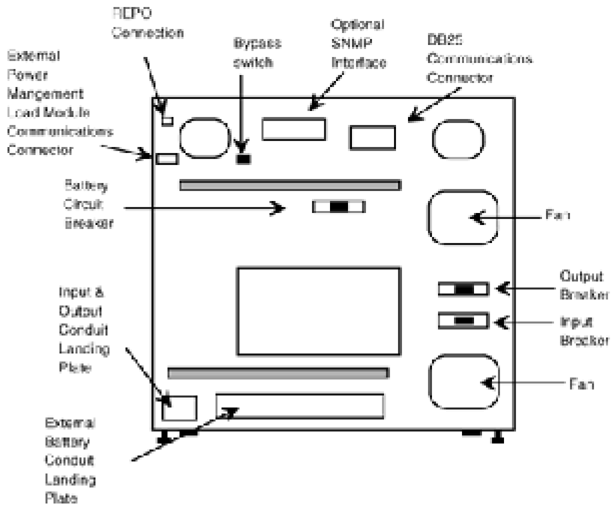

REPO Connection
External Power Mangement Load Module Communications Connector
Bypass switch
Optional SNMP Interface
DB25 Communications Connector
Battery Circuit Breaker
Fan
Output Breaker
Input Breaker
Input & Output Conduit Landing Plate
Fan
External Battery Conduit Landing Plate

Figure 16-13
Liebert UPS rear view.

As I said, it only took about an hour to get the UPS into the place I had reserved for it. It then took approximately an hour to get it unpacked and in its final resting place. Believe me, moving an 800-lb UPS unit is not something I want to do every day of the week. Now that you are reading this, take the time to do this level of planning for yourself. If you do, the people who work with you will be all the happier. If you do not, well, you just may have a sizable amount of weight arrive on pallet and have nowhere to put it. If you are new to the networking industry, take advantage of my experience and plan.

Another function of the Liebert UPS used in my network is its participation in the network via the SNMP interface. Liebert did an excellent job making the UPS functional with exiting SNMP protocol standards. It supports standard MIB definitions and interoperates well in the network. The SNMP interface is optional on the UPStation S UPS units from Liebert, and this is advantageous for those purchasing UPSs because not all networks will use an SNMP network management application. The crux of the matter is that it saves the customer money.

The 15-kVA Liebert UPS requires wiring into the load center of the facility in which it is operating. I recommend that you obtain a licensed electrician to assist you in preplanning, and especially during the installation of the UPS (or contact Liebert). Working with equipment like this is no trivial task, and it should not be played with because it can be dangerous. I don’t mean to promote fear; I simply want to emphasize that wiring in a UPS the size of mine is only for those who are trained in electricity. Consider Fig. 16-14.

Figure 16-14 shows three parts: the ac load center that feeds current to the facility where the network is located, a highlighted view of the UPS wired into the load center, and a conceptual view of how some of the equipment is connected to the UPS. This is the basic overview of how UPS fits into the network. It is more complex than this because of all the equipment the UPS protects. Because the UPS is capable of protecting the entire network, approximately four output feeds (lines with outlets) were required to accommodate all components.

This is an overview of the UPS used in this network; many more details could be included, but the purpose here is to assist you in conceptually understanding the placement and purpose of the UPS. Your situation will vary; however, consider how I implemented the UPS here as a reference point for thought.

16.6 Liebert Information on the UPStation S UPS

To do the Liebert UPS justice, I elected to present the Liebert documentation (with minor modifications) for the UPS I used. Consider this as a very good example of what you'll need with the UPS you decide to use. The information here is the documentation of the Liebert UPStation S UPS.

This specification defines the electrical and mechanical characteristics and requirements for a continuous-duty single-phase, solid-state, uninterruptible power supply system. The uninterruptible power supply system, hereafter referred to as the UPS, shall provide high-quality ac power for sensitive electronic equipment loads.

Standards

The UPS shall be designed in accordance with the applicable sections of the current revision of the following documents. Where a conflict arises between these documents and statements made herein, the statements in this specification shall govern.

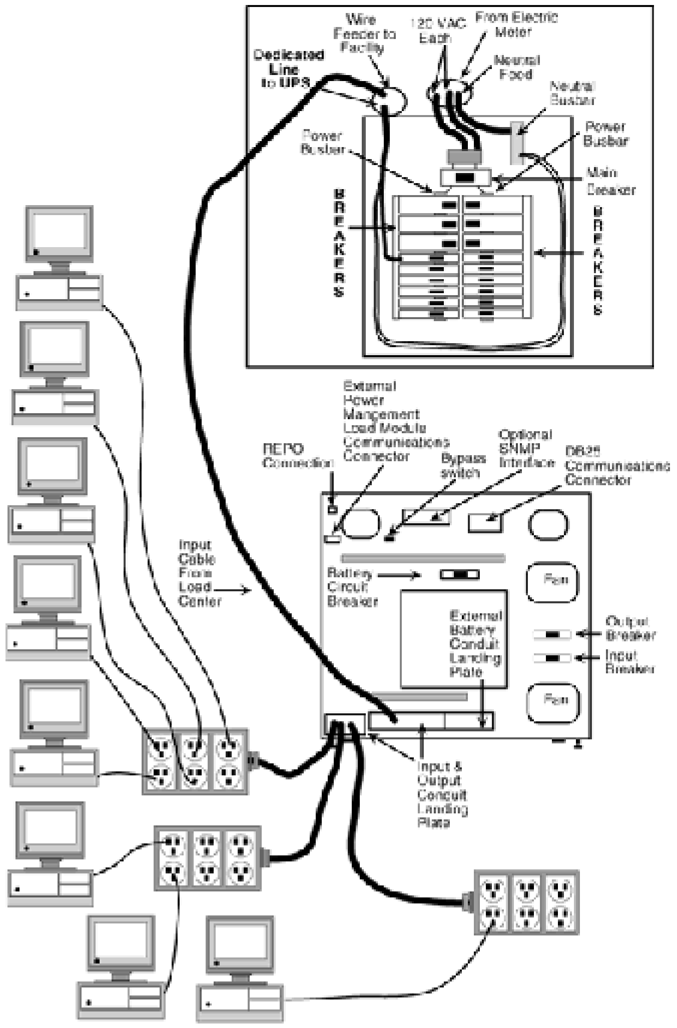


Figure 16-14
Conceptual view of ac supply.

ANSI C62.41-1980 (IEEE 587), Categories A and B

ASME

CSA 22.2, No. 107.1

FCC Part 15, Subpart J, Class A

National Electrical Code (NFPA 70)

NEMA PE-I

OSHA

UL Standard 1778

The UPS shall be UL-listed to UL Standard 1778, and shall be CSA-certified.

System Description (Modes of Operation)

The UPS shall be designed to operate as a true online system in the following modes:

A. *Normal.* The critical ac load is continuously supplied by the UPS inverter. The input converter derives power from a utility ac source and supplies dc power to the inverter. The battery charger shall maintain a float charge on the battery.

B. *Backup.* On failure of utility ac power, the critical ac load is supplied by the inverter, which obtains power from the battery. There shall be no interruption in power to the critical load on failure or restoration of the utility ac source.

C. *Recharge.* On restoration of utility ac power, after a utility ac power outage, the input converter shall automatically restart and resume supplying power to the inverter. Also the battery charger shall recharge the battery.

D. *Automatic restart.* On restoration of utility ac power, after a utility ac power outage and complete battery discharge, the UPS shall automatically restart and resume supplying power to the critical load. Also the battery charger shall automatically recharge the battery. This feature shall be enabled from the factory and shall be capable of being disabled by the user.

E. *Bypass.* The factory-installed bypass (optional on 3.5- to 6.0-kVA UPS modules) shall provide an alternate path for power to the critical load that shall be capable of operating in the following manner: (1) *Automatic.* In the event of an internal failure or should the inverter overload capacity be exceeded, the UPS shall perform an automatic transfer of the critical ac load from the inverter to the bypass source. (2) *Manual.* Should the UPS need to be taken out of service for limited maintenance or repair, manual activation of the bypass shall cause an immediate transfer of the critical ac load from the inverter to the bypass source. The input converter, inverter, and battery-charging operations shall be inhibited until the switch is moved back to the UPS position and the unit restarted.

System Performance Requirements

A. *Upgradability.* Select UPS systems shall be field-upgradable to higher power ratings as follows:

3.5-kVA modules shall be field-upgradable to 4.5-kVA modules,
3.5-kVA modules shall be field-upgradable to 6.0-kVA modules,
4.5-kVA modules shall be field-upgradable to 6.0-kVA modules.
8.0-kVA modules shall be field-upgradable to 10.0-kVA modules,
8.0-kVA modules shall be field-upgradable to 12.0-kVA modules,
10.0-kVA modules shall be field-upgradable to 12.0-kVA modules.
15.0-kVA modules shall be field-upgradable to 18.0-kVA modules.

B. *Isolation.* Input-to-output isolation shall be provided when operating in the UPS mode.

C. *Remote emergency power OFF.* The UPS shall provide provisions for remote emergency power-OFF capability.

AC Input to UPS

A. *Voltage configuration:* 176 to 264 V ac, single-phase, three-wire-plus-ground. Two-wire-plus-ground can be utilized for 3.5- to 6.0-kVA UPS modules without the optional bypass. The UPS modules (8 to 18 kVA only) shall also be capable of three-phase input to the rectifier. The UPS modules shall be capable of dual input to provide a separate power path for the bypass (optional on 8- to 18-kVA UPS modules only).

B. *Frequency:* 45 to 65 Hz.

C. *Input current distortion:* 5 percent total harmonic distortion (THD) maximum at full load. If three-phase input is utilized (available on 8- to 18-kVA units), maximum THD shall not exceed 30 percent.

D. *Input power factor:* 0.98 lagging minimum from 50 to 100 percent rated load.

E. *Inrush current:* 150 percent of full-load input current maximum for three cycles.

F. *Surge protection:* Sustains input surges without damage per criteria listed in ANSI C62.41-1980 (IEEE 587), Categories A and B.

AC Output, UPS Inverter

A. *Voltage configuration:* 208/120 V ac, single-phase, three-wire-plus-ground. Field-programmable to 240/120, 230/115, 220/127, 220/110, and 200/100 V ac. The UPS can also be ordered to the supply line to neutral voltages of 240, 230, and 220 V ac.

B. *Voltage regulation:* ±2 percent steady state.

C. *Frequency regulation:* field-selectable 50 or 60 Hz, ±0.01 percent.

D. *Frequency slew rate:* 1.0 H/s maximum. Field-selectable from 0.3, 0.5, 1, 2, or 3 Hz/s from the LCD display.

E. *Bypass frequency synchronization range:* ±1.0 Hz. Field-selectable from ±0.1 to ±5.0 Hz, in 0.1-Hz increments from the LCD display.

F. *Voltage distortion:* 3 percent THD maximum into a 100 percent linear load, 5 percent THD maximum into a 100% nonlinear load with crest factor ratio of 3:1.

G. *Load power factor range:* 0.5 lagging to 0.5 leading.

H. *Output power rating:* rated kVA at 0.8 lagging power factor for 3.5- and 4.5-kVA models; 0.67 to 0.77 lagging power factor for 6.0-kVA models depending on output voltage; 0.75 lagging power factor for 8.0- and 10.0-kVA models; 0.7 lagging power factor for 12.0-, 15.0-, and 18.0-kVA models.

I. *Inverter overload capability:* 105 percent continuously, 125 percent for 10 min, 150 percent for 10 s, 250 percent for 12 cycles. For units with the bypass option installed, the load shall be transferred to bypass when any of the above conditions are exceeded.

J. *Inverter output voltage adjustment:* ±5 percent manual adjustment from the LCD by qualified service personnel.

K. *Voltage transient response:* ±5 percent maximum for any load step up to and including 100 percent of the UPS rating.

L. *Transient recovery time:* to within 1 percent of steady-state output voltage within 50 ms.

Batteries

A. *Internal battery.* The battery shall consist of sealed, valve-regulated, reduced-maintenance, lead acid cells. Flame-retardant batteries can also be provided, which renders the UPS suitable for installation inside a computer room per requirements of UL Standard 1778.

B. *Reserve time* (with ambient temperature between 20 and 30°C).The UPS shall contain an internal battery system to provide the following reserve time:
.
5 kVA/2.8 kW—18 min at full load
4.5 kVA/3.6 kW—12 min at full load
6.0 kVA/4.0 kW—11 min at full load
8.0 kVA/6.0 kW—17 min at full load
10.0 kVA/7.5 kW—11 min at full load
12.0 kVA/8.4 kW—10 min at full load
15.0 kVA/10.5 kW—14 min at full load
18.0 kVA/12.6 kW—10 min at full load

The UPS shall contain provisions to interface with an external matching battery cabinet to extend reserve time capabilities.

C. *Battery recharge.* To prolong battery life, the UPS shall contain temperature-compensated battery charging. Recharge time shall be ten (10) times discharge time to 95 percent capacity. [Field-selectable to twenty (20) times discharge time to 95 percent capacity from the LCD display.]

Environmental Conditions

A. *Ambient temperature.* Operating: UPS 0 to 40°C; battery 20 to 30°C for optimum performance. Storage: UPS –30 to +70°C; battery 0 to 32°C for maximum 6 months.

B. *Relative humidity.* Operating: 0 to 95 percent noncondensing. Storage: 0 to 95 percent noncondensing.

C. *Altitude.* Operating: to 4000 ft. Derating or reduced operating temperature range required for higher altitudes. Storage: to 50,000 ft.

D. *Audible noise.* Noise generated by the UPS during normal operation shall not exceed 55 dBA measured at one meter (1 m) from the surface of the UPS.

E. *Electrostatic discharge.* The UPS shall be able to withstand a minimum 15 kV without damage and shall not affect the critical load.

User Documentation

The specified UPS system shall be supplied with one (1) user's manual. Manuals shall include installation drawings and instructions, a functional description of the equipment with block diagrams, safety precautions, illustrations, step-by-step operating procedures, and routine maintenance guidelines.

Warranty

The UPS manufacturer shall warrant the UPS against defects in materials and workmanship for 1 year. The warranty shall cover all parts for 1 year and on-site labor for 90 days. With the option al start-up provided by Customer Service and Support, the warranty shall cover all parts and onsite labor for 1 year. Extended warranty packages shall also be available.

Quality Assurance

A. *Manufacturer qualifications.* A minimum of 20 years' experience in the design, manufacture, and testing of solid-state UPS systems is required.

B. *Factory testing.* Before shipment, the manufacturer shall fully and completely test the system to assure compliance with the specification. These tests shall include operational discharge and recharge tests on the internal battery to guarantee rated performance.

Product Fabrication

All materials and components making up the UPS shall be new and of current manufacture, and shall not have been in prior service except as required during factory testing. The UPS shall be constructed of replaceable subassemblies. All active electronic devices shall be solid-state.

Product Wiring

Wiring practices, materials, and coding shall be in accordance with the requirements of the National Electrical Code (NFPA 70) and other applicable codes and standards.

Product Cabinet

The UPS unit, comprising input converter, battery charger, inverter, bypass, and battery consisting of the appropriate number of sealed battery cells, shall be housed in a single free-standing NEMA type 1 enclosure and meets the requirements of IP20. The UPS cabinet shall be cleaned, primed, and painted with the manufacturer's standard color. Casters and leveling feet shall be provided. UPS cabinet dimensions shall not exceed {select one: [9 in wide, 27 in deep, and 29 in high (for 3.5- to 6.0-kVA units)]; [18 in wide, 27 in deep, and 29 in high (for 8.0- to 12.0-kVA units)]; [27 in wide, 27 in deep, and 29 in high (for 15.0- to 18.0-kVA units)]}.

Product Cooling

The UPS shall be forced-air-cooled by internally mounted, variable-speed fans to reduce audible noise.

Components

A. *Input converter (general).* Incoming ac power shall be converted to a regulated dc output by the input converter for supplying dc power to the inverter. The input converter shall provide input power factor and input current distortion correction. The input converter shall provide input to output isolation by means of a high-frequency transformer.

B. *AC input current limit.* The input converter shall be provided with ac input current limiting, whereby the maximum input current shall be limited to 125 percent of the full load input current rating.

C. *Input protection.* The UPS shall have built-in protection against undervoltage, overcurrent, and overvoltage conditions including low-energy surges, introduced on the primary ac source and the bypass source. The UPS shall sustain input surges without damage per criteria listed in ANSI C62.41-1980 (IEEE 587). The UPS shall contain {select one: [input fuses (3.5 to 6.0 kVA)] or [an input circuit breaker (8.0 to 18.0 kVA)]} sized to supply full-rated load and to recharge the battery at the same time.

D. *Battery recharge.* To prolong battery life, the UPS shall contain two battery recharge rates, and charging voltage shall be temperature-compensated. The "turbo" mode of recharge shall be capable of recharging the battery to 95 percent capacity within 10 times the discharge time of the battery. The "slow" mode of recharge shall be capable of recharging the battery to 95 percent capacity within 20 times the discharge time. The factory default shall be "turbo" mode, and the microprocessor shall determine how often the faster recharge rate is employed to optimize battery life.

E. *Charger output filter.* The battery charger shall have an output filter to minimize ripple current into the battery. Under no conditions shall ripple current into the battery exceed 2 percent rms.

F. *Overvoltage protection.* There shall be dc overvoltage protection so that if the dc voltage exceeds the preset limit, the UPS shall shut down automatically and the critical load shall be transferred to bypass (if optional bypass is installed).

In addition:

A. *Inverter (general).* The UPS shall contain two independently controlled inverters and shall be of pulse-width-modulated (PWM) design capable of providing the specified ac output. The inverters shall convert dc power from the input converter output, or the battery, into precise regulated sine-wave ac power for supporting the critical ac load.

B. *Overload.* The inverter shall be capable of supplying current and voltage for overloads exceeding 100 percent and up to 150 percent of full-load current. A visual indicator and audible alarm shall indicate overload operation. For greater currents or longer time duration, the inverter shall have electronic current-limiting protection to prevent damage to components. The inverter shall be self-protecting against any magnitude of connected output over load. Inverter control logic shall sense and disconnect the inverter from the critical ac load without the requirement to clear protective fuses. For units supplied with the bypass option, the load shall be transferred to bypass when any of the above conditions are exceeded.

C. *Inverter dc protection.* The inverter shall be protected by the following dc shutdown levels: (1) dc overvoltage shutdown, (2) dc undervoltage shutdown (end of discharge), and (3) dc undervoltage warning (low battery reserve) shall be factory-set at 2 min and user-adjustable from 1 to 99 min from the LCD display.

D. *Inverter output voltage adjustment.* The inverter shall employ a manual control to adjust the output voltage from ±5 percent of the nominal value from the LCD by qualified service personnel.

E. *Output frequency.* The output frequency of the inverter shall be controlled by an oscillator. The oscillator shall hold the inverter output frequency to ±0.01 percent for steady-state and transient conditions. For units equipped with a bypass, the inverter shall track the bypass continuously, provided the bypass source maintains a frequency within the user-selected synchronization range. If the bypass source fails to remain within the selected range, the inverter shall revert to the internal oscillator.

F. *Output protection.* The UPS inverter shall employ electronic current limiting and an output circuit breaker. The main output breaker shall be rated for a minimum 10 kA/IC.

G. *Battery overdischarge protection.* To prevent battery damage from over-discharging, the UPS control logic shall automatically raise the shutdown voltage set point as discharge time increases beyond 15 min.

Display and Controls

A. *General.* The UPS shall be provided with a microprocessor-based unit status display and controls section designed for convenient and reliable user operation. The monitoring functions such as metering, status, and alarms shall be displayed on a 4-line × 16-character alphanumeric LCD display. Additional features of the LCD monitoring system shall include menu-driven display with text format, real-time clock (time and date), alarm history with date/timestamp, and battery-backed-up memory.

B. *Metering.* The following parameters shall be displayed: (1) input ac voltage line-to-line and line-to-neutral for each phase; (2) input ac current for each phase; (3) input frequency; (4) battery voltage; (5) battery charge/discharge current; (6) output ac voltage line-to-line and line-to-neutral for each phase; (7) output ac current for each phase; (8) output frequency; (9) percent of rated load being supplied by the UPS; (10) output kVA and kW for each phase; (11) battery time remaining during battery operation; (12) operating temperature of input converter, inverter, and internal battery.

C. *Status messages.* The following UPS status messages shall be displayed: (1) normal operation, (2) UPS on battery, (3) system shutdown, (4) startup sequence aborted, (5) battery test enabled/disabled, (6) system time set by operator, (7) load on bypass.

D. *Alarm messages.* The following alarm messages shall be displayed: (1) input power out of tolerance, (2) UPS output not synchronized to input, (3) output undervoltage, (4) input power single-phased, (5) output overvoltage, (6) incorrect input frequency, (7) output overcurrent, (8) input in current-limit charging batteries, (9) overcurrent detected in inverter, (10) charger failure, (11) control error—software timeout, (12) battery charger problem, (13) control error—internal test, (14) battery failed test, (15) critical power-supply failure, (16) can’t execute battery test—not recharged, (17) external shutdown (remote EPO activated), (18) can’t execute battery test at this time, (19) fan failure, (20) low-battery shutdown, (21) low-battery warning (adjustable 1 to 99 min), (22) dc bus overvoltage, (23) system shutdown due to overload, (24) PFC fault, (25) system shutdown impending due to overtemperature, (26) inverter fault, (27) inverter failure, (28) system shutdown—loss of control power, (29) overtemperature shutdown, (30) system output overloaded. An audible alarm shall be provided and activated by any of these alarm conditions. In units equipped with the bypass option, the following additional alarms shall be displayed: (1) bypass frequency out of tolerance, (2) bypass power out of tolerance, (3) fault load transferred to bypass, (4) load transferred to bypass due to overload, (5) load transferred to bypass due to dc overvoltage, (6) excessive retransfers attempted. An audible alarm shall be provided and activated by any of these alarm conditions.

E. *Controls.* UPS startup and shutdown operations shall be accomplished via the front LCD display panel. An advisory display and menu-driven user prompts shall be provided to guide the operator through system operation without the use of additional manuals. Push buttons shall be provided to display the status of the UPS and to test and reset visual and audible alarms. The UPS shall contain an output circuit breaker and a manual bypass switch (optional on 3.5- to 6.0-kVA) as additional user controls, which shall be located on the rear of the unit.

Online Battery Test

The UPS shall be provided with an online battery test feature. The test shall ensure the capability of the battery to supply power to the inverter while the load is supplied power in the normal mode. If the battery fails the test, the UPS shall display a warning message and sound an audible alarm. The battery test feature shall have the following user-selectable options, accessible from the LCD display: (1) dc bus voltage threshold (pass/fail value), (2) interval between tests (2 to 9 weeks), (3) duration of test (30 to 900 s; factory default is 30 s), (4) date and time of initial test, (5) enable/disable test.

Remote Emergency Power Off

Remote emergency power off capabilities shall be provided. A connector shall be provided for connection of a normally open contact supplied by the user. A 50-ft cable with the mating connector and pushbutton shall be available as an option.

Bypass (optional on 3.5- to 6.0-kVA UPS modules):

A. *General.* A bypass circuit shall be provided as an integral part of the UPS. The bypass shall have an overload rating of 300 percent-rated load for 10 cycles and 1000 percent for subcycle fault clearing. The bypass control logic shall contain an automatic transfer control circuit that senses the status of the inverter logic signals, and operating and alarm conditions. This control circuit shall provide a transfer of the load to the bypass source, without exceeding the transient limits specified herein, when an overload or malfunction occurs within the UPS.

B. *Automatic transfers.* The transfer control logic shall automatically activate the bypass, transferring the critical ac load to the bypass source, after the transfer logic senses one of the following conditions: (1) inverter overload capacity exceeded, (2) critical ac load overvoltage or undervoltage, or (3) UPS fault condition. For inverter overload conditions, the transfer control logic shall inhibit an automatic transfer of the critical load to the bypass source if one of the following conditions exists: (1) inverter/bypass voltage difference exceeding preset limits (+10, –15 percent of nominal), or (2) bypass frequency out of preset limits (±1.0 Hz, field-selectable from ±0.1 Hz to ±5.0 Hz, in 0.1-Hz increments). For UPS fault or output over/undervoltage conditions, the transfer control logic shall not inhibit automatic transfers of the critical load to the bypass source.

C. *Automatic retransfers.* Retransfer of the critical ac load from the bypass source to the inverter output shall be automatically initiated unless inhibited by manual control. The transfer control logic shall inhibit an automatic retransfer of the critical load to the inverter if one of the following conditions exists: (1) bypass out-of- synchronization range with inverter output, (2) overload condition exists in excess of inverter full-load rating, or (3) UPS fault condition present.

D. *Manual transfer.* Manual operation of the bypass shall directly connect the critical load to the input ac power source, bypassing the input converter, battery charger, inverter, and battery.

Internal Battery

Sealed, valve-regulated, low-maintenance, lead acid batteries shall be used as a stored-energy source for the specified UPS system. Flame-retardant batteries shall be available for computer-room applications. The battery shall be housed internal to the UPS cabinet, and sized to support the inverter at rated load and power factor, in an ambient temperature between 20 and 30°C, for a minimum of 10 min reserve time. The expected life of the battery shall be 5 years or a minimum 250 complete discharge cycles. For extended battery reserve time, external matching battery cabinets shall be available as an option (see Sec. 2.3.4 in Liebert UPStation S specifications).

Output Load Modules

Required on 3.5- to 6.0-kVA models. May be used on 8- to 12-kVA models only with configurable distribution module. Output load modules (maximum of four on 3.5- to 12.0-kVA units, maximum of eight on 8.0- to 12.0-kVA units optional configurable distribution module) shall be provided for field addition to the UPS cabinet. Each load module shall include user-specified receptacle(s), LED indicator lamp, and circuit-breaker protection. The following load modules shall be provided:

Receptacle	**Voltage (configuration)**	**Circuit breaker**
5-15R2	120 (L-N-G)	15A, 2-pole
L5-15R1	120 (L-N-G)	15A, 1-pole
6-15R1	208 or 240 (L-L-G)	15A, 2-pole
L6-15R1	208 or 240 (L-L-G)	15A, 2-pole
L5-15R2	120 (L-N-G)	15A, 2-pole
5-20R2	120 (L-N-G)	20A, 2-pole
L5-20R1	120 (L-N-G)	20A, 1-pole
L6-20R1	208 or 240 (L-L-G)	20A, 2-pole
L14-20R1	208/120 or 240/120 (L-L-N-G)	30A, 1-pole
L5-30R1	120 (L-N-G)	30A, 1-pole
L6-30R1	208 or 240 (L-L-G)	30A, 2-pole
L-14-30R1	208/120 or 240/120 (L-L-N-G)	30A, 2-pole
Hardwired (15A)	208/120 or 240/120 (L-L-N-G)	15A, 2-pole
Hardwired (20A)	208/120 or 240/120 (L-L-N-G)	20A, 2-pole
Hardwired (30A)	208/120 or 240/120 (L-L-N-G)	Not required

UPS Accessories (Optional Components)

Communication interface boards:

A. *General.* Communication interfaces shall be provided on field-installable, plug-in printed-circuit boards. Installation shall be from the top of the UPS cabinet. Two interface slots shall be provided to allow multiple communication capabilities.

B. *RS-232 interface board.* The RS-232 interface board shall facilitate communication interfaces for RS-232 communication, external modem, computer/LAN interface, AS/400 interface, and customer-configured relay interface.

1. *RS-232 interface port.* The RS-232 interface port shall transmit UPS status for display at a remote terminal, computer, or external modem (all by others), via a DB25 connector. The remote display shall mimic the information provided on the LCD monitoring and control panel, including status of all UPS alarms, input voltage, output voltage, percentage load, and battery time remaining.

2. *Autodial.* The UPS RS-232 interface port shall be capable of interfacing with an external modem to report alarm and status information to a remote location. The modem control system shall have the capability to store one telephone number. The UPS shall automatically dial the remote location in the event of the occurrence of any of the following alarms: (*a*) User-initiated UPS shutdown, (*b*) battery SCR fault (short or open circuit), (*c*) PFC voltage high, (*d*) dc bus undervoltage, (*e*) PFC voltage low, (*f*) ON battery, (*g*) PFC shutdown due to overtemperature, (*h*) low battery, (*i*) PFC fault (hardware fault), (*j*) battery failed test, (*k*) charger shutdown due to overtemperature, (*l*) UPS fault, (*m*) inverter shutdown due to overcurrent, (*n*) excessive retransfer attempts, (*o*) inverter shutdown due to overload, (*p*) control power-supply failure, (*q*) inverter shutdown due to under/overvoltage, (*r*) ROM test failed, (*s*) inverter shutdown due to overtemperature, (*t*) RAM test failed, (*u*) bypass shutdown due to overload (UPS shutdown), or (*v*) timeout fault (self-test at startup).

3. *Computer/LAN interface kits.* Computer/LAN interface kits shall consist of a 10-ft communication cable and software. Two separate software kits shall be available to operate in the following manner. Computer/LAN shutdown software shall run as a background task on the computer while monitoring the UPS. The software shall perform an unattended orderly shutdown of the computer operating system, when signaled by the UPS of a power failure or low battery condition. Power Surveillance software shall run as a background task on the computer while monitoring the UPS. The Power Surveillance software shall incorporate the shutdown capabilities of Computer/LAN shutdown software (except DOS) and shall also provide UPS monitoring and control capabilities from a standalone computer.

4. *IBM* AS/400* interface kits.* A shielded-cable compliant with NEMA Class 2 for plenum applications, with subminiature 9-pin D-type connector, shall be provided for connection to the IBM AS/400 signal interface *(IBM and AS/400 are registered trademarks of International Business Machines Corporation). Contacts rated for 1.0 A at 30 V ac or dc shall be provided to indicate a change of status of each of the following conditions: ON UPS, ON battery, low battery, or ON bypass. An optional splitter cable shall be available to allow connection to either RS-232, Power Surveillance, or external modem communication *and* Computer/LAN shutdown software or AS/400 interface simultaneously.

5. *Customer-configured relay interface.* An interface board shall be available that allows the customer to have UPS status/alarm conditions available for remote panel monitoring via relay contact closures. Programming is done via the LCD display.

C. *Simple Network Management Protocol (SNMP) interface board.* The SNMP interface board (agent) shall be provided to allow communication between the UPS and any network management system (NMS). The SNMP agent shall be installed in the UPS cabinet, and external "proxy" agent configurations shall not be allowed. The SNMP interface shall be available in Ethernet or Token-Ring TCP/IP message format to permit direct connection to the network. Ethernet connection shall be unshielded twisted pair (UTP). Token-Ring connection shall be unshielded twisted pair (UTP) and shielded twisted pair (STP). The SNMP agent shall include software that contains the standard library of commands in the management information base (MIB). An extended MIB shall be available to enable the user access to create custom UPS-variable interface screens. SNMP snap-in interfaces shall be available for major NOS platforms. The SNMP connection shall allow multiple network managers to both monitor the status and control many operational features of the UPS. Monitoring data supplied by the UPS shall include but not be limited to the input, output, and battery voltages and currents, battery condition, battery reserve time remaining, internal component temperatures, UPS loading, and UPS status and alarm indicators. UPS operational features shall include but not be limited to setting UPS operating parameters, turning the UPS off and on, and manually initiating battery tests.

Power Management Load Modules (Optional on 3.5- to 12.0-kVA Units)

Power management load modules (maximum of four on 3.5- to 12.0-kVA units, or maximum of eight on 8.0- to 12.0-kVA units) shall be provided for field addition to the UPS cabinet. Power management load modules shall provide the ability to individually control power through the output receptacles. The load modules (listed in Sec. 2.2.8 of the Liebert UPStation S specifications) are available as optional power management load modules with the exception of the hardwire modules. The output receptacles shall be capable of being turned on and off by either manual initiation or programmed time intervals. Power management load modules shall enable the user to (1) extend battery run time, by selectively turning off loads at predetermined time intervals; (2) conserve energy, by allowing the user to turn off "idle" equipment; (3) reset or reboot loads, by remotely turning the load off then on, enabling the user to reboot "locked up" equipment; (4) increase security, by remotely turning off loads to prohibit unauthorized personnel access; and (5) sequence load restarting, by selectively turning on loads at predetermined time intervals to minimize high inrush currents.

Power management programming shall be password-protected and can be accessed via the LCD display. The power management programming shall be user-specified, based on four elements: load module identification, events, action, and delay. Load modules shall be numerically identified and shall be capable of containing a user-defined alphanumeric label of up to 16 characters. An event shall be the triggering occurrence to cause the load module to take action. Events include but shall not be limited to system power-up, utility failure, utility restored, battery time remaining, time and date (day), and ON bypass. Programmable actions include but shall not be limited to output on (turns inverter on), output off (turn entire UPS off), turn on load [turns specified load module(s) on], and turn off load [turns specified load module(s) off]. Each action shall have an associated delay, user selected from 0 to 99 s or 0 to 99 min, prior to the execution of the action.

External Battery Cabinet

An external battery cabinet shall be provided for the UPS. The battery cabinet shall be used in parallel with the internal battery for extended power-outage reserve time. The battery cabinet shall contain (1 or 2) strings of sealed, valve-regulated, low-maintenance, lead acid cells, housed in a separate cabinet that matches the UPS cabinet styling. Flame-retardant batteries shall be available for computer-room applications. Intercabinet wiring with mating connectors shall be supplied with the external battery cabinet (for 3.5- to 12.0-kVA models only). The external battery cabinet shall contain two fuses per string, to provide individual string protection and isolation. The external battery cabinet shall increase power-outage reserve time to (__) min at full load, when operating in an ambient temperature between 20 and 30°C. Battery cabinets containing their own ac-dc chargers shall be available for applications requiring run times greater than 4 h.

Dual Input (Optional on 8.0- to 18.0-kVA UPS Modules)

The UPS shall be provided with a separate input terminal block to provide the UPS with dual input capabilities. This separate feed shall be single-phase, three-wire-plus-ground to match the UPS output configuration. The UPS inverter shall track the bypass source per parameters specified in Sec. 2.2.2.E of this (Liebert UPStation S) specification.

Field Quality Control

The following inspections and test procedures shall be performed by factory-trained field service personnel during the UPS startup:

Visual inspection:

1. Inspect equipment for signs of shipping or installation damage.

2. Verify installation per drawings.

3. Inspect cabinets for foreign objects.

4. Verify that neutral and ground conductors are properly sized and configured.

5. Inspect battery cases.

6. Inspect battery for proper polarity.

7. Verify that all printed-circuit boards are configured properly.

Mechanical inspection:

1. Check all control wiring connections for tightness.

2. Check all power wiring connections for tightness.

3. Check all terminal screws, nuts, and/or spade lugs for tightness.

Electrical inspection:

1. Check all fuses for continuity.

2. Confirm that input voltage and phase rotation is correct.

3. Verify that control transformer connections are correct for voltages being used.

4. Assure connection and voltage of the battery string(s).

Unit Startup and Site Testing

Site testing shall be provided by the manufacturer’s field service personnel if requested. Site testing shall consist of a complete test of the UPS system and the associated accessories supplied by the manufacturer. A partial battery discharge test shall be provided as part of the standard startup procedure. The test results shall be documented, signed, and dated for future reference.

Manufacturer's Field Service

Service Personnel: The UPS manufacturer shall directly employ a nationwide service organization, consisting of factory-trained customer engineers dedicated to the startup, maintenance, and repair of UPS and power equipment. The organization shall consist of factory-trained customer engineers working out of district offices in most major cities. An automated procedure shall be in place to ensure that the manufacturer is dedicating the appropriate technical support resources to match escalating customer needs.

The manufacturer shall provide a fully automated national dispatch center to coordinate field service personnel schedules. One toll-free number shall reach a qualified support person 24 h per day, 7 days per week, 365 days per year. If emergency service is required, callback response time from a local customer engineer shall be 20 min or less.

Replacement parts stocking: Parts shall be available through an extensive network to ensure around-the-clock parts availability throughout the country.

Replacement spare parts shall be stocked by local customer engineers with backup available from district service offices and the manufacturing location.

Customer support parts coordinators shall be on call 24 h per day, 7 days per week, 365 days per year for immediate parts availability.

UPS Maintenance Training

Maintenance training courses for customer employees shall be available by the UPS manufacturer. This training is in addition to the basic operator training conducted as a part of the system startup.

The training course shall cover UPS theory, location of subassemblies, safety, battery considerations, and UPS operational procedures. The course shall include ac-to-dc conversion and dc-to-ac inversion techniques as well as control, metering, and feedback circuits to the printed-circuit-board (PCB) level. Troubleshooting and fault isolation using alarm information and internal self-diagnostics shall be stressed.

Maintenance Contracts

A complete offering of preventive and full-service maintenance contracts for both the UPS system and battery system shall be available. An extended warranty and preventive maintenance package shall be available. Warranty and preventive maintenance service shall be performed by factory-trained customer engineers.

Note: *These guide specifications comply with the format outlined by the Construction Specifications Institute per CSI MP-2-1 and CSI MP-2-2. In correspondence, refer to Liebert document SL-24860 (R9/93).*

Liebert Corporation Year 2000 Product Compliance Statement

Liebert Corporation certifies that all Liebert software and firmware accompanying Liebert-manufactured air conditioning, power conditioning and distribution (PCD), and uninterruptible power supply (UPS) products are *year 2000-compliant.* Below is a list of products covered by this statement.

These Liebert products will function and operate as warranted in accordance with their performance specifications without regard for calendar year changes. Year-2000 compliant product design features consist of device functionality, date data century recognition, calculations that accommodate same-century and multicentury formulas and data values, and date data interface values that reflect the century.

Liebert product resident monitoring firmware does no sorting or processing by date; their products only store and retrieve alarms. The date generator will work properly irrespective of calendar year changes, so alarm histories will remain intact and correct. Embedded firmware in all microprocessor applications use time increments and does not process dates. Therefore, all products will operate properly, and all alarm dates will be displayed on the customer display correctly.

Liebert product-monitoring front-end software (SiteScan 2000W, SiteScan SS2W, and Alert, version 2.6) is presently not year-2000 compliant. This software will be corrected to be year-2000 compliant with the release of version 3.0. This version is scheduled for release in 1998, and will be available as an upgrade revision for all current SiteScan systems. Liebert Access Control Security Systems are not year-2000 compliant.

Product Compliance

Liebert is committed to providing its customers quality state-of-the-art products and timely solutions to current and future needs. The products listed below, when operated by Liebert embedded firmware, Liebert controls, and/or Liebert software, are year-2000 compliant:

Environmental

Atlas PEC	Dealermate	Logicool	Process Chiller
Challenger	Deluxe System 2	LS400	Quantum
Climate 3000	Deluxe System 3	Mini Air	Slimcool
CoolGuard	Deluxe System 4	Mini Tower	Small Systems Monitors
Condensers	Drycoolers	MiniMate	Spacemaster-SX
COP-SX	Industrial Cooling Series	MiniMate Plus	
CSU3000	Intelecool	Modular 3000	
Datamate	Leak Detection System	Modular Plus	
Data Pad	Little Glass House	Modupac Chiller	

UPS and Power Products

Accupower	Powersure Interactive	Series 4000	Sitenet SNMP Manager
AL 300	PPD6300	Series 7200	TVSS/Surge Suppressor
AL 3300	Precision Power Center	Series 7200 1 + 1	Transformers
AP 70X	Resource	Series 7200 HE	UPStation D
AP 200	Select	Series 7200 Single Phase	UPStation G
AP 400	Series AP100 (1xx)	Series 7400	UPStation G RT
AP 4300	Series AP300 (3xx)	Series 8000, 9000	UPStationGX
Data Wave	Series AP500 (5xx)	Static Transfer Switch	UPStation GXT
MicroUPS	Series AP600 (6xx)	SmartSwitch	UPStation S
PowerMate	Series 600T	Sitenet Integrator	UPStation S-3
Powersure	Series AP700 (70x)	Sitenet 1, and 2	

On-Site Access Control Software and Year 2000 Compliance

The Liebert OEM supplier has informed us that the On-Site Access Control software is *not* year-2000 compliant. This noncompliance includes Liebert On-Site/4, On-Site/10, On-Site/16, On-Site/64, and On-Site/128, the latest DOS version Liebert has been marketing. There are both external and internal causes of noncompliance.

1. *External to On-Site software:* The On-Site software relies on the system clock controlled by the BIOS (basic input/output system) of the host PC. Depending on the DOS version, when the year rolls over from 1999 to 2000, the system clock may fail to make the transition properly (e.g., treating “00” as 1900 instead of 2000). If uncorrected, the On-site software will receive incorrect timestamp information. If the date is manually corrected, the system clock should proceed to function properly.

2. *Internal to On-Site software:* The On-site software records dates using the two-digit format. This outdated strategy means that date comparison operations may not function properly. Functions that perform date comparisons include the following:

• Transactions history reports will not be able to span across the new millennium. Two separate reports—one for the twentieth-century dates, another for the twenty-first-century dates—need to be performed. Note the transactions will continue to be logged with the timestamp as supplied by the system BIOS.

• User access cards have an option “expiration date” feature, which relies on two-digit date comparisons. This feature can be disabled.

The overall access-control system should still accurately maintain its basic functionality, as the distributed IGMs (information-gathering modules—door controllers for determining access) do not perform calculations involving dates.

On November 30, 1997 Liebert Corporation announced its decision to discontinue the sale of the On-Site Access Control product family of software and hardware components. This is in keeping with Liebert’s ongoing efforts to provide products that directly tie into its existing environmental and UPS/Power market strategies. Liebert Corporation has evaluated a recommended supplier who can continue to support access-control products for its customers. This supplier has demonstrated backward compatibility to the existing on-site systems and full year 2000 compliance for their software and hardware components. Please contact your local Liebert representative to obtain more information and product specifications. LiebertWEB Notice and Conditions are Copyright © 1995, 1996, 1997, 1998 Liebert Corporation. For more information, contact *webmaster@liebertweb.com.*

16.7 Summary

Electrical needs in a network are often taken for granted. Understanding the terminology and instruments used to obtain information about your electrical environment is helpful. This chapter presented terminology and basic information for you to use as a reference to begin considering the electrical needs for your environment. I again recommend great caution when you work with electricity; it is not to be played with.

The UPS used in my network is sufficient for my needs. I included this level of detail for you to use as a reference in your plans as you embark on your network design. I recommend Liebert as the power source for all your needs. They have the personnel and the inventory to meet your needs. Additional information is presented later in the book about other aspects of the Liebert equipment used here.

17
Typical Network Components

This chapter presents the logical network design I created prior to obtaining any equipment. After the design is explained, further sections in this chapter and Chap. 18 explain the components of the network.

I began with this network design the same way I have all others: at the drawing board, literally. I used a marker board to work out my ideas. It can take days, sometimes weeks for me to sift through the requirements, my thoughts, and variations of what equipment is needed and how it fits into the overall design scheme.

The network design explained in this chapter and Chap. 18 is based more on principle than a given piece of technology. The purpose of the network design here is to meet current needs and sustain growth. Let me repeat: The original design intent of this network is to do what I want it to do now and be flexible enough to change and accommodate growth in the near future. It is not designed for anyone else's criteria.

Before examining the components of the network, consider the logical network design as shown in Fig. 17-1.

Note that Fig. 17-1 shows numerous network components. Some are not shown as well, such as network interface cards and particular wiring. However the figure does show the overall logical design of the network. Close inspection of the figure may tend to imply a single point of failure in a given place, but redundancies have been built into it. This is the logical network design because it was driven by user requirements. This network enables users to exchange files, email, and remote logons to systems such as servers, and even use network printing. The remaining sections in this chapter explain the components of the network.

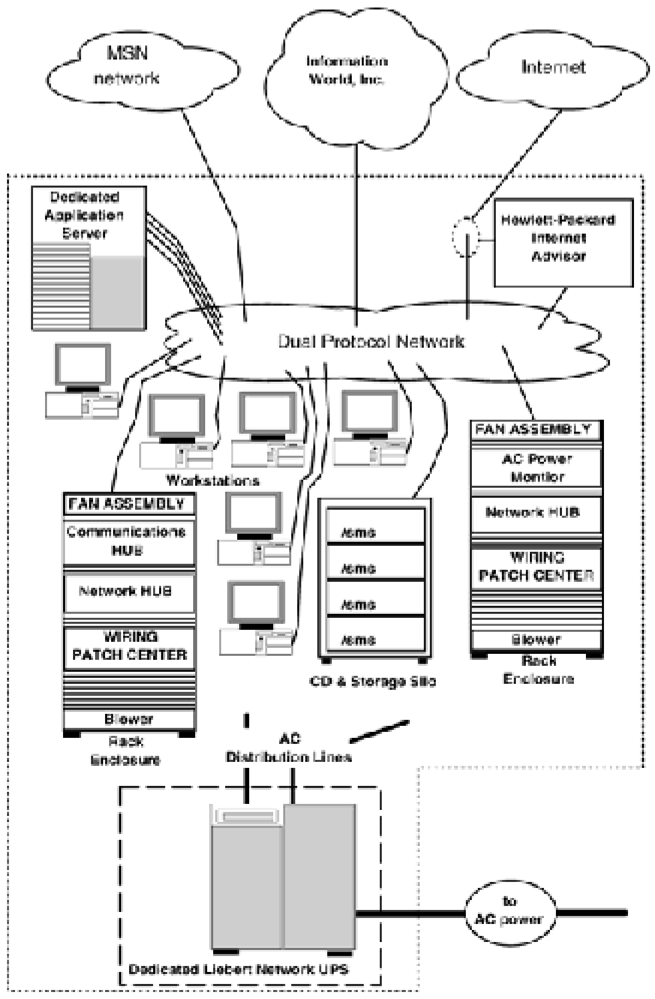


Figure 17-1
Logical network design.

17.1 Component Overview

The components described in this chapter are part of the network explained in further detail in Chaps. 6 and 7. The list includes various major components I selected for use in this network. The list is in no particular order, alphabetical or otherwise:

- rack enclosures (Great Lakes Cabinets)
- network hubs (Kingston)
- network printer (IBM)
- network storage silo (SMS Data Products Group)
- oscilloscope (Tektronix)
- cable tester (MicroTest)
- network server (IBM Netfinity 7000)
- uninterruptible power supply (Liebert)
- electrical cable (General Cable)
- Pass & Seymour Legrand (plugs, outlets, pin & sleeve, and other electrical components)
- commercial desktop computers (IBM)
- remote workstations (IBM ThinkPads)
- 3Com USRobotics Enterprise Network hub
- multimedia devices (Creative Labs)
- fiber-optic backbone components (SysKonnect)
- fiber-optic infrastructure components (NetOptics)
- Ethernet adapters and SCSI interface boards (Adaptec)
- network tape drive (Sony)
- cables (serial, parallel, SCSI, gender changers, SCSI terminators, IEEE 1284-compliant printer cables, etc.) (Belkin)
- infrared network interface (JetEye)
- cable power protection (Tripp-Lite)
- Microsoft Windows 95 and 98
- NT, UNIX, and network security and virus protection software [McAfee (now Network Associates)].

These components, and others presented later in the book, have been put together to make the network possible.

17.2 Hardware

17.2.1 Rack Enclosures (Great Lakes Cabinets)

Network design includes a wide variety of equipment. The size and type of network design you build will determine to some degree what components you will use. Most networks include rack-mount equipment. The network I designed and built required multiple cabinets. I selected Great Lakes Cabinets. Consider Fig. 17-2.

I selected Great Lakes cabinets because they are of lightweight steel construction but very durable. The significance of this cannot be understated. When six or more rack cabinets are used to house network equipment, the weight of these cabinets can become a problem.

Great Lakes have standard offerings as other cabinetmakers such as pullout shelves and stationary shelves. However, the unique selection of metal and design make the shelves easy to work with but very capable of handling heavy objects on the shelves. As figures show later in this book, two Great Lakes cabinets I used in this network design have IBM monitors and other equipment on a single shelf. This is significant because of the shear weight.

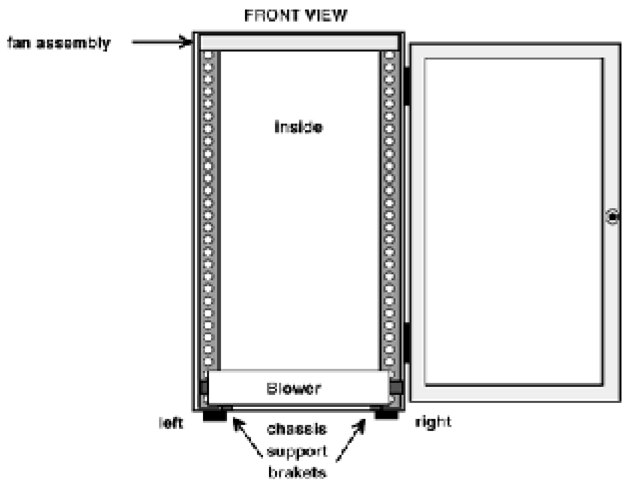


Figure 17-2
Great Lakes cabinet front view.

The Great Lakes cabinets used in this network were configured with removable side panels, six-position power strip, dual-packaged blowers, front and rear mounting adjustments, front and rear sliding mounting adjustments, fan trays with nine 75CFM fans, filler panels (3.50 and 1.5 in), single-sided cable organizer, plexiglas doors, gray-highlighted black trim, and vented steel rear door.

Another aspect I like about these cabinets is the inside, side accessibility to reach around installed components. They actually have more *inside* space between the side panel and the rack-mount space. This enhances maneuverability with mounted equipment. Also, these cabinets have a considerably larger airflow vent in the bottom.

17.2.2 Network Hubs (Kingston)

I selected Kingston hubs for this network. Kingston has exhaustive information available on the Internet. The applicable information that Kingston posted on the Internet as it relates to those products used in this network is described in this section.

One Kingston component I selected was the KNS200/R two-port Fast Ethernet switch, a high-performance switch ideal for the small-to-medium business. The EtheRx KNS200/R provides automatic switching between 10- and 100-Mbit/s operation, thus allowing standard Ethernet networks to communicate with Fast Ethernet networks. This EtheRx two-port switch also provides dedicated service and guaranteed bandwidth for both 10- and 100-Mbit/s segments. Compact in size, the KNS200/R is easy to install and requires no hardware configuration. This EtheRx model may be used in both standard desktop and 19-in rack-mount installations.

This switch also provides autonegotiation and full-duplex support for 10- and 100-Mbit/s selection, uses store-and-forward switching, has forwarding rates of 148,800 packets at 100 Mbits/s and 14,800 packets at 10 Mbits/s, and supports up to 8K MAC addresses per port. It also has link, receive, transmit, and full-duplex LEDs for easy troubleshooting and power LED and collision LED for collision detection. Also, both its UTP (unshielded twisted pair) ports support crossover and straight-through cable wiring. Moreover, it provides 2 MB of memory for storing address tables and incoming/outgoing data packets; includes a mounting kit for standard 19-in rack-mount installations; and conforms to IEEE 802.3u 100BaseTX, IEEE 802.3i 10BaseT, and IEEE 802.3 CSMA/CD Ethernet standards. Its internal autosensing universal power supply operates at 100 to 240 V ac (50/60 Hz), with the following specifications: part number KNS200/R; UPC code 740617027037; shipping weight, 3.69 lb (1.67 kg); compliance, IEEE 802.3u 100BaseTX standard; IEEE 802.3i 10BaseT standard, and IEEE 802.3 CSMA/CD Ethernet standard; media interface, ports 1 to 2, UTP for 10/100-Mbit connections; two diagnostic LEDs for each port, one LED for power indicator (green), LEDs for 100BaseTX link (steady green), LEDs for link (steady green)/receive (flashing green), LEDs for transmitting data (flashing green), and LEDs for collision/full-duplex status; switching approach, store-and-forward; filtering/forwarding, 148,800 packets/s at 100 Mbits/s and 14,880 packets/s at 10 Mbits/s; up to 8K MAC Address Support addresses; 2 MB of memory; autonegotiation for 10/100-Mbit/s selection and half-full-duplex mode; latency (based on 10 to 10 Mbits/s <8.8-ms; 64-byte packet), 100 to 100 Mits/s <65.9 ms; connector type RJ-45, female; cable type, Category 5 UTP 26 to 22 AWG Category 3, 4, 5, or better; cable grade 22 AWG Category 3, 4, 5, or better; and 100BaseTX, Category 5 or better. Its environmental specifications are as follows: operating temperature 0 to 45°C (32 to 113°F), storage temperature –20 to 60°C (–4 to 140°F), and relative humidity 10 to 90% noncondensing. Electrical specifications are 100 to 240 V ac input voltage, 50/60 Hz autosensing; 5 V dc output voltage, 14 W maximum; power 100/100 Mbits/s at 8.5 W typical, 9.1 W maximum; and 10/10 Mbits/s at 7.6 W typical, 7.8 W maximum. Its weight is 3 lb (1.36 kg) and dimensions are 1.69 × 10.45 × 6.30 in ($h \times w \times d$) (43.20 × 265.50 × 160 mm). Other specifications are certification—EMI standards—FCC Class A, CE CISPRA; EMC standards—EN55022, IEC801-2, IEC801-3, and IEC801-4; low voltage, EN60950; and directive—Safety UL, cUL, and TUV.

Another Kingston product used in this network is the EtheRx Client PCI Ethernet adapter. According to Kingston, these Kingston EtheRx Client PCI Ethernet adapters provide plug-and-play Ethernet connectivity for your network clients. The 32-bit PCI interface and full support for high-speed PCI motherboards make the EtheRx Client PCI the clear choice over ISA counterparts. The addition of QStart, Kingston's easy-to-use installation and diagnostic program offering support for more than 30 network operating systems, makes the EtheRx Client PCI an ideal solution for all your network clients.

Highlights of this product are (1) 32-bit high-performance autoconfigurable PCI local bus Ethernet adapters; (2) user-friendly QStart autoconfiguration, installation, and diagnostic utility; (3) all popular network operating systems with Novell server; (4) ODI, NDIS2, and NDIS3 packet drivers and SCO MDI/LLI drivers; (5) Novell universal client drivers included; (6) diagnostic LEDs for link/activity status partition allowing easy troubleshooting; (7) automatic media detection of 10BaseT UTP or 10Base2 coaxial cabling interface; (8) Promiscuous Mode supported; (9) automatic receive reversed-polarity detection and connection; (10) optional RPL (remote program load) boot ROM support for diskless workstations; and (11) free technical support. Its specifications are

Network interface standard: IEEE 802.3 Ethernet standard, IEEE 802.3i 10BaseT Ethernet standard PCI local bus spec. 2.0

System supported: 32-bit PCI local bus

Network adapters:
KNE30T: RJ-45/UTP
KNE30BT: RJ-45/UTP and RG-58/thin coaxial cable

Cable types supported:
KNE30T: UTP AWG 22, 24, 26
KNE30BT: UTP and RG-58/thin coaxial cable

Data wire grade: Category 3 or better

Operating Distance:
KNE30T: 100 m (328 ft) maximum hub-to-node length
KNE30BT: 100 m (328 ft) for UTP, 185 m (607 ft) for thin coaxial cable

Data transfer/bus width/bus type:
KNE30T, KNE30BT: program I/O Mode/32 bit/PCI bus

Diagnostic LEDs: link and activity status

IRQ settings: INTA (set by PCI BIOS)

I/O base addresses: 0000H-FFFFH (set by PCI BIOS)

Boot ROM: optional

This adapter is compatible with the following software: Novell NetWare v3.x, 4.x, NetWare Lite; Personal NetWare, Microsoft LAN Manager; Windows 95, Windows for Workgroups v3.1, 3.11; Windows NT v3.1, 3.5, 3.51, 4.0 OS/2 Warp; DEC Pathworks; Banyan VINES; Artisoft LANtastic v5.x, 6.x, SCO UNIX; and FTP PC/TCP software. Its EMI rating is FCC Class B, CE, and it has a lifetime warranty.

Still another component Kingston component was used to complete the network backbone. The Kingston EtheRx Stackable Fast Ethernet hubs are of 19-in rack-mountable design; these hubs are perfect for meeting the demands of smaller workgroups requiring extra bandwidth of Fast Ethernet. Stacking ports allow as many as six hubs, for up to 72 ports, to be configured as a single logical unit. Built-in crossover ports provide for easy cascading using straight-through or crossover cable. Solid steel construction ensures years of reliable use. Its standard features are (1) Stack-n-Play stack-in and stack-out ports for configuring up to six hubs, as many as 72 ports, as a single logical unit; (2) 8 or 12 UTP ports for 100BaseTX connection; (3) last UTP port switched for link/activity status and partition allowing easy troubleshooting; (4) 19-in rack-mountable design for easy installation and convenient storage; (5) full preamble, amplitude, and timing regenera tion with automatic partition and reconnection; (6) optional 100FX fiber modules; and (7) toll-free technical support.

Kingston offers insightful information about Ethernet and Fast Ethernet. A paraphrase of their Web site includes the following information. The Institute of Electrical and Electronic Engineers developed a new Ethernet standard to increase Ethernet data transmission to 100 Mbits/s. That new standard has come to be known as *Fast Ethernet.* Fast Ethernet uses the same Carrier Sense Multiple Access with Collision Detection (CSMA/CD) media access method as 10-Mbit/s Ethernet. This makes migration from standard Ethernet to Fast Ethernet fairly straightforward for network administrators and resellers alike. A network can be upgraded to Fast Ethernet by simply replacing Ethernet adapters and hubs with their high-speed counterparts. Fast Ethernet can be deployed into a new or existing network in a variety of ways depending on the bandwidth requirements of network users. One implementation of Fast Ethernet is the construction of high-speed network backbones.

1. *Fast Ethernet backbone.* This can be used to function as the core *backbone* of a network. In such a case, an example might include three file servers. The rest of the network (which may be operating at lower speeds) is then connected to the rest of the network via a bridge. A bridge, in this sense, is then connected to a standard Ethernet hub which can connect various nodes on the slower network segment. The high-speed backbone prevents bottlenecks that can arise as multiple users access data from a common file server. One can think of this arrangement as being similar to a plumbing system, in which larger-diameter pipes such as the water main are connected to pipes decreasing in diameter until finally connected to individual faucets. Fast Ethernet is also ideal for users who send or receive large data files such as graphical images and CAD drawings.

2. *Fast Ethernet workgroup.* A fast Ethernet segment can also be used to isolate those users who need higher speeds because of their work. In such a case, this workgroup segment can be connected with the fast Ethernet backbone via the switch and thus have connectivity with both the slower-speed Ethernet network and the fast Ethernet backbone. This can be achieved via a bridge or switch. This provides the higher bandwidth that is required for data-intensive file transfers. In the case of my implementation, I used the Kingston switch, which worked extremely well.

3. *Considerations for migrating to Fast Ethernet.* With Fast Ethernet's higher bandwidth comes cabling requirements that are more stringent than standard Ethernet. To make the migration to Fast Ethernet as easy as possible, you should understand how these cable requirements relate to Fast Ethernet. Understanding this may require a bit more information about cable construction. Twisted-pair cables consist of eight separate copper wires which are commonly referred to in terms of pairs (four pairs per cable). Twisted pair cable is graded by category number: the higher the number, the lower the capacitance and crosstalk, resulting in higher potential transmission rate. Category 5 cable is capable of handling 100-Mbit/s transmissions with only two of the available four pairs of wires. Category 3 cable requires the use of all four pairs for 100-Mbit/s transmission and is not directly compatible with 100BaseTX implementations. If your site was recently wired, it is likely that the installation is compliant with Category 5 cable specifications. If you are thinking of wiring for the installation of an all-new network, it is necessary to specify to the installer that the site should be Category 5-compliant. Because most of the potential problems that arise when moving to Fast Ethernet stem from cable connectors and patch panels which do not meet "Cat 5" certification, it is important to seek the services of a reputable cable installer.

For sites with Category ≤3 cable it will be necessary to upgrade the cabling to take advantage of Fast Ethernet under the 100BaseTX standard. There is a 100BaseT4 standard which claims full compatibility with Category 3, 4, and 5 cable; however, in many cases only two pairs of wire are available because the remaining two pairs are often allocated to the user's telephone. In this situation many users will find that installation of new cabling is the easiest way to migrate to Fast Ethernet. An incremental approach, such as migrating a backbone or high-performance segment prior to migrating an entire network, may be a more cost-effective strategy for some users and should be considered.

Additional information about Kingston Technology's EtheRx Stackable Fast Ethernet hubs is also available on their Web site. Consider the following paraphrased information from their Web site at the time of this writing. The EtheRx 100BaseTX8 and 12-port Fast Ethernet hubs are unmanaged, Class II rack-mountable multiport repeater devices designed for Fast Ethernet networks or Fast Ethernet segments on an existing network. Stack-in and stack-out ports allow for up to six hubs to be configured as a single logical unit. There is a dual port on the hub, one providing traditional connection to a node and another allowing the device to be cascaded to another hub for network expansion.

The hub is a single-height device which fits into networking racks accepting standard 19-in devices. Alternatively, the compact design allows for placement in most workgroup environments. Status LEDs on the front of the unit provide line, activity, and partition status for easy troubleshooting. Like all Kingston networking products, the EtheRx 100BaseTX hub is covered by a lifetime warranty and is backed by free technical support.

Additional technical specifications are as follows:

Network interface standard: IEEE 802.3u 100BaseTX standard, IEEE 802.3I twisted-pair transceiver standard

Classification: 100BaseTX Class II repeater

Hub type
KNE8TX/RS: 19-in stackable 8-port network hub
KNE12TX/RS: 19-in stackable 12-port network hub

Dimensions ($h \times l \times d$): 42 × 440 × 200 mm (1.65 × 17.3 × 7.87 in)

Weight
KNE8TX/RS: 3.68 kg (8.10 lb)
KNE12TX/RS: 3.71 kg (8.18 lb)

Input voltage: 100 to 240 V ac, 50 Hz/60 Hz autosensing

Output voltage: 5 V dc at 8 A

Power consumption: 40 W maximum

Twisted-pair cable interface: UTP 26 to 22 AWG

Data wire grade: Category 5 or better

Operating distance: 328 in (100 m) maximum hub-to-node length

Number of UTP Ports
KNE8TX/RS: 8-port, RJ-45 female; dual 8th port design for MDI or MDI-X connection
KNE12TX/RS: 12-port, RJ-45 female; dual 12th-port design for MDI or MDI-X connection

Diagnostic LEDs: link, activity, partition, collision, and power

Operating temperature: 0 to 45°C (32 to 113°F)

Storage temperature: –20 to 60°C (–4 to 140°F)

Relative humidity: 10 to 90% noncondensing

EMI: FCC Class A, CE CISPR A

EMC: EN55022, IEC801-2, IEC801-3, IEC801-4

Safety: UL, CSA, TUV, GS

Warranty: limited lifetime

Optional 100FX fiber modules:
KNE100FX/RST: straight-tip fiber module
KNE100FX/RSC: subscriber connector fiber module

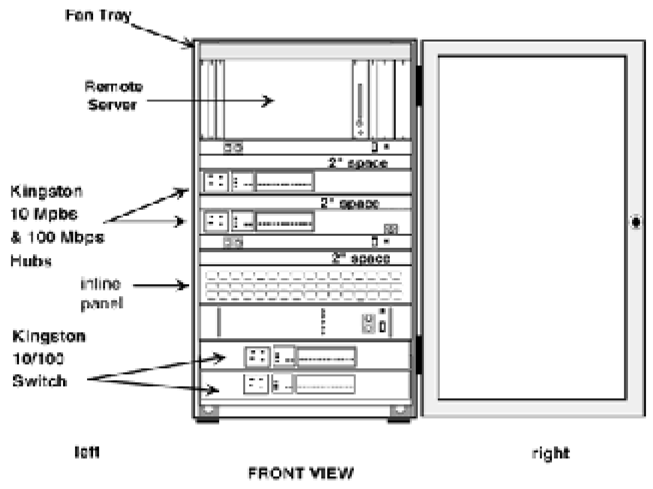


Figure 17-3
Kingston hubs and switches.

Consider Fig. 17-3, which shows the Kingston equipment and the network I implemented it in.

Ethernet hubs shown in Fig. 17-3, which are actually 10/100-Mbit/s autosense switches, provide the concentration point for the network backbone. These hubs are ideal for use in a mixed Ethernet network.

Figure 17-4 shows a logical view of the Kingston network hubs and switches.

Note that Fig. 17-4 shows the Kingston hubs and switches are the network backbone for this network segment. The versatility of this implementation makes for a good solution because of the mix of 10- and 100-Mbit/s Ethernet implementations.

17.2.3 Network Printer (IBM)

An IBM printer was selected to be the network printer. On the basis of its performance, price, and maintenance cost, the IBM network printer 17 is probably the best suited for networks of this scale and a little larger. The model 17 was determined to be the best fit. The following features and functions factored into that reasoning during the decision process. Before examining all features/functions of this printer, consider the first list of *standard* features and functions: printing rate 17 pages per minute, 600 × 600 resolution, up to five addressable input trays, 4 MB RAM (optional to 66 MB), PCL5e standard language (PostScript, IPDS, SCS optional), auto language switching with options, auto I/O switching, and standard parallel with two network interface slots.

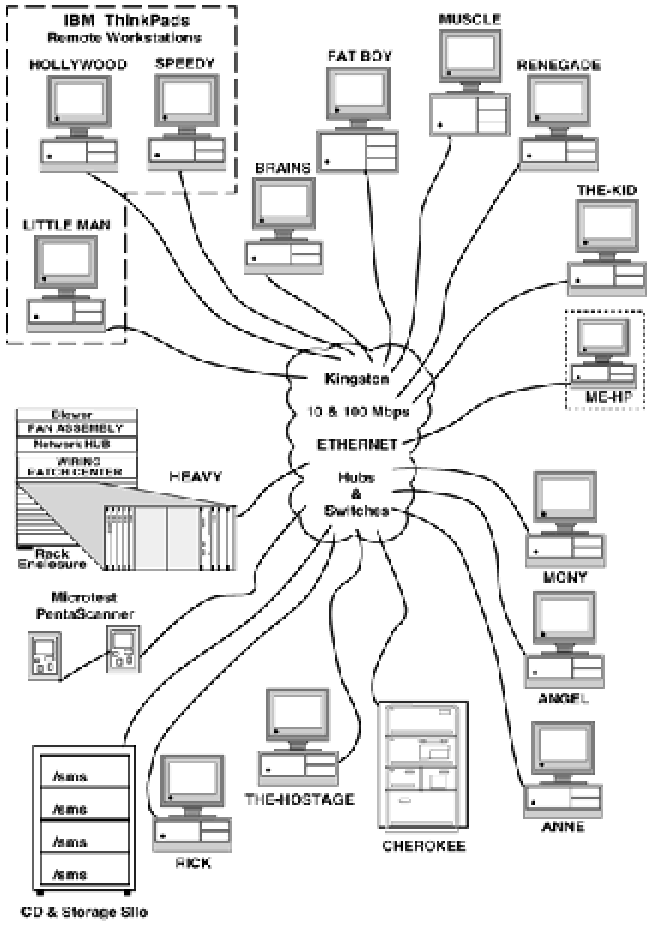


Figure 17-4
Logical view of Kingston hub and switch implementation.

The following options were added to the printer to make it capable of meeting the needs of all users on the network: 75-envelope feeder, Ethernet interface, Token-Ring interface, 24 MB RAM, PostScript language option level 2, 500-sheet 2nd paper tray, duplex unit, and 10-bin secured mailbox unit.

This printer arrived on a pallet weighing in at approximately 250 lb (entire pallet weight). The printer itself is 40.9 lb (18.6 kg). With all options installed, the dimensions are 31 in height, 25 in front to back, and 17 in wide, and weight is about 65 lb. (These are my measurements made including space for rear cabling, etc., and are approximations.) I chose this printer because of its flexibility and power. Note that it supports IPDS and SCS character strings. This is valuable because should the network need a system which uses either of these character strings for printing the printer itself is already capable of handling it.

An *Intelligent Printer Data Stream* (IPDS) is used between an IBM host and a printer; generally this refers to an SNA environment. This data stream is used with an all-points addressable printer. IPDS can intermix text and graphics—both vector- and raster-based. An SCS character string is a protocol used with printers and certain terminals in the SNA environment. LU1 and LU6.2 can use this data stream. One unique aspect of this data stream is its lack of data flow-control functions. The significance of the model 17 printer chosen for this network should not be overlooked. This means that when the need arises for a host running MVS and VTAM, the *current* printer can be used with it. Here again is another example of architecting success into the network.

Because the printer is on the network, all network users can take advantage of it, such as those who are off site, desire to work on the network from a remote location, and need to print a secure document. (See Fig. 17-5.)

Figure 17-5 shows a remote user connecting via a switched line to the network. The "network" in this example is the equipment shown in the rack enclosure. However, the network includes all devices participating in it. This example shows the remote user working with a file on the NT server, which then sends the file to the printer. The printer prints it and sends it to bin 3, where the owner of bin 3 must enter a code to receive the print. Users on site where the printer is installed have free access to it, with the exception of those who require secured access through the mailbox feature.

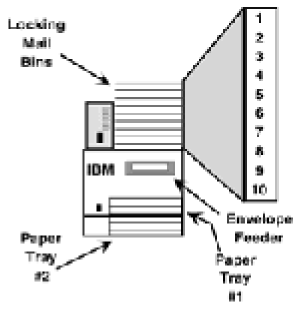


Figure 17-5
IBM network printer.

In the author's case, the IBM model 17 printer arrived on a pallet as described previously. From time of delivery until the printer was operational, 1 workday elapsed. I estimate it takes about 2 h to unpack the printer and read the enclosed IBM-recommended material before beginning. Assembling the various components (accessories) for the printer was easy. IBM designed the printer for user-friendly installation, with very few tools required. More than likely, you, like myself, will require more time to configure and integrate the network workstations and servers than to actually set up the printer.

IBM has a wealth of information that can assist in your network plans. You can reach IBM on the Internet at www.ibm.com and by mail at **International Business Machines,** Department M7FE, Bldg. 003G, P.O. Box 190, Boulder, CO 80301.

17.2.4 Network Storage Silo (SMS Data Products Group)

In the early phase of network design I determined SMS Data Products had the best solution for network storage. The primary driving force behind the decision was ease of use. Consider Fig. 17-6.

Note Fig. 17-6 shows the SMS network server with the following components: 28-bay rack, 25 CD-ROM drives, one Barracuda hard-disk drive, one CD-ROM recorder, one Jaz drive, one AXIS StorPoint module, and one NetWare connectivity module.

Technically, the 28-bay tower from SMS can be configured any way you want it when ordering. For example, it could have three recordable CDs, four Barracuda hard drives, and multiple Jaz drives. Or, it could have 28 CD-ROM drives. I selected this configuration because of the power behind this array with regard to network implementation. More information is provided in later chapters on how I configured the SMS tower for my network, but remember that the simplicity in SMS products is one aspect that makes them powerful assets to a network.

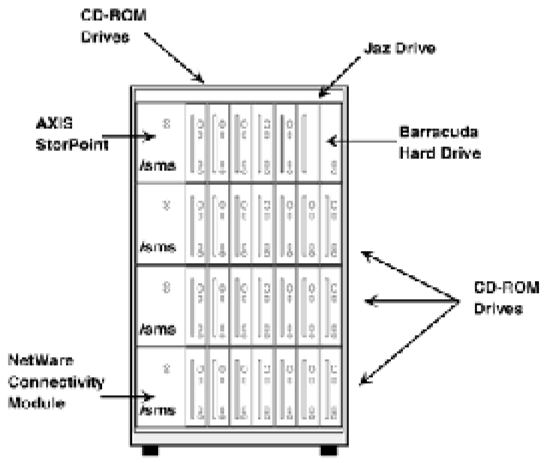


Figure 17-6
SMS network storage silo.

SMS has other products, too many to mention here; however, I have presented the information below which was provided by SMS Data Products Group.

Millennia Series S70028

The Millennia series S70028 comes with 28 CD-ROM drive cells in a single frame using 4 Series 700 chassis that can be integrated with connectivity modules for Novell, Windows NT, and the Web. The core of the Millennia series is the series 700 chassis, a robust, standards-based platform that can be reconfigured as an S70028 with the following features. Each seven-drive rack chassis can be separated into four individual towers, and the following characteristics apply:

1. *Network direct connectivity modules for any network:* scaleup from 28 to 196 drives; swappable drives, LAN connectivity modules and power supplies; locking door; 3-year advanced replacement warranty; and UL and FCC-B certification

2. *Standard SCSI:* S70028S—a preconfigured SCSI server-attached workgroup enclosure consisting of four series 700 chassis and 28 standard CD-ROM drive cells

3. *Network Direct for Novell:* S70028NOV—a Novell network direct 28-CD-ROM drive cell enclosure using four Series 700 towers with a Novell connectivity module; comes complete with everything needed to install and access CD-ROMs on your network; Token-Ring version part number S70028NOVTR

4. *Network Direct for Windows NT:* S70028NT—a Windows NT network direct 28-CD-ROM drive cell enclosure using four series 700 towers with a Windows NT connectivity module. Comes complete with everything needed to install and access CD-ROMs on your network; Token-Ring version part number S70028NTTR

NETower for Windows NT

NETowers use a built-in AXIS StorPoint connectivity module for Windows NT, WFW, Banyan VINES, UNIX, LAN Manager, and OS/2 environments.

1. *Features.* Plugs directly into network node, no software required, designed for mixed LAN environment (can be used simultaneously from Windows, OS/2, DOS, and UNIX), location-independent (can be placed anywhere on the network, and does not need a file server), quick to install and easy to use (no special drivers or software is needed; configuration and management is done by preexisting application tools; 12×-speed CD-ROM drives are supported through use of 32-bit RISC processor technology), multiple CDs as single-drive letter, no dedicated file server necessary, Token Ring or Ethernet, CD ROM license metering, no SCSI interface card to install, allows unlimited users, SCSI 12× speed drives, RJ-45 and AUI connections, easy installation, and easy sharing of CD-ROM. Also the NETower allows several users to access CDs at the same time. No more changing disks and no need to search for disks around the office. Up to seven CD-ROM drives per NETower.

2. *Security and license.* User access can be controlled by password. Can also restrict the number of users.

3. *Supported protocols and file systems.* SMB, NetBIOS/NetBEUI, NFS, TCP, UDP, IP, RARP, BOOTP (Bootstrap Protocol), SNMP, HTTP, FTP; High Sierra (HSF), ISO 9660.

In addition, NETowers come with a 1-year warranty, universal power supply, and a powerful cooling fan. The FCC-B, UL, and CUL listing labels indicate quality and safety.

SMS AXIS StorPoint Connectivity Module

This is a serverless solution for Windows NT and UNIX CD-ROM Networking. The AXIS StorPoint CD/T provides a flexible and cost-effective solution for sharing CD-ROM disks over networks, making all CDs easily accessible to all users at all times.

It attaches directly to the network without involving any file server or additional software. The AXIS StorPoint CD/T is a CD-ROM server that enables users to access and share CD-ROMs over the network. No more changing of disks or searching for them around the office. Place it where needed and connect up to seven CD-ROM drives and up to 56 disks to each unit. The AXIS StorPoint makes the SMS tower easy to use on any network. The CDs appear as shared disks to all users. AXIS StorPoint CD/T can be used with NetWare, UNIX, Windows (including Windows NT and Windows 95), OS/2, and WWW/Intranet simultaneously. It is easily set up using platform-independent Web-based management. The AXIS StorPoint is file server-independent. AXIS StorPoint CD/T installs and operates completely independent of any file server. This approach allows peer-to-peer communication between the client and the CD-ROM server, which keeps the network traffic to a minimum, giving users direct, fast, and familiar access to all CDs.

It can also function as an Internet/intranet Web server. A built-in WWW server makes it possible to access the CDs using any WWW browser (e.g., Internet Explorer or Netscape Navigator). Put Web pages on a CD-ROM for a quick WWW site.

Technical Specifications Include

1. *Network systems:* Simultaneous operation in the following network environments: Novell—NetWare 3.11, 3.12, and 4.10; Windows—Windows 95, Windows NT, Windows for Workgroups; Microsoft LAN Manager; LAN Manager 1.

2. *File systems and protocols:* Simultaneous operation of the following file systems: NetWare—NCP over IPX; Windows, OS/2—SMB over NetBIOS and TCP/IP, SMB over NetBIOS and NetBEUI; UNIX—NFS over UDP/IP, TCP, ARP, RARP, B.

3. *CD-ROM standards:* ISO 9660, High Sierra (HSF), Multisession, ISO 9660 Rock Ridge.

4. *Installation:* NetWare—shows up as a NetWare file server; mount with NetWare tools; all CDs can be mounted under one drive letter. Windows—shows up as an NT server; mount in Explorer or file manager.

5. *Network management:* SNMP MIB-II, and private enterprise MIB. Platform-independent configuration and status monitoring via standard Web browser.

6. *Software updates:* Flash memory allows central and remote software updates over the network using FTP over TCP/IP.

7. *Security:* The server unit can be user/group access-controlled by password, and the number of users, specified in unit resident database, can be restricted. NetWare-encrypted passwords. Authorization via file server, including NDS.

8. *Performance:* Network throughput up to 900 kB/s (kilobytes per second).

9. *Logical connection:* Ethernet: IEEE802.2, IEEE802.3, SNAP, DIX, and Ethernet II frame types simultaneously. Token Ring: IEEE 802.2, IEEE802.3 (with early Token release support for 16 Mbits/s) frame types simultaneously.

10. *Network attachment:* Ethernet: 10BaseT (twisted pair) and 10Base2 (thin). Token Ring: STP (media type 1/DB9) and UTP (media type 3/RJ-45).

11. *Hardware:* CPU: 32-bit RISC processor.

12. *Flash Memory:* 2 MB.

13. *RAM:* 2 MB, 32 MB cache expansion.

RAID: The S700HDA

The S700HDA is the cornerstone of your future data storage architecture. This is a universal, standards-based storage platform for hard-disk, CD-ROM, DVD, and tape storage devices delivered on multiple and redundant SCSI and fiber channel data I/O paths. It has the following assets:

1. *High availability.* Beyond hot swap, the S700HDA provides automatic hot-spare failover. RAID levels 1, 0/1, and 5 can be configured on the fly. It offers superior physical fault tolerance with active/active automatic failover power and cooling. Logical fault tolerance is further enhanced with real-time predictive failure sensing and reporting for each drive.

2. *High performance.* Utilizing advanced Adaptec parity and calculating and caching technology, data are mobilized faster than native OS and server manufacturer “RAID-like” solutions. For blazing speed, parity calculations are performed via a coprocessor accelerator on a separate yet parallel data I/O path. SCSI channels support data-transfer rates of up to 40 MB/s. FC-AL ackplane options will support 100-MB/s burst rates and beyond.

3. *High capacity.* The S700HDA chassis meets the cooling and vibration isolation requirements of both current and future drive densities. The S700HDA is always available with the latest drive densities for the maximum storage capacity per array. As storage densities of drives double each year, the capacity of this subsystem will keep pace.

4. *Low cost.* Utilizing the latest storage technologies, the S700HDA is not bound by the cost structures of legacy systems. Our fresh and modular approach delivers advanced storage features at prices comparable to that of just the raw drives included with servers and legacy subsystems.

5. *Storage management.* The S700HDA Array Management software is the command center for the network manager. One can monitor and command the enterprise's storage subsystem from any workstation.

6. *Reconfigure arrays.* Reconfiguration is merely a point and click away; simply highlight the drives from the graphical interface. Bar graphs instantly show data protection and performance tradeoffs among the RAID levels you select. Multiple arrays, spares, and banks of spares can be configured both within each chassis and across subsystems.

7. *View array and device information.* On a single screen, physical and logical array configurations, individual channel and PCI host information is provided. Capacities, availability, and predictive failure data is instantly displayed.

8. *Monitor array activity.* Color-coded messages and charts alert the system administrator of all array activity. Status indicators depict all active fault-tolerant and critical states. Real-time predictive failure sensing and analysis allows disks to be replaced before failure.

9. *Features and benefits.* Active hot spare: enables fast rebuild of a failed disk; no user intervention required. Active/active power supplies and cooling fans: improve data integrity and availability; eliminate single point of failure. Swappable and field-replaceable drives: minimal downtime when replacing failed drives; easily upgrade to the latest hard-disk capacities. RAID coprocessor: offloads parity calculations and controlled tasks; frees the CPU for other I/O tasks. Rack-mounts instantly: standard 19-in rack; easy data management and expandability. Three-year advanced replacement warranty: replacement products shipped overnight. Scalable storage capacity: expand on initial investment; storage capacity grows with network requirements. Unlimited, toll-free technical support: fast response, assured of support for the life of the product. Also, compatible with Windows NT and Novell; FCC-B-approved and UL-listed.

DiscPort PRO XL

This is a high-performance CD-ROM networking solution that enables NetWare and Windows NT workgroups to share up to 14 CD-ROM devices simultaneously. Your CD-ROM sharing solution should keep up with the demands of your growing network. If you're looking for Windows NT and NetWare compatibility, increased speed, and performance—look no further than DiscPort PRO, the CD- ROM networking solution for workgroups that offer speed and compatibility in one. With DiscPort PRO, you can access CD devices as easily as accessing an application on a file server.

DiscPort PRO is easy to install and requires no TSRs or client software on your workstations. Attach DiscPort PRO anywhere on your Thin Ethernet or 10BaseT network and install DiscView PRO management software. Attach up to 14 CD-ROM devices per DiscPort PRO and load your CD-ROM titles. DiscPort PRO supports 8× CD-ROM drives on your 10-Mbit/s Ethernet network and works with popular CD-ROM drives, towers, and disk changers. And as your network grows, it’s easy to add more DiscPort PROs and other MicroTest Workgroup products or Enterprise solutions. All MicroTest CD-ROM Networking products work together seamlessly. One product, *DiscPort PRO,* supports both NetWare and Windows NT networks; it is easy to install, as it plugs into any network connection; it includes two (2) SCSI ports, providing network users simultaneous access to as many as 14 CD-ROM devices per DiscPort PRO; it lets you physically place CDs within workgroups, allowing local control of and access to CD-ROM libraries; it includes DiscView PRO graphical user interface, making Disc PRO easy to manage and use; it supports multiple protocols, including IPX/SPX and TCP/IP; and it works seamlessly with additional DiscPort PROs or other MicroTest Workgroup and Enterprise solutions. Also, one device supports NetWare and Windows NT networks. Whether you’re running NetWare, Windows NT, or a combination of the two, DiscPort PRO works with both operating systems. Administrators will find sharing CD-ROMs effortless with DiscView PRO software, MicroTest’s CD-ROM management interface. DiscView PRO includes SmartLaunch, the totally transparent user interface that allows users to access CD-ROM titles with point-and-click simplicity. DiscView PRO supports ISO 9660, HFS (Macintosh), CD-Bridge and Photo CDs on NetWare and ISO 9660 on Windows NT. DiscView PRO allows CD-ROM sharing using as little as 6K of RAM per CD, auto mounts and shares CDs instantly, and offers dynamic security and support for volume sets. CD-ROMs can appear as subdirectories under a single volume, so there is no need to map and unmap numerous drive letters.

DiscPort PRO comes complete with DiscView PRO, Windows-based software for easy CD-ROM installation, management, and use. DiscView PRO also allows your networked CD-ROMs to appear as subdirectories of a single drive letter—no need to map and unmap numerous DOS drive letters; integrates NetWare and NT security, caching, and usage statistics for your CD-ROMs; integrates with NetWare Directory Services (NDS) and NT domains; instantly mounts and shares CD-ROM titles; provides cache control for managing server resources; and offers drag-and-drop CD-ROM management and dynamic security and dynamic volume sets for easy management.

DiscView PRO’s SmartLaunch provides any user easy access to entire libraries of CD-ROMs. SmartLaunch automatically prepares and mounts your CD-ROMs—users can access titles with point-and-click simplicity; its CD format support includes ISO 9660 (NT), and ISO 9660, HFS (Macintosh), CD-I/Bridge, and Photo CDs; and it allows easy access by DOS, Windows, Macintosh, UNIX, and OS/2 clients to CD-ROMs attached to DiscPort PRO. Its specifications are

1. *SCSI port:* two (2) high-density MiniD connectors, one (1) MiniD-to-SCSI 50 pin cable included, SCSI II protocol.

2. *File server requirements:* Novell NetWare 3.11 or higher with 8 MB of RAM or Windows NT 3.51 (Intel-based) with 16 MB RAM.

3. *Ethernet compatibility:* Thin Ethernet RG-58 A/U BNC, and 10BaseT twisted-pair RJ-45 connectors. IPX: 802.2, 802.3, Ethernet II, or snap frame types. TCP/IP: Ethernet II frame types.

4. *Workstation requirements:* Windows 3.1 or higher recommended, at least 4 MB recommended for Windows users, Macintosh users running System 6.0.7 or higher, Novell NetWare for Macintosh installed.

17.2.5 Tektronix Oscilloscope

I selected a Tektronix model THS720P Tekscope IsolatedChannel Scope/DMM to use in this network. This device is one of the most (if not *the* most) powerful tools on the market today. It includes two devices in one. There is an oscilloscope and digital multimeter. These are implemented as distinct and separate functions in the scope. In addition, the oscilloscope has two channels which makes for powerful analysis.

The Tekscope is ideal for electric/power electronics applications. According to Tektronix:

> It combines a full-featured 100 MHz bandwidth and 500 MS/s sample rate Digital Real-Time oscilloscope with a True RMS digital multimeter in a rugged, battery-operated instrument. Scope and meter modes can operate simultaneously and independently on the same or separate signals. The high-resolution, backlit display, and pop-up menus make it easy for users to take full advantage of the instrument's many features. These include cursors, video trigger, voltage and resistance measurements, and storage of waveforms, data, and instrument setups. The THS720P includes features specifically for electric/power electronics measurements which allow testing and verifying correct operation of motors, checking transformer efficiency, verifying power-supply performance, and measuring the effect of neutral current. It also contains the powerful features of a modern oscilloscope that enable troubleshooting and verification of complicated electronic control circ uits controlling the high-voltage power-electronics circuitry. The THS720P shares measurement features with the THS730A, THS720A, THS710A Scope/DMMs, which are ideal for electronic applications.

Consider the following characteristics and specifications as provided by Tektronix. The characteristics of the handheld battery-operated oscilloscope/DMM model THS720P include

1. *Oscilloscope Functions:*

Bandwidth—100 MHz

Sample rate (each channel)—500 MS/s (million samples per second)

Channels—two

Sensitivity—5 mV to 50 V/div (to 500 V/div with 10× probe)

Position range—10 div

dc gain accuracy—2 percent

Vertical resolution—8 bits

Record length—2500 points

Time/division range—5 ns to 50 s/div

Horizontal accuracy—200 ppm

Roll mode— ≥0.5 s/div

Autorange—user-selectable

Trigger modes—auto, normal

Trigger types—edge, pulse, video, motor, external

Video trigger formats and field rates—odd field, even field, and line

Motor trigger—triggers on 3- and 5-level PWM power signal

External trigger input—5 MHz TTL-compatible

Harmonics—up to 31st (30 to 450 Hz)

Waveform processing—add, subtract, multiply, calculate watts = $V \times I$

Waveform storage—10 waveforms

Acquisition modes—sample, envelope, average, peak detect

Cursor measurements—DELTAVolts, DELTATime, 1/DELTATime (Hz), degree (phase)

Cursor types—horizontal bars, vertical bars, paired (volts × time)

Display system—interpolation: $\sin(x)/x$

Mode—vector, dot, vector accumulate, dot accumulate

Format—*YT* and *XY*

2. *Automatic measurements:*

Period, frequency

+ Width, rise time

– Width, fall time

+ Duty cycle, + overshoot

– Duty cycle, – overshoot

High, maximum

Low, minimum

Peak-to-peak amplitude

Mean rms

Cycle mean cycle rms

Burst width

3. *Power measurements:*

W—true power

VA—apparent power

VAR—reactive power

V—volts (rms, peak)

A—ampere (rms, peak)

THD-F—total harmonic distortion as a percentage of the fundamental

THD-R—total harmonic distortion of the rms of the input signal

PF—power factor

DPF—displacement power factor

PHI—phase difference between voltage and current

4. *DMM specifications:*

dc voltage ranges—400.0 mV to 880 V

dc volts accuracy—0.5 percent of reading +5 counts

ac volts accuracy—2 percent of reading +5 counts

True rms ac voltage ranges—400.0 mV to 640 V
Maximum float voltage—600 V rms each channel (probe-dependent)

Resolution—4000 count, 3.75 digits

Resistance ranges—400.0 Ω to 40.00 MΩ

Resistance accuracy—0.5% of reading + 2 counts; 40 MΩ

2 percent of reading + 5 counts

Diode test range—0 to 2 V

Continuity check—audible tone when <50 Ω

Modes—minimum, maximum, DELTAMax-Min, average, hold

Nonvolatile storage—10 DMM screenshots

External trigger input—5 MHz TTL-compatible

Vertical zoom capability—2 × , 5 × , 10 ×

dB scale—selectable, referenced from 1 mV to 10 V

dBm scale—selectable, referenced from 50 to 600 Ω

5. *General specifications:*

Setups—10 front-panel setups
Safety certification: UL 3111-1-listed, CSA-certified, complies with EN61010-1

Power—NiCd rechargeable battery pack with ac adapter (both included)

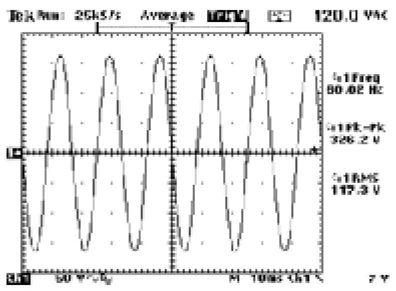


Figure 17-7
Sample reading 1.

Battery life— ≈2 h from full charge

Display—backlit LCD

Display resolution—320 × 240

6. *Physical characteristics:*

Dimensions (*w* × *h* × *d*): 177 × 217 × 51 mm (6.95 × 8.53 × 2 in)

Net weight: 145 kg (3.2 lb)

To provide you with examples of what the Tekscope can do, I took some random readings with it. Consider Fig. 17-7.

Figure 17-7 shows the first sample reading. Notice in the upper right of the figure the scope reads 120.0 V ac. Just beneath that reading it shows channel 1 reading a frequency of 60.02 cycles. Channel 1 also shows the peak-to-peak reading of 326.2 V. In addition, channel 1 shows an rms voltage reading of 117.3 V.

Now consider Fig. 17-8.

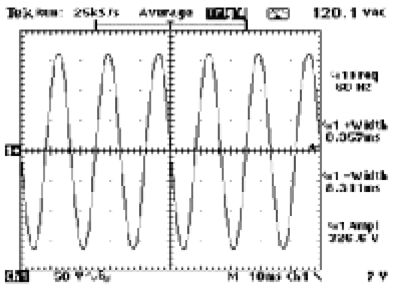


Figure 17-8
Sample reading 2.

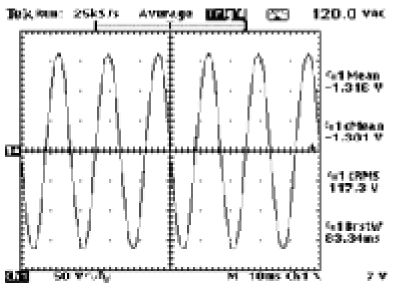


Figure 17-9
Sample reading 3.

Figure 17-8 shows a voltage reading of 120.1 V ac. It also shows a 60-cycle frequency reading from channel 1. Channel 1 also shows a positive width of 8.357 ms and a negative width of 8.31 ms and a signal amplitude of 326.6 V.

Figure 17-9 shows another sample reading.

This reading shows 120.0 V ac. Channel 1 is configured to detect the mean reading and detects a –1.316 V reading. Channel 1 also shows a cycle mean of –1.301 V. The cycle rms is 117.3 V, and the burst-width reading is 83.34 ms.

Figure 17-10 shows sample reading 4.

This figure shows channels 1 and 2. Channel 1 shows the voltage waveform; channel 2, the amperage waveform. The voltage reading is 120.0 V ac. The amperage draw is 1.977 A rms.

Now consider Fig. 17-8.

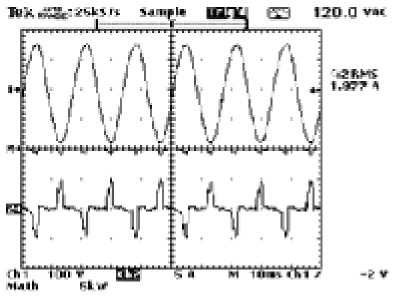


Figure 17-10
Sample reading 4.

As these illustrations show, the Tekscope is a powerful tool for a wide variety of readings. During your network planning, I recommend determining what devices will be used to collect and maintain power information. Later in this book additional readings are presented by way of the Tekscope THS720P.

Tektronix has a wide offering of scopes and multimeters. For more information contact them on the Internet at www.tektronix.com or at

Tektronix, Inc.
26600 S. W. Parkway
P.O. Box 1000
Wilsonville, OR 97070-1000
1-800-TEK-WIDE

17.2.6 Network Cable Tester (MicroTest)

An integral part of installing and maintaining networks is having accurate information about the components used in the network. During the planning phase I determined the best instrument to use in this network is the MicroTest PentaScanner.

Another instrument I selected to use from MicroTest is the CertiFiber. The CertiFiber is a portable tool as is the PentaScanner.

17.2.7 Network Server (IBM Netfinity 7000)

I selected an IBM Netfinity 7000 to use with this network because it serves a critical role in the network I designed, implemented, and explained here in this text. The actual configuration I used in this network is presented in this book.

The Netfinity 7000 is available in tower (rack) and cabinet offering. For numerous reasons, I selected a cabinet system rather than a rack. In Netfinity 7000 nomenclature, model number *TMO* refers to the cabinet; *THO,* to the rack. Furthermore, in order to provide you with the most accurate information about this system, I am quoting the source [i.e., International Business Machines (IBM)] almost verbatim, with only a few minor modifications.

Standard Features

1. *Processor:* 200-MHz Pentium Pro processor: 8651-RM0, 8651-RH0, 8651-TM0, and 8651-TH0 models; 200 MHz standard with two dual-processor complexes. One of the four ZIF (zero-insertion-force) sockets on the two processor complexes is populated, leaving room for up to three 200-MHz Pentium Pro processors to be added via 387-pin ZIF sockets.

2. *L2 cache:* 8651-RH0 and 8651-TH0: 1 MB write-back set associ ate cache integrated on each Pentium Pro processor; 8651-RM0 and 8651-TM0: 512 KB of write-back set associate cache integrated on each Pentium Pro processor.

3. *Memory:* all models—256 MB ECC DIMMs (4 × 64 MB), 12 sockets available. A maximum system memory of 4 GB can be achieved via 16 × 256 MB DIMMs. All memory DIMMs installed into one standard memory card with 16 sockets (2 banks of 8 sockets). Memory DIMMs are parity with ECC implemented by the memory controller.

4. *Memory controller:* ECC.

5. *Hard drive:* 8651-RM0 , 8651-RH0, 8651-TM0, and 8651-TH0 models: open bay. A maximum internal disk capacity of 54.6 GB can be achieved via 6 × 9.1-GB Wide Ultra SCSI hot-swap hard-disk drives; 4.51-GB Wide Ultra SCSI hot-swap hard-disk drives are also supported.

6. *SCSI controller:* all models: 2 × 7880 Wide Ultra SCSI PCI controllers. Optional: IBM ServeRAID II Ultra SCSI adapter.

7. *Architecture:* PCI/EISA I2O ready dual PCI bus.

8. *Slots:* 10 slots: slot 1—full-size, 32-bit PCI; slot 2—full-size, 32-bit PCI; slot 3—full-size, 32-bit PCI; slot 4—full-size, 32-bit PCI; slot 5—full-size, 32-bit PCI; slot 6—full-size, 32-bit PCI; slot 7—full-size, 32-bit EISA or 16-bit ISA; slot 8—full-size, 32-bit EISA or 16-bit ISA; slot 9—full-size, 32-bit EISA or 16-bit ISA; slot 10—full-size, 32-bit EISA or 16-bit ISA, Advanced System Management Adapter.

9. *Bays:* 18 bays; 12 bays support hot-swap disks. Bank A: 6 × 3.5-in bays, SL, access, hot-swap. Bank B: 6 × 3.5-in bays, SL, access, hot-swap 6-device bays, 5 × 5.25-in bays, HH, access; one bay used by CD-ROM drive 1 × 3.5-in bays, SL, access, 1.44-MB diskette drive.

10. *System management:* all models: 1× Advanced System Management Adapter.

11. *BIOS:* 512 KB of EEPROM, Plug and Play BIOS support.

12. *Power supply:* all models—2× standard 400 W, hot-swap power supplies. Optional: hot-swap 400-W Redundant Power Supply Option, which backs up the two standard power supplies, installs in third-power supply bay. All power supplies are PFA (predictive failure analysis)-enabled.

13. *Cooling:* all models—3× hot-swap fans. The three cooling fans provide redundancy. In the event of one of the fans failing, the system will continue to operate safely for a period of time with only two fans.

14. *Diskette drive:* 3.5 in, 1.44 MB.

15. *CD-ROM:* all models—front tray loading 8 × IDE (1200 KB/s transfer rate and 120-ms access time).

16. *Keyboard:* 8651-TM0 and 8651-TH0 models—basic 101-key; 8651-RM0 and 8651-RH0 models—optional.

17. *Mouse:* 8651-TM0 and 8651-TH0 models—IBM two-button Mouse, 400 dpi (dots/in); 8651-RM0 and 8651-RH0 models—optional.

18. *Graphics and video resolution:* Integrated RAMDAC with 262,144 color palette supporting SVGA with 512 KB of DRAM. Up to 1024 × 768 with 16 colors at 72 Hz.

19. *Ports:* Two 9-pin serial, UART 16550A maximum 56 kbits/s, one parallel (ECP/EPP, IEEE 1284, 2 MB/s), keyboard, mouse, SVGA, and two Wide Ultra SCSI PCI ports.

20. *Security:* The System Configuration Utility (SCU) is used to enable keyboard timer, disable keyboard and mouse inputs, enable power-on/reset administrator password, and disable writing to diskette drive. A mechanical lock on the front door of the system limits access to drive bays (tower models). Cover-intrusion switches on the top and the front covers provide cover-intrusion alarm warnings. An optional Netfinity Security Cover III can be installed to prevent unauthorized access to external cabling, adapter connectors, and ports on the backplane (tower models).

21. *Supported operating systems:* Microsoft Windows NT 3.51/4.0 Workstation, Microsoft Windows NT 3.51/4.0 Server, Novell NetWare 3.12/4.1/4.11/4.1 SMP, IBM OS/2 Warp 3.0, IBM OS/2 Warp 3.0 with Win-OS/2, IBM OS/2 Warp Connect 3.0, IBM OS/2 SMP V2.11, IBM OS/2 Warp Server V4, IBM OS/2 Warp Server Advanced V4 (includes SMP enhancements), IBM OS/2 LAN Server 4.0 Entry, IBM OS/2 LAN Server 4.0 Advanced, SCO UnixWare 2.x, SCO Open Server 5.0, SCO Open DeskTop/Open Server 3.0, and SunSoft Solaris 2.5.

22. *Standard software:* IBM ServerGuide, IBM Netfinity Manager 5.1, QAPlus/PRO, IBM AntiVirus, APC PowerChute Plus, and Lotus Domino 4.51 (single-processor edition).

23. *Support:* IBM TechConnect, IBM Netfinity Start Up Support, and IBM HelpCenter.

24. *Warranty:* Three-year on-site limited warranty: IBM on-site repair, 8:00 am to 5:00 pm Monday through Friday, with 8-hour average response time.

25. *Weight and dimensions:* Tower models (8651-TM0 and 8651-TH0)—weight 120 lb, height 19.0 in, width 19.0 in, depth 26 in. Rack model (8651-RM0 and 8651-RH0) —weight 120 lb, height 19.0 in, width 16.5 in, depth 24 in (11u required for a rack installation).

26. *Legal notices:* (*a*) MHz only measures internal clock speed, not application performance—many factors affect application performance; (*b*) when referring to hard-drive capacity, MB stands for million bytes and GB stands for billion bytes—total user-accessible capacity may vary depending on operating environments; (*c*) for terms and conditions or copies of IBM's limited warranty, call 800-772-2227 in the United States; limited warranty includes International Warranty Service in those countries where this statement of product is sold by IBM or IBM Business Partners (registration required); (*d*) Energy Star compliance—the EPA, as a matter of policy, does not endorse any particular company or its products; (*e*) battery life (and recharge times) will vary with screen brightness, applications, features, power management, battery conditioning, and other references; CD-ROM or hard-disk drive usage may also have a significant impact on battery life; (*f*) actual specifications may vary slightly depending on features and components; (*g*) unless otherwise indicated, prices shown are estimated reseller prices to end users; reseller prices may vary, and IBM reserves the right to change prices and product specifications and to discontinue marketing products without notice.

Netfinity Manager

IBM Netfinity Manager gives you total control over your entire Intel-based systems environment. It makes systems management easier by automating many processes. For example, you can monitor systems usage, log events, and be alerted in the event of a problem. You have the tools to manage what you want, when you want, wherever you are. You can (1) schedule asset management and set up routines to back up key systems, poll clients and servers for configuration information, and more; (2) head off problems before they start with predictive failure analysis alerting and configuration information; (3) reduce time and distance with remote access through IBM Netfinity Manager software and the Advanced System Management Adapter; (4) set up a remote Help Desk and take action to correct client and server systems problems remotely; (5) manage with remote control over the World Wide Web and connect to your servers and clients on any standard browser for total control; (6) Link to TME10, SMS, OpenView, and more and complete your total information technology management with integration with your system of choice; (7) manage your clusters with ease from a single console with IBM Netfinity Cluster Management for Microsoft Cluster Server (when available); (8) integrate your multiplatform, multiprotocol systems management solution easily; (9) reduce total cost of ownership by controlling many of the costs of operating your networked systems; and (10) maintain high availability and prevent problems with the IBM Advanced System Management Adapter's remote monitoring and control capabilities.

Netfinity Manager supports many of the industry's most popular PC operating systems such as Windows NT, Windows 95, Windows 3.1, Novell NetWare, OS/2, and network protocols such as NetBIOS, IPX, SNA (LU6.2), TCP/IP and serial. Netfinity Manager also provides these functions over an Internet connection with a Web browser.

With Netfinity Manager as the foundation for IBM's System Management solution, you can help your business realize true, long-term success through comprehensive control.

Table 17-1 lists correlating hardware and software as it relates to Netfinity.

Table 17-1 Netfinity Hardware and Software Information

Product type applications	Manufacturer	Part number	Product description
Backup	Seagate		Backup ExecNT for Windows NT; Server 6.1
Backup	Seagate		BackupExecNW; Backup Exec for NetWare Server 7.0
Backup	Cheyenne	OS2-SO17-PRO	Arcsolo for OS/2 (version 1.7)
Backup	Cheyenne	ARCserveSCO-20A	ARCserve/Open for SCO V2.0A
Backup	Cheyenne	NT-AS60EE	ARCserve for Windows NT 6.0
Backup	Cheyenne	N3-AS60-EE	ARCserve 6.0 for NetWare
Backup	IBM	84H3129	ADSM for OS/2 2.1 Network Enabler
Backup	Seagate	ALEO-220SY	Sytos Autoloader v2.2 for OS/2
Backup	Seagate	PREO-220SY	Sytos Premium for OS/2 2.2
Communication	IBM		EagleCommSrv, IBM Communication Server
Database	IBM	41H2114	DB2/2 (OS/2) V2.1.1
Database	IBM		EagleDBSrv, Database Server

Database	Microsoft		SQL Server 6.0, SQL Server (Back Office)
Database	Oracle		Oracle7NW, Oracle 7 Databse (NW)
Database	Oracle		WrkGrpEntOS_2, Workgroup Enterprise OS/2
Firewall	Checkpoint		CPFireWall, Firewall
Groupware	IBM		EagleNotesSrv, Lotus Notes Server
Groupware	Lotus		Notes4.xNT, Notes 4.1 for Microsoft Windows NT
Groupware	Lotus		Notes4.xNW, Notes 4.1 for Novell NetWare
Groupware	Lotus		Notes4.xOS2, Notes 4.1 for IBM Warp Server
Groupware	Lotus		Domino4.5NT, Notes Domino Server 4.5 for Microsoft Windows NT
Groupware	Lotus		Domino4.5NW, Notes Domino Server 4.5 for Novell NetWare
Groupware	Lotus		Domino4.5OS, Notes Domino Server 4.5 for OS/2
High Avail	Adaptec		Duralink, Duralink Redundant Network Link
Int. Server	IBM		EagleInetSrv, IBM Internet Server
Int. Server	Netscape		Suitespot3.0, Suitespot Server 3.0
Net Mgmt	IBM		PCSServices, IBM Netfinity 4.x, Services OS/2,NW,NT
Net Mgmt	IBM		PCSMgmt, IBM Netfinity 4.x Mgmt (OS/2,NW,NT)
Net Mgmt NW & NT	Intel	LDM2NW	Intel LAN Desk Manager 2.5
Net Mgmt	Microsoft	SMS 1.1	Systems Management Server (Back Office)
		Cables	
SCSI	IBM	76H5400	IBM Third Channel Cable
SCSI	IBM	94G7421	Netfinity PCI SCSI Controller to Bulkhead
SCSI	IBM	01K8017	IBM 0.8 mm to 68-pin SCSI Adapter
SCSI	IBM	70G9857	PC Server F/W to F/W External SCSI Cable
SCSI	IBM	70G9858	PC Server F/W to Fast External SCSI Cable
		Clustering	
Clustering	Microsoft	MCS1.0	MS Cluster Server 1.0
Clustering	IBM	94G7584	IBM Shared Disk Convenience Kit

Clustering	IBM	01k8018	IBM Netfinity Cluster Pack by Vinca
Clustering	IBM	94G6620	PC Server High Availability Solution for OS/2 Warp
Clustering	IBM	94G6621	PC Server High Availability Solution for Windows NT
Clustering	IBM	94G6622	PC Server High Availability Solution for NetWare

Controllers

UltraSCSI	Adaptec	AHA-3940AUW	PCI Fast/Wide Ultra SCSI Adapter
SCSI-2/RAID	DPT	PM3334UW_3	SmartRAID IV SCSI Raid Adapter (3-channel)
SSA/RAID	IBM	32H3811	IBM SSA RAID PCI Adapter
SSA/RAID	IBM	96H9835	IBM SSA RAID Cluster Adapter
UltraSCSI	Adaptec	AHA-2940UW	PCI Fast/Wide Ultra SCSI-2 Adapter
UltraSCSI	Adaptec	AHA-2940U	FCI Fast Ultra SCSI-2 Adapter
UltraSCSI	IBM	76H5401	IBM ServerRAID II 8 Mb/Battery-Backup Cache
UltraSCSI	IBM	76H3584	ServeRaID II Ultra SCSI Adapter
UltraSCSI	IBM	76H5407	PCI Ultra Wide SCSI Adapter

Ethernet LAN Adapters

Ethernet	IBM	13H9237	PCI Ethernet

Fast Ethernet LAN Adapters

Fast Ethernet	IBM	25H3501	10/100 ISA Ethernet
Fast Ethernet	Intel	PILA8465B	EtherExpress PRO/100 LAN
Fast Ethernet	Olicom	OC-2325	Fast Ethernet PCI/II 10/100
Fast Ethernet	3COM	3C905-TX	Fast EtherLink XL PCI
Fast Ethernet	Adaptec	ANA-6911_TX	Cogent 10/100 PCI Fast Ethernet
Fast Ethernet	Adaptec	ANA-6944A_TX	Cogent PCI Quartet 10/100 4-Port Ethernet
Fast Ethernet	IBM	86H2432	100/10 Etherjet PCI
Fast Ethernet	Intel	PILA8480	EtherExpress PRO/100 Server
Fast Ethernet	Olicom	OC-2326	Fast Ethernet PCI/II 10/100
Fast Ethernet	SMC	SMC9332BDT	EtherPower 10/100 32-bit
Fast Ethernet	SMC	SMC9432TX	EtherPower II 10/100 32-bit

High-speed Networking Products

ATM	3Com	3C975-F	ATMLink PCI-155 ATM
ATM	Fore	PCA-200EPC	ForeRunner PCA-200EPC PCI
ATM	Madge	32-01	Collage 155 ATM Adapter (Fiber)
FDDI	SysKonnect	SK5541	PCI FDDI Adapter (SAS-MIC)
FDDI	SysKonnect	SK5544	PCI FDDI Adapter (DAS)
ATM	IBM	85H9035	Turboways 25 PCI ATM APE-Bridge Adapter

Networking Hardware: Routers, Switches, and Hubs

Ethernet	IBM	8224	8224 Ethernet Hub
Ethernet	IBM	8237	8237 Ethernet Hub

Token Ring	IBM	8238	8238 Token Ring 16-port hub
External	IBM	2210-14T	2210-T14 25-Mbit ATM Router
External	IBM	2210-12E	2210-E12 Router
External	IBM	2210-12T	2210-T12 Router
Router/Interface	Sourcecom	Incarda_P	Incarda/P Server-based Router Adapter
ATM	IBM	42H0525	Turboways 25 ISA Adapter
Keyboard	IBM	76H0109	104-key standard keyboard (Raven Black)
Keyboard	IBM	13H6705	101-key enhanced keyboard with Trackpoint II Black
Monitor	IBM	6540-0x0	G42P Monitor (14 in, 48 kHz)
Monitor	IBM	6540-02x	G42L Monitor (14 in, 48 kHz)
Monitor	IBM	6541-02x	G51L Monitor (15 in, 54 kHz)
Monitor	IBM	6542103	G40 Color Monitor (14 in)
Monitor	IBM	6543303	G50 Color Monitor (15 in)
Monitor	IBM	6553503	P50 Color Monitor 6553
Monitor	IBM	6544403	G70 Color Monitor (17 in)
Monitor	IBM	6554673	P70 Color Monitor 6554
Monitor	IBM	6553-5xx	P50 Color Monitor 6553

Networking Hardware: Routers, Switches, and Hubs

Mouse	IBM	13H6690	Enhanced Mouse (Pearl White)
Mouse	IBM	13H6714	Enhanced Mouse (Raven Black)
Monitor	IBM	6546-x0x	G52P Monitor (15 in, 69 kHz)
Monitor	IBM	6547-x1x	G72S Monitor (17 in, 69 kHz)
Monitor	IBM	6547-x0x	G72P Monitor (17 in, 69 kHz)
Monitor	IBM	6546-x1x	G52S Monitor (15 in, 69 kHz)
Printer	HP	C3155A#ABA	Laserjet 5P
Printer	IBM	4772001	4772 Universal Financial Printer
Rack	IBM	9306900	IBM Netfinity Rack
Upgrade kit	IBM	94G7424	Tower-to-rack Conversion Kit
Upgrade kit	IBM	94G7425	Rack-to-Tower Conversion Kit

Storage Devices

Enclosure	IBM	7133-020	SSA Storage Enclosure—Rack-Mountable
		7133-020	
Enclosure	IBM	3519R01	3519 Rack Drawer Enclosure
Enclosure	IBM	7133-600	7133-600—SSA Storage Enclosure—Floor Standing
Enclosure	IBM	3517001	3517-001—SCSI Multi-Storage Enclosure
Enclosure	IBM	3517002	3517-002—SCSI Multi-Storage Enclosure
Enclosure	IBM	3527001	3527—IBM SSA Entry Storage Subsystem
Enclosure	IBM	3518001	3518—PC Server Enterprise Expansion Enclosure
Enclosure	IBM	35201RU	IBM NetFinity EXP10 Rack Storage Enclosure

Enclosure	Symbios-Logic		DS-20E Metastor RAID Storage Enclosure (Desktop)
Enclosure	Symbios-Logic		RM-20E Metastor RAID Storage Enclosure (Rack)
HDD/HS	IBM	27H1062	4.51-GB SL F/W SSA Hot-Swap HDD
HDD/HS	IBM	05J6413	2.25-GB SSA Hot-Swap HDD II
HDD/HS	IBM	05J6414	4.51-GB SSA Hot-Swap HDD II
HDD/HS	IBM	94G7429	4.51GB Wide Ultra SCSI Hot-Swap HDD
HDD/HS	IBM	94G7430	9.1GB Wide UltraSCSI Hot Swap HDD
HDD/HS	IBM	21H8734	9.1GB SL F/W SSA Hot-Swap HDD
Repeater	IBM	94G7426	Netfinity SCA Backplane Repeater Adapter Kit
Repeater	IBM	94G7585	IBM SCSI-2 F/W Enhanced Repeater Card
Tape Library	Exabyte	EXB-220	EXB-220 Tape Library
		Storage Devices	
Tape	IBM	76H0485	20/40-GB 8-mm Internal SCSI Tape Drive
Tape	IBM	01K1174	35/70-GB External DLT Tape Drive
Tape Library	IBM	3447106	DLT Tape Library (Rack-5U)
Tape	IBM	00K7900	35/70GB Internal DLT Tape Drive
Tape Library	IBM	3449356	8-mm Tape Library (Rack-15U)
Tape	Quantum	TH5AA-YF	DLT4000 40GB External SCSI-2 Tape Drive
Tape Library	IBM	3449355	8-mm Tape Library (Tower)
Tape Library	IBM	3447105	DLT Tape Library (Desktop)
		Token Ring LAN Adapters	
Token Ring	IBM	04H8095	AutoLANStreamer PCI T/R
Token Ring	IBM	41H8900	PCI Token Ring Adapter
Token Ring	Madge	51-02	Smart 16/4 PCI Ringnode
Token Ring	Olicom	OC-3137	Token-Ring PCI/II 16/4 Adapter
		System Upgrades	
DIMM	IBM	94G7384	64-Mb FPM ECC DIMM (4-8Mx72) 60-ns Memory Upgrade kit
DIMM	IBM	94G7385	128-MB FPM ECC DIMM (4-16Mx72) 60-ns Memory Upgrade kit
DIMM	IBM	94G7386	256-MB FPM ECC DIMM (4-32Mx72) 60-ns Memory Upgrade Kit
UPS	IBM	94G3135	APC Smart-UPS 1000
UPS	IBM	94G3136	APC Smart-UPS 1400
Security	IBM	94G7427	IBM Netfinity Security Cover III
Service	IBM	94G5570	Advanced Systems Management Adapter

Service	IBM	94G5571	Advanced Systems Management Power Unit
Service	IBM	94G7578	Advanced Systems Management Adapter
Hot-Plug	IBM	94G7150	400W Hot-Swap Redundant Power Supply
PP200MHz	IBM	94G6678	PC Server SMP 200-MHz Upgrade
Card	IBM	94G7387	Netfinity 200-MHz/1-MB Processor Card
PP200MHz	IBM	94G7147	Netfinity SMP 200-MHz/1-MB L2 Cache

System Upgrades

UPS	IBM	94G6674	APC Smart-UPS 1400RMB (120V)
UPS	IBM	94G6675	APC Smart-UPS 1400RMiB (230V)
UPS	IBM	94G6676	APC Smart-UPS 3000RMB (120V)
UPS	IBM	94G6677	APC Smart-UPS 3000RMiB (230V)

Wide-Area Networking

Modem/Bank	DEC	70001183	T1 Modem Bank (6-24 v.34 Modems)
Modem/Ext	Hayes	08-02349	Optima 28.8 v.34/V.FC Modem
Modem/Ext	USRobotics	001224-0	Courier v. Everything with v.34
SDN/Ext	3Com	3C871	ISDN External Digital Modem
Asynch	DEC	70001169	C/X16 System PCI DB25
ISDN/LAN	Ascend	P25-1UBR	Pipeline 25 ISDN Modem (Ethernet attach)
ISDN/Int	DEC	77000372	Datafire-U S1 Server ISDN Adapter
Multiprotocol	DEC	70001270	Sync/570i PCI (w/v.35 cable)
Multiprotocol	IBM	85X2706	ARTIC Realtime Interface Coprocessor
Multiprotocol	IBM	61G3862	ARTIC Multiport
Multiprotocol	IBM	06H3890	ARTIC Multiport 8-Port RS-232 w/1 MB
Multiprotocol	IBM	39H8058	ARTIC960/PCI
Multiprotocol	IBM	33F8791	ARTIC Multiport Model 2
Multiprotocol	Software Group	570PCI-NH	Netcom Highway (570 PCI)
X.25	IBM	71G6460	ARTIC X.25 Interface Coprocessor

Summary Information

Networks that utilize a server, especially an application or file server, require a server that is reliable, powerful, and expandable. The Netfinity 7000 used in this network is robust and expandable. Additional information is presented in forthcoming chapters about its configuration and use.

I recommend that you contact IBM for the most recent information about the servers they offer. They can be reached at www.ibm.com or at International Business Machines in Armonk, New York.

17.2.8 Uninterruptible Power Supply (Liebert)

Much research went into the power protection part of this network's design. Early on it was determined that a significantly large size UPS was needed to meet the requirements. After evaluating numerous UPSs from various vendors, the author chose Liebert.

Liebert is the best in power protection, at least this author thinks so. I have worked closely with Liebert equipment of all sizes in numerous data centers throughout the United States and abroad. Liebert has been around for decades and will be around for decades to come. The company is established, has considerable market penetration, well-trained and tenured employees, well-built and well-furnished physical plants, and, based on my experience, committed personnel at every level of the corporation. Liebert defines quality. Consider this firsthand evidence.

The large cooling systems Liebert offers are built in their facility. Liebert has certified welders on site who build each frame by hand—that's right, no mass production here in the sense of assembly lines and robots. The significance of this transcends the entire production cycle.

From the outset, devices built at Liebert are traced with an accompanying piece of paper with each employee's stamp during the production process. This signature literally follows the device to the end of the production cycle. During the production cycle numerous checkpoints (quality-control points) exist. If for any reason a product does not meet a preset standard, that product is literally stopped and does not continue until the problem is resolved.

Another aspect of Liebert's quality is *how* they build products. Liebert starts with *raw* elements. They don't work with prefabricated stuff; however, one noted exception exists. One printed-circuit board (PCB) is assembled in another location and included with a certain product at Liebert's facility. However, these PCBs undergo testing to verify predefined standards.

Liebert's attention to detail is incredible. Their painting process is not what most would think. They have tuned their painting process to a mixture between art and science. On some products they bake a powder onto the panels. This production process is monitored not only for quality but also for the control of the paint substance, so contamination does not leave the paint room.

A notable aspect of Liebert's attention to detail is the way their production floor is operated. No management exists. That's right; people work with other people to meet order requirements. It is the first team implementation I have ever seen that works well. Most teams are an averaging; Liebert's teams exploit the strength of each person. Change is driven by workers, not management.

Liebert uses technology where technology makes sense. For example, they implement a computer-controlled metal cutter. This enables them to compute the most number of cuts from a given piece of metal. This is one ingenious way to reduce production cost: employ people where they excel and employ machines where they excel.

This information is important to know when investing significant money into a device that is to serve network needs for a considerable time to come. A Liebert UPStation S was selected to meet the needs of this network. This line of UPSs can meet a wide variety of needs in networks. This UPS series ranges from 3.5 to 18 kVA in capacity. In this instance, I selected a 15-kVA UPS. The capability of this system can meet the needs of all equipment in the network I have built and explained in this book.

For further information contact Liebert at www.liebert.com or

Liebert Corporation
1050 Dearborn Drive
P.O. Box 29186
Colombus, OH 43229
Phone 614-841-5924

Liebert Corporation
19/F Causeway Bay Plaza 1
489 Hennessy Road, Causeway Bay
Hong Kong
Phone 852-2-572-2201

Liebert Corporation
Globe Park, Marlow
Buckinghamshire SL7 1YG
United Kingdom
Phone +144 1628 403200

17.2.9 General Cable

Because of the extensive infrastructure built in this network, considerable attention was given to the electrical requirements. The entire electrical infrastructure was designed and built. One aspect of that infrastructure included a requirement for flexible electrical cable for extension cords.

General Cable was selected for the network. Precisely 6-3 and 8-3 cable was used. These sizes were required to meet electrical standards for this network. The cable itself is very flexible and easy to work with; even 6-gauge cable is relatively easy to work with.

General Cable is distributed through Belden Cable. For more information, contact General Cable at

General Cable
4 Tesseneer Drive
Highland Heights, KY 41076
Phone 800-424-5666

17.2.10 Electrical Components (Pass & Seymour Legrand)

Electrical components including plugs, receptacles, flange outlets, pin and sleeve connections, and connectors were required for this network. After examination of a couple different electrical component suppliers, Pass & Seymour products were chosen. Pass & Seymour has the highest-grade electrical components in the industry.

Each time an electrical connection is made, resistance occurs. The more resistance, the greater the loss of electricity. The better the connection (where connections are made), the more *smoothly* the electric current can flow. In addition, the higher-quality components used in electrical connections, the safer the electrical infrastructure is.

I used the following Pass & Seymour products in this network: Commercial-grade duplex receptacles, boot connectors, L830 receptacles, L830 connectors, and pin and sleeve connectors.

For further information, contact Pass & Seymour at

Pass & Seymour Legrand
50 Boyd Ave
Syracuse, NY 13209
Phone 800-776-4035

7.2.11 Commercial Desktop Computers (IBM)

IBM personal computers are used in this network. The commercial desktop series used include the PC350 and XL series. A typical example of the general specifications for the Base system units (model 350s) used in this network include a 200-MHz Pentium MMX processor, a 2.6-GB hard disk (additional 3.0 GB hard drive), 16-MB nonparity EDO memory (additional 48 MB of RAM), and a 3½-in floppy disk drive.

Units used in this network employ a PCI Busmaster controller and SMART capabilities. These systems include PCI Enhanced IDE hard drives, universal serial bus ports, infrared, 64-bit PCI graphics, and wake-on LAN capability.

Functionality of the universal serial bus (USB) makes peripheral connectivity easier. The hot-connect ability enables peripheral devices to be connected in seconds. Such devices can be added or removed without reconfiguring or rebooting. Each USB port permits up to 127 USB-capable devices.

Some of the PCs used in this network have the capability for symmetrical multiprocessing (SMP) when dual processors are used. An L2 external CPU cache of 256 KB and Pipeline burst L3 cache are also used. The BIOS type is 256-KB Flash, SurePath.

The systems can accommodate up to 192 MB RAM at a speed of 60 ns deployed by 72-pin SIMMs. Their hard-disk size average seek time is 12 ms with a latency of approximately 5.8 ms. They support RAID and hot-swappable drive bays.

The graphic capabilities of these systems employ an S3 Trio64 V+ Graphics-type chipset. The result is SVGA graphics and a data width of 64-bit video RAM. Graphic resolution (with the standard video RAM) is 1280 × 1024 in 16 colors. The maximum resolution (with a maximum video RAM) is 1280 × 1024 with 65536 colors. The graphics bus interface uses PCI architecture.

The systems have a 200-W power-supply type for either 110 or 220 V with a universal manual switch. The heat and sound emissions are 48 dB. The typical weight of each cabinet is 28 lb, with dimensions 6.3 in height, 16.5 in width, and 17.6 in depth.

Systems used in this network include the following security features: (1) boot sequence control, (2) boot without keyboard or mouse, (3) cover key lock, (4) diskette boot inhibit, (5) diskette write protect (switch), (6) diskette I/O control, (7) hard-disk I/O control, (8) parallel I/O control, (9) power-on password, (10) secure fixed DASD, (11) secure removable media, (12) serial I/O control, (13) setup utility password (administrator password), and (14) U-bolt tie-down support.

The systems specifications used in this network also include the following product approvals and/or certifications according to IBM: BABT (UK); CE; CISPR-22 Class B; CSA C22.2 No. 950 (Canada); DEMKO (EN 60950); EIF (SETI) (EN 60950); Energy Saving Law (refer to N-B 1-9174-001); FCC Class B (US); IECEE CB Certificate and report to IEC-950 second edition; ISO 9241-3 Compliant; JATE; NEMKO (EN 60950); NS/G/1234/J/100003 (Telecommunications Safety only: no approval mark); OVE (EN 60950); Power Line Harmonics (refer to N-B 2-4700-017); SEMKO (EN 60950); TUV-GS (EN 60950); TUV-GS—ZH1/618; UL-1950 first edition; VCCI Class 2 (Japan).

In addition, IBM currently has a 3-year limited warranty period: first year, on-site service; second and third years, carry-in service, after third year, parts and labor.

The IBM desktop systems used in this network were shipped with preinstalled software. Some of these systems were reconfigured to meet the needs of the network. However, all respect, both legal and ethical, was given to manufacturers of hardware and software products. Each system used in this network is covered by either a site license or has a dedicated piece of software for each system; and, each system has one user. In the case of servers, workstations, or otherwise, each manufacturer's legal guidelines were followed. I strongly recommend these matters be factored into network design of any network. Simply stated, using an unpaid-for piece of software, unless it is clear that it is freeware, is stealing. It is no different from someone stealing a tangible item. Consider this when you design your network.

The Model 658842U IBM PC 300XL is designed with the latest 266-MHz Pentium II technology to handle demanding business applications in a networked environment. These are the high-end systems that deliver value and 2.5-GB hard drive to keep you ahead of the curve with the performance disk that power users demand. The IBM PC 300XL series includes open-bay models which can be custom-configured via IBM Authorized Assembler Program (AAP). Standard features are as follows:

1. *Processors:* Pentium II processors—233, 266, or 300 MHz with unified 512-KB L2 cache memory; 32 MB nonparity EDO memory (expandable memory to 384 MB), 32 MB, expandable to 384 MB (3 DIMMs), EDO/60 ns.

2. *Hard drives:* 2.5 or 4.2 GB EIDE with SMART or 4.3 GB Wide Ultra SCSI with SMART or open-bay PCI Bus master EIDE controller on planar SCSI models include a SCSI-2 Fast and Wide PCI Busmaster adapter.

Graphics and video resolution: S3 Trio64V2; 64-bit; 2 MB standard/maximum video DRAM; 256 colors at 1280 × 1024 resolution.

3. *Network features:* LANClient Control Manager supported, Wake-on LAN, Plug-In and Go, Flash over LAN (BIOS/CMOS), Plug-and-Play, CID.

4. *Network interface:* Integrated Intel EtherExpress 10/100 Mbits/s Ethernet with Wake-on LAN.

5. *CD-ROM:* Models available with 16×–8× (variable speed) CD-ROM. (Variable read rate; actual playback speed will vary and is often less than the maximum possible.)

6. *Audio:* models available with Crystal 4236B audio chip, 16-bit, support Sound Blaster Pro applications.

7. *Diskette drive:* 3.5 in 1.44-MB standard.

8. *Slots:* 3 shared PCI/ISA , 2 ISA.

9. *Bays:* three 3.5 in, two 5.25 in.

10. *BIOS:* Flash ROM.

11. *Architecture:* PCI local bus, ISA data bus.

12. *Ports:* serial (16550), enhanced parallel (ECP/EPP), two USB ports, SVGA video, EIDE controller, 10/100 Ethernet RJ-45, IrDA-2 compliant infrared, audio mic-in and line-out mini-jacks, keyboard, mouse.

13. *Keyboard/mouse:* IBM Cameo 104-key (rubber dome) and Enhanced Mouse.

14. *Power supply:* 200 W.

15. *Security features:* IBM AssetCare—serialization and laser etching of memory and processors, third-party registration available through Retainagoup Limited.

16. *IBM AntiVirus and ConfigSafe:* Vital Product Data (VPD) support, cover key lock, sliding front door lock, U-bolt anchor support, secure access openings, secure removable media, secure fixed DASD, diskette write protect, power-on password, configuration/administrator password, keyboard/mouse password, Wake-on LAN password prompt, boot sequence control, diskette boot inhibit, boot without keyboard/mouse, mouse-disable, I/O controls.

17. *Software and tools:* Windows 95 or Windows NT 4.0 preload available on models with a hard drive Lotus SmartSuite license, Microsoft NetMeeting (Windows 95 preload models only), LANClient Control Manager (downloadable via Internet), IBM Netfinity, Manager software, Intel LANDesk Client Manager, Artisoft CoSession, QAPlus System Support CD (Ready to Configure) with additional software/drivers.
18. *Limited warranty:* 3-year parts and 1-year labor.

19. *Legal notices:* (*a*) MHz only measures internal clock speed, not application performance—many factors affect application performance; (*b*) when referring to hard-drive capacity, MB stands for million bytes and GB stands for billion bytes; total user-accessible capacity may vary depending on operating environments; (*c*) for terms and conditions or copies of IBM's limited warranty, call 800-772-2227 in the United States; limited warranty includes International Warranty Service in those countries where this statement of product is sold by IBM or IBM Business Partners (registration required); (*d*) Energy Star compliance—the EPA, as a matter of policy, does not endorse any particular company nor its products; (*e*) battery life (and recharge times) will vary per on- screen brightness, applications, features, power management, battery conditioning and other references—CD-ROM or hard-disk drive usage may also have a significant impact on battery life; (*f*) actual specifications may vary slightly depending on features and components; (*g*) unless otherwise indicated, prices shown are estimated reseller prices to end users; reseller prices may vary, and IBM reserves the right to change prices and product specifications and to discontinue marketing products without notice.

For more information on IBM products, contact: www.ibm.com or International Business Machines, Armonk, New York.

17.2.12 Remote Workstations (IBM ThinkPads)

Multiple remote workstations were required to work with this network. After examination of notebook computers available, I selected IBM ThinkPads. The following is typical specifications for those ThinkPads used in this network.

ThinkPad 765D is a 166-MHz Pentium with the latest power, connectivity, and configuration flexibility to optimize effectiveness and maximize investment return. High-performance features may include large 13.3- or 12.1-in- high resolution displays (1024 × 768 with 65536 colors) with superb graphics, the latest Pentium processors with MMX technology, large (3-GB) hard drives, integrated infrared, and advanced multimedia. Other features are RAM, 2/32 MB nonparity EDO memory (expandable to 104 MB), 8 × CD-ROM, and MPEG-1 Microsoft Windows 95.

Standard features are as follows: pointing device type—TrackPoint III; standard diskette size—3.5-in floppy, 1.44 MB; optional diskette size—3.5-in 2.88 MB; diskette drive configuration external; keyboard type standard—full-size 84-key (tilt/palm rest space); keyboard type(s) selectable—numeric keypad, integrated; product approvals and certifications CISPR-22 Class B; CSA 4; C22.2 No. 950 (Canada); FCC Class B—Part 15; IEC-950; JATE; NOM (Mexico); SASO; UK-PTT; UL-1950; VCCI Class 2 (Japan). Limited warranty period is 3-years (system and type 3; battery, 1 year); customer carry-in repair or provided by ThinkPad EasyServ (North America only). Weight and dimensions: 7.7 lb, 2.2 × 11.7 × 9.3 in ($h \times w \times d$).

Legal notices are as follows:

1. MHz only measures internal clock speed, not application performance. Many factors affect application performance.

2. When referring to hard-drive capacity, MB stands for million bytes and GB stands for billion bytes. Total user-accessible capacity may vary depending on operating environments.

3. For terms and conditions or copies of IBM’s limited warranty, call 800-772-2227 in the United States. Limited warranty includes International Warranty Service in those countries where this statement of product is sold by IBM or IBM Business Partners (registration required).

4. Energy Star compliance: The EPA, as a matter of policy, does not endorse any particular company or its products.

5. Battery life (and recharge times) will vary based on screen brightness, applications, features, power management, battery conditioning, and other references. CD-ROM or hard-disk drive usage may also have a significant impact on battery life.

6. Actual specifications may vary slightly depending on features and components.

7. Unless otherwise indicated, prices shown are estimated reseller prices to end users. Reseller prices may vary. IBM reserves the right to change prices and product specifications and to discontinue marketing products without notice.

17.2.13 Network Hub (3Com USRobotics)

A 3Com USRobotics Enterprise Network hub was selected to use in this network. Data-communication equipment is the single most critical link in any network. This is true because it is the central point of attachment between remote users and a backbone network, regardless of the size of the backbone or location. It is also true if all users are in the same physical location. Data-communication equipment is central to networks; if the equipment fails, so does the network. At one time remote computing meant having a device in one location and a terminal attached to it by a wire. USRobotics revolutionized that definition by designing the Enterprise Network hub. This device, explained in greater detail later, is powerful.

Consider Fig. 17-11, which illustrates how the hub appears conceptually.

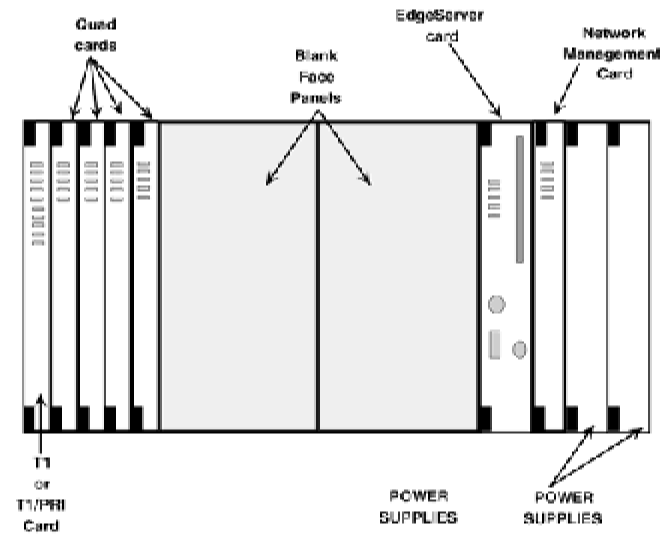


Figure 17-11
Enterprise Network hub.

When remote users or remote network(s) are concerned, multiple issues must be considered during the design phase. The minimum number of issues to be reviewed during your plans are security, reliability, maintenance, ease of use, internal protocol compatibility, expandability, internal design architecture, and interface-standard compatibility.

Security has become the single most important topic in networking, regardless of the type network or location. Networks can have a considerable degree of security built into the design if proper components are used that implement security. Where data-communication equipment is concerned, having a device that can provide a security firewall is best.

The USRobotics communication hub used in this network has three possible configurations regarding its function in the network. USRobotics refers to this as *gateway application cards.* USRobotics uses the following terminology and explanation: X.25, NETServer Card, and API Card. According to USRobotics the *X.25 card* provides access capability to packet-switched networks. This card uses an EIA-232/V.35 interface connection point. The *NETServer card* functions as either a router, terminal server, or both. Ethernet and Token-Ring NICs can be used with it. USRobotics refers to this card as the EdgeServe card. This card has Windows NT loaded onto it; more about the functionality of this card is explained later.

An *API card* is designed to let customers design their own applications by way of USRobotics software development kits. These panels can be removed and other cards inserted. A total of 17 slots exist; slot 1 is the T1 card. Beginning with slot number 2 are analog or digital quad modem cards. These cards have the equivalent of four modems on them. Slots 15 and 16 are the EdgeServer location. Slot 17 is where the network management card is located. The remaining slots house two power supplies. Although not shown in Fig. 17-11, the undercradle portion of the hub houses approximately 16 fans to cool the components.

Reliability is another important factor for any communication equipment. The design of the USRobotics hub has reliability built into it. One example supports this observation: The hub has two power supplies; but only one is required to operate the unit. These hubs build in redundancy even to the level of the power-supply unit.

Maintenance is another part of the equation for communication equipment. The hub used in this network has remote management capability, local management capability, and easy access for those components that may need removal.

Any communication device requires skill. Most require a fairly advanced level of skill to maximize use. The capability of any commu nication device has little to do with its ease of use, which is a design issue. With the hub used in this network, ease of use is designed into it. Ease of use can be measured in communication equipment by documentation provided, how thorough and detailed it is; by accessibility to configure ports, and the ability to use the equipment in a partially failed state (should that occur). My rule of thumb is: The more complex the functions a device offers, the simpler the documentation should be. The simple fact is that data-communication equipment is complex enough without humans adding another layer of complexity to it.

Another factor to analyze with data-communication equipment is the protocol compatibility. This includes evaluation of upper- and lower-layer protocols. Because this hub has the EdgeServer card in it, NetBEUI, TCP/IP, and IPX upper-layer protocols are supported. Token-Ring and Ethernet lower-layer protocols are supported as well. Regarding the USRobotics hub used in this network, the author has observed that the use of Token Ring and Ethernet is more than sufficient because these two protocols are dominant lower-layer protocols used in networks today.

Expandability is very important with data-communication equipment. The design of the Enterprise Network hub is such that any size network can be built around this technology. This is true because of how the equipment has been architected. It is possible with the 3Com USRobotics equipment to start a network with one or two Enterprise hubs and then continue to add them until racks of them are filled.

Internal design architecture is also very important to data-communication equipment. The internal architecture of data-communication gear is the proverbial pivot on which all communication transactions hinge. The internal communications bus and the incoming port architecture are the foundation of the device. These should be capable of handling a complete load on the device without causing hangs or system slowdown.

Interface-standard compatibility is another matter to examine when you are evaluating data-communication equipment. In this network, the hub has flexibility regarding how certain connections are made. In some instances options exist to make a connection. This alone makes for ease of use, installation, and maintenance. It also means some existing equipment at your site may be usable. That can save money.

For additional information you can contact 3Com USRobotics at www.3com.com or

3Com USRobotics
Network Systems Division
1800 Central Road
Mount Prospect, IL 60056-2293

17.2.14 Multimedia Components (Creative Labs)

Creative Labs was chosen as the vendor for multimedia equipment. In the past, buying IBM-compatible PCs, or clones, was not a problem. However, all things change. In the arena of multimedia, Creative Labs wrote the book on how to do it. Since multimedia equipment is primarily add-on at this point in time, systems do not depend on multimedia as they do the hard disk or monitor, for example.

However, some multimedia clone products exist. Many of these products attempt to copy what Creative Labs has already designed. In the arena of multimedia, clones are the incorrect way to invest money. The operational nature of some multimedia software is such that multimedia “clone” equipment may not be able to execute all the multimedia exploits. This may sound strange, but it is true. Today, systems typically have CD-ROMs, speakers, microphones, line outputs for amplifiers, line inputs for peripheral integration, and software that enables a user to create, play back, and listen to or see various data streams.

All the desktop systems in this network are IBM 350 series. I selected these because each one would be customized to deliver a robust workload. Another reason for choosing this series is the upgrade capability. It is the same with Creative Labs equipment.

In each system Creative Lab's equipment is the multimedia hardware and software. One system has a package of multimedia equipment from Creative Labs. The system includes an interface board, speakers, necessary cabling, microphone, CD-ROM, infrared remote control, software drivers, and various software titles for viewing and listening.

Creative Labs has designed the benchmark for multimedia systems. The significance of this should not be overlooked during the design phase of your network. Windows 95 and NT4.0 acknowledges most, if not all, Creative Labs hardware and software. It is plug-n-play-compatible. Another significant aspect of this equipment is its adaptability. Creative Labs is continually upgrading its equipment to stay inline with other vendors; however, they support equipment and systems that are not this year's product.

Multimedia is more than a CD-ROM and speakers. Today multimedia typically encompasses a digital video disk (DVD) and enhanced display support. More than at any other time, displays need powerful drivers and memory to store the screen of information to be presented.

Creative Labs is based in California, but has offices around the world. I recommend contacting the one closest to you for additional information about multimedia at www.soundblaster.com or

Creative Labs
1901 McCarthy Blvd.
Milpitas, CA 95035

Creative Technology Ltd.
67 Ayer Rajah Crescent #30-18
Singapore 0513

Creative Labs Technical Support
1523 Cimarron Plaza
Stillwater, OK 74075

Creative Labs Ltd.
Blanchardstown Industrial Park
Blanchardstown, Dublin 15
Ireland

17.2.15 Infrastructure Equipment (NetOptics)

The network designed and used in the writing of this book used fiber-optic equipment selected from NetOptics. I acquired the following components from NetOptics: SC-SC duplex fiber cables, SC-FDDI fiber cables, SC fiber splitters, and SC-based fiber patch panel. Figure 17-12 illustrates the implementation of the fiber in this network.

The implementation of fiber into the network adds greater bandwidth and more flexibility for application support such as video, voice, and multimedia. The NetOptics products I used were very well engineered and their support team extremely helpful. For more information about their products, contact them at

NetOptics
1180-A Miraloma Way
Sunnyvale, CA 94086
Phone: 408-737-7777

17.2.16 Fiber Distributed Data Interface Concentrator and Interface Cards (SysKonnect)

Because of the amount of network traffic, I selected FDDI to operate a significant portion of the network. I chose SysKonnect for the FDDI concentrator and FDDI interface cards.

SysKonnect has very useful information on their Web site, including the following:

> SysKonnect's SK-FDDI Concentrator II is a modular, manageable, and easily configurable connectivity platform for building high-speed FDDI workgroups and integrating your servers into the FDDI-ring. In addition to the impressive networking performance values FDDI offers in real use, it is the advanced security features which make FDDI the number-one choice for mission-critical applications.
>
> SysKonnect's modular: The base unit of the SK-5000S offers SK-FDDI Concentrator II two slots for inserting linecards. It combines the linecard ability with the compact design of the unit itself for enhanced functionality. The advanced features of FDDI functionality offer connectivity options and fast, convenient configuration.
>
> Besides all-copper backbone and all-fiber linecards, there are also linecards available offering mixed-media configurations. These modules
> have fiber interfaces (for A/B ports) and copper-interfaces for attaching additional FDDI stations.
>
> SK-FDDI Concentrator II Overview: Large variety of linecards allowing UTP, S/UTP, and fiber connections within one device. High port density with up to 16 fiber-ports and 24 copper ports per unit. Fully featured Class-A Station supporting Null-Attachment-, Single-Attachment-, and Dual-Attachment Concentrator configurations. In-Band configuration and management via BOOTP, TFTP, and TELNET. Out-of-band configuration via RS-232 port. All status indicators and connectors (except power-supply) on the front-panel.
>
> Innovative Web-Based: Besides supporting the traditional Management SMT and SNMP management, the SK-FDDI Concentrator II also has a built-in Web-server. This exceptional feature allows access to the concentrator with any common Web-browser. An administrator can handle all the management objects of the SK-FDDI Concentrator II directly from the workstation. Statistic information for the ports, SMT and device status can be viewed with the Web-browser. In addition to these monitoring features, it is also possible to do all the configuration of the concentrator with the Web-based management. So any configuration task, from changing single port configurations to complete software upgrades, can easily be done via the intranet or even the Internet.

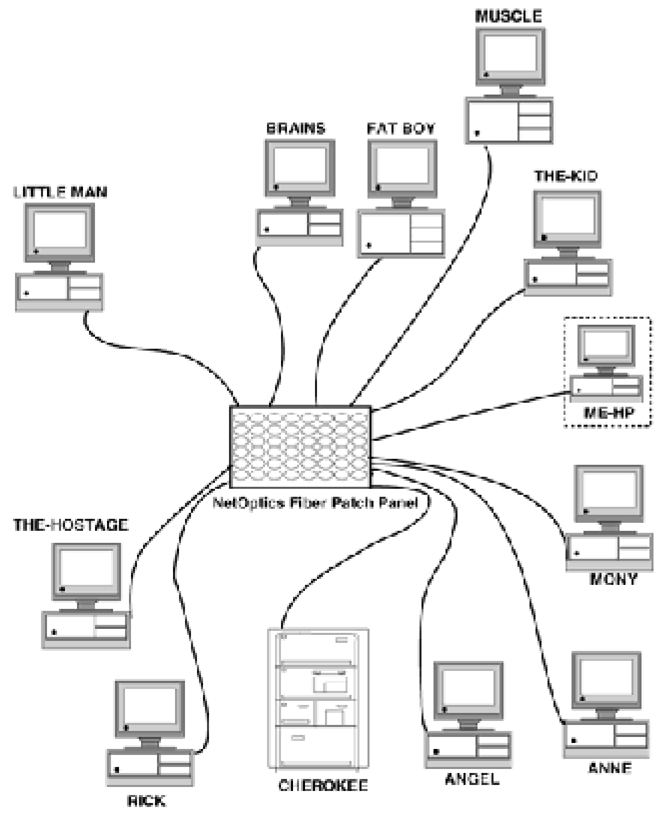


Figure 17-12
Fiber-optic hub and wiring infrastructure.

The following information is applicable to the concentrator I used. This information was helpful during the planning phase of the network. FDDI standards as defined by ISO 9314, ANSI X3T12, and TP-PMD (MLT-3) are as follows: (1) fiber-optic (MIC connector/duplex SC media types connector) UTP copper TP-PMD MLT-3 (RJ-45) cable; (2) SMT 7.3 (software upgradable); SNMP:—MIB II (RFC 1213), FDDI MIB (RFC 1512), management SysKonnect's private MIB Web-based management via integrated Web-server; (3) Flash-EEPROM for firmware update; out-of-band configuration via RS-232 interface in-band configuration via TELNET Remote configuration with BOOTP and TFTP; (4) 1 bicolor LED per port displaying port state, 4 bicolor LEDs displaying device state; (5) optical bypass Mini Din-6 control connection 256 KB SRAM; 2 MB Flash EEPROM MD Am79C830A Formac Plus; (6) compliance FCC Class A; EN 55022; Class A; EN 50082-1; IEC801-2,-3,-4 (1988) 1/0.5-kV compliance, UL 1950, CSA C22.2 No. 950 (cUL), VDE-GS, CE (EN 60950, CB-certification); (7) Storage: 1 to 60°C; Transport –20 to 60°C; operation 10 to 40°C; humidity storage 10 to 80%; transport 10 to 95% (noncondensing); operation: 10 to 80% [$w \times d$ = 430 × 88.8 mm (2 HU) × 290 mm × h]; 6 kg (13.23 lb) to 6.9 kg (15.21 lb) (depending on configuration); (8) power requirements 100 to 120/200 to 240 V~, 50 to 60 Hz; power consumption 65 to 120 W. Also, the SK-FDDI Concentrator II base unit SK-5000S has two slots for inserting linecards. At least one of these slots has to be equipped with a linecard to form a functional FDDI concentrator unit.

The following information provides additional references for FDDI information. SysKonnect has a wealth of informed people and data repositories on FDDI.

Media Access Control (MAC)

- ANSI X3.139, published 1987 (reaffirmed 1993)
- ISO 9314-2: published 1989

Enhanced MAC (MAC-2)

- Rev. 5.1 (11/22/93)
- ANSI X3.239-199x X3 letter ballot closed 4/20—no comments to BSR (4/21/94) for approval to publish
- ISO DIS 9314-9 Issue DIS letter ballot (per 7/93 meeting)
- Physical Layer Protocol (PHY)
- ANSI X3.148, published 1998
- ISO 9314-1, published 1989

Enhanced PHY (PHY-2)

- Rev. 5.2 (11/1/93)
- ANSI X3.231, published 1994
- ISO DIS 9314-7

Physical Layer, Medium Dependent (PMD)

- ANSI X3.166, published 1990

• ISO 9314-3

Station Management (SMT)

• Rev. 7.3a (6/7/94)

• ANSI X3.229, published 1994 Rev. 7.3a to be published shortly

• ISO DIS 9314-6 Issue DIS letter ballot (per 7/93 meeting)

Hybrid Ring Control (HRC)

• Rev. 6.2 (5/14/91)

• ANSI X3.186, published 1992

• ISO DIS 9314-5 DIS letter ballot (rev. 6.1) passed

• Rev. 6.3—5/28/92—ISO LB corrections plus 5 clarifications; editing completed for publishing of IS

Physical Layer Repeater (PHY-REP)

• Rev. 1.0.7 (10/13/94)

• Public review completed 9/19/95 with no comments

• Editorial comments approved and incorporated

Single Mode Fiber PMD (SMF-PMD)

• Rev. 4.2 (12/7/92)

• ANSI X3.184, published 1993

• ISO DIS 9314-4 DIS letter ballot to be issued

Low Cost Fiber PMD (LCF-PMD)

• Rev. 2.2 (9/27/95) ANSI X3.237-199x

• ISO CD 9314-9 issue CD letter ballot (per 7/93 meeting)

Twisted Pair PMD (TP-PMD)

• Rev. 2.2 (9/25/95) ANSI X3.263-199x

SONET Physical Layer Mapping (SPM)

• Request for withdrawal completed 12/15/95

• No document available

• Mapping standardized (T1.105-1992)

• X3T12 standard development on hold

Enhanced SMT Common Services (SMT-2-CS)

• Rev. 3.0 (12/8/93)

• ANSI X3.257-199x X3 public review ended 9/12/94

Enhanced SMT Isochronous Services (SMT-2-IS)

• Rev. 3.0 (12/8/93)

• ANSI X3.259-199x X3 public review ended 9/12/94

Enhanced SMT Packet Services (SMT-2-PS)

• Rev. 3.0 (12/8/93)

• ANSI X3.259-199x X3 public review ended 9/12/94

Conformance Test PICS Proforma (CT-PICS)

• Rev. 3.6 (4/12/95)

• ANSI (X3T9.5/92-098) X3.262-199x X3 public review ended 12/17/94

• ISO DIS 9314-13

Abstract Test Suite for MAC (MAC-ATS)

• Rev. 2.2 (9/8/95)

• ANSI X3.245-199x

• ISO DIS 9314-26

Abstract Test Suite for PHY (PHY-ATS)

• Rev. 2.6 (12/15/95)

• ANSI X3.248-199x

• ISO CD 9314-21 Issue CD letter ballot (per 7/93 meeting)

Abstract Test Suite for PMD (PMD-ATS)

• Rev. 2.5 (12/15/95) X3

• ANSI X3.255-199x edit for issuance of X3 letter ballot

• ISO CD 9314-20 Issue CD letter ballot (per 7/93 meeting)

Abstract Test Suite for SMT (SMT-ATS)

• Rev. 1.4 (1/5/96) (X3T9.5/92-102)

• Public review ended 3/5/96

Figure 17-13 shows a logical view of the network backbone built with the SysKonnect FDDI Concentrator. SysKonnect interface cards are also used in this network.

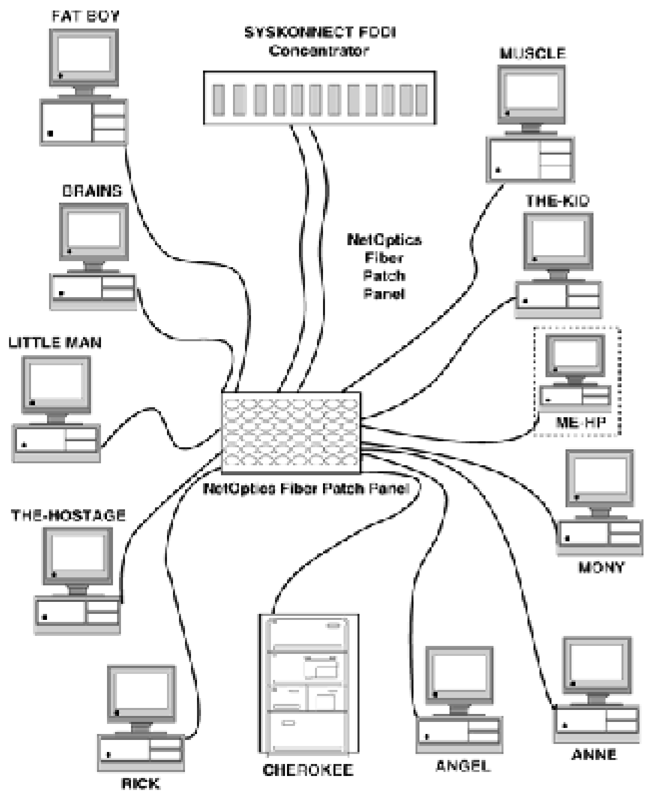

FAT BOY
SYSKONNECT FDDI Concentrator
MUSCLE
BRAINS
THE-KID
NetOptics Fiber Patch Panel
LITTLE MAN
ME-HP
NetOptics Fiber Patch Panel
THE-HOSTAGE
MONY
RICK
CHEROKEE
ANGEL
ANNE

Figure 17-13
SysKonnect Concentrator.

Note the implementation of the patch panel and the concentrator. This configuration makes for dynamic patching if necessary between different workstations and servers.

SysKonnect offers a full line of products that range in support from Windows NT to UNIX. Their technical support is unsurpassed. You can reach them at

SysKonnect
1922 Zanker Road
San Jose, CA 95122
Phone: 408-437-3800

17.2.17 Ethernet and SCSI adapters (Adaptec Corp.)

In the initial phase of network design I realized that devices using Small Computer Serial Interface (SCSI) were going to be part of the network. Consequently, I determined that the Adaptec Corporation would be the best company to use for SCSI adapters. Upon initial investigation with Adaptec, I discovered they offer some very powerful Ethernet adapters. Information on the SCSI and Ethernet adapters used in this network is provided here.

The Adaptec AHA-2940 SCSI adapter is used in this network. Some of these product family highlights include (1) 20-MB/s UltraSCSI data-transfer rate, (2) high-performance bus-mastering architecture for improved system performance, (3) tested for compatibility with more than 200 systems and peripherals from major manufacturers, (4) compatible with all major operating systems, and (5) easy plug-and-play installation.

The AHA-2940 Ultra host adapter delivers UltraSCSI speed for demanding professional applications and high-performance peripherals. With its bus-mastering architecture and advanced SCSI features, the AHA-2940 Ultra host adapter improves system performance, especially in multitasking environments. In addition, the AHA-2940 Ultra adapter offers unsurpassed system and peripheral compatibility.

With double the data-transfer speed using UltraSCSI technology, the AHA-2940 Ultra host adapter doubles the maximum rate on the SCSI bus from 10 to 20 MB/s. By increasing the throughput rate between peripherals and the desktop system, the AHA-2940 Ultra adapter accelerates system performance and user productivity.

In addition to UltraSCSI speed, the AHA-2940 Ultra host adapter features a high-performance bus-mastering architecture that regulates data movement directly between peripherals and system memory. The onboard Adaptec PhaseEngine RISC processor takes over I/O processing from the CPU, which frees it for other tasks. The result is lower CPU utilization and noticeably faster response time.

The AHA-2940 Ultra host adapter makes true multitasking possible. SCSI technology offers unique features, including disconnect/reconnect, tagged command queuing, and multithreaded I/O, which allow the system CPU to access data on multiple peripheral devices simultaneously.

All Adaptec host adapters are rigorously tested with hundreds of different SCSI peripherals and systems, which makes them the most com patible and reliable host adapters in the marketplace. The AHA-2940 Ultra adapter is also fully compatible with all popular desktop and network operating systems, including DOS, Windows, Windows NT, Windows 95, OS/2, NetWare, and UNIX.

Installing the AHA-2940 Ultra host adapter is quick and easy. Simply insert the host adapter in the PCI slot, and the system BIOS will assign resources to it. Using the SCSISelect utility, users can then fine-tune the performance of the adapter by making on-screen choices from the utility menu. Software automatically determines what hardware is present, loads the correct drivers, and modifies system configuration files. Completely compatible with Windows 95, the AHA-2940 Ultra host adapter utilizes advanced plug-and-play features that automatically assign IDs and control termination.

The AHA-2940 Ultra kit provides everything necessary for connecting high-performance SCSI peripherals to PCs and workstations. It features the AHA-2940AU host adapter board with an internal cable, complete user documentation, and Adaptec EZ-SCSI software for simple installation. Kit contents include

• AHA-2940AU PCI-to-UltraSCSI host adapter board

• Adaptec 7800 Family Manager Set software drivers for Windows NT, Windows 95, OS/2 2.x and 3.x, NetWare 3.x and 4.x, SCO UNIX 3.2.x, and UnixWare 1.x and 2.x Adaptec EZ-SCSI software for DOS, Windows, Windows NT, Windows 95, and Windows for Workgroups

• Includes applications such as SCSI Backup (Backup Basics), QuickScan, CD Player, Photo CD Viewer (Magic Lantern), Advanced Power Management, SCSI Interrogator, SCSI Disk Partitioner, SCSITutor, SCSIBench

• Provides device support for hard disks, removable disks, MO, CD-ROM, CD-Recordable, Photo CD, and tape drives and scanners

• Standard, 3-position internal SCSI ribbon cable

• Complete user documentation

The Adaptec AHA-2940 technical specifications are

Computer bus	PCI local-bus
Interface protocol	bus master DMA
Host bus data-transfer rate	≤MB/s burst rate
Peripheral bus	8-bit UltraSCSI
SCSI synchronous data rate	20 MB/s
SCSI asynchronous data rate	6 MB/s
Device protocol	SCSI-1, SCSI-2, SCSI-3, UltraSCSI
Advanced SCSI features	ASPI-compliant, multithreaded I/O (≤ 255 tasks simultaneously), scatter/gather, tagged queuing, disconnect/reconnect, synchronous and asynchronous Fast and Wide, bootable from attached disks
External connector	50-pin high-density
Electrical drivers	Single-ended, active, programmable via SCSISelect
Hard-disk capacity	Extended translation supports drive capacity of ≤8 GB/disk
Device support	≤7 disks under DOS 5.0
Electrical termination	Single-ended, active, software-controlled

The Adaptec physical and environmental specifications include 4.75 in (12 cm) length, 3.5 in (8.75 cm) height; operating temperature 0 to 55°C, storage temperature 255 to 85°C; humidity (operating) temperature 10 to 90%, noncondensing. Input/output environment is Windows 95, Windows 3.1, Windows NT, Windows for Workgroups, DOS, IBM OS/2 2.x and 3.x, NetWare 3.12 and 4.x, SCO UNIX 3.2.x, UnixWare 1.x and 2.x. Mean time between failures (MTBF) is 544,264.3 h (Bellcore, TR-NWT-000332, Method I, QL-I).

Another Adaptec component is used in the network: the AHA-2940 UW (ultrawide). The product highlights for this adapter are (1) 40-MB/s Ultra Wide SCSI data-transfer rate, (2) connection for up to 15 SCSI peripheral devices, (3) designed for true multitasking, and (4) tested for compatibility with hundreds of systems and peripherals from major manufacturers.

The AHA-2940 Ultra Wide host adapter is the ideal PCI-to-SCSI host adapter for entry-level servers and workstations. It moves data fast—up to 40 MB/s, and it connects up to 15 SCSI devices for expanded storage capacity. In addition, the AHA-2940 Ultra Wide adapter delivers unrivaled system and peripheral compatibility.

Combining UltraSCSI speed and 16-bit wide SCSI data transfers, the AHA-2940 Ultra Wide host adapter moves data on the SCSI bus at a maximum rate of 40 MB/s. By increasing the throughput rate between peripherals and the system CPU, the AHA-2940 Ultra Wide adapter accelerates system performance and user productivity.

The AHA-2940 Ultra Wide host adapter delivers expanded connectivity to meet the storage capacity requirements of server environments. It supports up to 15 SCSI devices simultaneously. Both 8- and 16-bit devices can be configured in any combination for maximum configuration flexibility.

The AHA-2940 Ultra Wide host adapter makes true multitasking possible. SCSI technology offers unique features, including disconnect/reconnect, tagged command queuing, and multithreaded I/O, which allow the system CPU to move data to and from multiple peripheral devices simultaneously.

All Adaptec host adapters are rigorously tested with hundreds of different SCSI peripherals and systems, which makes them the most compatible and reliable host adapters in the marketplace. The AHA-2940 Ultra Wide adapter is fully compatible with all popular desktop and network operating systems, including Microsoft Windows 3.1, DOS, Windows NT, Windows 95, OS/2, NetWare, and UNIX. Under Windows 95, the AHA-2940 Ultra Wide adapter utilizes advanced plug-and-play features that automatically assign IDs and control termination.

The AHA-2940 Ultra Wide kit provides everything necessary for connecting high-performance SCSI peripherals to workstations and entry-level servers. It features the AHA-2940 UW host adapter board with internal SCSI cables, and complete user documentation. The kit also includes Adaptec EZ-SCSI software for simple installation and the SCSISelect utility for easy on-screen performance tuning.

The AHA-2940 Ultra Wide complete kit includes (1) AHA-2940 UW PCI-to-Wide UltraSCSI host adapter board; (2) Adaptec 7800 Family Manager Set software drivers for Windows NT, Windows 95, OS/2 2.x and 3.x, NetWare 3.x and 4.x, SCO UNIX 3.2.x, and UnixWare 1.x and 2.x; and (3) Adaptec EZ-SCSI software for Windows NT, Windows 95, Windows for Workgroups, and DOS.

It also includes applications such as SCSI Tape Backup (Backup Basics), QuickScan, CD Player, photo CD Viewer (Magic Lantern), Advanced Power Management, SCSI Interrogator, SCSI Disk Partitioner, SCSITutor, SCSIBench and it provides device support for hard disks, removable disks, MO, CD-ROM, CD-Recordable, Photo CD and tape drives and scanners; one 3-position, 68-pin UltraSCSI internal ribbon cable and one 3-position, 50-pin UltraSCSI internal ribbon cable; complete user documentation; and a 5-year warranty card.

The technical specifications for the AHA-2940 UW are

Computer bus	PCI local-bus
Interface protocol	Bus master DMA
Host bus burst data rate	133 MB/s
Peripheral bus	8- and 16-bit Wide UltraSCSI
SCSI synchronous data rate	40 MB/s
SCSI asynchronous data rate	3.3 MB/s
Device Protocol	SCSI-1, SCSI-2, SCSI-3, Wide UltraSCSI

Advanced SCSI features	ASPI-compliant, multithreaded I/O (≤255 tasks simultaneously), scatter/gather, disconnect/reconnect, tagged command queuing, synchronous and asynchronous data transfer, bootable from attached disks
External connector	68-pin high-density
Hard-disk capacity	Extended translation supports drive capacity of ≤8 GB/disk
Device support	≤15 devices under DOS 5.0 and above
Electrical termination	Automatic, active, programmable via SCSISelect

The AHA-2940 UW operating environment includes Microsoft Windows 3.x, DOS, Windows NT, Windows 95, OS/2 2.x and 3.x, NetWare 3.12 and 4.x, SCO UNIX 3.2.x, and UnixWare 1.x and 2.x; the physical and environmental specifications are 6.87 in (17.0 cm) length, 3.87 in (9.5 cm) height; operating temperature 0 to 55°C, storage temperature –55 to 85°C; humidity (operating) 10 to 90%, noncondensing. MTBF 494,641 h (per Bellcore TR-NWT-000332,Issue 4, Method 1).

Another Adaptec product was used in the network discussed in later chapters: the COgent 4-port Ethernet adapter by Adaptec. The Adaptec 4-port Ethernet interface product highlights include

• Four connectors supporting four separate network segments, all at full bandwidth

• Each of the four channels operate at independent speeds for maximum flexibility (10, 20, 100, 200 Mbits/s)

• Full-duplex Fast Ethernet on UTP, for up to 800 Mbits/s throughput on one adapter

• NWay AutoSensing of maximum line speed on 10/100 TX adapters

• Duralink Failover offering FDDI-like port resiliency for optimum availability

Adaptec’s COgent Quartet adapters are the ultimate performance solution for PCI servers operating on Fast Ethernet and Ethernet networks. With four channels on a single board and full-duplex support, these adapters have the features today’s 32-bit Fast Ethernet and Ethernet servers need to handle user’s graphics, multimedia, database, and mission-critical applications.

These innovative adapters provide ample bandwidth for even the most demanding networks. They support up to four network segments on separate channels for a cumulative throughput of up to 800 Mbits/s (depending on model). The Quartet adapters multiply flexibility, performance, and cabling compatibility by four, and provide easy migration paths from Ethernet to Fast Ethernet network solutions.

The key features and benefits of the 4-port Ethernet adapters are:

1. *Scalability.* Using the Quartet adapter, IS organizations can increase the total network bandwidth without the need to buy a new server. Although the PCI bus offers 132 Mbits/s of bandwidth, typically only a limited number of slots are available thereby restricting the number of segments a server can handle. The Quartet adapters solve this problem by providing the power of four Fast Ethernet, or Ethernet, segments on a single adapter. These adapters use an on-board PCI-to-PCI bridge chip to extend the machine’s internal bus. Multiple Quartets can be installed on each server for even greater overall bandwidth.

2. *Flexible configuration.* The Quartet adapters support all four network channels at full cable bandwidth. Each channel is fully independent and is automatically configured by the system BIOS. Each of the adapters' four ID nodes can be customized at load time. The AutoSense feature of the 10/100-Mbit/s models allows the adapters to automatically detect the network's maximum line speed. No adapter configuration is necessary. For maximum flexibility, the AutoSense feature is compatible with all 10- and 100-Mbit/s devices, including those without NWay line speed negotiation. With the Quartet T4 model, cabling plant upgrades aren't necessary. They run on any UTP cable currently running 10BaseT and can be used in the future to run at 100 Mbits/s, providing an easy migration path.

3. *Resilient, redundant links.* Duralink Failover drivers included with each 10/100 TX adapter, ensure the integrity of mission-critical Ethernet server links. Duralink Failover establishes an FDDI-like port resiliency with both active and hot-standby links. In the event that the active link fails, the standby link is automatically activated to preserve the network link and maintain the server's availability. Duralink Manager, also included, is a data compilation tool that simplifies adapter monitoring and configuration.

4. *Full-duplex mode.* Full-duplex Fast Ethernet (FDFE) and full-duplex Ethernet (FDE) capabilities double the cumulative system throughput in Fast Ethernet and Ethernet networks. Servers using the Quartet adapters can handle, receive, and transmit requests simultaneously, over multiple ports so workstations benefit from much faster service.

5. *Designed for performance.* The Quartet adapters' efficient bus-master design minimizes CPU utilization for optimum throughput. Independent transmit and receive FIFO buffers (first-in, first-out memory) for each port onboard, and a powerful DMA capability mean consistently high performance, even during periods of peak network activity.

The following is a COgent-to-Adaptec Model Number Conversion Chart: COgent PCI Quartet

Previous model no.	**Bus speed**	**Interface 486, Pentium, other**	**Pentium-Pro**	
ANA-6944A/TX N/A	PCI 4 × 10/100 Mbits/s	RJ-45 (4)	X	XXXXXX
ANA-6944/T4	EM440T4-PCI	PCI 4 × 10/100 Mbits/s	RJ-45 (4)	X
ANA-6940/TX	EM400TX-PCI	PCI 4 × 100 Mbits/s	RJ-45 (4)	X
ANA-6904/BNC	EM964BNC	PCI 4 × 10 Mbits/s	BNC (4)	X
ANA-6904	EM964TP	PCI 4 × 10 Mbits/s	RJ-45 (4)	X

The 4-port Ethernet adapter technical specifications include

1. Systems supported: PCI Local Bus systems based on Intel, DEC Alpha, MIPS, and PowerPC processors.

2. Bus interface: PCI, 32-bit bus master (PCI 2.1 for ANA-6944A/TX).

3. Ethernet controller: DECchip LAN coprocessor chip (4).

4. FIFO buffer memory: (*a*) 10-Mbit/s models: 256 B transmit, 256 B receive (per port); (*b*) 100-Mbit/s models: (24 kB transmit, 4 kB receive per port).

5. Hardware interrupts: PCI interrupt A, supports shared interrupts.

6. Base IO or memory address: Assigned by BIOS.

7. PCI configuration space: Supports DWORD, WORD, and BYTE access.

8. Reliability: calculated MTBF >100,000 h.

9. Power requirements: 5 V at 2 A maximum (ANA-6904); 5 V at 3.5 A maximum (all other models).

10. Environmental operating range: 0 to 50°C.

11. Relative humidity: 5 to 85%, noncondensing.

12. Altitude: 3000 m maximum.

13. Interface connections: *10-Mbits/s models*—RJ-45 female (4), or BNC (4); RJ-45 supports Cat 3, 4, and 5 UTP and Type 1 STP; BNC supports RG-58 coax cable; *100-MBit/s models*—RJ-45 female (4); T4 model supports Cat 3, 4, and 5 UTP and Type 1 STP for 10BaseT and 100BaseT4 operation; TX models support Cat 3, 4, and 5 UTP and Type 1 STP for 10BaseT operation, and Cat 5 UTP and Type 1 STP for 100BaseT operation; full-duplex support on all RJ-45 connectors on all models for 10BaseT operation; additional full-duplex support on TX models for 100BaseTX operation.

14. Dimensions: 9.75 × 4 in to conform to PCI Long Card specifications (ANA-6904 models); 12 × 4.2 in to conform to PCI Long Card specifications (ANA-6944 models).

15. Standards compliance: IEEE 802.3 10Base2, IEEE 802.3 10BaseT, IEEE 802.3u 100BaseTX, IEEE 802.3u 100BaseT4, FCC Class A, CE Class A.

16. Drivers available: 3.5-in diskette—Novell NetWare 3.x, 4.x, and SFT III, Windows for Workgroups (NDIS 3.0), Windows 95, Windows NTO for Intel, DEC Alpha, PowerPC, and MIPS platforms, DOS NDIS (NDIS 2.0), OS/2 Warp (NDIS 2.0), MS LAN Manager, Artisoft LANtastic, Banyan VINES DOS client, FTP PC/TCP, DEC PathWorks, Sun Solaris, SCO OpenServer, SCO UnixWare (not all drivers are available for every adapter); Duralink for NetWare and Windows NT servers. [Duralink: 3.5-in diskette—Duralink Failover and Duralink Manager for ANA-6944A/TX (COgent EM 440TX PCI) at no charge.]

17. Diagnostic LEDs: link integrity for each channel (4); network activity for each channel (4).

18. Warranty: Adaptec’s COgent adapters are protected by a limited lifetime warranty.

For additional information about these and other Adaptec products, contact them at one of the following addresses:

Adaptec Asia
Block 1002
Jalan Bukit Merah #06-07
Singapore 159456
Phone: (65) 273-7300
Fax: (65) 273-0163

Adaptec Japan, Ltd.
Kiocho Hills, 4F
3-32 Kiocho, Chiyoda-ku, Tokyo, 102, Japan
Phone: (81) 3-5276-9882
Fax: (81) 3-5276-9884

Adaptec Europe
Dreve Richelle 161
Bldg. A, 2nd floor
B1410 Waterloo, Belgium
Phone: (32) 2-352-34-11
Fax: (32) 2-352-34-00

Additional Sales Offices
France
Phone: (33) 1-3452-3434
Fax: (33) 1-3452-3432

Germany
Phone: (49) 89-4564060
Fax: (49) 89-4560615

United Kingdom
Phone: (44) 1252-811200
Fax: (44) 1252-811212

United States (Miami)
Phone: (305) 265-1387
Fax: (305) 265-0399

Adaptec, Inc.
691 South Milpitas Boulevard
Milpitas, CA 95035
Phone: (408) 945-8600
Fax: (408) 262-2533

Additional sources are

• Literature: 1-800-934-2766 (USA and Canada) or 510-732-3829

• Presales support: 1-800-442-7274 (USA and Canada) or (408) 957-7274

• World Wide Web: http://www.adaptec.com

• Internet ftp server: *ftp.adaptec.com*

• Adaptec USA Bulletin Board Service (BBS): (408) 945-7727 (up to 28,800 bits/s, using 8 bits, 1 stop bit, no parity)

• Interactive fax: (303) 684-3400

Adaptec, the Adaptec logo, AHA, SCSISelect, EZ-SCSI, SCSI Interrogator, SCSI Tape Backup, SCSITutor, ThreadMark and Magic Lantern are trademarks of Adaptec, Inc., which may be registered in some jurisdictions. Microsoft, Windows, the Windows logo, and Windows 95 are registered trademarks and Windows NT and the Windows NT logo are trademarks of Microsoft Corporation used under license. All other trademarks used are owned by their respective owners. Information supplied by Adaptec, Inc., is believed to be accurate and reliable at the time of this printing, but Adaptec, Inc., assumes no responsibility for any errors that may appear in this document. Adaptec, Inc., reserves the right, without notice, to make changes in product design or specifications. Information is subject to change without notice.

17.2.18 Network Tape Drive (Sony)

Early in the design phase for this network I realized the need for a network tape drive after reviewing tape drive AIT, which forms the basis of Sony's new SDX series of tape drives and media. The ability to store large amounts of data and retrieve information quickly is critical in today's applications. Tape storage and retrieval has been revolutionized by Sony's award-winning Memory-in-Cassette (MIC) architecture. The MIC consists of a memory chip built into the data cartridge that holds the system's log and other user-definable information. Applications that benefit from MIC's capabilities include hierarchical storage management, video server, film editing, and real-time data acquisition.

SDX-300C features and specifications include

1. *High-speed data transfer.* With a recording density of 116 kB/in, the SDX-300C offers a sustained data-transfer rate of 3.0 MB/s (native) to over 6 MB/s (compressed). The drive uses a fast/wide SCSI with a burst transfer rate of 20 MB/s. The SDX-300C family has a large cartridge capacity; it can store 25 GB (native) or 50 GB (compressed) on a single 8-mm data cartridge.

2. *System design convenience.* The SDX-300C family includes a fast/wide SCSI interface with a 68-pin single-ended or differential interface or an 80-pin SCA connector and a self-cooling design.

3. *Exceptional reliability.* Utilizing Sony's extensive technology requires expertise in 8-mm tape recording. The SDX-300C products are designed to provide 200,000-h MTBF, an average head life of 50,000 h, and an average of 30,000 media uses.

4. *Memory-in-cassette.* Incorporating a flash memory chip within the data cartridge, the SDX-300C provides ultrafast media load and fast file search as well as the ability for applications to read and write to the memory chip.

5. *Media compatibility.* The SDX-300C family is designed exclusively for use with Sony's Advanced Metal Evaporated (AME) media technology incorporating pure cobalt metalization with a superdurable diamondlike carbon coating (DLC). AME tape is available with or without Memory-In-Cassette (MIC).

Specifications reflect 2:1 data compression. MTBF, head life, and media-use specifications are averages based on normal office environmental conditions. Actual experience may vary. Other specifications are

Drive type	3.5-in (8-mm) tape drive
Media	170-m AME tape: SDX-T3N (without MIC), SDX-T3C (with MIC)
Interface	SCSI-2 Fast/Wide, single-ended or differential
Data compression (SDX-300C)	IBM's Adaptive Lossless Data Compression (ALDC)
Capacity	25 GB (native); 50 GB (compressed)
Sustained transfer rate	3 MB/s (native); 6 MB/s (compressed)
Burst transfer rate	asynchronous: 5.0 MB/s; synchronous: 20.0 MB/s
Linear recording density	116,000 B/in
Recording block length	variable or fixed
Average media load time	20 s (without MIC); 10 s (with MIC)
Average access time	55 s (without MIC); 27 s (with MIC)

Search speed	60 in/s (without MIC); 120 in/s (with MIC)
Rewind speed	120 in/s (150 times nominal)
Drum rotational speed	4800 rpm (r/min)
Buffer size	4 MB
Uncorrectable error rate	Less than 10 to 17 bits
MTBF	200,000 power-on hours (POH)
Average R/W head life	50,000 tape contact hours
Media uses	30,000 end-to-end passes
Vibration (operating)	0.25 G peak, half sine wave of 50 to 500 Hz (swept)
Shock (operating)	5.0 G peak, half sine wave of 3-ms duration
Altitude	10,000 ft
Maximum wet-bulb temperature	26°C
Operating temperature	5 to 40°C (40 to 104°F)
Storage temperature	–40 to 70°C (–40 to 158°F)
Operating humidity	20 to 80% (noncondensing)
Power requirement	Dc 5 V ±5 percent, dc 12 V ±10 percent (internal); ac 100 to 200/220 to 240 V (external)
Power consumption	12 W (average, internal); 21 W (average, external)
Dimensions ($h \times w \times d$)	1.62 × 4.0 × 6.10 in (41.1 × 101.6 × 154.9 mm) internal; 2.28 × 7.44 × 10.31 in (57.9 × 189 × 261.9 mm) external
Weight	1 lb, 11 oz (765 g) internal; 4 lb, 14 oz (2.2 kg) external
Models	SDX-300C 68-pin single-ended; SDX-302C 80-pin SCA; SDX-310C 68-pin differential; SDX-S300C/NB external single-ended

Specifications reflect 2:1 data compression. MTBF, head life, and media-use specifications are averages based on normal office environmental conditions. Actual experience may vary. Nonmetric weights and measurements are approximate.

17.2.19 Computer Cabling (Belkin)

This network is no different from the rest; for example, there is always a need for special cabling and adapters. For this need I elected to use Belkin. Belkin has a wide variety of peripheral cabling and accessories, and they make the best products on the market.

The following is a brief list of the components I used from Belkin:

• DB9 female/female gender bender and DB9 male/male gender bender

- DB15 female/female gender bender and DB15 male/male gender bender
- SCSI-2 interface cable
- SCSI DB68 terminator
- DB25 male/male gender bender and DB25 female/female gender bender
- IEEE 1284 printer cable 20 ft and 35 ft
- DB25 serial cable
- DB9 serial cable
- Super VGA monitor extension cable

All these cables and accessories, and others, were used in this network. I selected Belkin because they make the best computer and printer cables on the market. They also have the best designed and manufactured gender benders.

I strongly recommend that you consider Belkin when you need gender benders and cables, especially when cables are required to meet specifications such as the IEEE 1284. You can reach them at www.belkin.com or

Belkin Components
501 West Walnut Street
Compton, CA 90220-5030

17.2.20 Infrared Network Interface (JetEye)

For an infrared network interface I selected JetEye, by Extended Systems. This device enables portable workstations to be moved about and perform network printing via their infrared port. It is probably one of the most cost effective methods of network connection to meet the need for printing available.

The JetEye features include (1) line-of-site infrared transmission for any infrared portable computer; (2) support for Windows NT, Windows 95, Windows 98, TCP/IP, NetWare, LAN Manager, and LAN Server; and (3) supports for operating distances up to 3 ft away.

The JetEye used in this network is dedicated to the ThinkPads which are remote workstations as well as portable. Each has been configured to operate with the JetEye infrared network interface.

For more information, contact

Extended Systems
5777 North Meeker Avenue
Boise, ID 83713
Phone 800-235-7576

17.2.21 Cable Power Protection (Tripp-Lite)

Power protection in any computing, network, or peripheral equipment is not optional. I have a standard reply when asked if such-and-such needs to be protected with power protection equipment: “If you can afford to throw away any or all of your equipment, replace it, and not be negatively impacted by downtime, then don’t use power protection equipment.” With that in mind, let us examine the power protection equipment used in this network.

I determined the following were needed to adequately protect the network components: (1) data shield parallel dataline surge suppressors, (2) data shield 10BaseT surge suppressors, (3) data shield dial-up line surge suppressors, (4) data shield AUI 802.3 surge suppressors, (5) data shield DB9 surge suppressors, and (6) data shield DB-25 serial surge suppressors.

The data shield protectors are used at all points in the network. These inline devices are used with serial, parallel, 10BaseT, AUI, modem cable, and DB-9 connections. Additionally, various electrical protection equipment also has telephone line spike/surge protection that is used to protect all incoming telephone lines.

Tripp-Lite has other power protection equipment. You can contact them at www.tripplite.com or

Tripp-Lite
500 North Orleans
Chicago, IL 60610

17.3 Software and Firmware

17.3.1 Microsoft Windows 95 and 98

Software used in this network includes Windows 95 and Windows 98 operating systems. Both work well with all the hardware and software integrated together.

In fact, Windows 3.1 is used in the network as well. Rather than pro vide a lengthy explanation of these operating systems, contact the following Web address for more information: www.microsoft.com.

17.3.2 Microsoft Windows NT

The network designed and explained in the following chapters includes Windows NT operating systems. The following NT versions are used: Server version 4.0, Workstation version 4.0, Enterprise Edition, Server version 5.0, and Workstation version 5.0.

You can contact Microsoft at the following to obtain more information: www.microsoft.com.

17.3.3 UNIX Operating System (SCO)

In addition to Microsoft operating systems, I used the Santa Cruz Operation (SCO) flavor of UNIX. SCO's Web site is very resourceful. The following information is pertinent to the topic of this book and current at the time of this publishing. This information was obtained from the SCO Web site on the Internet.

> UnixWare system solves year 2000 problem, Santa Cruz, CA (June 3, 1998)—SCO (NASDAQ:SCOC) today announced that its UnixWare operating system has solved the "Year 2000 Problem" for nearly 100 banks throughout the US. By upgrading the banks from their existing SVR4 technology-based UNIX systems to the UnixWare system, SCO VAR, Precision Computer Systems (PCs), has eliminated one of the most vexing software problems since the dawn of the Information Age.
>
> "For years banks have designed business-critical systems based on UNIX, and for a reason," said Ray Anderson, SCO's senior vice president, Marketing. "UNIX is trusted and stable, and banks cannot afford to run these environments on less reliable alternatives. Since early last year UnixWare has provided a clean year 2000 solution, offering compatibility with earlier UNIX systems and the value of standard Intel hardware."
>
> Located in Sioux Falls, South Dakota, PCS is a leading VAR and software provider to community banks nationwide. The banks require business-critical systems to handle customer checking accounts, savings deposits, loans and other crucial financial data. Most of the banks were running an early SVR4 technology-based UNIX system from Unisys. UnixWare to the Rescue.

The easiest, most effective solution for solving the problem was an upgrade to the SCO UnixWare system. PCS performed the installations after hours, and the banks' new systems were up and running. In addition to the UnixWare system, most banks purchased new hardware, such as an Unisys Aquanta Server, with either a Pentium 200, 233, or 266 MHz processor, and between 64 and 128 Mbytes of memory.

"Year 2000 issues are fast becoming the number one concern of many CIOs and IT professionals in the enterprise. Upgrading to our latest Aquanta servers and UnixWare provided these customers with a solution that resolved their Year 2000 issues," said Trevor Westoby, director of Marketing programs for Unisys Aquanta Business Systems.

Benefits of UnixWare System. The new systems provide the banks with a number of major benefits. The most crucial benefit is the fixing of the Year 2000 Problem. The new system solves the problem completely, enabling the banks to rest easy as the millennium approaches. Another major benefit is the increase in performance. The new system has reduced the processing time of daily updates, a critical banking function, by a factor of 10 to 1. This allows bankers to have immediate access to complete updated customer records.

"The UnixWare and Unisys combination not only saved many of our banking customers from certain disaster, but they will enjoy overall cost savings and more efficient processing," said Norm Meyer, National Sales Manager from PCS. "It's great to know that there's a relatively simple solution to the Year 2000 problem—install UnixWare."

What's the year 2000 Problem? The year 2000 Problem is a phrase used around the world to characterize a software issue that is deceptively simple, but has wide-ranging consequences. The heart of the problem is that many IT systems record dates using only two digits for the year. For example, 1984 is recorded as 84. This works as long as all dates are in the same century. In the year 2000 the natural sequence breaks down. Two-digit years become ambiguous—does 84 represent 1984 or 2084? Other problems include the possible continued use of twentieth century calendars (such as interpreting the year 2000 as 1900), and incorrect calculations of Leap Year days.

About SCO. SCO is the world's number-one provider of UNIX server operating systems, and the leading provider of network computing software that enables clients of all kinds—including, PCs, character terminals, and NCs—to have Webtop access to business-critical applications running on servers of all kinds. SCO designed Tarantella software, the world's first application broker for network computing. SCO sells and supports its products through a worldwide network of distributors, resellers, systems integrators, and OEMs. For more information, see SCO's WWW home page at: http://www.sco.com/.

SCO, The Santa Cruz Operation, the SCO logo, Tarantella, the Tarantella logo and UnixWare are trademarks or registered trademarks of The Santa Cruz Operation, Inc., in the United States and other countries. UNIX is a registered trademark of The Open Group in the United States and other countries. All other brands or products are or may be products or services of their respective owners.

17.3.4 Network Security and Virus Protection Software [McAfee (now Network Associates)]

Computer and network security is probably the single most important issue today. I predict it will become three to five times as important as i t is today. Viruses, bots, and all sorts of antidata objects exist within the Internet. Most people have no idea how vulnerable parts of the Internet are. Even "service providers" are more vulnerable than they will admit. The sad fact is every company, yes, *every* company, I know of has disinformation arsenals to make people feel secure. Management at the higher levels in most corporations operates in ignorance of these matters because it is legally safe for them and prudent that they do so; ask and this statement will be denied, but remember where you read it. I point this out because I do not want you to be ignorant. There is no magic *program* or anything else that can make networks safe. Good programs exist, the ones chosen and implemented in this network are an example; however, no single program can make your network 100 percent immune.

Remember this during the design phase of your network. Networks can have security designed into them from the outset. Security in your network needs to be factored into every area from electricity provision, telephone access, and every other aspect that categorizes your network.

McAfee software suite was selected to meet the needs of this network. Part of the reason for this is the amount of anti-virus programs and information they have—and it works. The second reason for selecting McAfee was because of the frequency by which they update their antivirus software. At present McAfee has over 250 highly technical documents available on viruses; likewise they claim to have information about the 1,000 most common viruses. When the security analysis for this network was complete, the following software packages were selected and are used in this network:

VirusScan	McAfee Service Desk
Desktop Security Suite	NetShield
Commuter	WebScan
QuickBackup	PCCrypto

These products have been implemented to varying degrees on each system. Each program's benefits and highlights are presented here.

1. *VirusScan.* This program may well be the most popular anti-virus software in the marketplace today. It operates with Windows 3.1, 95, Windows NT4.0, DOS, and OS/2. It is software that, once installed, operates automatically on power-up. It can be used at will once a system is operational. This program requires minimal space but does a professional job. This program is NCSAA certified.

2. *Desktop Security Suite.* This program also operates with Windows 3.1, 95, and NT4.0. This suite of programs includes antivirus software, backup abilities, and encryption technology. The virus pro gram is VirusScan. QuickBackup operates with ZIP, Jaz, the Internet, or rewritable CD-ROMs. The backup program enables hourly backup or on demand, whichever is best for you. The cryptographic part of the suite provides 160-bit encryption. This part of the suite enables users to encrypt files before they are sent over the Internet. The PC cryptographic part also permits network traffic to be encrypted between Windows-based computers and those running UNIX.

3. *Commuter.* Commuter is more than a communication software. It also includes virus protection, desktop storage management, electronic mail, personal information organizer, calendar, to-do list, and a contact manager.

4. *QuickBackup.* This backup program works with Windows 95 and NT4.0. It enables transparent backup of files to SCSI,ZIP, and Jaz drives. An icon-driven program makes for ease of use. The program installs quickly and works well. It does provide encryption protection and Internet support.

5. *Service Desk.* The service desk product is powerful. It is actually multiple products in one box. It works with Windows 3.1, 95, and NT4.0. The product lets customer support personnel have access to information about the customer and make a remote connection to a system reported with a problem. The package comes with the ability to distribute software. The package also includes a system diagnostic part for support personnel to use with customers.

6. *NetShield.* NetShield uses McAfee's proprietary code, called *Code Trace*, *Code Matrix*, and *Code Poly*. The product actually operates in an NT environment in native mode. The program takes full advantage of NT's server/client remote task distribution capability. The product supports real-time scanning while operation of other tasks run.

7. *WebScan.* The WebScan product is designed to detect viruses within a browser. It examines downloads and email attachments, making it a power addition to any desktop or laptop system communicating in networks today. It also provides a “cybersitter” that blocks out unwanted Web sites and chat groups. Coverage of the program includes examination of `.doc, lzip. exe. zrc. arj.,` and other file types.

8. *PCCrypto.* PCCrypto is used to secure documents and other data files created by anyone using computers. It can encrypt graphics, spreadsheets, and text documents. It uses a 160-bit blowfish encryption mechanism. The package consumes a minimal amount of space and is one of the most powerful, if not the most powerful, tools of its kind on the market today.

I recommend that you dedicate a system for testing software, initially. Then I recommend installing McAfee VirusScan. Then scan each and every diskette you have. That’s right. It does not matter if it takes someone 2 months. Even new diskettes out of the box. Every diskette you receive from a manufacturer must be scanned. You say: “But Ed, isn’t this going a little too far”? I’ll say this. One year (I won’t say which) I bought some software; it was new. It came from the original vendor in shrink-wrapped plastic. The shrink-wrapped diskettes were enclosed in a box with a seal on it. The box with a seal on it was shrink-wrapped. I didn’t think a thing about it. Within 10 minutes after opening the software, it brought one of my systems to its knees. The diskette had a virus. How do I know? I checked it personally. How do I know the system it went into was “clean”? It was and is my benchmark system. Furthermore, those who know me know that nobody, not anyone, puts a disk in my systems except me. It took me 2 days to recover the system. Consider this next time you stick a new program on diskette in your system. You can pay now or gamble, but remember the odds are against you.

McAfee has other products that may meet your needs. I recommend you contact them. My experience with them has always been pleasant and their staff very informative. They can be reached at www.mcafee.com or

McAfee
2710 Walsh Avenue
Santa Clara, CA 95051
Phone: 408-988-3832

McAfee (UK) Ltd.
Hayley House, London Road
Bracknell, Berkshire
GR12 2TH United Kingdom

McAfee Canada
178 Main Street
Unionville, Ontario
Canada L3R 2G9

McAfee Europe B. V.
Orlypein 81—Busitel 1
1043 DS Amsterdam
The Netherlands

McAfee France S. A.
50 rue de Londres
75008 Paris
France

McAfee Deutschland GmbH
Industriestrasse 1
D-82110 Germering
Germany

17.4 Summary

This chapter has presented the equipment used in the author’s network. It is important to do most of the initial network planning on marker board and/or paper. With TCP/IP and SNA protocols, the planning phase is important. Once this is done, one can begin to acquire equipment and build the network. I think it is best to design the network, acquire the equipment, and then build the network. This approach makes the installation, operation, and maintenance much easier.

18
Typical Network Components and Configuration

This chapter focuses on some specific components used in the network and the role each plays in the overall network functionality. The components listed in Chap. 17 were obtained and integrated into the network. However, prior to beginning this work I took ample time to plan the network. TCP/IP networks can be complex to manage, but they don't have to be. A little planning goes a long way.

18.1 Network Design

I began with this network design the same way I have all others: at the drawing board—literally. I use a marker board to work out my ideas. It can take days, sometimes weeks, for me to sift through the requirements, my thoughts, and variations of what equipment is needed and how it fits into the overall design scheme.

The network design explained in this chapter is based more on principle and specific product components that implement certain technology. In other words, I design networks on the basis of principles and fundamentals, not the hardware or software highlight or fad of the week or month. The purpose of the network design here is to meet current needs and sustain growth. Let me repeat: The original design intent of this network is to do what *I want it to do now* and be flexible enough to change and accommodate growth in the near future. *It is not designed for anyone else's criteria.* In general, any network design should be approached with this in mind, and in this case, I created the network for use as an example of how TCP/IP networks operate.

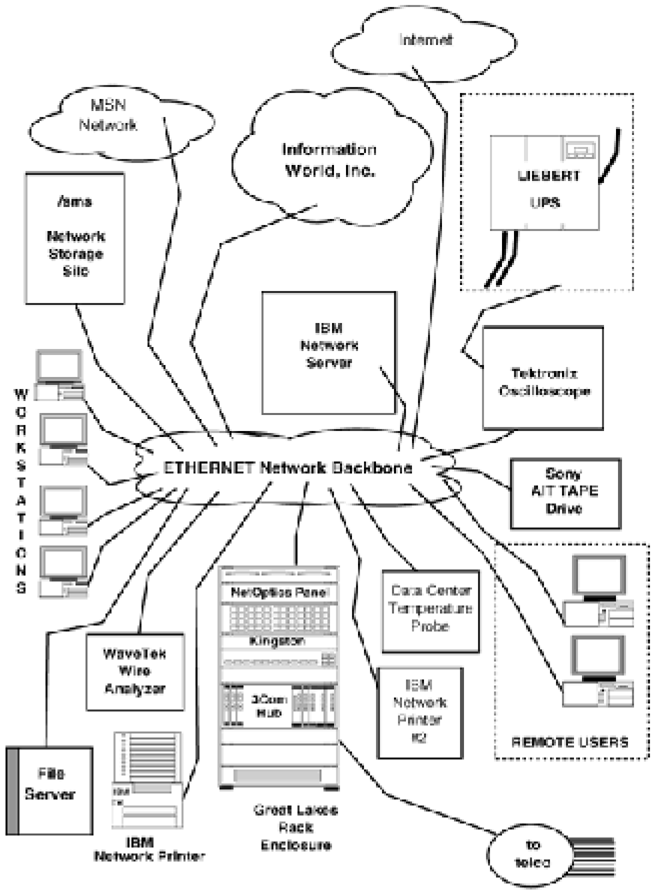


Figure 18-1
Highlighted network view.

Before examining the components of the network, consider the highlighted network view shown in Fig. 18-1.

Note the numerous network components shown in Fig. 18-1. Not all the network components are shown but will be later in this chapter and the next, including components such as network interface cards, network wiring, and electrical wiring. However, the figure does show the overall logical design of the network. It is a good beginning point to understand the overall network design. Inspection of the figure may tend to imply a single point of failure in a given place, but redundancies have been built into it. Further details are presented later.

This is the logical network design because it was driven by user requirements. This network enables users to exchange files; email; and remote logons to systems such as servers, search network server repositories, and network printing.

18.2 Component Overview

Before examining each component individually, in this section let us take a look at the network components and their vendors:

- Temperature probe (Fluke)
- Oscilloscope/digital multimeter (Tektronix)
- Network analyzer (Internet Advisor: Hewlett-Packard)
- Rack enclosure (Great Lakes Cabinets)
- Commercial desktop computers (IBM)
- Remote workstations (IBM ThinkPads)
- Data-communications network hub (3Com USRobotics)
- Ethernet network hubs (Kingston)
- Patch panel (NetOptics)
- Network power infrastructure (wiring) (Thomas & Betts)
- Network server (IBM Netfinity 7000)
- Operating system software (Microsoft and UNIX)
- Network printer (IBM)
- Network and computer security (McAfee)
- Multimedia components (Creative Labs)
- Network storage (SMS Data Products Group)
- Network wiring analyzer (Microtest)
- Troubleshooting network analyzer (Hewlett-Packard)
- Network tape drive (Sony)
- Power protection (Liebert)
- Computer cables and accessories (Belkin)

• Software troubleshooting tools (Microsoft)

• Miscellaneous devices and tools

These components, and others presented later, have been put together to make the network possible.

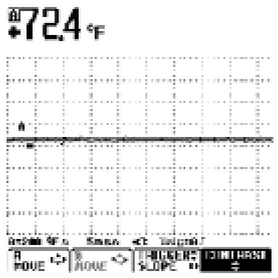


Figure 18-2
Initial temperature reading 1.

18.3 Temperature Probe (Fluke)

One important aspect of network planning and network maintenance after a network is installed is knowing the temperature of the operating environment. Consider Fig. 18-2.

Figure 18-2 shows an initial reading at the beginning of a business day in the network data center. Note that the temperature reading is displayed in degrees Fahrenheit.

Consider Fig. 18-3.

The reading in Fig. 18-3 was taken with the same instrument and in the same location, as in Fig. 18-2, a couple of hours after all equipment was powered on and operating. Note that the temperature has risen by approximately 4 degrees. Now consider Fig. 18-4.

The reading shown in Fig. 18-4 was taken with the same instrument, in the same place, only later during the same day. Note that the tem perature now has climbed 6 degrees, and is now 82.3°F. Since the initial reading just a short time earlier in the day the room temperature has increased by 10 degrees.

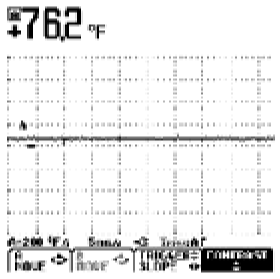


Figure 18-3
Temperature reading 2.

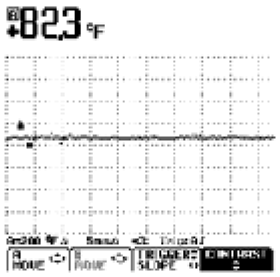


Figure 18-4
Temperature reading 3.

At the time of this reading no cooling fans, blowers, or air circulation was operating. The room was an average room with normal central air conditioning and ventilation. Now consider Fig. 18-5.

Before midday the room temperature is at a shocking 93.3°F. At this point air blowers and cooling fans were turned on, and airflow to the room where the readings were taken increased. This temperature is far above what should be normal operating temperature.

Normally, the ambient temperature would not reach this level. However, in certain places where equipment is concentrated in a given area, the area temperature around such space can be much higher than the room temperature say 15 or 20 ft away in another location in the same room. The significance of these readings should not be overlooked. It proves a point that airflow, cooling, and various air-circulation equipment is important to maintain stable and normal operating temperatures throughout a room in which equipment is concentrated.

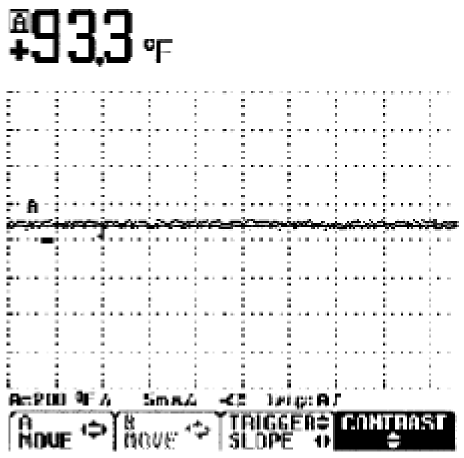


Figure 18-5
Temperature reading 4.

This is a good example of why rack-mount cabinets are important. With well-designed rack-mount cabinets, equipment can be placed such that the blowers and fans create a positive airflow and thus maintain temperatures at a level safe for the equipment and personnel working with it.

Parallel to this need to monitor temperature surrounding equipment is another important aspect of network design: free space around equipment to provide adequate airflow. Many vendors supply this free-space recommendation with the specifications for their equipment. Be sure to ask for this information if it is not stated with each piece of equipment you plan to install in your network.

The instrument used to obtain the temperature here was provided by Fluke, who can be reached at www.fluke.com or

Fluke Corporation
P.O. Box 9090
Everett, WA 98206

or

Fluke Europe B.V.
P.O. Box 680
7600 AR, Alemelo
The Netherlands

18.4 Oscilloscope/Digital Multimeter (Tektronix)

Because of the complexity of the network electrical planning, implementation, and maintenance, I determined that I needed a powerful tool to work with a range of readings; in short, I needed an oscilloscope and a digital multimeter. I also needed the ability to upload readings taken onto a computer for storage and printout. After researching the available products in the marketplace I selected the Tektronix model THS720P Tekscope Isolated Channel Scope/DMM to use in this network. It includes two devices in one: an oscilloscope and a digital multimeter. These are implemented as distinct and separate functions. In addition, the oscilloscope has two channels which makes for powerful analysis.

Some features, capabilities, and specifications about this tool are presented here. It is a 100-MHz-bandwidth and 500-MS/s (million samples per second)-sample-rate digital real-time oscilloscope/true rms digital multimeter. The scope and meter modes can operate simultaneously and independently on the same or separate signals. Some of the features it includes are cursors; video trigger; voltage and resistance measurements; and storage of waveforms, data, and instrument setups. It can be used to test correct operation of motors, transformer efficiency, and power-supply performance, and to measure the effect of neutral current. Some of the functions and specifications of the Tektronix scope include

1. *General properties.* Bandwidth—100 MHz; sample rate (each channel)—500 MS/s; channels—two; sensitivity—5 mV to 50 V/div (to 500 V/div with 10× probe); position range—10 div; dc gain accuracy—2%; vertical resolution—8 bits; record length—2500 points; time/division range—5 ns to 50 s/div; horizontal accuracy—200 ppm; roll mode >/=0.5 s/div; autorange—user-selectable; trigger modes—auto, normal; trigger types—edge, pulse, video, motor, external; video trigger formats and field rates—odd field, even field, and line; motor trigger—triggers on 3- and 5-level PWM power signal; external trigger input—5 MHz TTL-compatible; harmonics—up to 31st (30 to 450 Hz); waveform processing—add, subtract, multiply, calculate watts=$V \times I$; waveform storage—10-waveforms capacity; acquisition modes—sample, envelope, average, peak detect; cursor measurements—DELTAvolts, DELTAtime, 1/DELTAtime (Hz), degree (phase); cursor types—horizontal and vertical bars, paired (volts and time); display system—interpolation, sin(x)/x; mode—vector, dot, vector accumulate, dot accumulate; format—*YT* and *XY.*

2. *Automatic measurements.* Period: +width, –width, +duty cycle, –duty cycle, high, low. Frequency: rise and fall time, +overshoot, –overshoot, maximum, minimum. Peak-to-peak amplitude: mean cycle-mean burst width, rms, cycle rms.

3. *Power measurements.* W—true power; VA—apparent power; VAR—reactive power; V—volts (rms, peak), A—amperes (rms, peak); THD-F—total harmonic distortion as a percentage of the fundamental, THD-R—total harmonic distortion of the rms of the input signal; PF—power factor, DPF—displacement power factor; PHI—phase difference between voltage and current.

4. *DMM specifications.* Dc voltage ranges—400.0 mV to 880 V; dc volts accuracy—0.5% of reading+5 counts; ac volts accuracy—2% of reading+5 counts; true rms ac voltage ranges—400.0 mV to 640 V; maximum float voltage—600 V rms each channel (probe-dependent); resolution—4000-count, 3.75 digits; resistance ranges—400.0 Ω to 40.00 MΩ; resistance accuracy—0.5% of reading+2 counts, 40 MΩ 2% of reading+5 counts; diode test range—0 to 2 V; continuity check—audible tone when <50 Ω; modes—minimum, maximum, $\delta_{\max} - \Delta_{\min}$, average, hold; nonvolatile storage—10 DMM screenshots; external trigger input—5 MHz TTL-compatible; vertical zoom capability—2×, 5×, 10×; dB scale—selectable, referenced from 1 mV to 10 V; dBm scale—selectable, referenced from 50 to 600 Ω.

5. *General specifications.* Setups—10 front-panel setups; safety certification—UL 3111-1-listed, CSA-certified, complies with EN61010-1; power—NiCd rechargeable battery pack and ac adapter; battery life—~2 h from full charge; display—backlit LCD; display resolution—320 × 240.

To set up a baseline for future reference of the electrical characteristics for the network, I took sample readings prior to network equipment implementation. Consider Fig. 18-6.

Figure 18-6 is a random voltage reading in the data center prior to installation of the Liebert UPS power conditioning equipment. Note that the ac voltage is 119.4 and the frequency reading 59.95. Approximately 30 min later another reading was made. Consider Fig. 18-7.

Figure 18-7 shows a reading of 121.2 V ac. It also shows a waveform of 118.4 V ac rms. Still another reading was taken a short time later in the same place, with the same tool. Consider Fig. 18-8.

Figure 18-8 shows a reading of 120.3 V ac and a 59.95 frequency reading. These readings are in close proximity to those shown in Fig. 18.7. Actually, in this particular network environment, the power supplied by the utility company is relatively stable.

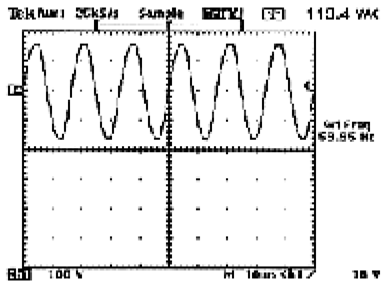


Figure 18-6
Baseline electrical reading 1.

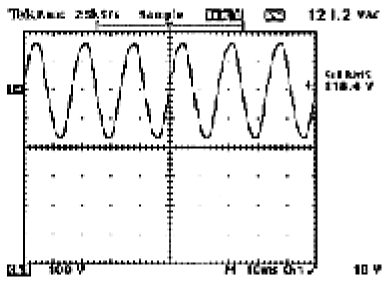


Figure 18-7
Baseline electrical reading 2.

I recommend that you take readings like these at your site to obtain a baseline reference for the future. Your site might require 10 times the number of readings to establish a better snapshot of the electrical characteristics of the location in which your network equipment is installed. Regardless of how many snapshots you feel are needed, this is a valuable tool to establish a baseline for power quality and then continue to take readings like these periodically over time.

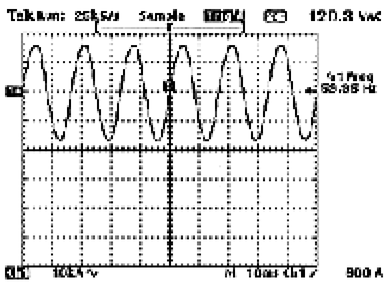


Figure 18-8
Baseline electrical reading 3.

18.5 Network Analyzer (Internet Advisor: Hewlett-Packard)

This network is similar to other networks in that it requires a network analyzer for adequate information to be obtained to maintain it properly. If you are relatively new to networking, you may want to refer back to this section later for it contains information that is considered advanced.

I selected the Internet Advisor for this network because of its strength and effectiveness. Consider the following information. The HP Internet Advisor (J2522B) provides comprehensive Ethernet testing. It is a portable, full-featured network-analysis solution that allows users to install, support, and maintain local area network (LAN), wide area network (WAN), and asynchronous transfer mode (ATM) networks by providing features that make troubleshooting any network easier. Some of these features are a portable platform with a rugged built-in 486 PC, commentators that help solve network problems quickly and effectively, seven-layer decoding of all major protocols, comprehensive network statistics, and traffic generation. Other features include

1. *Mainframe characteristics.* Intel 486 DX4, 100-MHz processor; 16 MB of PC memory; VGA color display, 26.4-cm (10.4-in) diagonal or active-matrix display (as an optional feature); 814-MB hard drive (as standard) or 1.4-GB hard drive (as an optional feature); 3.5-in flexible (floppy) disk drive; PCMCIA slot (type I/II); built-in mouse; serial, parallel, and external VGA monitor ports; Windows 95.

2. *General specifications.* Data rate 10 Mbits/s; two RJ-45 connectors with hub logic allowing for testing in switched environments; AUI connector for universal Ethernet testing through external transceivers.

3. *Hardware filtering.* Hardware timestamp with 100-ns resolution.

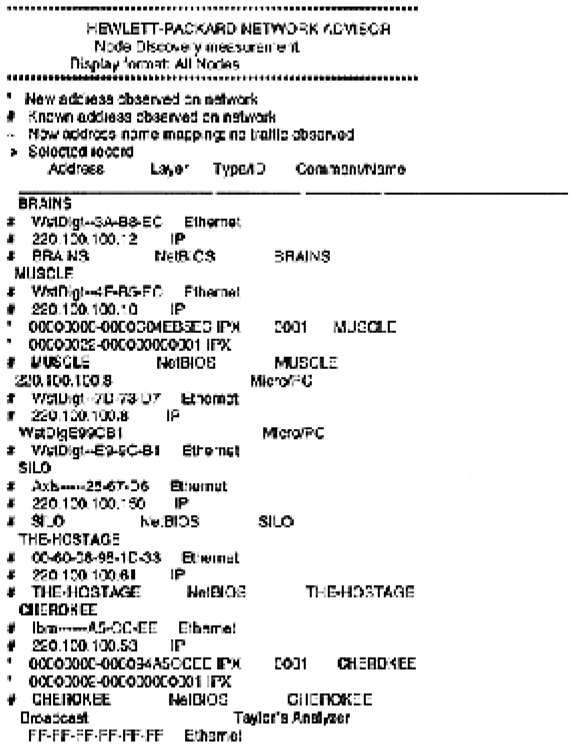
4. *Physical characteristics.* weight-6.2 kg (13.7lb); dimensions—8.5 × 30 × 29 cm (3.4 × 12 × 11.5 in) (*h* × *w* × *d*); display—26.5-cm diagonal (10.4-in) passive DSTNcolor LCD VGA; optional 26.5-cm diagonal (10.4-in) active-matrix TFT color VGA.

5. *Operating conditions.* (*a*) *Temperature*—operating, 5 to 40°C (41 to 104°F); nonoperating –25 to 60°C (–13 F to 140°F); (*b*) *humidity*—operating 20 to 80% relative humidity (RH) to 40°C noncondensing; (*c*) *storage*—10 to 90% RH to 60°C noncondensing; (*d*) *power requirements*—100 to 240 V ac, 50 to 60 Hz, 75 W maximum; (*e*) warranty—3 years; (*f*) *regulatory compliances*—bears the CE and CSA marks.

The particular model I used was the HP J2522B Internet Advisor LAN-Ethernet with Opt 221 26.4 cm active matrix color display, HP J2514A Deluxe Carrying Case, HP J2533A Token Ring-Parallel-Port Adapter (International), and HP J2531A Internet Reporter-LAN.

Consider Fig. 18-9.

Figure 18-9 provides a lot of information. The purpose of this trace is to obtain the systems on the network. The name of the screen, *node discovery measurement,* appears at the top of the figure; the names of the network computers participating in the network I created—BRAINS, CHEROKEE, MUSCLE, SILO, and THE-HOSTAGE, appear within the body of the figure, followed by the IP addresses and functions (broadcast, etc.) of each.

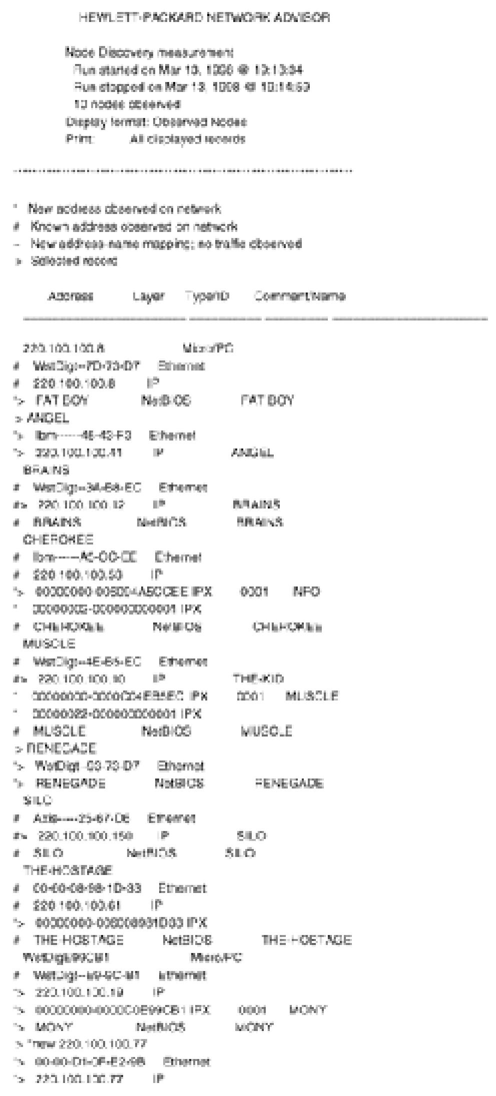


```
****************************************************************
            HEWLETT-PACKARD NETWORK ADVISOR
              Node Discovery measurement
          Display format: All Nodes
****************************************************************
*  New address observed on network
#  Known address observed on network
-- New address name mapping: no traffic observed
>  Selected record
      Address          Layer    Type/ID      Common/Name
 ___________________________ __________ _________ ____________________
 BRAINS
#  WstDigt--3A-B8-EC    Ethernet
#  220.100.100.12       IP
#  BRAINS             NetBIOS          BRAINS
 MUSCLE
#  WstDigt--4F-RG-FC    Ethernet
#  220.100.100.10       IP
*  0000000C-00C00MEB3ED IPX      0001     MUSCLE
*  0000002E-00C0000C0001 IPX
#  MUSCLE             NetBIOS          MUSCLE
 220.100.100.8                       Micro/PC
#  WstDigt--7D-73-D7    Ethernet
#  220.100.100.8        IP
 WstDigE99CB1                          Micro/PC
#  WstDigt--E9-9C-B1    Ethernet
 SILO
#  Axis----25-67-D6    Ethernet
#  220.100.100.150      IP
#  SILO            NetBIOS          SILO
 THE-HOSTAGE
#  00-60-08-95-1D-33    Ethernet
#  220.100.100.61       IP
#  THE-HOSTAGE        NetBIOS          THE-HOSTAGE
 CHEROKEE
#  Ibm-----A5-OC-EE    Ethernet
#  220.100.100.53       IP
*  0000000C-000034A5OCEE IPX      0001     CHEROKEE
*  0000000E-00C0000C0001 IPX
#  CHEROKEE           NetBIOS          CHEROKEE
 Broadcast                        Taylor's Analyzer
   FF-FF-FF-FF-FF-FF    Ethernet
```

Figure 18-9
Node discovery measurement.

Now consider Fig. 18-10.

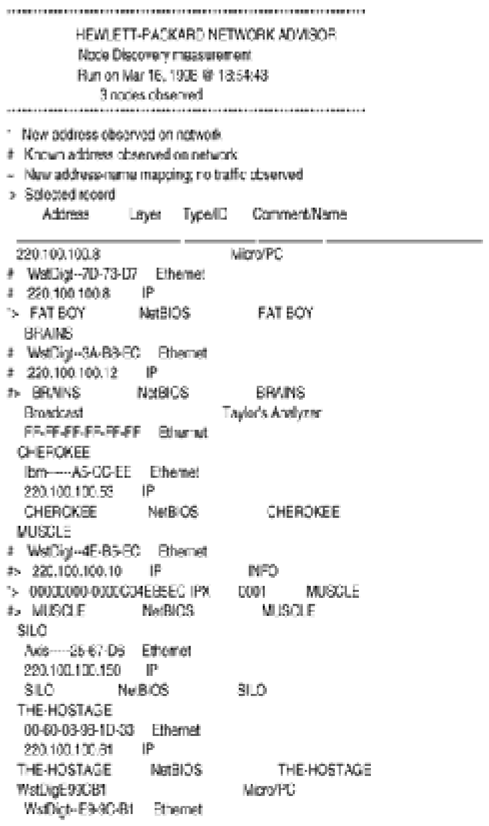
Figure 18-10 shows additional network information in a later snapshot of nodes on the network at a later time. Note the systems partici pating on the network at this point in time: BRAINS, MUSCLE, SILO, THE-HOSTAGE, and CHEROKEE.



```
            HEWLETT-PACKARD NETWORK ADVISOR

          Node Discovery measurement
            Run started on Mar 13, 1998 @ 10:13:34
            Run stopped on Mar 13, 1998 @ 10:14:53
            10 nodes observed
          Display format: Observed Nodes
          Print:        All displayed records

-------------------------------------------------------------------

*  New address observed on network
#  Known address observed on network
-  New address-name mapping; no traffic observed
>  Selected record

     Address          Layer    TypeID     Comment/Name
   ____________________ _________ ________ ___________________

  220.100.100.8                 Micro/PC
#   WstDigt--7D-73-D7    Ethernet
#   220.100.100.8        IP
*>  FAT BOY             NetBIOS          FAT BOY
> ANGEL
*>  Ibm-----46-43-F3    Ethernet
*>  220.100.100.41       IP              ANGEL
  BRAINS
#   WstDigt--34-68-EC    Ethernet
#>  220.100.100.12       IP              BRAINS
#   BRAINS             NetBIOS          BRAINS
  CHEROKEE
#   Ibm-----AC-OC-CE    Ethernet
#   220.100.100.50       IP
*>  00000000-00800-A5CCEE IPX     0001     NFO
*   00000002-000000000001 IPX
#   CHEROKEE            NetBIOS          CHEROKEE
  MUSCLE
#   WstDigt--4E-B5-EC    Ethernet
#>  220.100.100.10       IP              THE-KID
*   00000000-0000C04EB5EC IPX     0001     MUSCLE
*   00000022-000000000001 IPX
#   MUSCLE             NetBIOS          MUSCLE
> RENEGADE
*>  WstDigt--03-73-D7    Ethernet
*>  RENEGADE            NetBIOS          RENEGADE
  SILO
#   Aztk-----25-67-D6    Ethernet
#>  220.100.100.150      IP              SILO
#   SILO               NetBIOS          SILO
  THE-HOSTAGE
#   00-00-08-98-1D-33    Ethernet
#   220.100.100.61       IP
*>  00000000-00800891D33 IPX
#   THE-HOSTAGE         NetBIOS          THE-HOSTAGE
  WstDigt99CB1                  Micro/PC
#   WstDigt--e9-9C-B1    Ethernet
*>  220.100.100.19       IP
*>  00000000-0000C0E99CB1 IPX     0001     MONY
*>  MONY               NetBIOS          MONY
> *new 220.100.100.77
*>  00-00-D1-0F-E2-9B    Ethernet
*>  220.100.100.77       IP
```

Figure 18-10
Additional network information.

The difference between Figs. 18.9 and 18.10 reflects the time when I took the reading from the network. This level of information is powerful for measuring network loads and network nodes using the network and for obtaining tangible information to create baseline readings for future reference. Consider Fig. 18-11.



```
-----------------------------------------------------------
            HEWLETT-PACKARD NETWORK ADVISOR
            Node Discovery measurement
            Run on Mar 16, 1998 @ 13:54:43
                3 nodes observed
-----------------------------------------------------------
*   New address observed on network
#   Known address observed on network
~   New address-name mapping; no traffic observed
>   Selected record
       Address          Layer     Type/ID      Comment/Name
  ________________________ __________ _________ _______________________
  220.100.100.8                           Micro/PC
#   WstDigt--7D-73-D7     Ethernet
#   220.100.100.8         IP
*>  FAT BOY             NetBIOS               FAT BOY
    BRAINS
#   WstDigt--3A-B9-EC     Ethernet
#   220.100.100.12        IP
#>  BRAINS              NetBIOS               BRAINS
    Broadcast                               Taylor's Analyzer
    FF-FF-FF-FF-FF-FF     Ethernet
  CHEROKEE
    Ibm------A5-CC-EE     Ethernet
    220.100.100.53        IP
    CHEROKEE             NetBIOS               CHEROKEE
  MUSCLE
#   WstDigt--4E-B5-EC     Ethernet
#>  220.100.100.10        IP                 INFO
*>  00000000-0000C04EB5EC IPX      0001         MUSCLE
#>  MUSCLE              NetBIOS               MUSCLE
  SILO
    Axis-----25-67-D5     Ethernet
    220.100.100.150       IP
    SILO              NetBIOS               SILO
  THE-HOSTAGE
    00-60-08-98-1D-33     Ethernet
    220.100.100.81        IP
  THE-HOSTAGE          NetBIOS               THE-HOSTAGE
  WstDigtE99CB1                             Micro/PC
    WstDigt--E9-9C-B1     Ethernet
```

Figure 18-11
Network information take 2.

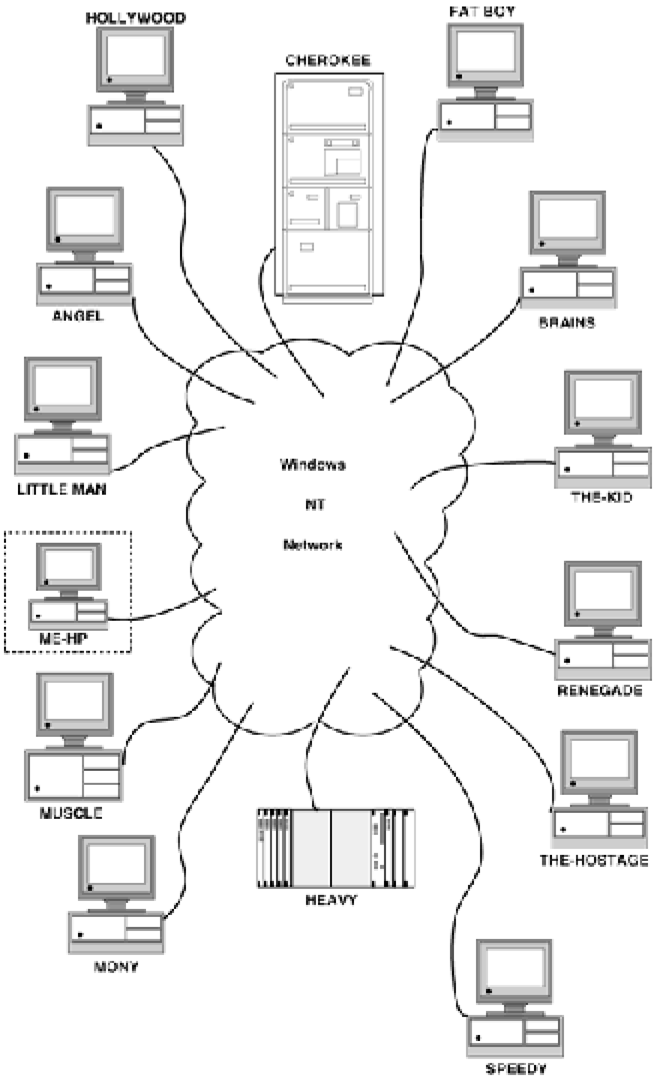


Figure 18-12
Overview of network nodes.

Figure 18-11 shows the addition of FAT-BOY on the network. The previous hosts are also participating on the network as well. The protocols used on the network—NetBIOS, Ethernet, and IP—are also shown in Fig. 18-11. To put this information into perspective, consider Fig. 18-12.

Figure 18-12 shows an overview of the network nodes participating in the network I designed. Note that the ME-HP node is surrounded by a broken-line box. The Internet Advisor is capable of listening to the communication of all nodes on this network. Now consider the information presented below.

Network Commentator for ETHERNET

TCP: Close Connection	[Normal] Mar 13@19:45:05.8432606
220.100.100.53	<—> 220.100.100.10
CHEROKEE	**MUSCLE**
Port: 1144	LOC-SRV 135
Tx Packets: 6	5

```
  Low Window: 0              0
  Retrans: 0                 0
  Connection Duration: 0:00:00.0081942
  Frame Number(s): 220
TCP: Close Connection        [Normal] Mar 13@19:45:05.8527809
  220.100.100.53             <—> 220.100.100.10
CHEROKEE                     MUSCLE
  Port: 1145                 1034
  Tx Packets: 6              5
  Low Window: 0              0
  Retrans: 0                 0
  Connection Duration: 0:00:00.0068831
  Frame Number(s): 231
NOV: Routing Information     [Normal] Mar 13@19:45:10.5468778
Reply
  To Node: FF-FF-FF-FF-FF-
  FF, Broadcast
  00000000-006094A5CCEE      —> 00000000-FFFFFFFFFFFF
  Network number: 00000002
  Number of hops: 1          Number of ticks: 2
  Frame Number: 235
NOV: General Service         [Normal] Mar 13@19:45:40.0371709
Query
  From Node: 00-00-C0-E9-
  9C-B1, WstDigE99CB1
  00000000-0000C0E99CB1      —> 00000000-FFFFFFFFFFFF
  Server Type: File Server
  Frame Number: 272
NOV: Nearest Service         [Normal] Mar 13@19:46:05.1400663
Query
  From Node: 00-00-C0-4E-
  B5-EC, MUSCLE
  00000000-0000C04EB5EC      —> 00000000-FFFFFFFFFFFF
  Server Type: File Server
  Frame Number: 314
```

NOV: Nearest Service Query [Normal] Mar 13@19:46:05.852107

From Node: 00-00-C0-4E-B5-EC, MUSCLE

00000000-0000C04EB5EC —> 00000000-FFFFFFFFFFFF

Server Type: File Server

Frame Number: 319

NOV: Nearest Service Query [Normal] Mar 13@19:46:06.57323

From Node: 00-00-C0-4E-B5-EC, MUSCLE

00000000-0000C04EB5EC —> 00000000-FFFFFFFFFFFF

Server Type: File Server

Frame Number: 321

NOV: Nearest Service Query [Normal] Mar 13@19:46:07.2943876

From Node: 00-00-C0-4E-B5-EC, MUSCLE

00000000-0000C04EB5EC —> 00000000-FFFFFFFFFFFF

Server Type: File Server

Frame Number: 322

NOV: Nearest Service Query [Normal] Mar 13@19:46:08.0155127

From Node: 00-00-C0-4E-B5-EC, MUSCLE

00000000-0000C04EB5EC —> 00000000-FFFFFFFFFFFF

Server Type: File Server

Frame Number: 325

NOV: Nearest Service Query [Normal] Mar 13@19:46:08.7365608

From Node: 00-00-C0-4E-B5-EC, MUSCLE

00000000-0000C04EB5EC —> 00000000-FFFFFFFFFFFF

Server Type: File Server

Frame Number: 326

NOV: Nearest Service Query [Normal] Mar 13@19:46:09.4577

From Node: 00-00-C0-4E-B5-EC, MUSCLE

00000000-0000C04EB5EC —> 00000000-FFFFFFFFFFFF

```
   Server Type: File Server
   Frame Number: 327
NOV: Nearest Service      [Normal] Mar 13@19:46:10.1788107
Query
   From Node: 00-00-C0-4E-B5-EC, MUSCLE
   00000000-0000C04EB5EC  —> 00000000-FFFFFFFFFFFF
   Server Type: File Server
   Frame Number: 329
NOV: Routing Information  [Normal] Mar 13@19:46:10.5455269
Reply
   To Node: FF-FF-FF-FF-FF-FF, Broadcast
   00000000-006094A5CCEE  —> 00000000-FFFFFFFFFFFF
   Network number: 00000002
   Number of hops: 1         Number of ticks: 2
   Frame Number: 330
NOV: Nearest Service      [Normal] Mar 13@19:46:10.8999035
Query
   From Node: 00-00-C0-4E-B5-EC, MUSCLE
   00000000-0000C04EB5EC  —> 00000000-FFFFFFFFFFFF
   Server Type: File Server
   Frame Number: 331

NOV: Nearest Service Query  [Normal] Mar 13@19:46:11.6210437
   From Node: 00-00-C0-4E-B5-EC, MUSCLE
   00000000-0000C04EB5EC   —> 00000000-FFFFFFFFFFFF
   Server Type: File Server
   Frame Number: 332
NOV: General Service Query  [Normal] Mar 13@19:46:12.3421574
   From Node: 00-00-C0-4E-B5-EC, MUSCLE
   00000000-0000C04EB5EC   —> 00000000-FFFFFFFFFFFF
   Server Type: File Server
   Frame Number: 333
NOV: General Service Query  [Normal] Mar 13@19:46:13.0632036
   From Node: 00-00-C0-4E-B5-EC, MUSCLE
   00000000-0000C04EB5EC   —> 00000000-FFFFFFFFFFFF
   Server Type: File Server
```

```
  Frame Number: 334
NOV: General Service Query  [Normal] Mar 13@19:46:13.7843639
  From Node: 00-00-C0-4E-B5-EC, MUSCLE
  00000000-0000C04EB5EC     —> 00000000-FFFFFFFFFFFF
  Server Type: File Server
  Frame Number: 335
NOV: General Service Query  [Normal] Mar 13@19:46:14.5054544
  From Node: 00-00-C0-4E-
  B5-EC, MUSCLE
  00000000-0000C04EB5EC     —> 00000000-FFFFFFFFFFFF
  Server Type: File Server
  Frame Number: 344
TCP: Close Connection       [Normal] Mar 13@19:46:27.4038468
  220.100.100.8             <—> 220.100.100.150
  220.100.100.8             SILO
  Port: 1043                NETBIOS 139
  Tx Packets: 10            8
  Low Window: 0             0
  Retrans: 0                0
  Connection Duration: 0:00:02.0613844
  Frame Number(s): 407
NOV: Nearest Service Query  [Normal] Mar 13@19:46:39.9998894
  From Node: 00-00-C0-E9-
  9C-B1, WstDigE99CB1
  00000000-0000C0E99CB1     —> 00000000-FFFFFFFFFFFF
  Server Type: File Server
  Frame Number: 472
NOV: Routing Information    [Normal] Mar 13@19:47:10.544317
Reply
  To Node: FF-FF-FF-FF-FF-
  FF, Broadcast
  00000000-006094A5CCEE     —> 00000000-FFFFFFFFFFFF
  Network number: 00000002
  Number of hops: 1         Number of ticks: 2
  Frame Number: 488
TCP: Reset Connection       [Warning] Mar 13@19:47:19.720187
220.100.100.12              —> 220.100.100.53
  BRAINS                    CHEROKEE
  Port: NETBIOS 139         1111
```

```
   Tx Packets: 1                0
   Low Window: 0                0
   Retrans: 0                   0
   Connection Duration: 0:00:00.0
   Frame Number(s): 489
NOV: General Service Query     [Normal] Mar 13@19:47:39.960695
   From Node: 00-00-C0-E9-
   9C-B1, WstDigE99CB1
   00000000-0000C0E99CB1        —> 00000000-FFFFFFFFFFFF
   Server Type: File Server
   Frame Number: 562
TCP: Reset Connection          [Warning] Mar 13@19:47:47.6089335
   220.100.100.41               —> 220.100.100.53
                                   CHEROKEE
   Port: NETBIOS 139            1136
   Tx Packets: 1                0
   Low Window: 0                0
   Retrans: 0                   0
   Connection Duration: 0:00:00.0
   Frame Number(s): 566
TCP: Close Connection          [Normal] Mar 13@19:48:07.5697698
   220.100.100.19               <—> 220.100.100.150
                                   SILO
   Port: 1029                   NETBIOS 139
   Tx Packets: 10               8
   Low Window: 0                0
   Retrans: 0                   0
   Connection Duration: 0:00:02.0748997
   Frame Number(s): 658
NOV: Routing Information       [Normal] Mar 13@19:48:10.5430294
Reply
   To Node: FF-FF-FF-FF-FF-
   FF, Broadcast
   00000000-006094A5CCEE        —> 00000000-FFFFFFFFFFFF
   Network number: 00000002
   Number of hops: 1            Number of ticks: 2
   Frame Number: 697
TCP: Reset Connection          [Warning] Mar 13@19:48:31.2433202
```

```
   220.100.100.77              —> 220.100.100.53
                                 CHEROKEE
   Port: NETBIOS 139           1143
   Tx Packets: 1               0
   Low Window: 0               0
   Retrans: 0                  0
   Connection Duration: 0:00:00.0
   Frame Number(s): 1545
NOV: Nearest Service Query     [Normal] Mar 13@19:48:39.9249748
   From Node: 00-00-C0-E9-9C-B1, WstDigE99CB1
00000000-0000C0E99CB1          —> 00000000-FFFFFFFFFFFF
   Server Type: File Server
   Frame Number: 1858
TCP: Reset Connection          [Warning] Mar 13@19:48:49.3872264
   220.100.100.61              —> 220.100.100.53
THE-HOSTAGE                    CHEROKEE
   Port: NETBIOS 139           1139
   Tx Packets: 1               0
   Low Window: 0               0
   Retrans: 0                  0
   Connection Duration: 0:00:00.0
   Frame Number(s): 1899
NOV: Nearest Service Query     [Normal] Mar 13@19:49:01.7757016
   From Node: 00-00-C0-4E-B5-EC, MUSCLE
   00000000-0000C04EB5EC       —> 00000000-FFFFFFFFFFFF
   Server Type: File Server
   Frame Number: 1920
NOV: Nearest Service Query     [Normal] Mar 13@19:49:02.4938427
   From Node: 00-00-C0-4E-B5-EC, MUSCLE
   00000000-0000C04EB5EC       —> 00000000-FFFFFFFFFFFF
   Server Type: File Server
   Frame Number: 1921
NOV: Nearest Service Query     [Normal] Mar 13@19:49:03.2149768
   From Node: 00-00-C0-4E-B5-EC, MUSCLE
   00000000-0000C04EB5EC       —> 00000000-FFFFFFFFFFFF
   Server Type: File Server
   Frame Number: 1922
NOV: Nearest Service Query     [Normal] Mar 13@19:49:03.936104
```

```
   From Node: 00-00-C0-4E-B5-EC, MUSCLE
   00000000-0000C04EB5EC    —> 00000000-FFFFFFFFFFFF
   Server Type: File Server
   Frame Number: 1923
NOV: Nearest Service Query   [Normal] Mar 13@19:49:04.6572385
   From Node: 00-00-C0-4E-B5-EC, MUSCLE
   00000000-0000C04EB5EC    —> 00000000-FFFFFFFFFFFF
   Server Type: File Server
   Frame Number: 1924
NOV: Nearest Service Query   [Normal] Mar 13@19:49:05.3783097
   From Node: 00-00-C0-4E-B5-EC, MUSCLE
   00000000-0000C04EB5EC    —> 00000000-FFFFFFFFFFFF
   Server Type: File Server
   Frame Number: 1925
NOV: Nearest Service Query   [Normal] Mar 13@19:49:06.0994503
   From Node: 00-00-C0-4E-B5-EC, MUSCLE
   00000000-0000C04EB5EC    —> 00000000-FFFFFFFFFFFF
   Server Type: File Server
   Frame Number: 1926
NOV: Nearest Service Query   [Normal] Mar 13@19:49:06.8205179
   From Node: 00-00-C0-4E-B5-EC, MUSCLE
   00000000-0000C04EB5EC    —> 00000000-FFFFFFFFFFFF
   Server Type: File Server
   Frame Number: 1927
NOV: Nearest Service Query   [Normal] Mar 13@19:49:07.5416253
   From Node: 00-00-C0-4E-B5-EC, MUSCLE
   00000000-0000C04EB5EC    —> 00000000-FFFFFFFFFFFF
   Server Type: File Server
   Frame Number: 1932
NOV: Nearest Service Query   [Normal] Mar 13@19:49:08.2627509
   From Node: 00-00-C0-4E-B5-EC, MUSCLE
   00000000-0000C04EB5EC    —> 00000000-FFFFFFFFFFFF
   Server Type: File Server
   Frame Number: 1933
NOV: General Service Query   [Normal] Mar 13@19:49:08.9838831
   From Node: 00-00-C0-4E-B5-EC, MUSCLE
   00000000-0000C04EB5EC    —> 00000000-FFFFFFFFFFFF
   Server Type: File Server
   Frame Number: 1934
NOV: General Service Query  [Normal] Mar 13@19:49:09.7049542
   From Node: 00-00-C0-4E-B5-EC, MUSCLE
   00000000-0000C04EB5EC     —> 00000000-FFFFFFFFFFFF
   Server Type: File Server
   Frame Number: 1935
NOV: General Service Query  [Normal] Mar 13@19:49:10.426067
   From Node: 00-00-C0-4E-B5-EC, MUSCLE
   00000000-0000C04EB5EC     —> 00000000-FFFFFFFFFFFF
   Server Type: File Server
   Frame Number: 1937
NOV: Routing Information    [Normal] Mar 13@19:49:10.5417939
Reply
   To Node: FF-FF-FF-FF-FF-FF, Broadcast
   00000000-006094A5CCEE     —> 00000000-FFFFFFFFFFFF
   Network number: 00000002
   Number of hops: 1         Number of ticks: 2
   Frame Number: 1938
NOV: General Service Query  [Normal] Mar 13@19:49:11.1471537
   From Node: 00-00-C0-4E-B5-EC, MUSCLE
   00000000-0000C04EB5EC     —> 00000000-FFFFFFFFFFFF
   Server Type: File Server
   Frame Number: 1939
NOV: General Service Query  [Normal] Mar 13@19:49:39.8876314
   From Node: 00-00-C0-E9-9C-B1, WstDigE99CB1
   00000000-0000C0E99CB1     —> 00000000-FFFFFFFFFFFF
   Server Type: File Server
   Frame Number: 2002
```

Note the wide variety of information obtained. First, the protocol used is identified; that is, `NOV` for Novell and `TCP` for TCP/IP. NetBIOS is also indicated as being in use. IP addresses with their functions, such as the *general service query* and *close connection,* are also shown. A timestamp that yields the hour is also indicated. This and other information provided shows a wealth of insights into different aspects of the network operation. This level of information is critical to have after the network is designed and installed. What is helpful is to see the information that can be obtained prior to your network design. In the case of my network, the HP Internet Advisor is indispensable.

Another type of information that the Internet Advisor can obtain is the 802.3/Ethernet Decode. Figure 18-13 has detailed information on all frames; consider the synthesis of this level of information.

```
**************** *****  HEWLETT PACKARD NETWORK ADVISOR    *****
*****           *****
*****           Measurement:    802.3/Ethernet Decode        *****
*****           Print Type:    All Frames                    *****
*****           Open Views:    Detailed                      *****
*****           Display Mode:    Viewing All Frames          *****
*****           Print Date:    03/13/98                      *****
*****           Print Time:    20:4:31                       *****
*****           *****
******************************************************************
___________________________________
   Frame: 1                      Time: Mar 13@20:02:56.9192136 Length: 64
Destination address             Broadcast                    Broadcast
Source address                220.100.100.8          Individual, global
Type          08-06            ARP
>Data size          46
Frame check sequence          8E-69-77-43
___________________________________
   Frame: 2                      Time: Mar 13@20:02:58.4297056 Length: 64
Destination address              Broadcast                    Broadcast
Source address              220.100.100.8          Individual, global
Type          08-06           ARP
>Data size          46
Frame check sequence          1E-DE-8C-55
___________________________________
   Frame: 3                      Time: Mar 13@20:02:59.9403448 Length: 64
Destination address              Broadcast                    Broadcast
Source address              220.100.100.8          Individual, global
Type          08-06           ARP
>Data size        46
Frame check sequence          8E-69-77-43
___________________________________
```

Figure 18-13
HP Internet Advisor measured by 802.3/Ethernet Decode.

```
  Frame: 4           Time: Mar 13@20:03:01.6672512 Length: 65
Destination address          03-00-00-00-00-01          Group, local
Source address  220.100.100.8             Individual, global
Length  47
>Data size         47
Frame check sequence  F4-31-89-29
___________________________________
  Frame: 5           Time: Mar 13@20:03:01.6676236 Length: 65
Destination address  220.100.100.8             Individual, global
Source address          CHEROKEE             Individual, global
Length  47
>Data size          47
Frame check sequence  3A-9A-CD-F5
___________________________________
  Frame: 6           Time: Mar 13@20:03:01.6679466 Length: 64
Destination address          CHEROKEE             Individual, global
Source address 220.100.100.8          Individual, global
Length  3
>Data size          3
Padding:
  00-06-04-00-01-00-00-C0 7D-73-D7-DC-64-64-08-00
  00-00-00-00-00-DC-64-64 4D-02-53-BF-8B-50-18-15
  38-81-A1-00-00-00-00-00 23-FF-53
Frame check sequence  20-62-02-76
___________________________________
  Frame: 7           Time: Mar 13@20:03:01.6681270 Length: 64
Destination address  220.100.100.8             Individual, global
Source address          CHEROKEE             Individual, global
Length  3
>Data size          3
Padding:
  00-00-00-00-00-00-00-00 00-00-00-00-00-00-00-00
  00-00-00-00-00-00-00-00 00-00-00-00-00-00-00-00
  00-00-00-00-00-00-00-00 00-00-00
Frame check sequence  66-1B-6E-95
___________________________________
  Frame: 8           Time: Mar 13@20:03:01.6683975 Length: 64
Destination address          CHEROKEE             Individual, global
Source address          220.100.100.8          Individual, global
Length  4
>Data size          4
Padding:
  00-FF-EF-0A-17-30-00-00 00-30-00-43-48-45-52-4F
  4B-45-45-20-20-20-20-20 20-20-20-46-41-54-20-42
  4F-59-20-20-20-20-20-20 20-20
Frame check sequence  47-41-9D-7D
---------------------------------------
```

```
Frame: 9           Time: Mar 13@20:03:01.6685602 Length: 64
Destination address   220.100.100.8          Individual, global
Source address          CHEROKEE          Individual, global
Length  4
>Data size          4
Padding:
  00-00-00-00-00-00-00-00 00-00-00-00-00-00-00-00
  00-00-00-00-00-00-00-00 00-00-00-00-00-00-00-00
  00-00-00-00-00-00-00-00 00-00
Frame check sequence  49-07-D3-43

___________________________________
  Frame: 10             Time: Mar 13@20:03:01.6688476 Length: 64

Destination address     CHEROKEE          Individual, global
Source address     220.100.100.8          Individual, global
Length  18
>Data size          18
Padding:
  00-00-00-00-00-00-DC-64 64-4D-02-53-BF-8B-50-18
  15-38-81-A1-00-00-00-00 00-23-FF-53
Frame check sequence  93-AD-15-BE

___________________________________
  Frame: 11             Time: Mar 13@20:03:01.6690630 Length: 64

Destination address    220.100.100.8          Individual, global
Source address    CHEROKEE          Individual, global
Length  4
>Data size          4
```

```
Padding:
  00-00-00-00-00-00-00-00 00-00-00-00-00-00-00-00
  00-00-00-00-00-00-00-00 00-00-00-00-00-00-00-00
  00-00-00-00-00-00-00-00 00-00
Frame check sequence  BF-EA-F4-CD

--------------------------------------
  Frame: 12             Time: Mar 13@20:03:01.6691318 Length: 64

Destination address          220.100.100.8          Individual, global
Source address          CHEROKEE          Individual, global
Length  18
>Data size          18
Padding:
  00-00-00-00-00-00-00-00 00-00-00-00-00-00-00-00
  00-00-00-00-00-00-00-00 00-00-00-00
Frame check sequence  A4-37-97-40
___________________________________
  Frame: 13             Time: Mar 13@20:03:01.6694761 Length: 64

Destination address          CHEROKEE          Individual, global
Source address          220.100.100.8          Individual, global
Length  4
>Data size          4
```

```
Padding:
  00-FF-EF-0A-17-30-00-00 00-30-00-43-48-45-52-4F
  4B-45-45-20-20-20-20-20 20-20-20-46-41-54-20-42
  4F-59-20-20-20-20-20-20 20-20
Frame check sequence  FD-84-2C-50

___________________________________
  Frame: 14             Time: Mar 13@20:03:01.6701895 Length: 190

Destination address          CHEROKEE             Individual, global
Source address          220.100.100.8  Individual, global
Length  172          Partial packet store
>Data size          86
>Sliced data

___________________________________
  Frame: 15             Time: Mar 13@20:03:01.6706443 Length: 123

Destination address          220.100.100.8  Individual, global
Source address          CHEROKEE             Individual, global
Length  105          Partial packet store
>Data size          86
>Sliced data

___________________________________
  Frame: 16             Time: Mar 13@20:03:01.6710735 Length: 64

Destination address          CHEROKEE             Individual, global
Source address          220.100.100.8          Individual, global
Length  4
>Data size          4
Padding:
  00-FF-EF-0A-17-30-00-00 00-30-00-43-48-45-52-4F
  4B-45-45-20-20-20-20-20 20-20-20-46-41-54-20-42
  4F-59-20-20-20-20-20-20 20-20
Frame check sequence  5D-42-26-5E
___________________________________
  Frame: 17             Time: Mar 13@20:03:01.6725487 Length: 189

Destination address          CHEROKEE             Individual, global
Source address          220.100.100.8          Individual, global
Length  171          Partial packet store
>Data size          86
>Sliced data

___________________________________
  Frame: 18             Time: Mar 13@20:03:01.6833829 Length: 130

Destination address          220.100.100.8          Individual, global
Source address          CHEROKEE             Individual, global
Length  112          Partial packet store
>Data size          86
>Sliced data

___________________________________
  Frame: 19             Time: Mar 13@20:03:01.6838061 Length: 64

Destination address          CHEROKEE             Individual, global
```

```
Source address          220.100.100.8          Individual, global
Length  4
>Data size          4
Padding:
  00-FF-EF-0A-17-30-00-00 00-30-00-43-48-45-52-4F
  4B-45-45-20-20-20-20-20 20-20-20-46-41-54-20-42
  4F-59-20-20-20-20-20-20 20-20
Frame check sequence  AB-AF-01-D0

___________________________________
  Frame: 20             Time: Mar 13@20:03:01.6843194 Length: 145

Destination address          CHEROKEE          Individual, global
Source address          220.100.100.8  Individual, global
Length  127          Partial packet store
>Data size          86
>Sliced data

___________________________________
  Frame: 21             Time: Mar 13@20:03:01.6953938 Length: 568

Destination address          220.100.100.8          Individual, global
Source address          CHEROKEE          Individual, global
Length  550          Partial packet store
>Data size          86
>Sliced data

___________________________________
  Frame: 22             Time: Mar 13@20:03:01.6967575 Length: 64

Destination address          CHEROKEE          Individual, global
Source address          220.100.100.8  Individual, global
Length  4
>Data size          4
```

```
Padding:
  00-FF-EF-0A-17-30-00-00 00-30-00-43-48-45-52-4F
  4B-45-45-20-20-20-20-20 20-20-20-46-41-54-20-42
  4F-59-20-20-20-20-20-20 20-20
Frame check sequence  EB-22-14-CC

___________________________________

  Frame: 23             Time: Mar 13@20:03:01.7274710 Length: 64

Destination address          CHEROKEE          Individual, global
Source address          220.100.100.8  Individual, global
Length  18
>Data size          18
Padding:
  FF-53-4D-42-25-00-00-00 00-00-00-80-00-00-00-00
  00-00-00-00-00-00-00-00 01-08-BB-14
Frame check sequence  F9-E9-49-5B

___________________________________

  Frame: 24             Time: Mar 13@20:03:01.7276453 Length: 64

Destination address          220.100.100.8          Individual, global
Source address          CHEROKEE          Individual, global
```

```
Length  4
>Data size       4
Padding:
  00-00-00-00-00-00-00-00 00-00-00-00-00-00-00-00
  00-00-00-00-00-00-00-00 00-00-00-00-00-00-00-00
  00-00-00-00-00-00-00-00 00-00
Frame check sequence  A4-57-F8-98

___________________________________

  Frame: 25             Time: Mar 13@20:03:03.3579381 Length: 64

Destination address            Broadcast             Broadcast
Source address          220.100.100.8          Individual, global
Type          08-06          ARP
>Data size          46
Frame check sequence  20-1A-22-FA

___________________________________

  Frame: 26             Time: Mar 13@20:03:03.5655960 Length: 65

Destination address          03-00-00-00-00-01                    Group,
local
Source address          220.100.100.8          Individual, global
Length  47
>Data size          47
Frame check sequence  E5-04-DF-65

___________________________________

  Frame: 27             Time: Mar 13@20:03:03.5668637 Length: 65

Destination address          220.100.100.8          Individual, global
Source address          SILO          Individual, global
Length  47
>Data size          47
Frame check sequence  99-9F-D2-6F
```

```
____________________________________
  Frame: 28             Time: Mar 13@20:03:03.5672027 Length: 64

Destination address                    SILO          Individual, global
Source address          220.100.100.8          Individual, global
Length  3
>Data size          3
Padding:
  00-06-04-00-01-00-00-C0 7D-73-D7-DC-64-64-08-00
  00-00-00-00-00-DC-64-64 4D-20-20-20-46-41-54-20
  42-4F-59-20-20-20-20-20 20-20-20
Frame check sequence  AD-D6-F0-E6

____________________________________
  Frame: 29             Time: Mar 13@20:03:03.5678517 Length: 64

Destination address          220.100.100.8          Individual, global
Source address          SILO          Individual, global
Length  3
>Data size          3
Padding:
  FF-FF-FF-FF-FF-FF-FF-FF FF-FF-FF-FF-FF-FF-FF-FF
```

```
FF-FF-FF-FF-FF-FF-FF-FF FF-FF-FF-FF-FF-FF-FF-FF
  FF-FF-FF-FF-FF-FF-FF-FF FF-FF-FF
Frame check sequence  56-42-97-30

____________________________________
  Frame: 30             Time: Mar 13@20:03:03.5681241 Length: 64

Destination address          SILO                   Individual, global
Source address          220.100.100.8          Individual, global
Length  4
>Data size          4
Padding:
  00-FF-EF-0A-17-31-00-00 00-31-00-53-49-4C-4F-20
  20-20-20-20-20-20-20-20 20-20-20-46-41-54-20-42
  4F-59-20-20-20-20-20-20 20-20
Frame check sequence  45-C9-1F-6B

____________________________________
  Frame: 31             Time: Mar 13@20:03:03.5762576 Length: 64

Destination address          220.100.100.8          Individual, global
Source address          SILO          Individual, global
Length  4
>Data size          4
Padding:
  FF-FF-FF-FF-FF-FF-FF-FF FF-FF-FF-FF-FF-FF-FF-FF
  FF-FF-FF-FF-FF-FF-FF-FF FF-FF-FF-FF-FF-FF-FF-FF
  FF-FF-FF-FF-FF-FF-FF-FF FF-FF
Frame check sequence  B5-8D-1C-0D

____________________________________
  Frame: 32             Time: Mar 13@20:03:03.5765530 Length: 64

Destination address          SILO                   Individual, global
Source address          220.100.100.8          Individual, global
```

```
Length  18
>Data size          18
Padding:
  00-00-00-00-00-00-DC-64 64-4D-20-20-20-46-41-54
  20-42-4F-59-20-20-20-20 20-20-20-20
Frame check sequence  34-5F-09-87

___________________________________

  Frame: 33             Time: Mar 13@20:03:03.5768360 Length: 64

Destination address           220.100.100.8          Individual, global
Source address          SILO          Individual, global
Length  4
>Data size          4
Padding:
  10-65-00-00-40-06-E3-53 DC-64-64-96-DC-64-64-0C
  00-8B-04-95-BC-08-EC-B0 02-47-4D-3C-50-10-10-00
  1B-60-00-00-FF-FF-FF-FF FF-FF
Frame check sequence  5E-04-1A-D6

___________________________________

  Frame: 34             Time: Mar 13@20:03:03.5772064 Length: 64
```

```
Destination address           220.100.100.8          Individual, global
Source address SILO          Individual, global
Length  18
>Data size          18
Padding:
  FF-FF-FF-FF-FF-FF-FF-FF FF-FF-FF-FF-FF-FF-FF-FF
  FF-FF-FF-FF-FF-FF-FF-FF FF-FF-FF-FF
Frame check sequence  AD-5B-8C-60

___________________________________

  Frame: 35             Time: Mar 13@20:03:03.5774858 Length: 64

Destination address           SILO                    Individual, global
Source address          220.100.100.8          Individual, global
Length  4
>Data size          4
Padding:
  00-FF-EF-0A-17-31-00-00 00-31-00-53-49-4C-4F-20
  20-20-20-20-20-20-20-20 20-20-20-46-41-54-20-42
  4F-59-20-20-20-20-20-20 20-20
Frame check sequence  04-FA-BD-01

___________________________________

  Frame: 36             Time: Mar 13@20:03:03.8540212 Length: 71

Destination address           CHEROKEE                    Individual, global
Source address          220.100.100.8          Individual, global
Length  53
>Data size          53
Frame check sequence  50-18-B5-38

___________________________________

  Frame: 37             Time: Mar 13@20:03:03.8543697 Length: 71

Destination address           220.100.100.8          Individual, global
Source address          CHEROKEE          Individual, global
```

```
Length  53
>Data size          53
Frame check sequence  14-F9-A7-A2

___________________________________
  Frame: 38             Time: Mar 13@20:03:03.8547697 Length: 64

Destination address          CHEROKEE                    Individual, global
Source address          220.100.100.8          Individual, global
Length  18
>Data size          18
Padding:
  4F-20-20-20-20-20-20-20 20-20-20-20-20-46-41-54
  20-42-4F-59-20-20-20-20 20-20-20-20
Frame check sequence  DE-8D-26-6E

___________________________________
  Frame: 39             Time: Mar 13@20:03:03.8549544 Length: 64

Destination address          220.100.100.8          Individual, global
Source address          CHEROKEE          Individual, global
```

```
Length 4
>Data size                   4
Padding:
  00-00-00-00-00-00-00-00 00-00-00-00-00-00-00-00
  00-00-00-00-00-00-00-00 00-00-00-00-00-00-00-00
  00-00-00-00-00-00-00-00 00-00
Frame check sequence  09-8A-C6-5F

___________________________________
  Frame: 40             Time: Mar 13@20:03:03.8552476 Length: 64

Destination address          CHEROKEE                    Individual, global
Source address          220.100.100.8          Individual, global
Length  18
>Data size          18
Padding:
  FF-53-4D-42-71-00-00-00 00-00-01-80-00-00-00-00
  00-00-00-00-00-00-00-00 01-08-00-00
Frame check sequence  FC-6F-C1-6A

___________________________________
  Frame: 41             Time: Mar 13@20:03:03.8554348 Length: 64

Destination address          220.100.100.8          Individual, global
Source address          CHEROKEE                    Individual, global
Length  4
>Data size          4
Padding:
  00-00-00-00-00-00-00-00 00-00-00-00-00-00-00-00
  00-00-00-00-00-00-00-00 00-00-00-00-00-00-00-00
  00-00-00-00-00-00-00-00 00-00
Frame check sequence  7F-7D-CA-E9

___________________________________
  Frame: 42             Time: Mar 13@20:03:03.8557614 Length: 64

Destination address          CHEROKEE                    Individual, global
```

```
Source address          220.100.100.8          Individual, global
Length  3
>Data size          3
Padding:
  0B-0E-00-FF-EF-14-00-00 00-04-00-00-00-05-30-4F
  20-20-20-20-20-20-20-20 20-20-20-20-46-41-54-20
  42-4F-59-20-20-20-20-20 20-20-20
Frame check sequence  3C-F0-22-3F

___________________________________
  Frame: 43             Time: Mar 13@20:03:03.8559136 Length: 64

Destination address          220.100.100.8          Individual, global
Source address          CHEROKEE                    Individual, global
Length  3
>Data size          3
Padding:
  00-00-00-00-00-00-00-00 00-00-00-00-00-00-00-00
  00-00-00-00-00-00-00-00 00-00-00-00-00-00-00-00
```

```
00-00-00-00-00-00-00-00 00-00-00
Frame check sequence  66-1B-6E-95

___________________________________
  Frame: 44             Time: Mar 13@20:03:04.8565401 Length: 64

Destination address            Broadcast                  Broadcast
Source address          220.100.100.8          Individual, global
Type          08-06          ARP
>Data size          46
Frame check sequence  36-08-6B-61

___________________________________
  Frame: 45             Time: Mar 13@20:03:06.3670765 Length: 64

Destination address            Broadcast                  Broadcast
Source address          220.100.100.8          Individual, global
Type          08-06          ARP
>Data size          46
Frame check sequence  20-1A-22-FA

___________________________________
  Frame: 46             Time: Mar 13@20:03:06.7809019 Length: 64

Destination address            Broadcast                  Broadcast
Source address          BRAINS          Individual, global
Type          08-06          ARP
>Data size          46
Frame check sequence  CE-8F-B9-E2

___________________________________
  Frame: 47             Time: Mar 13@20:03:06.7812445 Length: 64

Destination address          BRAINS          Individual, global
Source address          220.100.100.8          Individual, global
Type          08-06          ARP
>Data size          46
Frame check sequence  8A-03-34-7C
```

```
___________________________________
  Frame: 48             Time: Mar 13@20:03:06.7816335 Length: 126

Destination address          220.100.100.8          Individual, global
Source address          BRAINS          Individual, global
Type          08-00          IP
>Data size          86
>Sliced data

___________________________________
  Frame: 49             Time: Mar 13@0:03:06.7823637 Length: 105

Destination address          BRAINS          Individual, global
Source address          220.100.100.8          Individual, global
Type          08-00          IP
>Data size          86
>Sliced data

___________________________________
  Frame: 50             Time: Mar 13@20:03:06.7834375 Length: 140
```

```
Destination address                    220.100.100.8          Individual,
global
Source address          BRAINS          Individual, global
Type          08-00          IP
>Data size          86
>Sliced data
___________________________________
  Frame: 51             Time: Mar 13@20:03:06.7841119 Length: 97

Destination address          BRAINS          Individual, global
Source address          220.100.100.8          Individual, global
Type          08-00          IP
>Data size          79
Frame check sequence  6B-39-72-1F
___________________________________
  Frame: 52             Time: Mar 13@20:03:06.7974856 Length: 161

Destination address          220.100.100.8          Individual, global
Source address          BRAINS          Individual, global
Type          08-00          IP
>Data size          86
>Sliced data
___________________________________
  Frame: 53             Time: Mar 13@20:03:06.7990992 Length: 687

Destination address          BRAINS          Individual, global
Source address          220.100.100.8          Individual, global
Type          08-00          IP
>Data size          86
>Sliced data
___________________________________
  Frame: 54             Time: Mar 13@20:03:06.8081367 Length: 140

Destination address          220.100.100.8          Individual, global
Source address          BRAINS          Individual, global
Type          08-00          IP
>Data size          86
>Sliced data
```

```
  Frame: 55            Time: Mar 13@20:03:06.8088484 Length: 97

Destination address          BRAINS         Individual, global
Source address        220.100.100.8           Individual, global
Type          08-00          IP
>Data size          79
Frame check sequence  C0-B3-14-66
```

This snapshot (Fig. 18-13) from the network consists of 55 frames. Note that the destination and source addresses identify the directional flow of data. This is important because knowing this can help you understand *how* and *what* two nodes are communicating on a network. The level of information that can be obtained here can also help you understand the flow of data between hosts. Length and type of data frames also provide information about the hosts communicating, and this information can also be used to deduce network traffic loads on the network at given intervals. Having information that can be referenced that includes date and times of the traffic provides a reference point for aspects of network maintenance such as future planning for expanding the network and adding or removing various segments of the network. Later, additional information was provided on the Internet Advisor that I obtained during the operation and maintenance of my network. I recommend that you contact HP for additional information. They can be reached at www.hp.com or

Hewlett-Packard
5070 Centennial Blvd.
Colorado Springs, CO 80919

18.6 Rack Enclosure (Great Lakes Cabinets)

Great Lakes Cabinets were used in this network. In order to accommodate all the network equipment, four cabinets were required. Three 72-in cabinets and one 48-in cabinet were required. Consider Fig. 18-14.

This illustration shows a front view of the rack enclosure. The enclosure has a blower at the bottom and fan tray at the top. The circulation of this cabinet met the needs of the equipment used in this network. Now consider Fig. 18-15.

Figure 18-15 is similar to Fig. 18-14; however, chassis support brackets run the length of the cabinet. As Figs. 18.14 and 18.15 show, there are vertical panel mounting rails in the front and rear of the enclosure. It is possible to mount equipment in the enclosure without rear vertical panel mounting rails and chassis support brackets. This is a judgment call you need to make. The enclosure is sturdy enough to hold considerable weight.

Great Lakes Cabinets provides a variety of options that can be used in the cabinets they offer. My decision to use front and rear vertical mounting rails was due to weight concerns of the equipment mounted in the enclosure. Your situation could differ; however, such vertical rails and their chassis support brackets will make the enclosure even stronger than it already is; it is an added strength. Figure 18-16 illustrates the network with these cabinets installed.

Figure 18-16 shows four cabinets and other equipment. Most of the network equipment was rack-mounted in these cabinets. However, some network equipment is not rack-mountable. Great Lakes Cabinets have a unique benefit. Both sides can be worked with inside with relative ease because of the space between the mounting rails and the side walls. Both sides of the enclosure are vented to assist in air circulation. The front and rear doors are both keyed and lockable. They are also removable! This flexibility makes the cabinet accessible from every side. The top has a circle of machined holes to aid in circulation. The Great Lakes Cabinets arrived at the location where the network was physically assembled in a tractor-trailer. Each cabinet weighed approximately 250 lb. Great Lakes Cabinets shipping included all the required parts to assemble the enclosure with the least amount of tools.

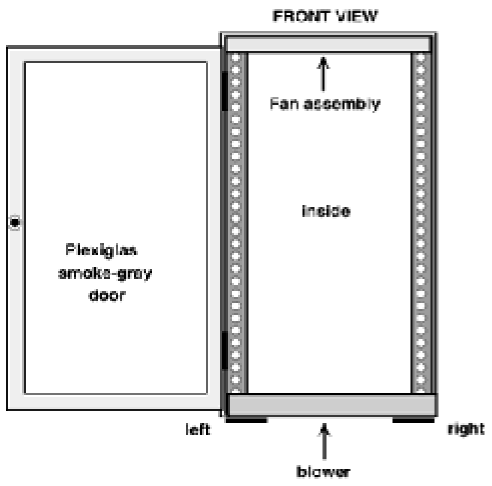


Figure 18-14
Network rack enclosure (front view).

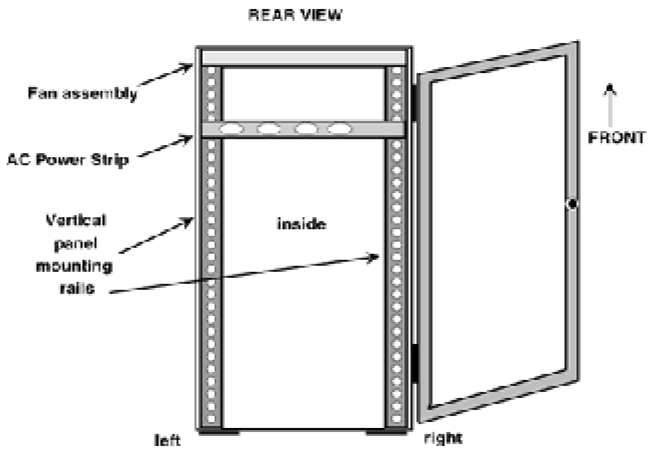


Figure 18-15
Network rack enclosure (rear view).

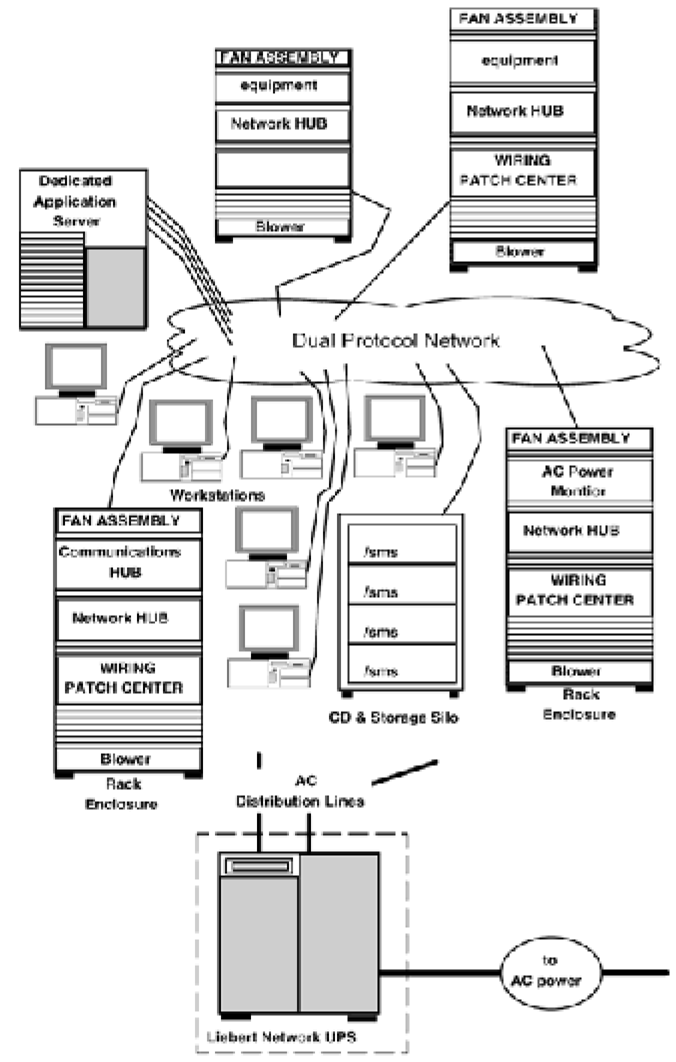


Figure 18-16
Conceptual view of network with cabinets.

If you desire more information about rack enclosures, regardless of your size requirements, contact Great Lakes Cabinets at

Great Lakes Cabinets
P.O. Box 551
Edinboro, PA 16412

This enclosure is a good example of ensuring that your facility can accommodate the weight of the equipment. These cabinets arrived within a day of another pallet of equipment. All together they weighed over 1000 lb and required approximately 10 ft^2 of floor space.

18.7 Commercial Desktop Computers (IBM)

IBM Personal Computers are used in this network. The commercial desktop series used include the PC350 and XL series. A typical example of the general specifications for the Base system units (model 350s) used in this network includes a 200-MHz Pentium MMX processor, a 2.6-GB hard disk (plus an additional 3.0-GB hard drive), 16 MB of nonparity EDO memory (plus an additional 48 MB of RAM), and a 3 1/2-in floppy-disk drive.

Figure 18-17 illustrates these systems and other select network components.

Figure 18-17 shows multiple desktops and a network server. The system named *MUSCLE* in this figure is also a server; it is a dedicated file server. More information on these systems include the following specifications. Units used in this network employ a PCI Busmaster controller and SMART capabilities. These systems include: PCI Enhanced IDE hard drives, Universal Serial Bus ports, infrared and 64-bit PCI graphics, and wake-on LAN capability. The functionality of the Universal Serial Bus (USB) facilitates peripheral connectivity. The hot-connect ability enables peripheral devices to be connected in seconds. Such devices can be added or removed without reconfiguring or rebooting. Each USB port permits up to 127 USB-capable devices.

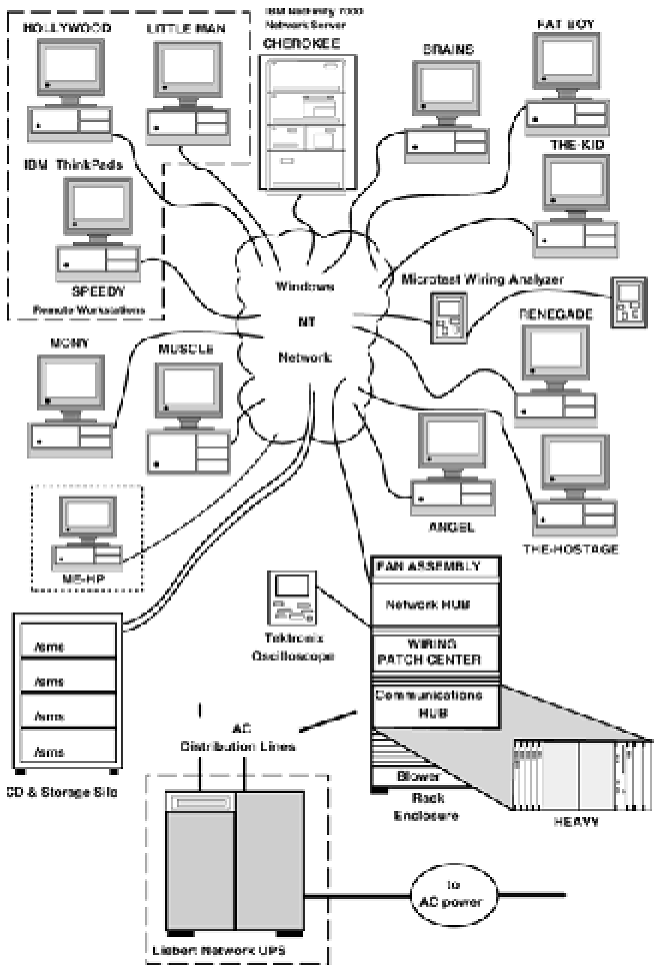


Figure 18-17
Conceptual view of commercial desktops and network.

Some of the PCs used in this network have the capability for symmetrical multiprocessing (SMP) when dual processors are used. An L2 external CPU cache of 256K and Pipeline burst L3 cache are also included. The BIOS type is 256K Flash, SurePath.

The systems can accommodate up to 192 MB RAM at a speed of 60 ns deployed by 72-pin SIMMs. The hard-disk-size average seek time is 12 ms with a latency of approximately 5.8 ms. They support RAID and hot-swappable drive bays.

The graphic capabilities of these systems employs an S3 Trio64 V+Graphics type chipset. The result is SVGA graphics and a data width of 64-bit video RAM. Graphic resolution (with the standard video RAM) is 1280 × 1024 in 16 colors. The maximum resolution (with a maximum video RAM) is 1280 × 1024 with 65536 colors. The graphics bus interface uses PCI architecture.

The systems have a 200-W power-supply type for either 110 or 220 V with a universal manual switch. The heat and sound emissions are 48 dB. The typical weight of each cabinet is 28 lb; height, 6.3 in; width, 16.5 in; and depth, 17.6 in.

Systems used in this network include the following security features:

Boot sequence control
Boot without keyboard or mouse
Cover key lock
Diskette boot inhibit
Diskette write protect (switch)
Diskette I/O control
Hard-disk I/O control
Parallel I/O control
Power-on password
Secure fixed DASD
Secure removable media
Serial I/O control
Setup utility password (administrator password)
U-bolt tie-down support

The system specifications used in this network also include the following product approvals and/or certifications according to IBM: BABT (UK); CE; CISPR-22 Class B; CSA C22.2 No. 950 (Canada); DEMKO (EN 60950); EIF (SETI) (EN 60950); Engery Saving Law (refer to N-B 1-9174-001); FCC Class B (US); IECEE CB Certificate and report to IEC-950, second edition; ISO 9241-3 compliance; JATE; NEMKO (EN 60950); NS/G/1234/J/100003 (Telecommunications Safety only: no approval mark); OVE (EN 60950); Power Line Harmonics (refer to N-B 2-4700-017); SEMKO (EN 60950); TUV-GS (EN 60950); TUV-GS = ZH1/618; UL-1950, first edition; and VCCI Class 2 (Japan).

In addition, IBM’s current warranty is a limited warranty period and type 3: 3-year warranty with repair on-site first year, carry-in second and third years, then parts and labor.

The IBM desktop systems used in this network were shipped with preinstalled software. Some of these systems were reconfigured to meet the needs of the network. However, all respective legal and ethical consideration was given to manufacturers of hardware and software products. Each system used in this network is covered by either a site license or has a dedicated piece of software; and each system has only one user. In the case of servers, workstations, or otherwise, each manufacturer’s legal guidelines were followed. I strongly recommend that these matters be factored into network design of any network. Simply stated: Using an unpaid piece of software, unless it is clear that it is freeware, is stealing. It is no different from someone stealing a tangible item. Consider this when you design your network.

The IBM PC 300XL (model 658842U) is designed with the latest characteristics: 266-MHz Pentium II technology to handle demanding business applications in a networked environment. These are the high-end systems that deliver value and 2.5-GB hard drives to keep you ahead of the curve with the performance disks that power users demand. The IBM PC 300XL series includes open-bay models which can be custom-configured via the IBM Authorized Assembler Program (AAP). Standard features are as follows:

1. Processors: Pentium II processors—233, 266, or 300 MHz with unified 512K L2 cache memory; 32-MB nonparity EDO memory (memory expandable to 384 MB) (3 DIMMs), EDO/60 ns.

2. *Hard drives:* 2.5- or 4.2-GB EIDE with SMART or 4.3-GB Wide Ultra SCSI with SMART or open-bay PCI bus-master EIDE controller on planar; SCSI models include an SCSI-2 Fast and Wide PCI bus-master adapter.

3. *Graphics and video resolution:* S3 Trio64V2; 64-bit; 2MB standard/maximum video DRAM; 256 colors at 1280 × 1024 resolution.

4. *Network features:* LANClient Control Manager supported, Wake-on LAN, Plug-in and Go, Flashover LAN (BIOS/CMOS), Plug-and-Play, CID.

5. *Network interface:* Integrated Intel EtherExpress 10/100-Mbit/s Ethernet with Wake-on LAN.

6. *CD-ROM:* Models available with 16×-8× (variable-speed) CD-ROM (variable read rate; actual playback speed will vary and is often less than the maximum possible).

7. *Audio:* models available with Crystal 4236B audio chip, 16-bit, support Sound Blaster Pro applications.

8. *Diskette drive:* 3.5-in 1.44-MB standard.

9. *Slots:* Three shared PCI/ISA, two ISA.

10. *Bays:* Three 3.5-in, two 5.25 in.

11. *BIOS:* Flash ROM.

12. *Architecture:* PCI local bus, ISA data bus.

13. *Ports:* Serial (16550), enhanced parallel (ECP/EPP), two USB ports, SVGA video, EIDE controller, 10/100 Ethernet RJ-45, IrDA-2-compliant infrared, audio mic-in and line-out minijacks, keyboard, mouse.

14. *Keyboard/mouse:* IBM Cameo 104-key (rubber dome) and Enhanced Mouse.

15. *Power supply:* 200 W.

16. *Security features:* IBM AssetCare—serialization and laser etching of memory and processors, third-party registration available through Retainagoup Limited.

17. *IBM AntiVirus and ConfigSafe:* Vital Product Data (VPD) support, cover key lock, sliding front door lock, U-bolt anchor support, secure access openings, secure removable media, secure fixed DASD, diskette write protect, power-on password, configuration/administrator password, keyboard/mouse password, Wake-on LAN password prompt, boot sequence control, diskette boot inhibit, boot without keyboard/mouse, mouse-disable, I/O controls.

18. *Software and tools:* Windows 95 or Windows NT 4.0 preload available on models with a hard-drive Lotus SmartSuite license, Microsoft NetMeeting (Windows 95 preload models only), LANClient Control Manager (downloadable via Internet), IBM Netfinity Manager software, Intel LANDesk Client Manager, Artisoft CoSession, QAPlus System Support CD (Ready to Configure) with additional software and drivers.

19. *Limited warranty:* 3-year parts and 1-year labor.

Legal notices are as follows:

1. MHz measures only internal clock speed, not application performance. Many factors affect application performance.

2. When referring to hard-drive capacity, MB stands for million bytes and GB stands for billion bytes. Total user-accessible capacity may vary depending on operating environments.

3. For terms and conditions or copies of IBM's limited warranty, call 800-772-2227 in the United States. Limited warranty includes International Warranty Service in those countries where this statement of product is sold by IBM or IBM Business Partners (registration required).

4. Energy Star compliance—the EPA, as a matter of policy, does not endorse any particular company or its products.

5. Battery life (and recharge times) will vary with screen brightness, applications, features, power management, battery conditioning, and other references. CD-ROM or hard-disk drive usage may also have a significant impact on battery life.

6. Actual specifications may vary slightly depending on features and components.

7. Unless otherwise indicated, prices shown are estimated reseller prices to end users. Reseller prices may vary. IBM reserves the right to change prices and product specifications and to discontinue marketing products without notice.

For more information on IBM products, contact www.ibm.com or International Business Machines in Armonk, New York.

18.8 Remote Workstations (IBM ThinkPads)

Multiple remote workstations were required to work with this network. Figure 18-18 illustrates these systems and the network.

Remote workstations were used because mobile use was required in my network. I chose IBM ThinkPads because of their features, reliability, and value for the money. The following are typical specifications for those (765D) ThinkPads used in this network: (1) the latest power, connectivity, and 166-MHz Pentium with configuration flexibility to optimize effectiveness and maximize investment return; (2) high-performance features such as large (13.3- or 12.1-in) high-resolution (1024 × 768 with 65,536 colors) displays with superb graphics, the latest Pentium processors with MMX technology, large hard drives, and integrated infrared and advanced multimedia; and (3) 3-GB hard-disk RAM 2/32 MB with nonparity EDO memory (expandable to 104 MB), 8 × CD-ROM MPEG-1, Microsoft Windows 95. Standard features include the following:

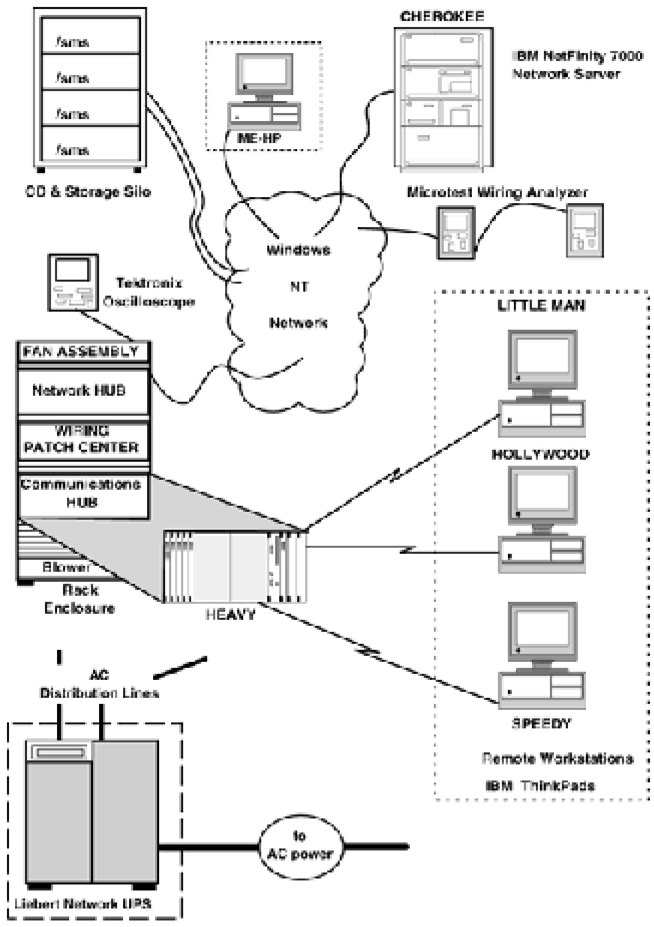


Figure 18-18
Conceptual view of remote workstations.

Pointing device type:	TrackPoint III
Standard diskette size:	3.5-in 1.44-MB
Optional diskette size:	3.5-in 2.88-MB
Diskette drive configuration:	external
Keyboard type standard:	full-size 84 key (tilt/palm rest space)
Keyboard type(s) selectable:	numeric keypad—integrated
Product approvals and certifications:	CISPR-22 Class B; CSA C22.2 No. 950 (Canada); FCC Class B—Part 15; IEC-950; JATE; NOM (Mexico); SASO; UK-PTT; UL-1950; VCCI Class 2 (Japan)
Warranty: limited period, and type 3:	3-year (system battery 1 year) customer carry-in repair or provided by ThinkPad EasyServ (North America only)
Weight:	7.7 lb
Height:	2.2 in

Width:	11.7 in
Depth:	9.3 in

Legal notices are as follows:

1. MHz measures only internal clock speed, not application performance. Many factors affect application performance.

2. When referring to hard-drive capacity, MB stands for million bytes and GB stands for billion bytes. Total user-accessible capacity may vary depending on operating environments.

3. For terms and conditions or copies of IBM's limited warranty, call 800-772-2227 in the United States. Limited warranty includes International Warranty Service in those countries where this statement of product is sold by IBM or IBM Business Partners (registration required).

4. Energy Star compliance—the EPA, as a matter of policy, does not endorse any particular company or its products.

5. Battery Life (and recharge times) will vary with screen brightness, applications, features, power management, battery conditioning, and other references. CD-ROM or hard-disk drive usage may also have a significant impact on battery life.

6. Actual specifications may vary slightly depending on features and components.

7. Unless otherwise indicated, prices shown are estimated reseller prices to end users. Reseller prices may vary. IBM reserves the right to change prices and product specifications and to discontinue marketing products without notice.

18.9 Data-Communications Network Hub (3Com USRobotics)

A USRobotics Enterprise Network hub was selected to use in this network. Data-communication equipment is the single most critical link in any network. This is true because it is the central point of attachment between remote users and a backbone network, regardless of the size of the backbone or location. It is also true if all users are in the same physical location. Data-communication equipment is central to networks; as they fail, so does the network. At one time remote computing meant having a device in one location and a terminal attached to it by a wire. USRobotics revolutionized that definition by designing the Enterprise Network hub. This device, explained in greater detail here, is powerful.

Consider Fig. 18-19, which illustrates the network designed in Dallas and remote users and a remote network, both located in Chicago.

Note that remote users in Fig. 18-19 are connecting directly into the Dallas network via the communications hub. In this case the remote users use their modems to connect directly to the hub.

Where remote users or remote network(s) are concerned, multiple issues must be considered during the design phase. The following list is the minimum number of issues to be reviewed during your plans: security, reliability, maintenance, ease of use, internal protocol compatibility, expandability, internal design architecture, and interface-standard compatibility.

Security has become the single most important topic in networking, regardless of the type network or location. Networks can have a considerable degree of security built into the design if proper components are used to implement security. Where data-communication equipment is concerned, having a device that can provide a security firewall is best. Consider Fig. 18-20.

Figure 18-20 shows a secure firewall implemented in the communications hub. Remote users in this illustration are required to sign onto the hub. It is a point of isolation. Other devices on the network require sign-ons and passwords as well.

The USRobotics communication hub used in this network has three possible configurations regarding its function in the network. USRobotics refers to these as *gateway application cards.* USRobotics uses the following terminology and explanation: X.25, NETServer card, and API card. According to USRobotics, the *X.25 card* provides access capability to packet-switched networks. This card uses an EIA-232/V.35 interface connection point. The *NETServer card* functions as either a router, a terminal server, or both. Ethernet and Token Ring NICs can be used with it. USRobotics refers to this card as the EdgeServe card.

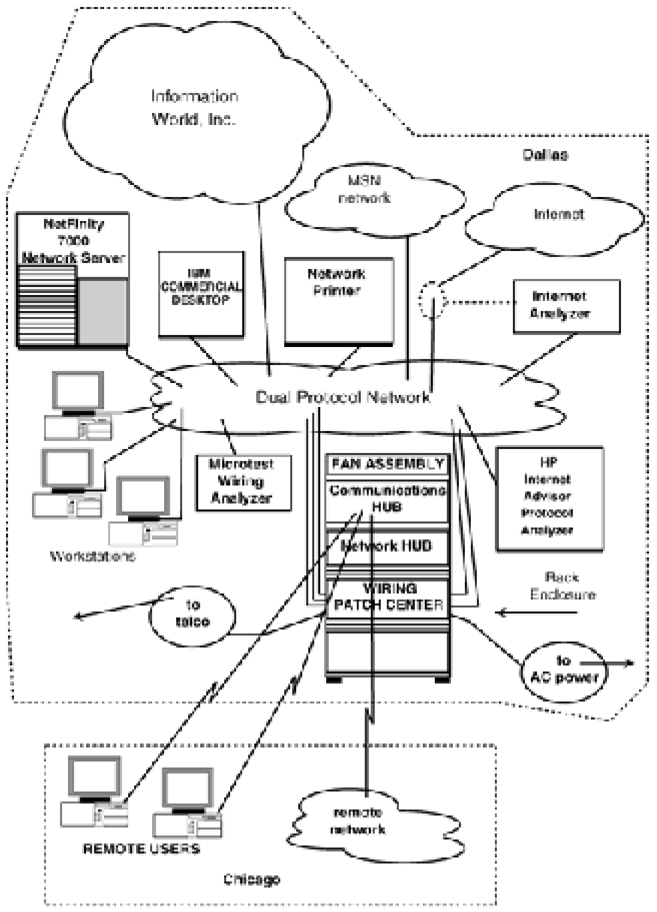


Figure 18-19
Data-communications hub.

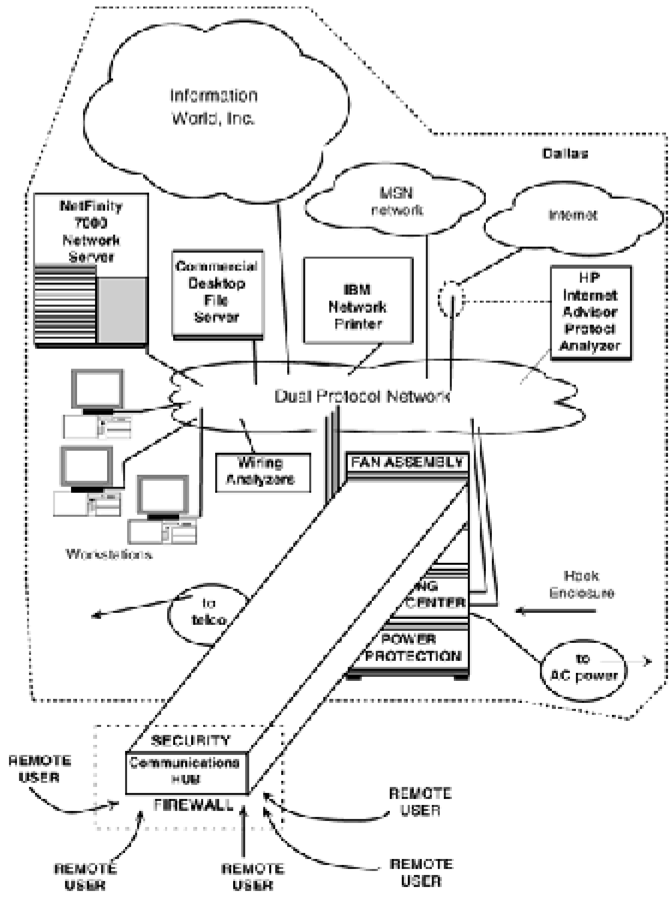


Figure 18-20
Network firewall.

This card has Windows NT loaded onto it, and the functionality this card provides is explained later. The *API card* is designed to let customers design their own applications using USRobotics software development kits. Figure 18-21 illustrates the Enterprise hub.

Note that Fig. 18-21 shows the hub with blank face panels. These panels can be removed and other cards inserted. There are 17 slots. Slot number 1 is the T1 card. Beginning with slot 2, there are analog or digital quad modem cards. These cards have the equivalent of four modems on them. Slots 15 and 16 are the EdgeServer locations. Slot 17 is where the network management card is located. The remaining slots house two power supplies. Although not shown in Fig. 18-21, the undercradle portion of the hub houses approximately 16 fans to cool the components.

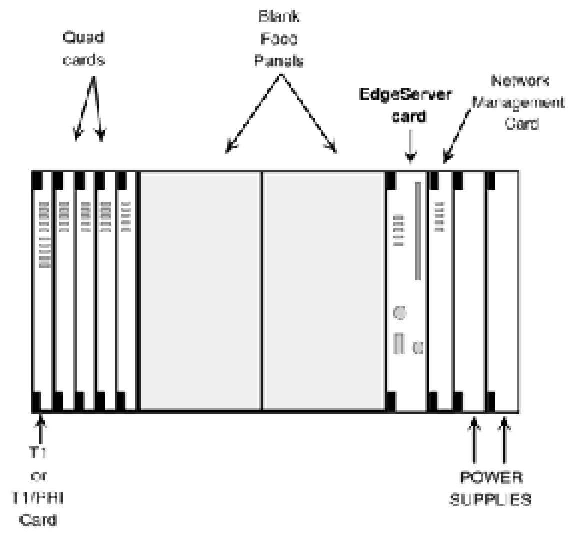


Figure 18-21
Network hub.

Reliability is another important factor for any communication equipment. The design of the USRobotics hub has reliability built into it. One example supports the observation that the hub has two power supplies, but only one is required to operate the unit. They have built-in redundancy even to the level of power-supply unit.

Maintenance is another part of the equation for communication equipment. The hub used in this network has remote management capability, local management capability, and easy access to those components that may need to be removed.

Any communication device requires skill. Most require fairly advanced levels of skill to maximize use. The capability of any commu nication device has little to do with its ease of use, which is a design issue. With the hub used in this network, ease of use is designed into it. Ease of use can be measured in communication equipment by documentation provided to determine how thorough and detailed it is, by accessibility to configure ports, and by the ability to use the equipment in a practically failed state (should that occur). My rule of thumb is: The more complex the functions offered by a device, the simpler the documentation should be. The simple fact is that data-communication equipment is complex enough without humans adding another layer of complexity to it.

Another factor to analyze with data-communication equipment is protocol compatibility. This includes evaluation of upper- and lower-layer protocols. Because this hub has the EdgeServer card in it, NetBEUI, TCP/IP, and IPX upper-layer protocols are supported. Token Ring and Ethernet lower-layer protocols are supported as well. An author's note regarding the USRobotics hub used in this network: Use of Token Ring and Ethernet is more than sufficient because these two protocols are dominant lower-layer protocols used in networks today.

Expandability is very important with data-communication equipment. Design of the USRobotics Enterprise Network hub is such that any size network can be built around this technology. This is true because of how the equipment has been architected. It is possible with the USRobotics equipment to start a network with one or two Enterprise hubs and then continue to add them until racks are filled with them.

Internal design architecture is also very important to data-communication equipment. The internal architecture of data-communication gear is the proverbial pivot on which all communication transactions hinge. The internal communications bus and the incoming port architecture constitute the foundation of the device. These should be capable of handling a complete load on the device without causing hangs or system slowdown.

Interface-standard compatibility is another matter to examine when you are evaluating data-communication equipment. In this network, the hub has flexibility regarding the manner in which certain connections are made. In some instances options exist to make a connection. This alone makes for ease of use, installation, and maintenance. It also means that some existing equipment at your site may be usable. That can save money.

The 3Com USRobotics Enterprise Network hub has the EdgeServer card installed in it. This card has the following: 1.44 floppy drive, mouse port, keyboard port, display port, SCSI port, minimum of 800-MB hard drive, minimum 100-MHz processor, and 10BaseT capability.

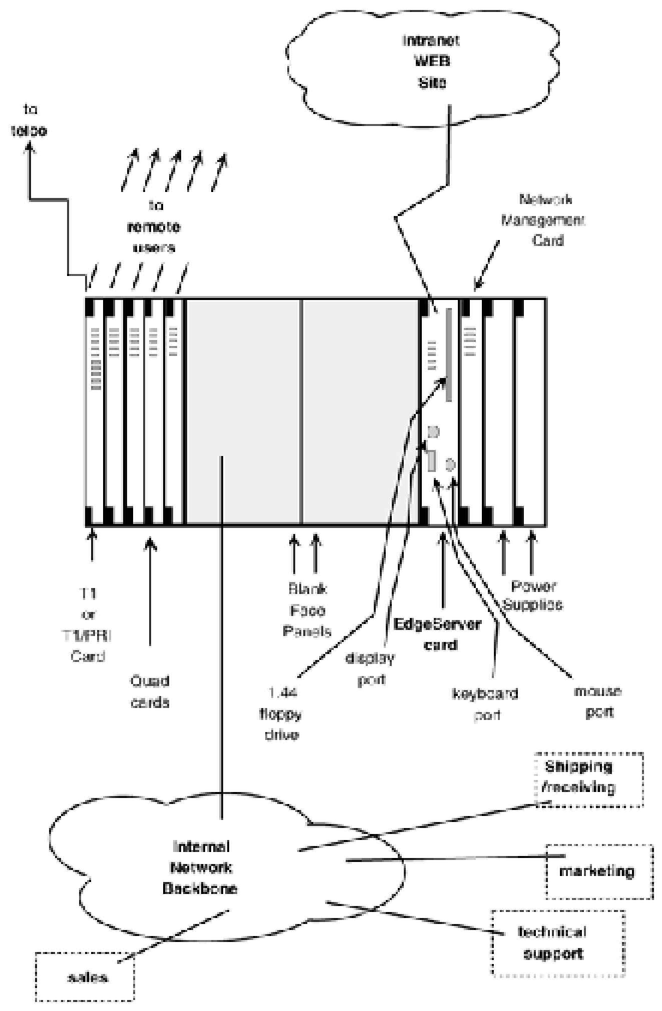


Figure 18-22
Hub functionality.

The EdgeServer card also has Microsoft Windows NT Server 4.0 installed on it. Conceptually the card and some of its functionality appears as shown in Fig. 18-22.

The advantage of having NT Server on the EdgeServer card is manyfold. First, when remote users access the communications hub for information purposes only, they can be stopped there and not access other systems that are part of the network. Second, remote users who require access to other systems can use the EdgeServer as a *gateway*, if you will, to access the network behind it. The EdgeServer card can function as an excellent firewall to protect the assets behind it while permitting access to it.

Still another powerful feature of the EdgeServer card is the SCSI port it has located on the back. This feature makes it possible to connect a CD-ROM drive to it. Documentation is provided with the Enterprise Network hub, but it is also provided via CD, which makes it convenient to access when manuals are not easily accessible.

A network management card is also part of the hub's component configuration. The card supports Ethernet and Token Ring as lower-layer protocols. It is a separate card, and it provides a console port that can be used for (1) *remote access*, permitting dialing into the hub from a remote site; (2) *local access*, enabling management to access the hub locally with an RJ-45 and DB-25 cable with a null-modem adapter provided with the hub; and (3) software download, to aid in the management aspect of the hub.

The network management card supports 10BaseT, 10Base5, and 10Base2 connection points for Ethernet cable flexibility. Token Ring cable support includes shielded twisted-pair (also called *IBM type 1*) and unshielded twisted-pair (also called *IBM type 3*) cable.

The network management card does not run SNMP management agents directly on the card; however, the support for SNMP is not compromised. The network management card technically functions as a proxy agent, but the functionality and management ability with SNMP operation features the same support.

The Enterprise hub used in this network has a T1 card. It operates as a *primary rate interface* (PRI). The T1 card is managed by the *Total Control Manager* (TCM). It is SMP-based and works with MS Windows. The card itself is easily configurable. Either a dumb terminal, remote PC, LAN PC, or direct-connect PC can work with configuration management parameters. Its operands function within the SNMP MIB standards; both GET and SET operations can be issued against the T1 card.

The T1 front panel includes LEDs to indicate the operational status of the card. Those LEDs include

ALARM	This LED is activated on existence of any of the following states: alarm indication signal, frame slip, out of frame, excessive CRC errors, change of frame alignment, line format violation, or frame alignment error.
CARRIER	This LED indicates whether a carrier is present; an unframed signal LED indicates if an out-of-frame or loss-of-signal condition occurs or if a signal is reported "not present."
LOOPBACK	This LED indicates if a test is in operation, initiated from the local telephone company.
RUN/FAIL	This LED indicates the operational mode of the T1 card; that is, if it is operating normal or in a "critical" mode due to a hardware and/or software fail condition.

The Enterprise Network hub modems are either analog or digital. Consider Fig. 18-23, which shows the Enterprise Network hub with analog modems (quad cards) connected to analog telephone lines. The gateway interface card in this illustration shows a generic connectivity to a network. The gateway connectivity portion of the Enterprise Network hub does not necessarily require a Windows NT Server; however, this implementation is popular.

Figure 18-24 shows a different hub implementation.

This illustration shows a T1 link to the telephone company. It also shows digital modems in use after a signal is in the hub. The gateway aspect of the hub indicates that users can have access outbound to a network if such is configured. Technically, the network access a remote user has is configurable for either analog or digital modem connectivity.

The Enterprise Network hub is only one of many products offered by 3Com USRobotics. For additional information, contact them at www.usr.com or

3Com USRobotics
Corporate Systems Division
8100 North McCormick Blvd.
Skokie, IL 60076

The data-communication hub used in this network provides robust throughput for remote users. Because of its ability to implement security well, a considerable degree of secure access has been obtained. Internal (local) users are also able to access the network because it functions as a node on the network, hence it is *seen* by other systems as a Windows NT node on the network.

18.10 Ethernet Network Hubs (Kingston)

Most networks have some form of hub or device that serves as the vehicle for the lower-layer network protocols. In this case Kingston's hubs were selected.

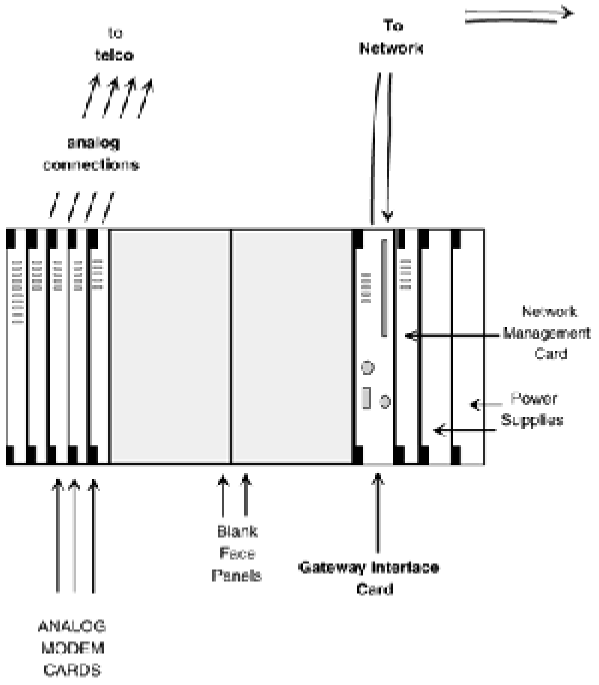


Figure 18-23
Network hub analog functionality.

The hubs in this network are 12- and 16-port. These are sufficient for operation of this network. The inherent design permits them to be "stacked" (no pun intended). They are RJ-45, multiport, stackable, rack-mount enclosures. Figure 18-25 illustrates how these appear inside the rack enclosure.

This illustration shows two hubs with cable connections in front, which makes it easier to access the hubs. One RJ-45-type cable is used to connect two or more of these hubs together. As shown in Fig. 18-26, ac cabling takes place at the rear of the hubs.

These hubs are straightforward and easy to use. They have LEDs on the front to indicate the status of each port. The documentation and support are also very good. Anyone who is already familiar with network hubs will find these hubs easy to use.

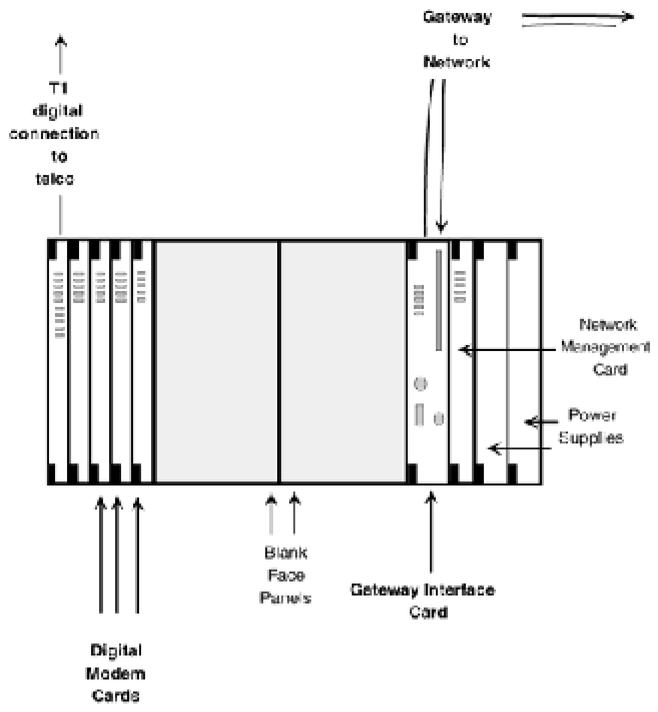


Figure 18-24
Network hub digital functionality.

18.11 Patch Panel (NetOptics)

Chaos will exist if wiring of any network is not planned for. Believe me, I speak from experience of walking through layers of cable behind equipment that was not labeled. With that thought in mind, the design for wiring in this network was examined from multiple angles, including considerations as to the reliability of the cablemaker and, in the actual implementation whether it would be easy to reconfigure equipment once installed and also how one would *know* which cables go where. With this level of thought and after performing multiple designs on the marker board, I decided to use Hubble equipment for the rack enclosure as well as for wiring.

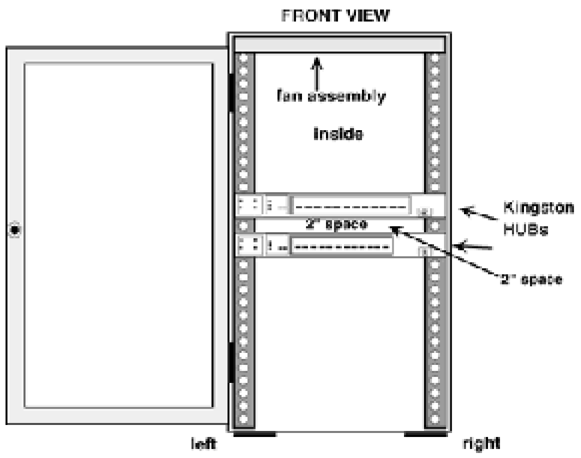


Figure 18-25
Kingston network hubs (front view).

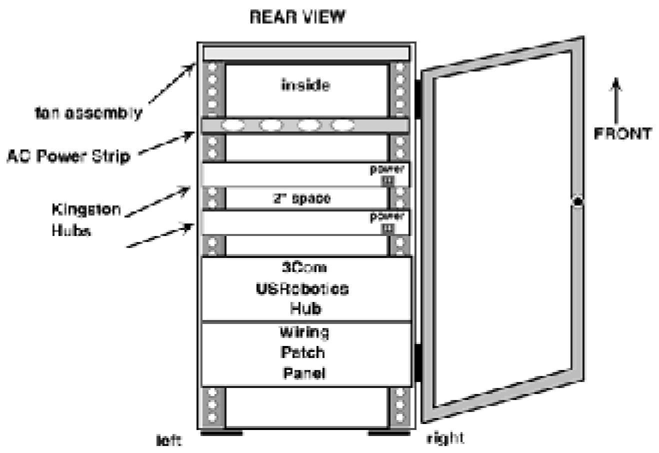


Figure 18-26
Kingston hubs (rear view).

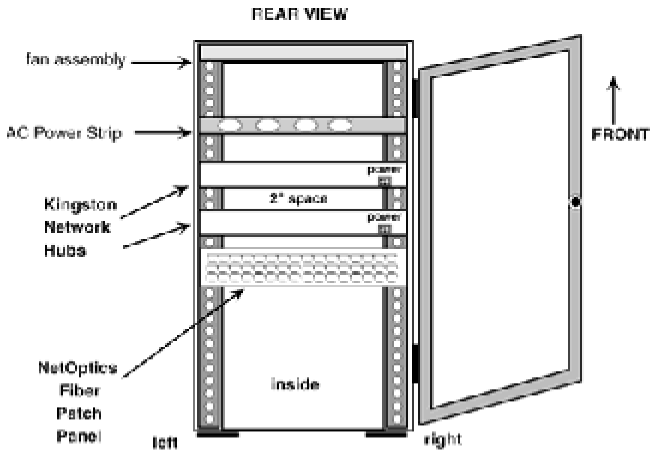


Figure 18-27
NetOptics fiber patch panel.

Figure 18-27 shows an inside view of the enclosure from the rear and also shows that the panel has SC connectors on what is considered the back side.

Figure 18-28 shows both front and rear views of the inline panel. The significance of this inline panel will become very clear later.

NetOptics offers fiber patch panels with various connectors, such as SC or ST.

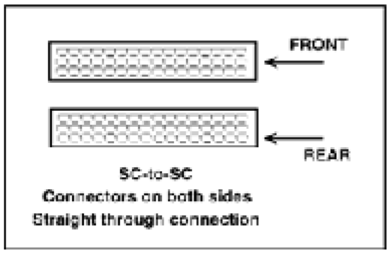


Figure 18-28
NetOptics patch panel.

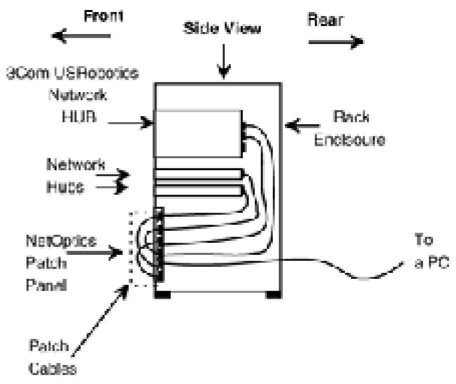


Figure 18-29
NetOptics panel patch (side view).

Figure 18-29 shows a side view of the rack enclosure. Note the USRobotics Enterprise Network hub, Kingston hubs, NetOptics Panel, and PC connections. Look closely at the area inside the broken-line rectangular box. This illustrates that any combination of connections can be achieved from the front of the enclosure. How? All equipment is connected into the rear of the inline panel. Actual physical configuration is then made via patch cables from the front. In this network a 48-port inline panel is used; however, Hubble has larger panels as well.

This example of component and wiring design is what I mean by designing ease of use, flexibility, and expandability into the network from the outset. Go one step further with Fig. 18-29 and consider this—with a MicroTest cable tester, any cables can be tested *from the front of all equipment* with relative ease. The same is true for troubleshooting or monitoring upper-layer protocols. The *Hewlett-Packard Internet Advisor* can be easily connected to the inline panel to monitor network traffic and other operating parameters. To make life even better, Hubble wiring is color-coded. During the installation of this equipment, colors, numbers, and labels were assigned to all ports, equipment, and other entry points to the network. There is no reason why a network design should not be documented completely from the very beginning. A well-designed network will be such that anyone who understands the technology will be able to perform any applicable work function.

If you have any questions about inline panels, wiring, fiber connections, surface-mount housing, jacks, or other components required to design your network, contact

NetOptics
1180-A Miraloma Way
Sunnyvale, CA 94086

18.12 Network Power Infrastructure (Thomas & Betts)

Another aspect to the network is its power infrastructure. Many installations build a network within an existing facility. Rarely will a facility and a network be built at the same time. If you are building a network in an existing facility, it would serve you well to do some site planning with the facilities manager, who in most cases is versed not only in the electrical but also the structural, air, and cooling capabilities within the facility.

In the case of my network, I built the entire infrastructure from the ground up—no pun intended. Because of the scope of my project, I met with a couple of Thomas & Betts representatives on more than one occasion. This meeting was enlightening for me because the T&B representative brought some topics out that had not been discussed before in regard to the facility I built.

The final result was my decision to use the following T&B products to make the facility functional: load centers, circuit breakers, dual-gang cover plates, metal box housings for receptacles and switches, wirewrap ties, metal couplings to secure wiring, NEMA (National Electrical Manufacturers Assoc.) twist lock cover plates, 6- and 8-gauge connectors, and quad boxes.

Figure 18-30 is a sample view of the network electrical infrastructure. The electrical components shown in Fig. 18-30 were from Thomas & Betts. The work was supplied by me. It is a sample of the network; not all hosts are shown.

Figure 18-31 shows another aspect of the electrical infrastructure of the network.

Regardless of the location, the power infrastructure is fundamental to the network. Without it networks will not operate. That’s right. Without the power infrastructure and electrical power, no networks of any significance known to humankind today will operate. With this in mind, you should be aware of how critical it is to know the infrastructure, and be prepared to make changes if any are required.

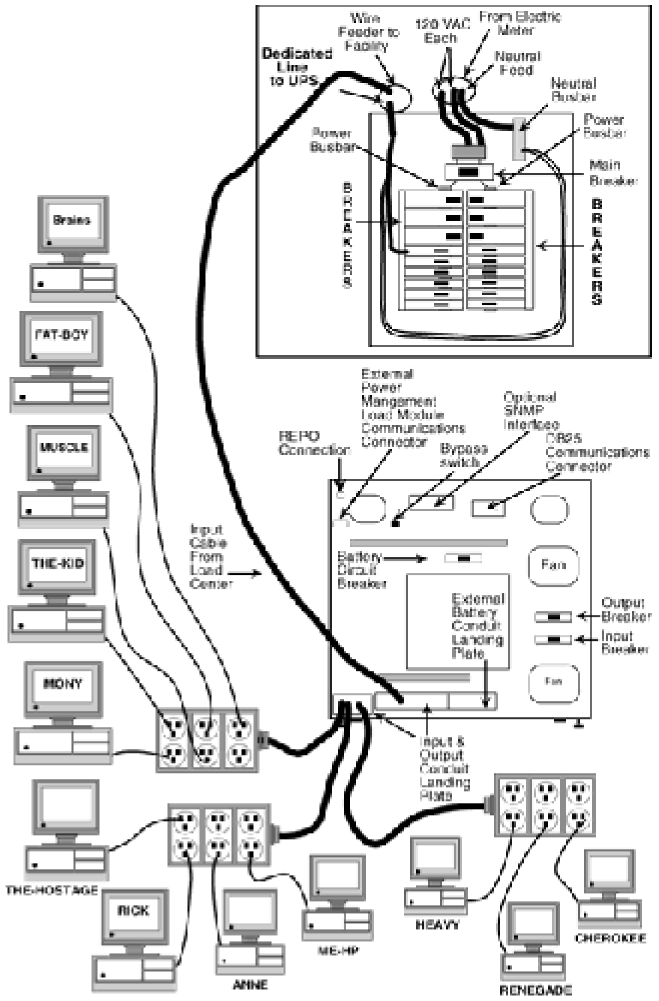


Figure 18-30
Conceptual view of the electrical infrastructure.

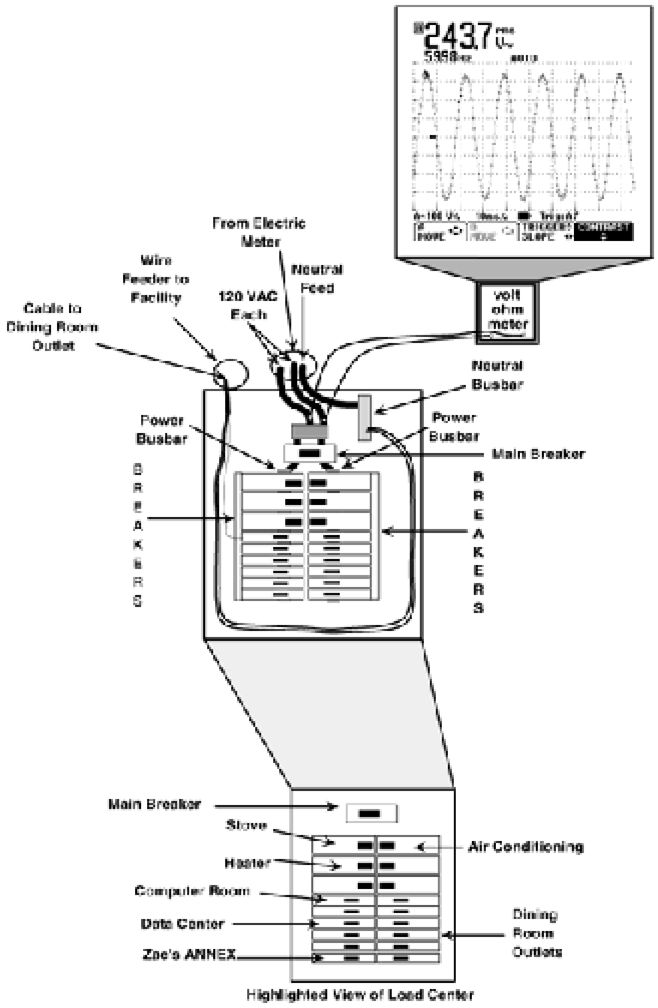


Figure 18-31
Highlighted view of a different perspective of the electrical infrastructure.

18.13 Network Server (IBM NetFinity 7000)

The network server is one of the most critical components of the network. I chose a NetFinity 7000 to be the server of choice in this network. Here are its specifications again so you don't have to go back to previous sections to refresh your memory:

Three (3) 200-MHz Pentium processors
0.75 GB of RAM (750 MB RAM)
39-MB hard disk (RAID support)
RAID implemented
Two 200-W power supplies
512-kB L2 cache
Network management interface board
Ethernet Network interface card
IBM's standard software bundle

Figure 18-32 shows a conceptual view of how the NetFinity 7000 is implemented and some features it provides to users, both local and remote. The network server shows the following applications available to all network users: DB2 server, Lotus Domino, Lotus SmartSuite, and RAID Manager.

These applications are for all users except the RAID manager; this application is for the user with administrative privileges. The NetFinity 7000 used in this network includes the configuration which is a result of the diagnostic report shown in Fig. 18-33.

The CHEROKEE, as the network server is called in my network, is more than capable of not only providing applications to network users but also performing certain file server functions. Implementation of RAID with the server makes it a strong component in the network and a safe server to use for sensitive data.

18.14 Operating System Software

Windows 95, Windows 98, Windows NT4.0, Windows NT 5.0, and SCO UNIX were used in the network. To understand the use of these operating systems, consider the information extracted from two sample workstations on the network. Figure 18-34 is a two-part report. The first part is a system summary report from FAT-BOY (a Windows 95-based computer on the network). The second is a report from ANGEL (a Windows NT workstation on the network).

Windows NT could be the standard against which other operating systems are judged for some time to come. From appearances, it has the “look and feel” interface of Windows 95. This version of NT has robust features and is separated into what is called the *NT server* and the *NT workstation*, which share common characteristics such as advanced file-handling systems backup capabilities, C2 security, graphical user interface (GUI) tools, network capabilities, remote access capabilities, and TCP/IP.

More detailed information can be obtained from Microsoft. I recommend you contact them, at www.microsoft.com.

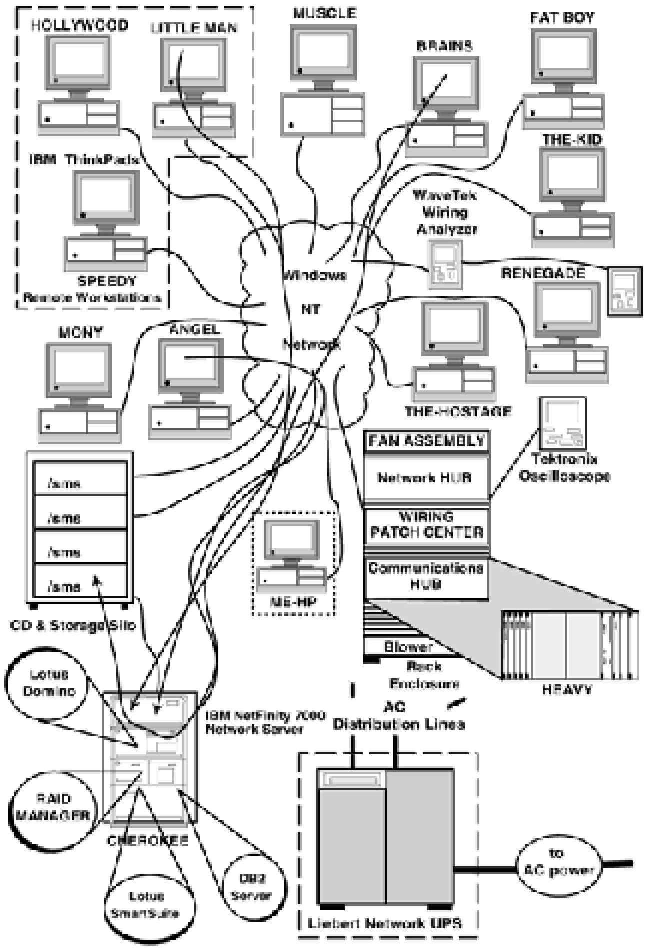


Figure 18-32
NetFinity 7000 network server.

```
————————————
OS Version Report
————————————
Microsoft (R) Windows NT (TM) Server
Version 4.0 (Build 1381: Service Pack 3) x86 Multiprocessor Free
Registered Owner: Ed Taylor
Product Number: 51222-270-1251174-59767
————————————
System Report
————————————
System: AT/AT COMPATIBLE
Hardware Abstraction Layer: MPS 1.4 - APIC platform
BIOS Date: 10/20/97
BIOS Version: BIOS Version 1.00.14.CD0
Processor list:
  0: x86 Family 6 Model 1 Stepping 9 GenuineIntel ~198 Mhz
  1: x86 Family 6 Model 1 Stepping 9 GenuineIntel ~198 Mhz
  2: x86 Family 6 Model 1 Stepping 9 GenuineIntel ~198 Mhz
————————————
Video Display Report
————————————
BIOS Date: 01/07/94
BIOS Version: CL-GD542X VGA BIOS Version 1.41
Adapter:
  Setting: 640 × 480 × 256
```

```
          60 Hz
  Type: cirrus compatible display adapter
  String: Cirrus Logic Compatible
  Memory: 512 KB
  Chip Type: CL 5424
  DAC Type: Integrated RAMDAC
Driver:
  Vendor: Microsoft Corporation
  File(s): cirrus.sys, vga.dll, cirrus.dll, vga256.dll, vga64K.dll
  Version: 4.00, 4.0.0
────────────
Drives Report
────────────
C:\ (Local - NTFS) CHEROKEE Total: 0KB, Free: 0KB
D:\ (Removable - FAT) Total: 1,045,200KB, Free: 1,045,184KB
E:\ (Local - NTFS) CHEROKEE-E Total: 37,556,692KB, Free:
     36,256,336KB
G:\ (CDROM - CDFS) IE_DISP_R1_0 Total: 189,020KB, Free: 0KB
H:\ (CDROM - CDFS) PUBS1310 Total: 366,396KB, Free: 0KB
I:\ (CDROM - CDFS) ROCKYSERVICEP Total: 178,218KB, Free: 0KB
────────────
Memory Report
────────────
```

Figure 18-33
Diagnostics report for \\CHEROKEE.

```
Handles: 2,088 Threads: 169
Processes: 28
Physical Memory (K)
  Total: 785,840
  Available: 707,232
  File Cache: 15,748
────────────
Services Report
────────────
Alerter     Running (Automatic)
Computer Browser              Running (Automatic)
EventLog (Event log)          Running (Automatic)
Server      Running (Automatic)
Workstation (NetworkProvider)         Running (Automatic)
License Logging Service               Running (Automatic)
TCP/IP NetBIOS Helper        Running (Automatic)
Messenger   Running (Automatic)
Netfinity Support Program             Running (Automatic)
Net Logon (RemoteValidation)          Running (Automatic)
Norton SpeedDisk             Running (Automatic)
Plug and Play (PlugPlay)              Running (Automatic)
Remote Access Connection Manager (Network)          Running (Automatic)
Remote Access Server (Network)        Running (Automatic)
Remote Procedure Call (RPC) Locator           Running (Automatic)
Remote Procedure Call (RPC) Service           Running (Automatic)
Spooler (SpoolerGroup)          Running (Automatic)
Telephony Service          Running (Manual)
────────────
Drivers Report
────────────
AFD Networking Support Environment (TDI)          Running (Automatic)
```

```
aic78xx (SCSI miniport)           Running (Boot)
Remote Access Mac (NDIS)          Running (Automatic)
atapi (SCSI miniport)          Running (Boot)
Beep (Base)       Running (System)
Cdfs (File system)             Running (Disabled)
Cdrom (SCSI CDROM Class)       Running (System)
cirrus (Video)          Running (System)
Disk (SCSI Class)       Running (Boot)
Diskperf (Filter)       Running (Boot)
Intel 82557-based PRO Adapter Driver (NDIS)          Running (Automatic)
Fastfat (Boot file system)     Running (Disabled)
Floppy (Primary disk)          Running (System)
i8042 Keyboard and PS/2 Mouse Port Driver (Keyboard Port) Running
          (System)
ipsraidn (SCSI miniport)       Running (Boot)
Keyboard Class Driver (Keyboard Class)              Running (System)
KSecDD (Base)          Running (System)
Modem (Extended base)          Running (Boot)
Mouse Class Driver (Pointer Class)                  Running (System)
Msfs (File system)          Running (System)
Mup (Network)               Running (Manual)
NetBEUI Protocol (PNP_TDI)          Running (Automatic)
```

```
Microsoft NDIS System Driver (NDIS)          Running (System)
Microsoft NDIS TAPI driver (NDIS)                    Running (System)
Remote Access WAN Wrapper (NDISWAN)          Running (Automatic)
NetBIOS Interface (NetBIOSGroup)                    Running (Automatic)
WINS Client(TCP/IP) (PNP_TDI)          Running (Automatic)
NetFin (Extended base)          Running (Automatic)
Npfs (File system)          Running (System)
Ntfs (File system)          Running (Disabled)
Null (Base)          Running (System)
NWLink IPX/SPX Compatible Transport Protocol (PNP_TDI) Running
(Automatic)
NWLink NetBIOS (PNP_TDI)          Running (Automatic)
NWLink SPX/SPXII Protocol          Running (Manual)
Parallel (Extended base)          Running (Automatic)
Parport (Parallel arbitrator)          Running (Automatic)
ParVdm (Extended base)          Running (Automatic)
Remote Access Auto Connection Driver (Streams Drivers) Running
(Automatic)
Remote Access ARP Service (PNP_TDI)          Running (Automatic)
Rdr (Network)          Running (Manual)
Scsiscan (SCSI Class)          Running (System)
Serial (Extended base)          Running (Automatic)
Srv (Network)          Running (Automatic)
TCP/IP Service (PNP_TDI)          Running (Automatic)
────────────
IRQ and Port Report
────────────
Devices                                               Vector Level Affinity
────────────
MPS 1.4 - APIC platform          8          8 0x00000007
MPS 1.4 - APIC platform          0          0 0x00000007
MPS 1.4 - APIC platform          1          1 0x00000007
MPS 1.4 - APIC platform          2          2 0x00000007
MPS 1.4 - APIC platform          3          3 0x00000007
MPS 1.4 - APIC platform          4          4 0x00000007
MPS 1.4 - APIC platform          5          5 0x00000007
MPS 1.4 - APIC platform          6          6 0x00000007
MPS 1.4 - APIC platform          7          7 0x00000007
MPS 1.4 - APIC platform          8          8 0x00000007
MPS 1.4 - APIC platform          9          9 0x00000007
MPS 1.4 - APIC platform          10         10 0x00000007
MPS 1.4 - APIC platform          11         11 0x00000007
MPS 1.4 - APIC platform          12         12 0x00000007
MPS 1.4 - APIC platform          13         13 0x00000007
MPS 1.4 - APIC platform          14         14 0x00000007
MPS 1.4 - APIC platform          15         15 0x00000007
MPS 1.4 - APIC platform          16         16 0x00000007
MPS 1.4 - APIC platform          17         17 0x00000007
MPS 1.4 - APIC platform          18         18 0x00000007
MPS 1.4 - APIC platform          19         19 0x00000007
MPS 1.4 - APIC platform          20         20 0x00000007
MPS 1.4 - APIC platform          21         21 0x00000007
MPS 1.4 - APIC platform          22         22 0x00000007
```

```
MPS 1.4 - APIC platform          23          23 0x00000007
MPS 1.4 - APIC platform          24          24 0x00000007
MPS 1.4 - APIC platform          25          25 0x00000007
MPS 1.4 - APIC platform          26          26 0x00000007
MPS 1.4 - APIC platform          27          27 0x00000007
MPS 1.4 - APIC platform          28          28 0x00000007
MPS 1.4 - APIC platform          29          29 0x00000007
MPS 1.4 - APIC platform          30          30 0x00000007
MPS 1.4 - APIC platform          31          31 0x00000007
MPS 1.4 - APIC platform          32          32 0x00000007
MPS 1.4 - APIC platform          33          33 0x00000007
MPS 1.4 - APIC platform          34          34 0x00000007
MPS 1.4 - APIC platform          35          35 0x00000007
MPS 1.4 - APIC platform          36          36 0x00000007
MPS 1.4 - APIC platform          37          37 0x00000007
MPS 1.4 - APIC platform          38          38 0x00000007
MPS 1.4 - APIC platform          39          39 0x00000007
MPS 1.4 - APIC platform          40          40 0x00000007
MPS 1.4 - APIC platform          41          41 0x00000007
MPS 1.4 - APIC platform          42          42 0x00000007
MPS 1.4 - APIC platform          43          43 0x00000007
MPS 1.4 - APIC platform          44          44 0x00000007
MPS 1.4 - APIC platform          45          45 0x00000007
MPS 1.4 - APIC platform          46          46 0x00000007
MPS 1.4 - APIC platform          47          47 0x00000007
MPS 1.4 - APIC platform          61          61 0x00000007
MPS 1.4 - APIC platform          65          65 0x00000007
MPS 1.4 - APIC platform          80          80 0x00000007
MPS 1.4 - APIC platform          193         193 0x00000007
MPS 1.4 - APIC platform          225         225 0x00000007
MPS 1.4 - APIC platform          253         253 0x00000007
MPS 1.4 - APIC platform          254         254 0x00000007
MPS 1.4 - APIC platform          255         255 0x00000007
i8042prt                         1           1 0xffffffff
i8042prt                         12          12 0xffffffff
Serial                           4           4 0x00000000
Serial                           3           3 0x00000000
E100B                            9           9 0x00000000
Floppy                           6           6 0x00000000
aic78xx                          11          11 0x00000000
aic78xx                          10          10 0x00000000
```

```
aic78xx                          5           5 0x00000000
aic78xx                          15          15 0x00000000
atapi                            0           14 0x00000000
ipsraidn                         11          11 0x00000000
────────────
Devices                                         Physical Address Length
────────────
MPS 1.4 - APIC platform          0x00000000 0x0000000010
MPS 1.4 - APIC platform          0x00000020 0x0000000002
MPS 1.4 - APIC platform          0x00000040 0x0000000004
MPS 1.4 - APIC platform          0x00000048 0x0000000004
MPS 1.4 - APIC platform          0x00000061 0x0000000001
```

```
MPS 1.4 - APIC platform          0x00000070 0x0000000002
MPS 1.4 - APIC platform          0x00000080 0x0000000010
MPS 1.4 - APIC platform          0x00000092 0x0000000001
MPS 1.4 - APIC platform          0x000000a0 0x0000000002
MPS 1.4 - APIC platform          0x000000c0 0x0000000010
MPS 1.4 - APIC platform          0x000000d0 0x0000000010
MPS 1.4 - APIC platform          0x000000f0 0x0000000010
MPS 1.4 - APIC platform          0x00000400 0x0000000010
MPS 1.4 - APIC platform          0x00000461 0x0000000002
MPS 1.4 - APIC platform          0x00000464 0x0000000002
MPS 1.4 - APIC platform          0x00000480 0x0000000010
MPS 1.4 - APIC platform          0x000004c2 0x000000000e
MPS 1.4 - APIC platform          0x000004d0 0x0000000002
MPS 1.4 - APIC platform          0x000004d4 0x000000002c
MPS 1.4 - APIC platform          0x00000c84 0x0000000001
i8042prt                         0x00000060 0x0000000001
i8042prt                         0x00000064 0x0000000001
Parport                          0x00000378 0x0000000003
Serial                           0x000003f8 0x0000000007
Serial                           0x000002f8 0x0000000007
E100B                            0x0000e4e0 0x0000000014
Floppy                           0x000003f0 0x0000000006
Floppy                           0x000003f7 0x0000000001
aic78xx                          0x0000ec00 0x0000000100
aic78xx                          0x0000e800 0x0000000100
aic78xx                          0x0000dc00 0x0000000100
aic78xx                          0x0000d800 0x0000000100
atapi                            0x000001f0 0x0000000008
atapi                            0x000003f6 0x0000000001
ipsraidn                         0x0000fc00 0x0000000100
cirrus                           0x000003b0 0x000000000c
cirrus                           0x000003c0 0x0000000020
────────────
DMA and Memory Report
────────────
Devices                    Channel         Port
────────────
Floppy                    2                   0
────────────
Devices                                         Physical Address Length
────────────
MPS 1.4 - APIC platform          0xfec00000 0x00000400
MPS 1.4 - APIC platform          0xfec08000 0x00000400
E100B                            0xfe0ff000 0x00000014
E100B                            0xfe0ff000 0x00000014
aic78xx                          0xfe8ff000 0x00001000
```

```
aic78xx                          0xfe8fe000 0x00001000
aic78xx                          0xfe1ff000 0x00001000
aic78xx                          0xfe1fe000 0x00001000
ipsraidn                         0xfeafe000 0x00002000
cirrus                           0x000a0000 0x00020000
```

```
Environment Report
────────────
System Environment Variables
  ComSpec = C:\WINNT\system32\cmd.exe
  NUMBER_OF_PROCESSORS = 3
  OS = Windows_NT
  Os2LibPath = C:\WINNT\system32\os2\dll;
  Path = E:\Notes;C:\WINNT\system32;C:\WINNT;C:\WNETFIN
  PROCESSOR_ARCHITECTURE = x86
  PROCESSOR_IDENTIFIER = x86 Family 6 Model 1 Stepping 9, GenuineIntel
  PROCESSOR_LEVEL = 6
  PROCESSOR_REVISION = 0109
  windir = C:\WINNT
Environment Variables for Current User
  TEMP = C:\TEMP
  TMP = C:\TEMP
────────────
Network Report
────────────
Your Access Level: Admin & Local (total control)
Workgroup or Domain: INFO
Network Version: 4.0
LanRoot: INFO
Logged On Users: 1
Current User (1): Administrator
  Logon Domain: INFO
  Logon Server: CHEROKEE
Transport: NetBT_E100B1, 00-60-94-A5-CC-EE, VC's: 0, Wan: Wan
Transport: Nbf_E100B1, 00-60-94-A5-CC-EE, VC's: 0, Wan: Wan
Transport: NwlnkNb, 00-60-94-A5-CC-EE, VC's: 0, Wan: Wan
Character Wait: 3,600
Collection Time: 250
Maximum Collection Count: 16
Keep Connection: 600
Maximum Commands: 5
Session Time Out: 45
Character Buffer Size: 512
Maximum Threads: 17
Lock Quota: 6,144
Lock Increment: 10
Maximum Locks: 500
Pipe Increment: 10
Maximum Pipes: 500
Cache Time Out: 40
Dormant File Limit: 45
Read Ahead Throughput: 4,294,967,295
Mailslot Buffers: 3
Server Announce Buffers: 20
Illegal Datagrams: 5
Datagram Reset Frequency: 60
Bytes Received: 3,395
```

```
SMB's Received: 33
```

```
                             I. FAT-BOY
******************** SYSTEM SUMMARY ********************
Windows version: 4.00.950 Computer Name: FAT-BOY
Processor Type: Pentium
System BUS Type: ISA
BIOS Name: IBM
BIOS Date: 05/29/96
BIOS Version: Unknown
Machine Type: IBM PC/AT
Math Co-processor: Not Present
Registered Owner: Ed Taylor
Registered Company: Information World, Inc.
******************** IRQ SUMMARY ********************
IRQ Usage Summary:3#dr 00 - System timer
01 - Standard 101/102-Key or Microsoft Natural Keyboard
02 - Programmable interrupt controller
03 - Sportster 33600 Fax PC Plug and Play
04 - Communications Port (COM3)
05 - Creative Labs Sound Blaster 16 Plug and Play
06 - Standard Floppy Disk Controller
07 - ECP Printer Port (LPT1)
08 - System CMOS/real time clock
09 - PCI Card
09 - S3
10 - SMC EtherEZ (8416)
11 - IBM MPEG Interactive Video Player
12 - Standard PS/2 Port Mouse
13 - Numeric data processor
14 - Standard Dual PCI IDE Controller
14 - Primary IDE controller (single fifo)
15 - Standard Dual PCI IDE Controller
15 - Secondary IDE controller (single fifo) 3F
******************** I/O PORT SUMMARY ********************
I/O Port Usage Summary:3 # 0000h-001Fh - Direct Memory Access
Controller
0020h-0021h - Programmable interrupt controller
002Eh-002Fh - Motherboard resources
0040h-0043h - System timer
0060h-0060h - Standard 101/102-Key or Microsoft Natural Keyboard
0061h-0061h - System speaker
0064h-0064h - Standard 101/102-Key or Microsoft Natural Keyboard 3
**************** System Resource Report ********************
3F 0070h-0071h - System CMOS/real time clock 3 # 0081h-0083h -
Direct Memory Access Controller
0087h-0087h - Direct memory access controller
0089h-008Bh - Direct memory access controller
008Fh-008Fh - Direct memory access controller
00A0h-00A1h - Programmable interrupt controller
00C0h-00DFh - Direct memory access controller
00F0h-00FFh - Numeric data processor
0100h-0100h - Creative SB32 PnP
```

Figure 18-34
Windows system resource and workstation reports.

```
0170h-0177h - Standard Dual PCI IDE Controller
0170h-0177h - Secondary IDE controller (single fifo)
01F0h-01F7h - Primary IDE controller (single fifo)
01F0h-01F7h - Standard Dual PCI IDE Controller
0200h-0207h - Gameport Joystick
0200h-0203h - IBM MPEG Interactive Video Player
0220h-023Fh - SMC EtherEZ (8416)
0240h-024Fh - Creative Labs Sound Blaster 16 Plug and Play
0270h-0273h - IO read data port for ISA Plug and Play enumerator
02F8h-02FFh - Sportster 33600 Fax PC Plug and Play
0300h-0301h - Creative Labs Sound Blaster 16 Plug and Play
0376h-0376h - Standard Dual PCI IDE Controller
0376h-0376h - Secondary IDE controller (single fifo)
0388h-038Bh - Creative Labs Sound Blaster 16 Plug and Play
03B0h-03BBh - S3
03BCh-03BEh - ECP Printer Port (LPT1)
03C0h-03DFh - S3
03E8h-03EFh - Communications Port (COM3)
03F0h-03F5h - Standard Floppy Disk Controller
03F6h-03F6h - Primary IDE controller (single fifo)
03F6h-03F6h - Standard Dual PCI IDE Controller
0660h-0663h - Creative Advanced Wave Effects Synthesis for AWE 32
0CF8h-0CFFh - PCI bus
1000h-101Fh - PCI Card
FFF0h-FFF7h - Standard Dual PCI IDE Controller
FFF8h-FFFFh - Standard Dual PCI IDE Controller 3F
****************** UPPER MEMORY USAGE SUMMARY **********************
Memory Usage Summary: 3 #00000000h-0009FFFFh - System board exten-
sion for PnP BIOS
000A0000h-000AFFFFh - S3
000B0000h-000BFFFFh - S3
000C0000h-000C7FFFh - S3
000E0000h-000FFFFFh - System board extension for PnP BIOS
00100000h-00FFFFFFh - System board extension for PnP BIOS
40000000h-43FFFFFFh - S3
**************** System Resource Report ****************************
DMA USAGE SUMMARY
DMA Channel Usage Summary:3#
02 - Standard Floppy Disk Controller
03 - Creative Labs Sound Blaster 16 Plug and Play
04 - Direct memory access controller
06 - Creative Labs Sound Blaster 16 Plug and Play
07 - IBM MPEG Interactive Video Player3F
******************** MEMORY SUMMARY ********************
640 KB Total Conventional Memory
32288 KB Total Extended Memory
******************** DISK DRIVE INFO ********************
A: Floppy Drive, 3.5" 1.44M 80 Cylinders 2 Heads
512 Bytes/Sector 18 Sectors/TrackC: Fixed Disk 1664960K
Total 935744K Free 826 Cylinders 64 Heads
512 Bytes/Sector 63 Sectors/TrackD: CD-ROM Drive
************************ SYSTEM DEVICE INFO *********************
Class: Other devices
```

```
Device: Creative SB32 PnP
Resources:
I/O: 0100h-0100h Class: Other devices Device: IBM MPEG Interactive
Video layer
Resources:
IRQ: 11
I/O: 0200h-0203h
DMA: 07 Class: Other devices Device: PCI Card
Resources: IRQ: 09
I/O: 1000h-101Fh Class: Network adapters System Resource Report
******************************
Device: SMC EtherEZ (8416)
Resources: IRQ: 10
I/O: 0220h-023Fh Class: Network adapters Device: Dial-Up Adapter
No resources used. DISABLED DEVICE Class: Ports (COM & LPT)
Device: Generic IRDA Compatible Device
No resources used. Class: Ports (COM & LPT) Device: ECP Printer
Port (LPT1)
Resources:
IRQ: 07
I/O: 03BCh-03BEh
Device drivers:
C:\WINDOWS\SYSTEM\lpt.vxd
File size: 0 bytes.
Manufacturer: Microsoft Corporation
File version: 4.00.950
Copyright: Copyright + Microsoft Corp. 1992–1995 Class: Ports (COM & LPT)
Device: Communications Port (COM3)
Resources: IRQ: 04
I/O: 03E8h-03EFh
Device drivers: C:\WINDOWS\SYSTEM\serial.vxd
File size: 18572 bytes.
Manufacturer: Microsoft Corporation
File version: 4.00.950
Copyright: Copyright Microsoft Corp. 1992-1995
C:\WINDOWS\SYSTEM\serialui.dll
File size: 12032 bytes.
Manufacturer: Microsoft Corporation
File version: 4.00.950
Copyright: Copyright Microsoft Corp. 1993–1995
Class: Mouse Device: Standard PS/2 Port Mouse
Resources: IRQ: 12
************************* System Resource Report*************************
Device drivers: C:\WINDOWS\SYSTEM\mouse.drv
File size: 7712 bytes.
Manufacturer: Microsoft Corporation
File version: 9.01.0.000
Copyright: Copyright Microsoft Corp. 1990-1995
C:\WINDOWS\SYSTEM\msmouse.vxd
File size: 15804 bytes.
Manufacturer: Microsoft Corporation
File version: 4.00.950
```

```
Copyright: Copyright Microsoft Corp. 1988–1995
Class: Hard disk controllers Device: Secondary IDE controller (single fifo)
Resources: IRQ: 15
I/O: 0170h-0177h
I/O: 0376h-0376h Class: Hard disk controllers Device: Primary IDE controller
(single fifo)
Resources: IRQ: 14
I/O: 01F0h-01F7h
I/O: 03F6h-03F6h
*DISABLED DEVICE* Class: Hard disk controllers
Device: Standard IDE/ESDI Hard Disk Controller
No resources used. Class: Hard disk controllers Device: Standard Dual PCI DE
Controller
Resources:
IRQ: 14
IRQ: 15
I/O: 01F0h-01F7h
I/O: 03F6h-03F6h
I/O: 0170h-0177h
I/O: 0376h-0376h
I/O: FFF0h-FFF7h
I/O: FFF8h-FFFFh
Class: Floppy disk controllers Device: Standard Floppy Disk Controller
Resources: IRQ: 06
**************************System Resource Report********************
I/O: 03F0h-03F5h DMA: 02 Class: Display adapters Device: S3
Resources: IRQ: 09
I/O: 03B0h-03BBh
I/O: 03C0h-03DFh
MEM: 000C0000h-000C7FFFh
MEM: 000A0000h-000AFFFFh
MEM: 000B0000h-000BFFFFh
MEM: 40000000h-43FFFFFFh
Device drivers:
C:\WINDOWS\SYSTEM\orchid.cp0
Driver file possibly missing.
No version information.
C:\WINDOWS\SYSTEM\orchidf.cp0
Driver file possibly missing.
No version information.
C:\WINDOWS\SYSTEM\s3.drv
File size: 57632 bytes.
Manufacturer: Microsoft Corporation
File version: 4.00.950
Copyright: Copyright + Microsoft Corp. 1992–1995
C:\WINDOWS\SYSTEM\s3.vxd
File size: 17087 bytes.
Manufacturer: Microsoft Corporation
File version: 4.00.950
Copyright: Copyright + Microsoft Corp. 1988–1995
C:\WINDOWS\SYSTEM\supervga.drv
```

```
File size: 52320 bytes.
Manufacturer: Microsoft Corporation
File version: 4.00.950
Copyright: Copyright Microsoft Corp. 1991–1995
Class: CDROM Device: TEAC CD-56E
No resources used. Class: Monitor Device: Plug and Play Monitor
(VESA DDC)
No resources used. Class: Modem Device: Sportster 33600 Fax PC Plug
and Play
Resources:System Resource Report
IRQ: 03 I/O: 02F8h-02FFh Class: Modem Device: Courier Dual
Standard V.34 Ready Fax
No resources used. Class: System devices
Device: I/O read data port for ISA Plug and Play enumerator
Resources: I/O: 0270h-0273h
Class: System devices Device: PCI standard ISA bridge
No resources used. Class: System devices Device: PCI standard host
CPU bridge
No resources used. Class: System devices Device: Motherboard
resources
Resources: I/O: 002Eh-002Fh
Class: System devices Device: Motherboard resources
No resources used. Class: System devices Device: System board
extension for PnP BIOS
Resources: MEM: 000E0000h-000FFFFFh Class:
System devices Device: System board extension for PnP BIOS
Resources: MEM: 00000000h-0009FFFFh
MEM: 00100000h-00FFFFFFh Class: System devices Device: PCI bus
Resources: I/O: 0CF8h-0CFFh
Device drivers:
C:\WINDOWS\SYSTEM\pci.vxd
File size: 24535 bytes.
****************System Resource Report**********************
Manufacturer: Microsoft Corporation File version: 4.00.950
Copyright: Copyright Microsoft Corp. 1988–-1995 Class: System devices
Device: Numeric data processor
Resources: IRQ: 13
I/O: 00F0h-00FFh Class: System devices Device: System speaker
Resources: I/O: 0061h-0061h Class: System devices
Device: System CMOS/real time clock
Resources: IRQ: 08
I/O: 0070h-0071h Class: System devices Device: System timer
Resources: IRQ: 00
I/O: 0040h-0043h Class: System devices Device:
Direct memory access controller
Resources:
I/O: 0000h-001Fh
I/O: 0081h-0083h
I/O: 0087h-0087h
I/O: 0089h-008Bh
I/O: 008Fh-008Fh
I/O: 00C0h-00DFh
```

```
DMA: 04 Class: System devices Device: Programmable interrupt con-
troller
Resources: IRQ: 02
I/O: 0020h-0021h
I/O: 00A0h-00A1h Class: System devices
Device: Advanced Power Management support System Resource
Report No resources used.
Class: System devices Device: Plug and Play BIOS
No resources used.
Device drivers:
  C:\WINDOWS\SYSTEM\VMM32\bios.vxd
No version information.Class: System devices Device: System board
No resources used. Class:
Keyboard Device: Standard 101/102-Key or Microsoft Natural Keyboard
Resources: IRQ: 01
I/O: 0060h-0060h
I/O: 0064h-0064h
Device drivers:
C:\WINDOWS\SYSTEM\keyboard.drv
File size: 12688 bytes.
Manufacturer: Microsoft Corporation
File version: 4.00.950
Copyright: Copyright Microsoft Corp. 1991–1995
C:\WINDOWS\SYSTEM\VMM32\vkd.vxd
No version information.
*DISABLED DEVICE* Class: Sound, video and game controllers
Device: Gameport Joystick
No resources used.
Device drivers:
C:\WINDOWS\SYSTEM\vjoyd.vxd
File size: 20590 bytes.
Manufacturer: Microsoft Corporation
File version: 4.00.950
Copyright: Copyright Microsoft Corp. 1994–1995
C:\WINDOWS\SYSTEM\msjstick.drv
File size: 7744 bytes.
Manufacturer: Microsoft Corporation
File version: 4.0.950
Copyright: Copyright Microsoft Corp. 1991–1995
Class: Sound, video and game controllers Device: Creative Advanced
Wave Effects Synthesis for AWE 32
****************System Resource Report************************
I/O: 0660h-0663h Device drivers:
C:\WINDOWS\SYSTEM\sbawe.vxd
File size: 40014 bytes.
Manufacturer: Creative Technology Ltd.
File version: 4.00.466
Copyright: Copyright (c) 1993-95 Creative Technology Ltd.
C:\WINDOWS\SYSTEM\sbawe32.drv
File size: 23216 bytes.
Manufacturer: Creative Technology Ltd.
File version: 4.00
```

```
Copyright: Copyright (c) 1993–1995 Creative Technology Ltd.
C:\WINDOWS\SYSTEM\synthgm.sbk
File size: 34832 bytes.
No version information. Class: Sound, video and game controllers
Device: Creative Labs Sound Blaster 16 Plug and Play
Resources: IRQ: 05
I/O: 0240h-024Fh
I/O: 0300h-0301h
I/O: 0388h-038Bh
DMA: 03
DMA: 06
Device drivers:
C:\WINDOWS\SYSTEM\cspman.dll
File size: 17776 bytes.
Manufacturer: Creative Technology Ltd.
File version: 4.00
Copyright: Copyright Creative Technology Ltd. 1994–1995
C:\WINDOWS\SYSTEM\sb16.vxd
File size: 54363 bytes.
Manufacturer: Creative Technology Ltd.
File version: 4.00.493
Copyright: Copyright Creative Technology Ltd. 1994–1995
C:\WINDOWS\SYSTEM\sbfm.drv
File size: 4128 bytes.
Manufacturer: Creative Technology Ltd.
File version: 4.00
Copyright: Copyright Creative Technology Ltd. 1994–1995
C:\WINDOWS\SYSTEM\sb16snd.drv
File size: 46000 bytes.
Manufacturer: Creative Technology Ltd.
File version: 4.00
Copyright: Copyright Creative Technology Ltd. 1994–1995
C:\WINDOWS\SYSTEM\wfm0200.acv
****************System Resource Report*****************
File size: 13456 bytes. Manufacturer: Creative Technology Ltd.
File version: 4.00
Copyright: Copyright Creative Technology Ltd.
C:\WINDOWS\SYSTEM\wfm0200a.csp
File size: 2238 bytes.
No version information.
C:\WINDOWS\SYSTEM\wfm0201.acv
File size: 5184 bytes.
Manufacturer: Creative Technology Ltd.
File version: 4.00
Copyright: Copyright Creative Technology Ltd.
C:\WINDOWS\SYSTEM\wfm0201a.csp
File size: 6776 bytes.
No version information.
C:\WINDOWS\SYSTEM\wfm0202.acv
File size: 9056 bytes.
No version information.
C:\WINDOWS\SYSTEM\wfm0202a.csp
```

```
File size: 9004 bytes.
No version information.
C:\WINDOWS\SYSTEM\wfm0203.acv
File size: 9056 bytes.
Manufacturer: Creative Technology Ltd.
File version: 4.00
Copyright: Copyright Creative Technology Ltd.
C:\WINDOWS\SYSTEM\wfm0203a.csp
File size: 9004 bytes.
No version information. Class: Sound, video and game controllers
Device: Gameport Joystick
Resources:
I/O: 0200h-0207h
Device drivers:
C:\WINDOWS\SYSTEM\vjoyd.vxd
File size: 20590 bytes.
Manufacturer: Microsoft Corporation
File version: 4.00.950
Copyright: Copyright Microsoft Corp. 1994–1995
C:\WINDOWS\SYSTEM\msjstick.drv
File size: 7744 bytes.
Manufacturer: Microsoft Corporation
File version: 4.0.950
Copyright: Copyright Microsoft Corp. 1991–1995
Class: Disk drives System Resource Report
Device: GENERIC IDE DISK TYPE: No resources used.
Class: Disk drives Device: GENERIC NEC FLOPPY DISK
No resources used. Class: Printer Device: HP ColorPro
No resources used.
Device drivers:
C:\WINDOWS\SYSTEM\HPPLOT.DRV
Driver file possibly missing.
No version information.
C:\WINDOWS\SYSTEM\HPPLOT.HLP
Driver file possibly missing.
No version information. Class: Printer Device: IBM Network Printer 17 PCL
No resources used.
Device drivers:
C:\WINDOWS\SYSTEM\IBMPCL.HLP
File size: 26652 bytes.
No version information.
C:\WINDOWS\SYSTEM\IBMPCLR.DLL
File size: 9504 bytes.
Manufacturer: The IBM Printing Systems Company
File version: 3.10
Copyright: Licensed Materials-Property of IBM. Copyright IBM Corp. 1
C:\WINDOWS\SYSTEM\MTETAB.DLL
File size: 3840 bytes.
Manufacturer: Softel vdm
File version: 2.00
Copyright: Copyright Softel vdm 1995
C:\WINDOWS\SYSTEM\IBMPCLFI.DLL
```

```
File size: 592 bytes.
Manufacturer: IBM CORP.
File version: 3.10.130
Copyright: Copyright IBM Corporation, 1996. Copyright Microsoft
C:\WINDOWS\SYSTEM\IBM4317F.HLP
File size: 27094 bytes.
No version information.
C:\WINDOWS\SYSTEM\IBMPCL.DRV
File size: 7104 bytes.
Manufacturer: The IBM Printing Systems Company
File version: 3.10
*************** System Resource Report************************
Copyright: Licensed Materials-Property of IBM. Copyright IBM Corp. 1
Class: Printer Device: HP ColorPro
No resources used.
Device drivers:
C:\WINDOWS\SYSTEM\HPPLOT.DRV
Driver file possibly missing.
No version information.
C:\WINDOWS\SYSTEM\HPPLOT.HLP
Driver file possibly missing.
No version information. Class: Printer Device: IBM 2390 PS/1
No resources used.
Device drivers:
C:\WINDOWS\SYSTEM\IBM239X.DRV
File size: 30560 bytes.
Manufacturer: Microsoft Corporation
File version: 4.00.950
Copyright: Copyright Microsoft Corp. 1991–1995
C:\WINDOWS\SYSTEM\UNIDRV.DLL
File size: 416 bytes.
Manufacturer: Microsoft Corporation
File version: 4.00.950
Copyright: Copyright Microsoft Corp. 1991-1995
C:\WINDOWS\SYSTEM\UNIDRV.HLP
File size: 15343 bytes.
No version information.
C:\WINDOWS\SYSTEM\ICONLIB.DLL
File size: 12176 bytes.
Manufacturer: Microsoft Corporation
File version: 4.00.950
Copyright: Copyright Microsoft Corp. 1991–1995
Class: Printer Device:No resources used. Class:
Printer Device: IBM 4019 LaserPrinter PS39
No resources used.
Device drivers:
C:\WINDOWS\SYSTEM\IB401939.SPD
File size: 10431 bytes.
No version information.
************System Resource Report*******************************
C:\WINDOWS\SYSTEM\PSCRIPT.DRVFile size: 65520 bytes.
Manufacturer: Microsoft Corporation
```

```
File version: 4.00.950
Copyright: Copyright Microsoft Corp. 1991–1995
C:\WINDOWS\SYSTEM\PSCRIPT.HLP
File size: 20439 bytes.
No version information.
C:\WINDOWS\SYSTEM\PSCRIPT.INI
File size: 328 bytes.
No version information.
C:\WINDOWS\SYSTEM\TESTPS.TXT
File size: 2640 bytes.
No version information.
C:\WINDOWS\SYSTEM\APPLE380.SPD
File size: 6046 bytes.
No version information.
C:\WINDOWS\SYSTEM\FONTS.MFM
File size: 30183 bytes.
No version information.
C:\WINDOWS\SYSTEM\ICONLIB.DLL
File size: 12176 bytes.
Manufacturer: Microsoft Corporation
File version: 4.00.950
Copyright: Copyright Microsoft Corp. 1991–1995
C:\WINDOWS\SYSTEM\PSMON.DLL
File size: 28672 bytes.
Manufacturer: Microsoft Corporation
File version: 4.00.950
Copyright: Copyright Microsoft Corp. 1981–1995 Class: Printer
Device: IBM 2390 PS/1
No resources used.
Device drivers:
C:\WINDOWS\SYSTEM\IBM239X.DRV
File size: 30560 bytes.
Manufacturer: Microsoft Corporation
File version: 4.00.950
Copyright: Copyright Microsoft Corp. 1991–1995
C:\WINDOWS\SYSTEM\UNIDRV.DLL
File size: 416 bytes.
Manufacturer: Microsoft Corporation
File version: 4.00.950
Copyright: Copyright Microsoft Corp. 1991–1995
C:\WINDOWS\SYSTEM\UNIDRV.HLP
File size: 15343 bytes.
***********System Resource Report************************
No version information.
C:\WINDOWS\SYSTEM\ICONLIB.DLL
File size: 12176 bytes.
Manufacturer: Microsoft Corporation
File version: 4.00.950
Copyright: Copyright Microsoft Corp. 1991–1995
Class: Printer Device: Apple LaserWriter Plus
No resources used.
Device drivers:
C:\WINDOWS\SYSTEM\APPLE380.SPD
```

```
File size: 6046 bytes.
No version information.
C:\WINDOWS\SYSTEM\PSCRIPT.DRV
File size: 65520 bytes.
Manufacturer: Microsoft Corporation
File version: 4.00.950
Copyright: Copyright Microsoft Corp. 1991–1995
C:\WINDOWS\SYSTEM\PSCRIPT.HLP
File size: 20439 bytes.
No version information.
C:\WINDOWS\SYSTEM\PSCRIPT.INI
File size: 328 bytes.
No version information.
C:\WINDOWS\SYSTEM\TESTPS.TXT
File size: 2640 bytes.
No version information.
C:\WINDOWS\SYSTEM\FONTS.MFM
File size: 30183 bytes.
No version information.
C:\WINDOWS\SYSTEM\ICONLIB.DLL
File size: 12176 bytes.
Manufacturer: Microsoft Corporation
File version: 4.00.950
Copyright: Copyright Microsoft Corp. 1991–1995
C:\WINDOWS\SYSTEM\PSMON.DLL
File size: 28672 bytes.
Manufacturer: Microsoft Corporation
File version: 4.00.950
Copyright: Copyright + Microsoft Corp. 1981–1995 Class: Printer
Device: IBM 4019 LaserPrinter PS39
No resources used.
Device drivers:
C:\WINDOWS\SYSTEM\IB401939.SPD
**************** System Resource Report ***************
File size: 10431 bytes.
No version information.
C:\WINDOWS\SYSTEM\PSCRIPT.DRV
File size: 65520 bytes.
Manufacturer: Microsoft Corporation
File version: 4.00.950
Copyright: Copyright Microsoft Corp. 1991-1995
C:\WINDOWS\SYSTEM\PSCRIPT.HLP
File size: 20439 bytes.
No version information.
C:\WINDOWS\SYSTEM\PSCRIPT.INI
File size: 328 bytes.
No version information.
C:\WINDOWS\SYSTEM\TESTPS.TXT
File size: 2640 bytes.
No version information.
C:\WINDOWS\SYSTEM\APPLE380.SPD
File size: 6046 bytes.
```

```
No version information.
C:\WINDOWS\SYSTEM\FONTS.MFM
File size: 30183 bytes.
No version information.
C:\WINDOWS\SYSTEM\ICONLIB.DLL
File size: 12176 bytes.
Manufacturer: Microsoft Corporation
File version: 4.00.950
Copyright: Copyright Microsoft Corp. 1991–1995
C:\WINDOWS\SYSTEM\PSMON.DLL
File size: 28672 bytes.
Manufacturer: Microsoft Corporation
File version: 4.00.950
Copyright: Copyright Microsoft Corp. 1981–1995
```

II. Angel's Report

```
Microsoft Diagnostics Report For \\ANGEL
────────────
OS Version Report
────────────
Microsoft (R) Windows NT (TM) Workstation
Version 4.0 (Build 1381: Service Pack 1) x86 Uniprocessor Free
Registered Owner: ANGEL, Information World, Inc.
Product Number: 17597-OEM-0023333-22375
────────────
System Report
────────────
System: AT/AT COMPATIBLE
Hardware Abstraction Layer: PC Compatible Eisa/Isa HAL
BIOS Date: 07/24/97
BIOS Version: BIOS Version 0.11.01.DU0M- Beta
Processor list:
  0: x86 Family 6 Model 3 Stepping 3 GenuineIntel ~265 Mhz
────────────
Video Display Report
────────────
BIOS Date: 07/21/97
BIOS Version: S3 86C775/86C785 Video BIOS. Version 1.01.11- C2.08.05
          S3 86C775/86C785 Video BIOS. Version 1.01.11-C2.08.05
          S3 86C775/86C785 Video BIOS. Version 1.01.11-C2.08.05
          S3 86C775/86C785 Video BIOS. Version 1.01.11-C2.08.05
Adapter:
  Setting: 640 × 480 × 256  60 Hz
  Type: s3mini compatible display adapter
  String: S3 Compatible Display Adapter
  Memory: 2 MB
  Chip Type: S3 Trio64V2
  DAC Type: S3 SDAC
Driver:
  Vendor: S3 Incorporated
  File(s): s3mini.sys, s3disp.dll
```

```
Version: 1.03.10, 4.0.0
Drives Report
────────────
C:\ (Local - FAT) ANGEL Total: 1,023,824KB, Free: 783,600KB
  Serial Number: 1939 - BFE
  Bytes per cluster: 512
  Sectors per cluster: 32
  Filename length: 255
D:\ (Local - FAT) ANGEL-D Total: 2,096,320KB, Free: 2,090,208KB
  Serial Number: 2651 - C03
  Bytes per cluster: 512
  Sectors per cluster: 64
  Filename length: 255
E:\ (Local - FAT) Total: 999,632KB, Free: 999,616KB
  Serial Number: 3055 - C06
  Bytes per cluster: 512
  Sectors per cluster: 32
  Filename length: 255
────────────
Memory Report
────────────
Handles: 1,281
Threads: 124
Processes: 19

Physical Memory (K)
  Total: 32,180
  Available: 6,072
  File Cache: 6,784

Kernel Memory (K)
  Total: 5,104
  Paged: 4,184
  Nonpaged: 920

Commit Charge (K)
  Total: 21,316
  Limit: 67,788
  Peak: 22,084

Pagefile Space (K)
  Total: 44,032
  Total in use: 640
  Peak: 640


C:\pagefile.sys
  Total: 44,032
  Total in use: 640
  Peak: 640
────────────
Services Report
────────────
```

```
Alerter Stopped (Manual)
  C:\WINNT40\System32\services.exe
  Service Account Name: LocalSystem
  Error Severity: Normal
  Service Flags: Shared Process
  Service Dependencies:
          LanmanWorkstation
Computer Browser          Running (Automatic)
  C:\WINNT40\System32\services.exe
  Service Account Name: LocalSystem
  Error Severity: Normal
  Service Flags: Shared Process
  Service Dependencies:
          LanmanWorkstation
          LanmanServer
          LmHosts
ClipBook Server Stopped (Manual)
  C:\WINNT40\system32\clipsrv.exe
  Service Account Name: LocalSystem
  Error Severity: Normal
  Service Flags: Own Process
  Service Dependencies:
          NetDDE
DHCP Client (TDI)          Stopped (Disabled)
  C:\WINNT40\System32\services.exe
  Service Account Name: LocalSystem
  Error Severity: Normal
  Service Flags: Shared Process
  Service Dependencies:
          Tcpip
          Afd
          NetBT
IBM DMI Service Layer (DMI Service Layer)          Running (Automatic)
  C:\sva\dmi\bin\dmislsrv.exe
  Service Account Name: LocalSystem
  Error Severity: Normal
  Service Flags: Own Process, Interactive
EventLog (Event log)          Running (Automatic)
  C:\WINNT40\system32\services.exe
  Service Account Name: LocalSystem
  Error Severity: Normal
  Service Flags: Shared Process
Server Running (Automatic)
  C:\WINNT40\System32\services.exe
  Service Account Name: LocalSystem
  Error Severity: Normal
  Service Flags: Shared Process
  Group Dependencies:
          TDI
Workstation (NetworkProvider)          Running (Automatic)
  C:\WINNT40\System32\services.exe
  Service Account Name: LocalSystem
  Error Severity: Normal
```

```
Service Flags: Shared Process
  Group Dependencies:
          TDI
PC System Monitor          Running (Automatic)
  C:\sva\dmi\bin\lm78cint.exe
  Service Account Name: LocalSystem
  Error Severity: Normal
  Service Flags: Own Process, Interactive
  Group Dependencies:
          DMI Service Layer
TCP/IP NetBIOS Helper          Running (Automatic)
  C:\WINNT40\System32\services.exe
  Service Account Name: LocalSystem
  Error Severity: Normal
  Service Flags: Shared Process
  Group Dependencies:
          NetworkProvider
Messenger          Running (Automatic)
  C:\WINNT40\System32\services.exe
  Service Account Name: LocalSystem
  Error Severity: Normal
  Service Flags: Shared Process
  Service Dependencies:
          LanmanWorkstation
          NetBios
Network DDE (NetDDEGroup)          Stopped (Manual)
  C:\WINNT40\system32\netdde.exe
  Service Account Name: LocalSystem
  Error Severity: Normal
  Service Flags: Shared Process
  Service Dependencies:
          NetDDEDSDM
Network DDE DSDM          Stopped (Manual)
  C:\WINNT40\system32\netdde.exe
  Service Account Name: LocalSystem
  Error Severity: Normal
  Service Flags: Shared Process
Net Logon (RemoteValidation)          Stopped (Manual)
  C:\WINNT40\System32\lsass.exe
  Service Account Name: LocalSystem
  Error Severity: Normal
  Service Flags: Shared Process
  Service Dependencies:
          LanmanWorkstation
          LmHosts
Norton SpeedDisk          Running (Automatic)
  C:\Program Files\Norton Speed Disk Trial\SDSRV.EXE
  Service Account Name: LocalSystem
  Error Severity: Normal
  Service Flags: Own Process
NT LM Security Support Provider          Stopped (Manual)
  C:\WINNT40\System32\SERVICES.EXE
```

```
Service Account Name: LocalSystem
  Error Severity: Normal
  Service Flags: Shared Process
Plug and Play (PlugPlay)          Running (Automatic)
  C:\WINNT40\system32\services.exe
  Service Account Name: LocalSystem
  Error Severity: Normal
  Service Flags: Shared Process
Remote Access Autodial ManagerRunning (Automatic)
  C:\WINNT40\system32\rasman.exe
  Service Account Name: LocalSystem
  Error Severity: Normal
  Service Flags: Shared Process
  Service Dependencies:
          RasMan
Remote Access Connection Manager (Network)          Running (Manual)
  C:\WINNT40\system32\rasman.exe
  Service Account Name: LocalSystem
  Error Severity: Normal
  Service Flags: Shared Process, Interactive
  Service Dependencies:
          tapisrv
Remote Access Server (Network) Stopped (Manual)
  C:\WINNT40\system32\rassrv.exe
  Service Account Name: LocalSystem
  Error Severity: Normal
  Service Flags: Own Process
  Service Dependencies:
          LanmanServer
          RasMan
          NetBios
          NetBT
Directory Replicator          Stopped (Manual)
  C:\WINNT40\System32\lmrepl.exe
  Service Account Name: LocalSystem
  Error Severity: Normal
  Service Flags: Own Process
  Service Dependencies:
          LanmanWorkstation
          LanmanServer
Remote Procedure Call (RPC) Locator          Stopped (Manual)
  C:\WINNT40\System32\LOCATOR.EXE
  Service Account Name: LocalSystem
  Error Severity: Normal
  Service Flags: Own Process
Remote Procedure Call (RPC) Service          Running (Automatic)
  C:\WINNT40\system32\RpcSs.exe
  Service Account Name: LocalSystem
  Error Severity: Normal
  Service Flags: Own Process
Schedule          Stopped (Manual)
  C:\WINNT40\System32\AtSvc.Exe
  Service Account Name: LocalSystem
```

```
Error Severity: Normal
  Service Flags: Own Process
Spooler (SpoolerGroup)          Running (Automatic)
  C:\WINNT40\system32\spoolss.exe
  Service Account Name: LocalSystem
  Error Severity: Normal
  Service Flags: Own Process, Interactive
Telephony Service          Running (Manual)
  C:\WINNT40\system32\tapisrv.exe
  Service Account Name: LocalSystem
  Error Severity: Normal
  Service Flags: Own Process
UPS          Stopped (Manual)
  C:\WINNT40\System32\ups.exe
  Service Account Name: LocalSystem
  Error Severity: Normal
  Service Flags: Own Process
____________
Drivers Report
____________
Abiosdsk (Primary disk)          Stopped (Disabled)
  Error Severity: Ignore
  Service Flags: Kernel Driver,
  Shared Process
AFD Networking Support Environment (TDI) Running          (Automatic)
  C:\WINNT40\System32\drivers\afd.sys
  Error Severity: Normal
  Service Flags: Kernel Driver,
  Shared Process
Aha154x (SCSI miniport)          Stopped (Disabled)
  Error Severity: Normal
  Service Flags: Kernel Driver,
          Shared Process
Aha174x (SCSI miniport)          Stopped (Disabled)
  Error Severity: Normal
  Service Flags: Kernel Driver,
          Shared Process
aic78xx (SCSI miniport)          Stopped (Disabled)
  C:\WINNT40\system32\drivers\aic78xx.sys
  Error Severity: Normal
  Service Flags: Kernel Driver,
  Shared Process
Always (SCSI miniport)          Stopped (Disabled)
  Error Severity: Normal
  Service Flags: Kernel Driver,
  Shared Process
ami0nt (SCSI miniport)          Stopped (Disabled)
  Error Severity: Normal
  Service Flags: Kernel Driver,
  Shared Process
amsint (SCSI miniport)          Stopped (Disabled)
  Error Severity: Normal
```

```
Service Flags: Kernel Driver,
  Shared Process
Arrow (SCSI miniport)          Stopped (Disabled)
  Error Severity: Normal
  Service Flags: Kernel Driver,
  Shared Process
Remote Access Mac (NDIS)          Running (Automatic)
  C:\WINNT40\system32\drivers\asyncmac.sys
  Error Severity: Normal
  Service Flags: Kernel Driver,
  Shared Process
atapi (SCSI miniport)          Stopped (Disabled)
  Error Severity: Normal
  Service Flags: Kernel Driver,
          Shared Process
Atdisk (Primary disk)          Stopped (Disabled)
  Error Severity: Ignore
  Service Flags: Kernel Driver,
  Shared Process
ati (Video)          Stopped (Disabled)
  Error Severity: Ignore
  Service Flags: Kernel Driver,
  Shared Process
Beep (Base)          Running (System)
  Error Severity: Normal
  Service Flags: Kernel Driver,
  Shared Process
BusLogic (SCSI miniport)          Stopped (Disabled)
  Error Severity: Normal
  Service Flags: Kernel Driver,
  Shared Process
Busmouse (Pointer Port)          Stopped (Disabled)
  Error Severity: Ignore
  Service Flags: Kernel Driver,
          Shared Process
Cdaudio (Filter)          Stopped (System)
  Error Severity: Ignore
  Service Flags: Kernel Driver,
  Shared Process
Cdfs (File system)          Running (Disabled)
  Error Severity: Normal
  Service Flags: File System Driver, Shared Process
  Group Dependencies:
          SCSI CDROM Class
Cdrom (SCSI CDROM Class)          Running (System)
  Error Severity: Ignore
  Service Flags: Kernel Driver,
  Shared Process
  Group Dependencies:
          SCSI miniport
Changer (Filter)          Stopped (System)
  Error Severity: Ignore
  Service Flags: Kernel Driver,
Shared Process
cirrus (Video)          Stopped (Disabled)
  Error Severity: Ignore
  Service Flags: Kernel Driver,
  Shared Process
Cpqarray (SCSI miniport)          Stopped (Disabled)
  Error Severity: Normal
  Service Flags: Kernel Driver,
  Shared Process
cpqfws2e (SCSI miniport)          Stopped (Disabled)
  Error Severity: Normal
  Service Flags: Kernel Driver,
  Shared Process
cs32ba11 (Base)          Running (System)
  C:\WINNT40\System32\drivers\cs32ba11.SYS
  Error Severity: Normal
  Service Flags: Kernel Driver,
  Shared Process
dac960nt (SCSI miniport)          Stopped (Disabled)
  Error Severity: Normal
  Service Flags: Kernel Driver,
  Shared Process
dce376nt (SCSI miniport)          Stopped (Disabled)
  Error Severity: Normal
  Service Flags: Kernel Driver,
  Shared Process
Delldsa (SCSI miniport)          Stopped (Disabled)
  Error Severity: Normal
  Service Flags: Kernel Driver,
  Shared Process
Dell_DGX (Video)          Stopped (Disabled)
  Error Severity: Ignore
  Service Flags: Kernel Driver,
  Shared Process
Disk (SCSI Class)          Running (Boot)
  Error Severity: Ignore
  Service Flags: Kernel Driver,
  Shared Process
  Group Dependencies:
          SCSI miniport
Diskperf (Filter)          Stopped (Disabled)
  Error Severity: Normal
  Service Flags: Kernel Driver,
  Shared Process
DptScsi (SCSI miniport)          Stopped (Disabled)
  Error Severity: Normal
  Service Flags: Kernel Driver,
  Shared Process
dtc329x (SCSI miniport)          Stopped (Disabled)
  Error Severity: Normal
  Service Flags: Kernel Driver,
  Shared Process
```

```
Intel 82557-based PRO Adapter Driver (NDIS) Running (Automatic)
  C:\WINNT40\System32\drivers\e100b.sys
  Error Severity: Normal
  Service Flags: Kernel Driver,
Shared Process
et4000 (Video)          Stopped (Disabled)
  Error Severity: Ignore
  Service Flags: Kernel Driver,
Shared Process
Fastfat (Boot file system)          Running (Disabled)
  Error Severity: Normal
  Service Flags: File System Driver,
  Shared Process
Fd16_700 (SCSI miniport)          Stopped (Disabled)
  Error Severity: Normal
  Service Flags: Kernel Driver,
  Shared Process
Fd7000ex (SCSI miniport)          Stopped (Disabled)
  Error Severity: Normal
  Service Flags: Kernel Driver,
  Shared Process
Fd8xx (SCSI miniport)          Stopped (Disabled)
  Error Severity: Normal
  Service Flags: Kernel Driver,
  Shared Process
flashpnt (SCSI miniport)          Stopped (Disabled)
  Error Severity: Normal
  Service Flags: Kernel Driver,
  Shared Process
Floppy (Primary disk)          Running (System)
  Error Severity: Ignore
  Service Flags: Kernel Driver,
  Shared Process
Ftdisk (Filter)          Stopped (Disabled)
  Error Severity: Ignore
  Service Flags: Kernel Driver,
  Shared Process
i8042 Keyboard and PS/2 Mouse Port
  Driver (Keyboard Port) Running          (System)
  System32\DRIVERS\i8042prt.sys
  Error Severity: Normal
  Service Flags: Kernel Driver,
  Shared Process
Inport (Pointer Port)          Stopped (Disabled)
  Error Severity: Ignore
  Service Flags: Kernel Driver,
  Shared Process
Jazzg300 (Video)          Stopped (Disabled)
  Error Severity: Ignore
  Service Flags: Kernel Driver,
  Shared Process
Jazzg364 (Video)          Stopped (Disabled)
  Error Severity: Ignore
```

```
Service Flags: Kernel Driver,
  Shared Process
Jzvxl484 (Video)          Stopped (Disabled)
  Error Severity: Ignore
  Service Flags: Kernel Driver,
  Shared Process
Keyboard Class Driver (Keyboard Class)          Running (System)
  System32\DRIVERS\kbdclass.sys
  Error Severity: Normal
  Service Flags: Kernel Driver,
  Shared Process
KSecDD (Base)          Running (System)
  Error Severity: Normal
  Service Flags: Kernel Driver,
  Shared Process
LM78          Running (Automatic)
  \??\C:\WINNT40\system32\drivers\lm78nt.sys
  Error Severity: Normal
  Service Flags: Kernel Driver,
          Shared Process
mga (Video)          Stopped (Disabled)
  Error Severity: Ignore
  Service Flags: Kernel Driver,
          Shared Process
mga_mil (Video)          Stopped (Disabled)
  Error Severity: Ignore
  Service Flags: Kernel Driver,
  Shared Process
mitsumi (SCSI miniport)          Stopped (Disabled)
  Error Severity: Normal
  Service Flags: Kernel Driver,
          Shared Process
mkecr5xx (SCSI miniport)          Stopped (Disabled)
  Error Severity: Normal
  Service Flags: Kernel Driver,
          Shared Process
Modem (Extended base)          Stopped (Manual)
  Error Severity: Ignore
  Service Flags: Kernel Driver,
  Shared Process
Mouse Class Driver (Pointer Class)          Running (System)
  System32\DRIVERS\mouclass.sys
  Error Severity: Normal
  Service Flags: Kernel Driver,
  Shared Process
Msfs (File system)          Running (System)
  Error Severity: Normal
  Service Flags: File System Driver,
  Shared Process
Mup (Network)          Running (Manual)
  C:\WINNT40\System32\drivers\mup.sys
  Error Severity: Normal
```

```
Service Flags: File System Driver,
  Shared Process
Ncr53c9x (SCSI miniport)          Stopped (Disabled)
  Error Severity: Normal
  Service Flags: Kernel Driver,
  Shared Process
ncr77c22 (Video)          Stopped (Disabled)
  Error Severity: Ignore
  Service Flags: Kernel Driver,
  Shared Process
Ncrc700 (SCSI miniport)          Stopped (Disabled)
  Error Severity: Normal
  Service Flags: Kernel Driver,
  Shared Process
Ncrc710 (SCSI miniport)          Stopped (Disabled)
  Error Severity: Normal
  Service Flags: Kernel Driver,
  Shared Process
Microsoft NDIS System Driver (NDIS)
  Service Flags: Kernel Driver,
  Shared Process
Microsoft NDIS TAPI driver (NDIS)          Running (System)
  C:\WINNT40\system32\drivers\ndistapi.sys
  Error Severity: Normal
  Service Flags: Kernel Driver,
  Shared Process
Remote Access WAN Wrapper (NDISWAN)          Running (Automatic)
  C:\WINNT40\system32\drivers\ndiswan.sys
  Error Severity: Normal
  Service Flags: Kernel Driver,
  Shared Process
NetBIOS Interface (NetBIOSGroup)          Running (Manual)
  C:\WINNT40\System32\drivers\netbios.sys
  Error Severity: Normal
  Service Flags: File System Driver,
  Shared Process
  Group Dependencies:
          TDI
WINS Client(TCP/IP) (PNP_TDI)          Running (Automatic)
  C:\WINNT40\System32\drivers\netbt.sys
  Error Severity: Normal
  Service Flags: Kernel Driver,
Shared Process
  Service Dependencies:
          Tcpip
NetDetect          Stopped (Manual)
  C:\WINNT40\system32\drivers\netdtect.sys
  Error Severity: Normal
  Service Flags: Kernel Driver,
  Shared Process
Npfs (File system)          Running (System)
  Error Severity: Normal
  Service Flags: File System Driver,
Shared Process
Ntfs (File system)          Stopped (Disabled)
  Error Severity: Normal
  Service Flags: File System Driver,
  Shared Process
Null (Base)          Running (System)
  Error Severity: Normal
  Service Flags: Kernel Driver,
  Shared Process
Oliscsi (SCSI miniport)          Stopped (Disabled)
  Error Severity: Normal
  Service Flags: Kernel Driver,
  Shared Process
Parallel (Extended base)          Running (Automatic)
  Error Severity: Ignore
  Service Flags: Kernel Driver,
  Shared Process
  Service Dependencies:
          Parport
  Group Dependencies:
          Parallel arbitrator
Parport (Parallel arbitrator)          Running (Automatic)
  Error Severity: Ignore
  Service Flags: Kernel Driver,
  Shared Process
ParVdm (Extended base)          Running (Automatic)
  Error Severity: Ignore
  Service Flags: Kernel Driver,
  Shared Process
  Service Dependencies:
          Parport
  Group Dependencies:
          Parallel arbitrator
PCIDump (PCI Configuration)          Stopped (System)
  Error Severity: Ignore
  Service Flags: Kernel Driver,
  Shared Process
Pcmcia (System Bus Extender)          Stopped (Disabled)
  Error Severity: Normal
  Service Flags: Kernel Driver,
  Shared Process
PIIXIDE (SCSI miniport)          Running (Boot)
  C:\WINNT40\system32\drivers\PIIXIDE.SYS
  Error Severity: Normal
  Service Flags: Kernel Driver,
  Shared Process
PnP ISA Enabler Driver (Base)          Stopped (System)
  Error Severity: Ignore
  Service Flags: Kernel Driver,
  Shared Process
psidisp (Video)          Stopped (Disabled)
  Error Severity: Ignore
```

```
Service Flags: Kernel Driver,
  Shared Process
Ql10wnt (SCSI miniport)          Stopped (Disabled)
  Error Severity: Normal
  Service Flags: Kernel Driver,
  Shared Process
qv (Video)          Stopped (Disabled)
  Error Severity: Ignore
  Service Flags: Kernel Driver,
  Shared Process
Remote Access Auto Connection Driver
  Streams Drivers) Running          (Automatic)
  C:\WINNT40\system32\drivers\rasacd.sys
  Error Severity: Normal
  Service Flags: Kernel Driver,
  Shared Process
Remote Access ARP Service (PNP_TDI)          Running (Automatic)
  C:\WINNT40\system32\drivers\rasarp.sys
  Error Severity: Normal
  Service Flags: Kernel Driver,
          Shared Process
  Service Dependencies:
          TCPIP
Rdr (Network)          Running (Manual)
  C:\WINNT40\System32\drivers\rdr.sys
  Error Severity: Normal
  Service Flags: File System Driver,
  Shared Process
s3 (Video)          Stopped (Disabled)
  Error Severity: Ignore
  Service Flags: Kernel Driver,
  Shared Process
S3Inc (Video)          Running (System)
  System32\DRIVERS\s3mini.sys
  Error Severity: Normal
  Service Flags: Kernel Driver,
  Shared Process
Scsiprnt (Extended base)          Stopped (Automatic)
  Error Severity: Ignore
  Service Flags: Kernel Driver,
  Shared Process
  Group Dependencies:
          SCSI miniport
Scsiscan (SCSI Class)          Stopped (System)
  Error Severity: Ignore
  Service Flags: Kernel Driver,
  Shared Process
  Group Dependencies:
          SCSI miniport
Serial (Extended base)          Running (Automatic)
  Error Severity: Ignore
  Service Flags: Kernel Driver,
  Shared Process
```

```
Sermouse (Pointer Port)          Stopped (Disabled)
  Error Severity: Ignore
  Service Flags: Kernel Driver,
  Shared Process
Sfloppy (Primary disk)          Stopped (System)
  Error Severity: Ignore
  Service Flags: Kernel Driver,
  Shared Process
  Group Dependencies:
          SCSI miniport
Simbad (Filter)          Stopped (Disabled)
  Error Severity: Normal
  Service Flags: Kernel Driver,
  Shared Process
slcd32 (SCSI miniport)          Stopped (Disabled)
  Error Severity: Normal
  Service Flags: Kernel Driver,
  Shared Process
Sparrow (SCSI miniport)          Stopped (Disabled)
  Error Severity: Normal
  Service Flags: Kernel Driver,
  Shared Process
Spock (SCSI miniport)          Stopped (Disabled)
  Error Severity: Normal
  Service Flags: Kernel Driver,
  Shared Process
Srv (Network)          Running (Manual)
  C:\WINNT40\System32\drivers\srv.sys
  Error Severity: Normal
  Service Flags: File System Driver,
  Shared Process
symc810 (SCSI miniport)          Stopped (Disabled)
  Error Severity: Normal
  Service Flags: Kernel Driver,
  Shared Process
T128 (SCSI miniport)          Stopped (Disabled)
  Error Severity: Normal
  Service Flags: Kernel Driver,
  Shared Process
T13B (SCSI miniport)          Stopped (Disabled)
  Error Severity: Normal
  Service Flags: Kernel Driver,
  Shared Process
TCP/IP Service (PNP_TDI)          Running (Automatic)
  C:\WINNT40\System32\drivers\tcpip.sys
  Error Severity: Normal
  Service Flags: Kernel Driver,
  Shared Process
tga (Video)          Stopped (Disabled)
  Error Severity: Ignore
  Service Flags: Kernel Driver,
  Shared Process
```

```
tmv1 (SCSI miniport)          Stopped (Disabled)
  Error Severity: Normal
  Service Flags: Kernel Driver,
  Shared Process
Ultra124 (SCSI miniport)          Stopped (Disabled)
  Error Severity: Normal
  Service Flags: Kernel Driver,
  Shared Process
Ultra14f (SCSI miniport)          Stopped (Disabled)
  Error Severity: Normal
  Service Flags: Kernel Driver,
  Shared Process
Ultra24f (SCSI miniport)          Stopped (Disabled)
  Error Severity: Normal
  Service Flags: Kernel Driver,
  Shared Process
v7vram (Video)          Stopped (Disabled)
  Error Severity: Ignore
  Service Flags: Kernel Driver,
          Shared Process
VgaSave (Video Save)          Running (System)
  C:\WINNT40\System32\drivers\vga.sys
  Error Severity: Ignore
  Service Flags: Kernel Driver,
  Shared Process
VgaStart (Video Init)          Stopped (System)
  C:\WINNT40\System32\drivers\vga.sys
  Error Severity: Ignore
  Service Flags: Kernel Driver,
  Shared Process
Wd33c93 (SCSI miniport)          Stopped (Disabled)
  Error Severity: Normal
  Service Flags: Kernel Driver,
  Shared Process
wd90c24a (Video)          Stopped (Disabled)
  Error Severity: Ignore
  Service Flags: Kernel Driver,
  Shared Process
wdvga (Video)          Stopped (Disabled)
  Error Severity: Ignore
  Service Flags: Kernel Driver,
  Shared Process
weitekp9 (Video)          Stopped (Disabled)
  Error Severity: Ignore
  Service Flags: Kernel Driver, Shared Process
Xga (Video)          Stopped (Disabled)
  Error Severity: Ignore
  Service Flags: Kernel Driver, Shared Process
────────────
IRQ and Port Report
────────────
Devices          Vector Level Affinity
────────────
```

```
i8042prt          1          1 0xffffffff
i8042prt          12         12 0xffffffff
Serial            4          4 0x00000000
cs32ba11          55         7 0x00000001
```

```
E100B             10         10 0x00000000
Floppy            6          6 0x00000000
PIIXIDE           0          14 0x00000000
PIIXIDE           0          15 0x00000000
────────────
Devices          Physical Address Length
────────────
i8042prt          0x00000060 0x0000000001
i8042prt          0x00000064 0x0000000001
Parport           0x000003bc 0x0000000003
Serial            0x000003f8 0x0000000007
cs32ba11          0x00000530 0x0000000008
cs32ba11          0x00000388 0x0000000004
E100B             0x0000ff40 0x0000000014
Floppy            0x000003f0 0x0000000006
Floppy            0x000003f7 0x0000000001
LM78              0x00000295 0x0000000002
PIIXIDE           0x000001f0 0x0000000008
PIIXIDE           0x0000ffa0 0x0000000008
PIIXIDE           0x00000170 0x0000000008
PIIXIDE           0x0000ffa8 0x0000000008
S3Inc             0x000003c0 0x0000000010
S3Inc             0x000003d4 0x0000000008
VgaSave           0x000003b0 0x000000000c
VgaSave           0x000003c0 0x0000000020
VgaSave           0x000001ce 0x0000000002
────────────
DMA and Memory Report
────────────
Devices          Channel Port
-----------------------------
cs32ba11          1          0
cs32ba11          0          0
Floppy            2          0
────────────
Devices          Physical Address Length
────────────
E100B             0xffbef000 0x00000014
E100B             0xffbef000 0x00000014
S3Inc             0x000a0000 0x00010000
S3Inc             0x000c0000 0x00008000
VgaSave           0x000a0000 0x00020000
-----------------------------
Environment Report
-----------------------------
System Environment Variables
  ComSpec = C:\WINNT40\system32\cmd.exe
  Os2LibPath = C:\WINNT40\system32\os2\dll;

Path 5 C:\WINNT40\SYSTEM32;C:\WINNT40;C:\SVA\DMI\BIN
  windir = C:\WINNT40
  OS = Windows_NT
  PROCESSOR_ARCHITECTURE = x86
  PROCESSOR_LEVEL = 6
  PROCESSOR_IDENTIFIER = x86 Family 6 Model 3 Stepping 3,GenuineIntel
  PROCESSOR_REVISION = 0303
  NUMBER_OF_PROCESSORS = 1
  help = c:\ipfwin\help
  ipf_path = c:\ipfwin
Environment Variables for Current User
  TEMP = C:\TEMP
  TMP = C:\TEMP
────────────
Network Report
────────────
Your Access Level: Admin & Local (total control!)
Workgroup or Domain: INFO
Network Version: 4.0
LanRoot: INFO
Logged On Users: 1
Current User (1): Administrator
  Logon Domain: ANGEL
  Logon Server: ANGEL
  Transport: NetBT_E100B1,00-60-94-45-43-F3, VC's: 0, Wan: Wan
Character Wait: 3,600
Collection Time: 250
Maximum Collection Count: 16
Keep Connection: 600
Maximum Commands: 5
Session Time Out: 45
Character Buffer Size: 512
Maximum Threads: 17
Lock Quota: 6,144
Lock Increment: 10
Maximum Locks: 500
Pipe Increment: 10
Maximum Pipes: 500
Cache Time Out: 40
Dormant File Limit: 45
Read Ahead Throughput: 4,294,967,295
Mailslot Buffers: 3
Server Announce Buffers: 20
Illegal Datagrams: 5
Datagram Reset Frequency: 60
Log Election Packets: False
Use Opportunistic Locking: True
Use Unlock Behind: True
Use Close Behind: True
Buffer Pipes: True
Use Lock, Read, Unlock: True
Use NT Caching: True
Use Raw Read: True
Use Raw Write: True
Use Write Raw Data: True
Use Encryption: True
Buffer Deny Write Files: True
Buffer Read Only Files: True
Force Core Creation: True
512 Byte Max Transfer: False
Bytes Received: 1,028
SMB's Received: 9
Paged Read Bytes Requested: 0
Non Paged Read Bytes Requested: 0
Cache Read Bytes Requested: 0
Network Read Bytes Requested: 0
Bytes Transmitted: 1,133
SMB's Transmitted: 9
Paged Read Bytes Requested: 0
Non Paged Read Bytes Requested: 0
Cache Read Bytes Requested: 0
Network Read Bytes Requested: 0
Initially Failed Operations: 0
Failed Completion Operations: 0
Read Operations: 0
Random Read Operations: 0
Read SMB's: 0
Large Read SMB's: 0
Small Read SMB's: 0
Write Operations: 0
Random Write Operations: 0
Write SMB's: 0
Large Write SMB's: 0
Small Write SMB's: 0
Raw Reads Denied: 0
Raw Writes Denied: 0
Network Errors: 0
Sessions: 2
Failed Sessions: 0
Reconnects: 0
Core Connects: 0
LM 2.0 Connects: 0
LM 2.x Connects: 0
Windows NT Connects: 2
Server Disconnects: 0
Hung Sessions: 0
Use Count: 2
Failed Use Count: 0
Current Commands: 0
Server File Opens: 0
Server Device Opens: 0
Server Jobs Queued: 0
Server Session Opens: 0
Server Sessions Timed Out: 0
Server Sessions Errored Out: 0
Server Password Errors: 0
Server Permission Errors: 0
Server System Errors: 0
Server Bytes Sent: 249
Server Bytes Received: 477
Server Average Response Time: 0
Server Request Buffers Needed: 0
Server Big Buffers Needed: 0
```

18.15 Network Printer (IBM)

An IBM printer was selected for the printer. Model 17 was determined to be the best fit. The following features and functions factored into that reasoning during the decision process. Before examining all features and functions of this printer, consider the first list of *standard* features and functions: 17 pages per minute, 600×600 resolution, up to five addressable input trays, 4 MB RAM (optional to 66 MB), PCL5e standard language (PostScript, IPDS, SCS optional), automatic language switching with options, automatic I/O switching, and standard parallel with two network interface slots.

The following options were added to the printer to make it capable of meeting the needs of all users on the network: 75-envelop feeder, Ethernet interface, Token Ring interface, 24 MB RAM, PostScript language option level 2, 500-sheet second paper tray duplex unit, and 10-bin secured mailbox unit.

This printer arrived on a pallet weighing in at approximately 250 lb (entire pallet weight). The printer itself is 40.9 lb (18.6 kg). With all options installed the dimensions are 31 high, 25 in front to back, and 17 in wide, with a weight of about 65 lb. (These are my measurements, which include space for rear cabling, etc., and are approximations.)

The printer was chosen because of its flexibility and power. Note that it supports IPDS and SCS character strings. This is valuable because should the network need a system which uses either of these character strings for printing, the printer itself is already capable of handling it.

Intelligent Printer Data Stream (IPDS) is used between an IBM host and a printer; generally this refers to an SNA environment. This data stream is used with an all-points addressable printer. IPDS can intermix text and graphics—both vector- and raster-based. An SCS character string is a protocol used with pritners and certain terminals in the SNA environment. LU1 and LU6.2 can use this data stream. One unique aspect of this data stream is its lack of data-flow control functions. The significance of the model 17 printer chosen for this network should not be overlooked. This means that when the need arises for a host running MVS and VTAM, the *current* printer can be used. Here again is another example of architecting success into the network.

Because the printer is on the network all network users can take advantage of it; this includes off-site users who want to work on the network from a remote location, and print something to someone and have that document secure (see Fig. 18-35).

Figure 18-35 shows a remote user connecting via a switched line to the network. The “network” in this example is viewed as the equipment in the rack enclosure. However, the network includes all devices participating in it. This example shows *HOLLYWOOD* sending a file to the network printer. All the remote users have the same capability to interact with the network printer.

Users onsite where the printer is installed have free access to it, with the exception of those who require secured access through the mailbox feature.

Knowing actual *electrical printer* requirements is important. Consider the information here. These are actual readings I took specifically for the purpose of use here as an example. I used the Tektronix THS720P oscilloscope for these readings. Consider Fig. 18-36.

Figure 18-36 shows the voltage and frequency readings of the line supplying voltage to the printer. These readings were taken just minutes prior to turning on the printer to make these readings.

Figure 18-37 shows not only the voltage and frequency reading but also the amperage reading as shown by channel 2 (Ch 2). Note that the reading is 5.249 A. This reading was taken on immediate power-on of the printer.

Figure 18-38 shows amperage and voltage readings with the drum cycled to heat itself back to printing temperature. Note that the amperage draw is 6.961 A.

Figure 18-39 shows still another reading. I took this reading while monitoring the oscilloscope for a period of about 10 min. Actually, I was somewhat surprised. This is what I call a “random” reading, meaning that there was no meaningful correlation to anything I did and this reading.

In fact, I was somewhat surprised at the reading myself, but it is real and is actual documentation of a real event with the electrical consid erations for this printer. I should add that no other devices were drawing current on the line in which I took these readings. The oscilloscope itself was running on battery power, and the printer was the only device powered by the line providing power to the printer. Now consider Fig. 18-40.

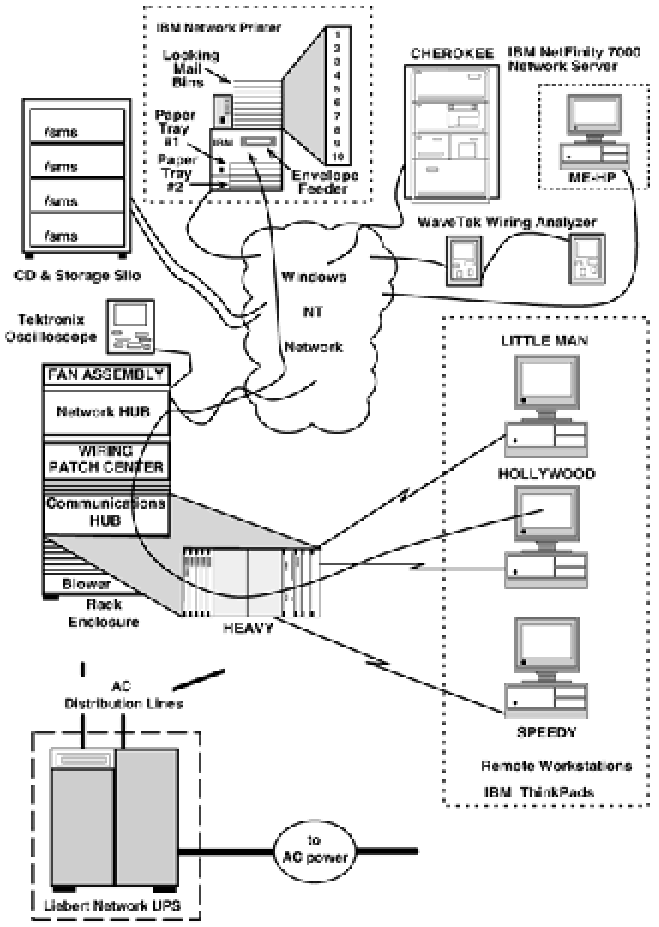


Figure 18-35
Remote users using the network printer.

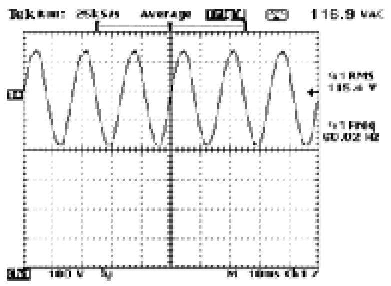


Figure 18-36
Line voltage readings.

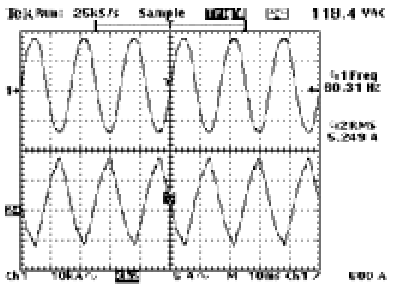


Figure 18-37
Amperage and voltage readings on power-on.

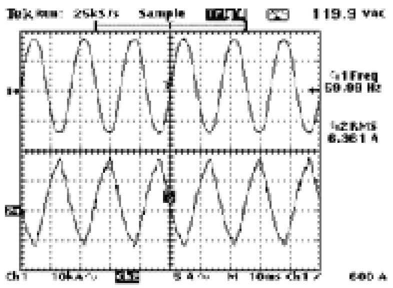


Figure 18-38
Amperage and voltage readings with drum heating.

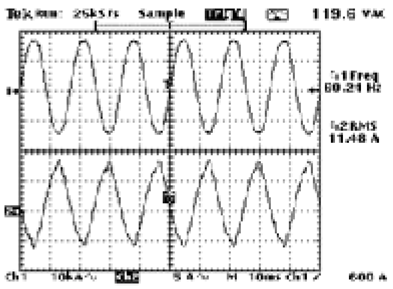


Figure 18-39
Random amperage and voltage readings.

I decided to take a temperature reading while measuring the environment around the printer. I repositioned the temperature probe near the printer. The temperature of the area near the printer was 77.9°F.

The significance of this electrical information should not be understated. You need to know the electrical requirements of your network equipment. You may think that computers and network devices don't use much electricity, and for the most part this is true. However, the electrical requirements of each device should be known. I presented this information here so that you could use it as an example of how to plan for your network printer.

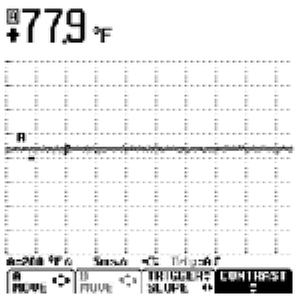


Figure 18-40
Temperature readings near the printer.

In the author's case, the IBM model 17 printer arrived on a pallet as described previously. From time of delivery until the printer was operational, 1 workday elapsed: I estimate about 2 h for unpacking the printer and reading the material IBM recommends before beginning. Assembling the various components (accessories) for the printer was easy. IBM designed the printer so that only a minimal number of tools are needed to install it. More than likely you, like myself, will spend more time configuring it and integrating the network workstations and servers than actually setting up the printer.

The weight and amount of space the printer requires is also important. I hope this information encourages you to think through your plans prior to receiving equipment at your site.

The printer used in this network is more than adequate for the needs here. I suggest that you contact IBM (at www.ibm.com or International Business Machines in Armonk, New York) and receive information to assist you in your plans for a network printer.

18.16 Network and Computer Security (McAfee)

Computer and network security is probably the single most important issue today. I predict it will become 3 to 5 times more important than it is today at some time in the future. Viruses, bots, and all sorts of antidata objects exist within the Internet. Most people have no idea how vulnerable parts of the Internet are. Even "service providers" are more vulnerable than they will admit. The sad fact is that every company, yes *every* company, regardless of size, has disinformation arsenals to make people feel secure. Higher-level management in most corporations operates in ignorance of these matters because it is legally safe and prudent for them to do so; ask and this statement will be denied, but remember where you read it. I point this out because I do not want you to be ignorant. There is no magic *program* or anything else that can make networks safe. Good programs exist, and the ones chosen and implemented in this network are examples; however, no single program can make your network 100 percent immune to security problems. Remember this during the design phase of your network. Networks can have security designed into them from the outset. Security in your network needs to be factored into every area from electricity provision, telephone access, and every other aspect that categorizes your network.

McAfee software suite was selected to meet the needs of this network. Part of the reason for this is the amount of antivirus programs and information they have—and it works. The second reason for selecting McAfee was the fact that they frequently update their antivirus software. At present McAfee has over 250 highly technical documents available on viruses; likewise, they claim to have information about the 1000 most common viruses. When the security analysis for this network was complete, the following software packages were selected and are used in this network: VirusScan, Desktop Security Suite, Commuter, QuickBackup, McAfee Service Desk, NetShield, WEBScan, and PCCrypto.

These products have been implemented on each system to varying degrees. The benefits and highlights of each program are presented here.

1. *VirusScan.* This program may well be the most popular antivirus software in the marketplace today. It operates with Windows 3.1, 95, Windows NT4.0, DOS, and OS/2. It is software that, once installed, operates automatically on power-up. It can be used at will once a system is operational. This program requires minimal space but does a professional job. This program is NCSAA-certified.

2. *Desktop Security Suite.* This program also operates with Windows 3.1, 95, and NT4.0. This suite of programs includes antivirus software, backup abilities, and encryption technology. The virus program is VirusScan. QuickBackup operates with ZIP, Jaz, the Internet, or rewritable CD-ROMs. The backup program enables hourly backup or on demand, whichever is best for you. The cryptographic part of the suite provides 160-bit encryption and enables users to encrypt files before they are sent over the Internet. The PC cryptographic part also permits network traffic to be encrypted between Windows-based computers and those running UNIX.

3. *Commuter.* Commuter is more than just a communication software. It also includes virus protection, desktop storage management, electronic mail, a personal information organizer, a calendar, a to-do list, and a contact manager.

4. *QuickBackup.* This backup program works with Windows 95 and NT4.0. It enables transparent backup of files to SCSI, ZIP, and Jaz drives. An icon-driven program makes for ease of use. The program installs quickly and works well. It does provide encryption protection and Internet support.

5. *Service Desk.* This product is powerful. It is actually multiple products in one box. It works with Windows 3.1, 95, and NT4.0. The product lets customer support personnel have access to information about the customer and make a remote connection to a system report ed with a problem. The package comes with the ability to distribute software. The package also includes a system diagnostic part for support personnel to use with customers.

6. *NetShield.* Netshield uses McAfee's proprietary code, called *Code Trace*, *Code Matrix*, and *Code Poly*. The product actually operates in an NT environment in native mode. The program takes full advantage of NT's server/client remote task distribution capability. The product supports real-time scanning during operation of other tasks.

7. *WEBScan.* The WEBScan product is designed to detect viruses within a browser. It examines downloads and email attachments, making it a power addition to any desktop or laptop system communicating in networks today. It also provides a cybersitter that blocks out unwanted Websites and chatgroups. Coverage of the program includes examination of `.doc, lzip. exe. zrc. arj.,` and other file types.

8. *PCCrypto.* PCCrypto is used to secure documents and other data files created by anyone using computers. It can encrypt graphics, spreadsheets, and text documents. It uses a 160-bit blowfish encryption mechanism. The package consumes a minimal amount of space and is one of the most powerful, if not *the* most powerful, tools of its kind on the market today.

I recommend that you dedicate a system for testing software initially and then install McAfee VirusScan. Then scan each and every diskette you have, even new diskettes fresh out of the box. Every diskette you receive from a manufacturer must be scanned. You say: “But Ed, isn’t this going a little too far?” I’ll say this. One year (I won’t say which) I bought some software; it was new. It came from the original vendor in shrink-wrapped plastic. The shrink-wrapped diskettes were enclosed in a box with a seal on it. The sealed box was also shrink-wrapped. I didn’t think anything about it. Within 10 min after opening the software, it brought one of my systems to its knees. The diskette had a virus. How do I know? I checked it personally. How do I know the system it went into was “clean” ? It was and is my benchmark system. Furthermore, those who know me know that nobody, not anyone, puts a disk in my systems except me. It took me 2 days to recover the system. Consider this next time you stick a new program on diskette in your system. You can pay now or gamble, but remember the odds are against you.

McAfee has other products that may meet your needs. I recommend you contact them. My experience with them has always been pleasant, and their staff are very informative. They can be reached at www.mcafee.com or

McAfee
2710 Walsh Avenue
Santa Clara, CA 95051
408-988-3832

18.17 MultiMedia Components (Creative Labs)

Creative Labs was chosen as the vendor for multimedia equipment. In the past, buying IBM compatible, or clones, was not a problem. However, all things change. In the arena of multimedia, Creative Labs wrote the book on how to do it. Since multimedia software is primarily add-on at this point in time, systems do not depend on it as they do the hard disk or monitors, for example.

However, some multimedia clone products exist. Many of these products attempt to copy what Creative Labs has already designed. In the arena of multimedia, clones are the incorrect way to invest money. The operational nature of some multimedia software is such that multimedia “clone” equipment may not be able to execute all the exploits of multimedia. This may sound strange, but it is true.

Today, systems typically have CD-ROMs, speakers, microphones, line outputs for amplifiers, and line inputs for peripheral integration, as well as software that enables a user to create, play back, and listen to or see various data streams.

All the desktop systems in this network are IBM 350 series. I selected these because each one would be customized to deliver a robust workload. Another reason for choosing this series is the upgrade capability. The same applies for Creative Labs equipment. All systems in this network also have Creative Lab multimedia hardware and software. One system has a package of multimedia equipment from Creative Labs. The system includes an interface board, speakers, necessary cabling, microphone, CD-ROM, infrared remote control, software drivers, and various software titles for viewing and listening.

Creative Labs has designed the benchmark for multimedia systems. The significance of this should not be overlooked during the design phase of your network. Windows 95 and NT4.0 acknowledge most, if not all, Creative Labs hardware and software, which are plug-and-play-compatible. Another significant aspect of this equipment is its adaptability. Creative Labs is continually upgrading its equipment to stay in line with other vendors; however, they support equipment and systems that are not this year's product.

Multimedia equipment is more than a CD-ROM and speakers. Today this equipment typically encompasses a digital videodisk (DVD) and enhanced display support. More than at any other time, displays need powerful drivers and memory to store the screen of information to be presented.

Creative Labs is based in California, but has offices around the world. I recommend contacting the one closest to you for additional information about multimedia, either on the Internet at www.soundblaster.com or

Creative Labs
1901 McCarthy Blvd.
Milpitas, CA 95035

Creative Labs Technical Support
1523 Cimarron Plaza
Stillwater, OK 74075

Creative Technology Ltd.
67 Ayer Rajah Crescent 03-18
Singapore 0513

Creative Labs Ltd.
Blanchardstown Industrial Park
Blanardstown, Budlin 15
Ireland

18.18. Network Storage (SMS Data Products Group)

Network storage is a big topic today and will become bigger in the coming months and years. As mentioned previously in this book, I selected an/SMS Data Products Group product to meet the needs of my network.

A couple of factors came together in my decision to use an /SMS product. First, their products are simple. Don't be deceived; most things in life that are powerful are simple. /SMS has an entire line of products that are very simple in terms of their requirements to install and operate. This simplicity is important because in case you haven't figured it out, networking is complex enough without adding complexity to the network! The second factor that contributed greatly in my decision to use /SMS for network storage is the people factor. Many interactions occurred between myself and /SMS personnel before I received the "silo" (my term for their network storage device). Every person I spoke with at /SMS was not only very courteous but also very knowledgeable. For example, in one of the first few conversations I had a question about some SCSI configuration. The salesperson I spoke with displayed a lot of integrity by replying: "Ed, I don't know the answer to that, but [person's name] does; let me have [that person] call you." That one comment told me a lot about the company and the integrity of the person I spoke with. When I later spoke to the technical person, after about 2 minutes I realized I was speaking with someone who was *very* knowledgeable about SCSI and all the connectivity matters of their equipment. This person was not trying to impress me, this person is a quality technical person. In fact, this person's words were confirmed by the Adaptec Corporation (the major company behind SCSI equipment).

Another factor for determining the selection of /SMS equipment is the fact that they have been in business for two decades at the time of this writing. Yes, that is important. Remember, this industry (*networking* as it is known today) is a relatively new phenomenon. For this company to have been in business in this area since the mid-1970s says something about the company itself. Other factors also contributed; suffice it to say that all things pointed to selecting /SMS as the network storage device provider of choice.

Consider Fig. 18-41.

Figure 18-41 shows the /SMS server. Note that it is connected to the network and to the network server. This is because it has dual-port connectivity and supports 10/100 Ethernet speeds. The connection to the network server is SCSI. The system used in this network has not only CDs but also a Barracuda hard drive and a Jaz drive. Now consider Fig. 18-42.

Figure 18-42 shows an/SMS system connected to the NetFinity 7000 and the NetFinity connected to the network via a four-port network adapter. This is a possible configuration for the /SMS system. I configured the silo to operate this way, and its functionality was in no way negatively affected by this type of network connection. As shown in Fig. 18-42, the network users can access the silo if they have access to it through CHEROKEE.

/SMS has a variety of network products. The ones used in this example are representative of what the company has to offer. I recommend contacting them for further information at www.sms.com or

/SMS Data Products Group
1501 Farm Credit Drive
McLean, VA 22102
800-331-1767

18.19 Network Wiring Analyzer (MicroTest)

Networks require wiring to some extent to function. In the case of my network, I determined that I needed a powerful wiring analyzer to make sure that the wiring used was in proper working order and met the specifications. As a result, I had to select a wiring analyzer. I selected the MicroTest PentaScanner.

This tool is powerful. It consists of battery-powered, hand-held units. It comes with software that enables the information captured about a given piece of cable to be uploaded onto a computer and saved and/or printed for further analysis.

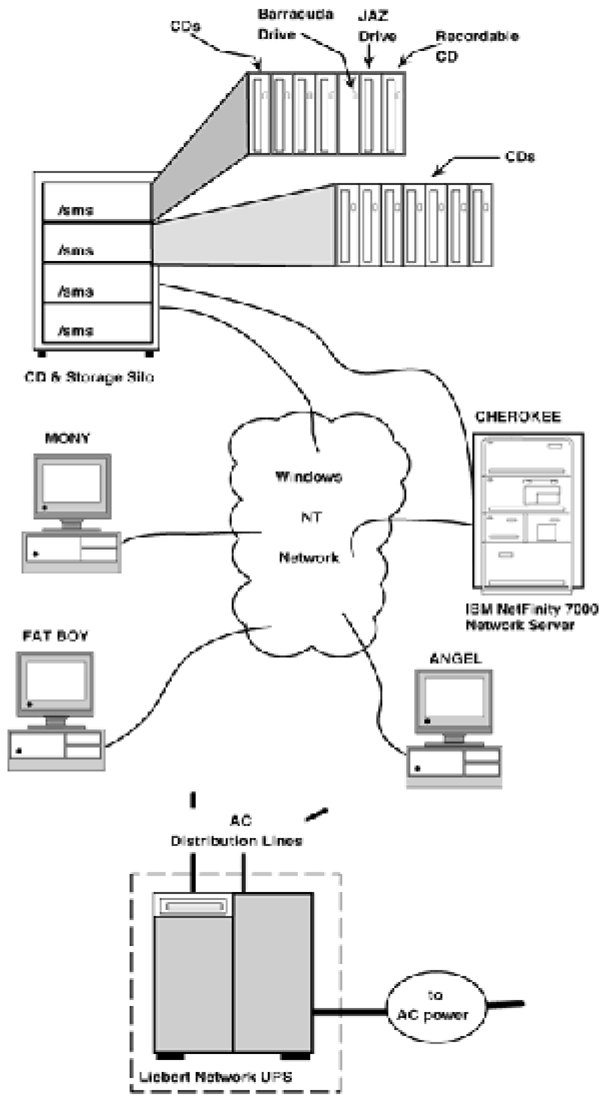


Figure 18-41
Network storage.

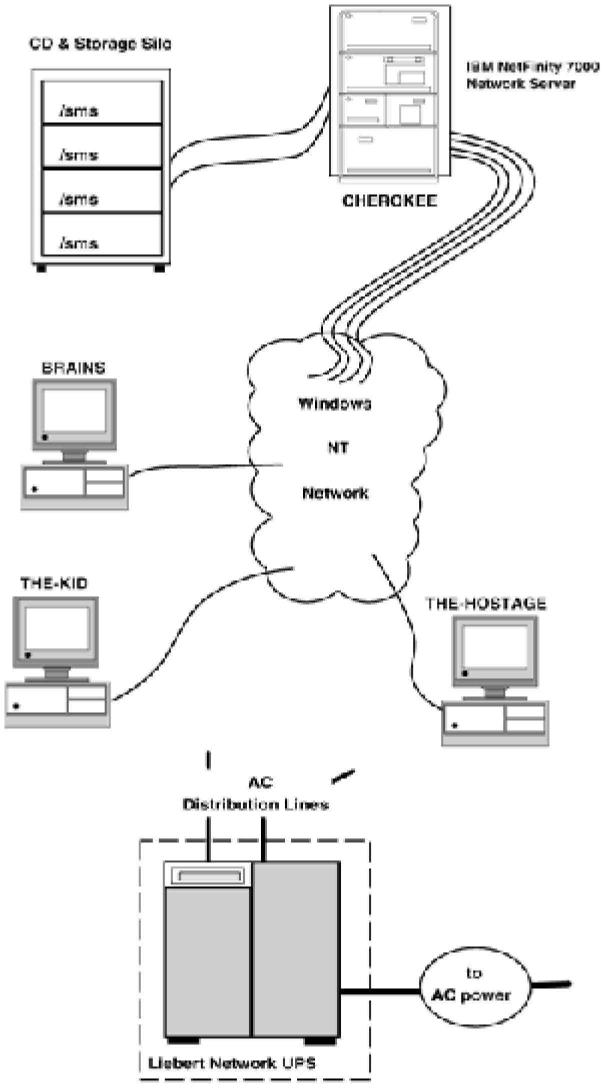


Figure 18-42
Network storage connections.

18.20 Troubleshooting Network Analyzer (Hewlett-Packard)

To design a network requires a certain type of skill, a marriage between the abstract and practical, if you will. To keep a network operational at its peak is another thing.

Obtaining information about an operational network is important. Being sure the information is accurate is more important. Examine Fig. 18-43, which is an actual snapshot on the network nodes participating at the time I took this sample with the HP Internet Advisor.

Consider the following information. It is a network stack decode performed by the Internet Advisor. Notice it explains some functions of the CHEROKEE.

```
***********************************************************
*****  HEWLETT-PACKARD NETWORK ADVISOR              *****
*****                                    *****
*****  Measurement:   Brief Network Stack Decode      *****
*****  Print Type:    All Frames                  *****
*****  Open Views:        Summary                *****
*****  Display Mode:  Viewing All Frames            *****
*****  Print Date:    03/20/98                   *****
*****  Print Time:    16:40:27                   *****
***********************************************************
1       36:18.087 WellfleetE8-8F-1C 01-80-C2-00-00-00 LLC C S = 42 D = UI
2       36:20.025 WellfleetE8-8F-1C 01-80-C2-00-00-00 LLC C S = 42 D = 42 UI
3       36:22.025 WellfleetE8-8F-1C 01-80-C2-00-00-00 LLC C S = 42 D = 42 UI
4       36:24.004 00000000-CHEROKEE 00000000-Broadcast IPX RIP Response:
          1  network
5       36:24.025 WellfleetE8-8F-1C 01-80-C2-00-00-00 LLC C S = 42 D = 42 UI
6       36:26.025 WellfleetE8-8F-1C 01-80-C2-00-00-00 LLC C S = 42 D =42 UI
7       36:28.025 WellfleetE8-8F-1C 01-80-C2-00-00-00 LLC C S = 42 D = 42 UI
8       36:29.413 WstDigt—93-73-D7 Broadcast ARP C PA = [220.100.100.10]
9       36:30.025 WellfleetE8-8F-1C 01-80-C2-00-00-00 LLC C S = 42 D = 42 UI
10      36:30.915 WstDigt—93-73-D7 Broadcast ARP C PA = [220.100.100.10]
11      36:32.025 WellfleetE8-8F-1C 01-80-C2-00-00-00 LLC C S = 42 D = 42 UI
12      36:32.418 WstDigt—93-73-D7 Broadcast ARP C PA = [220.100.100.10]
13      36:34.025 WellfleetE8-8F-1C 01-80-C2-00-00-00 LLC C S = 42 D = 42 UI
14      36:36.025 WellfleetE8-8F-1C 01-80-C2-00-00-00 LLC C S = 42 D = 42 UI
15      36:38.025 WellfleetE8-8F-1C 01-80-C2-00-00-00 LLC C S = 42 D = 42 UI
16      36:40.025 WellfleetE8-8F-1C 01-80-C2-00-00-00 LLC C S = 42 D = 42 UI
17      36:42.025 WellfleetE8-8F-1C 01-80-C2-00-00-00 LLC C S = 42 D = 42 UI
18      36:44.025 WellfleetE8-8F-1C 01-80-C2-00-00-00 LLC C S = 42 D = 42 UI
19      36:46.025 WellfleetE8-8F-1C 01-80-C2-00-00-00 LLC C S = 42 D = 42 UI
20      36:48.086 WellfleetE8-8F-1C 01-80-C2-00-00-00 LLC C S = 42 D = 42 UI
21      36:50.024 WellfleetE8-8F-1C 01-80-C2-00-00-00 LLC C S = 42 D = 42 UI
22      36:52.025 WellfleetE8-8F-1C 01-80-C2-00-00-00 LLC C S = 42 D = 42 UI
23      36:52.552 IP Multicast        Broadcast            BOOTP Request
24      36:54.024 WellfleetE8-8F-1C 01-80-C2-00-00-00 LLC C S = 42 D = 42 UI
 25      36:56.024 WellfleetE8-8F-1C 01-80-C2-00-00-00 LLC C S = 42 D = 42 UI
 26      36:58.024 WellfleetE8-8F-1C 01-80-C2-00-00-00 LLC C S = 42 D = 42 UI
 27      37:00.024 WellfleetE8-8F-1C 01-80-C2-00-00-00 LLC C S = 42 D = 42 UI
 28      37:02.024 WellfleetE8-8F-1C 01-80-C2-00-00-00 LLC C S = 42 D = 42 UI
 29      37:04.024 WellfleetE8-8F-1C 01-80-C2-00-00-00 LLC C S = 42 D = 42 UI
 30      37:06.024 WellfleetE8-8F-1C 01-80-C2-00-00-00 LLC C S = 42 D = 42 UI
 31      37:08.024 WellfleetE8-8F-1C 01-80-C2-00-00-00 LLC C S = 42 D = 42 UI
 32      37:09.487 CHEROKEE      Broadcast       ARP C PA = [220.100.100.10]
 33      37:10.024 WellfleetE8-8F-1C 01-80-C2-00-00-00 LLC C S = 42 D = 42 UI
 34      37:10.987 CHEROKEE      Broadcast       ARP C PA = [220.100.100.10]
 35      37:12.024 WellfleetE8-8F-1C 01-80-C2-00-00-00 LLC C S  42 D = 42 UI
 36      37:12.487 CHEROKEE      Broadcast       ARP C PA = [220.100.100.10]
 37      37:13.987 CHEROKEE      Broadcast       ARP C PA = [220.100.100.10]
```

```
38      37:14.024 WellfleetE8-8F-1C 01-80-C2-00-00-00 LLC C S = 42 D = 42 UI
39      37:15.487 CHEROKEE      Broadcast       ARP C PA = [220.100.100.10]
40      37:16.024 WellfleetE8-8F-1C 01-80-C2-00-00-00 LLC C S = 42 D = 42 UI
41      37:16.987 CHEROKEE      Broadcast        ARP C PA = [220.100.100.10]
42      37:17.539 CHEROKEE      220.100.100.255       NETB Datagram
43      37:18.086 WellfleetE8-8F-1C 01-80-C2-00-00-00 LLC C S = 42 D = 42 UI
44      37:18.487 CHEROKEE      Broadcast        ARP C PA = [220.100.100.10]
45      37:18.851 00000000-CHEROKEE 00000000-Broadcast SMB C
46      37:19.347 CHEROKEE 03-00-00-00-00-01 NETB Datagram CHEROKEE-
        JSPNRMPTGSBSSDIR
47      37:19.987 CHEROKEE      Broadcast         ARP C PA = [220.100.100.10]
48      37:20.024 WellfleetE8-8F-1C 01-80-C2-00-00-00 LLC C S = 42 D = 42 UI
49      37:21.487 CHEROKEE      Broadcast         ARP C PA = [220.100.100.10]
50      37:21.846 CHEROKEE      03-00-00-00-00-01      NETB Name query SILO-1
51      37:21.846 00000000-CHEROKEE 00000000-Broadcast NETB Find Name
          SILO-1
52      37:22.024 WellfleetE8-8F-1C 01-80-C2-00-00-00 LLC C S = 42 D = 42 UI
53      37:22.409 00000000-CHEROKEE 00000000-Broadcast NETB Find Name
         SILO-1
54      37:22.971 00000000-CHEROKEE 00000000-Broadcast NETB Find Name
         SILO-1
55      37:22.987 CHEROKEE       Broadcast             ARP C PA = [220.100.100.10]
56      37:24.003 00000000-CHEROKEE 00000000-Broadcast IPX RIP Response: 1
         network
57      37:24.024 WellfleetE8-8F-1C 01-80-C2-00-00-00 LLC C S = 42 D = 42 UI
58      37:24.487 CHEROKEE Broadcast         ARP C PA = [220.100.100.10]
59      37:25.987 CHEROKEE Broadcast          ARP C PA = [220.100.100.10]
60      37:26.024 Wellfleet E8-8F-1C 01-80-C2-00-00-00 LLC C = 42 D = 42 UI
61      37:27.487 CHEROKEE Broadcast          ARP C PA = [220.100.100.10]
62      37:28.024 Wellfleet E8-8F-1C 01-80-C2-00-00-00 LLC C S = 42 D = 42 UI
63      37:28.346 CHEROKEE 220.100.100.255 NETB C ID = 33218 Query Name =
        JSPNRMPTGSBSSDIR
64      37:28.987 CHEROKEE Broadcast           ARP C PA = [220.100.100.10]
65      37:29.096 CHEROKEE 220.100.100.255                NETB C ID = 33218 Query
        Name = JSPNRMPTGSBSSDIR
66      37:29.648 CHEROKEE 03-00-00-00-00-01           SMB C Transaction name
67      37:29.846 CHEROKEE 220.100.100.255                  NETB C ID = 33218 Query
        Name = JSPNRMPTGSBSSDIR
68      37:30.024 Wellfleet E8-8F-1C 01-80-C2-00-00-00 LLC C S = 42 D = 42 UI
```

```
69      37:30.487 **CHEROKEE** Broadcast            ARP C PA = [220.100.100.10]
70      37:30.846 **CHEROKEE** 220.100.100.255            NETB C ID = 33220 Query
                 Name = **SILO-1**
71      37:30.851 **CHEROKEE** Broadcast            ARP C PA = [220.100.100.175]
72      37:31.300 **CHEROKEE** Broadcast            ARP C PA = [220.100.100.8]
73      37:31.987 **CHEROKEE** Broadcast             ARP C PA = [220.100.100.10]
74      37:32.024 Wellfleet E8-8F-1C 01-80-C2-00-00-00 LLC C S = 42 D = 42 UI
75      37:33.487 **CHEROKEE** Broadcast              ARP C PA = [220.100.100.10]
76      37:34.024 Wellfleet E8-8F-1C 01-80-C2-00-00-00 LLC C S = 42 D = 42 UI
77      37:34.987 **CHEROKEE** Broadcast               ARP C PA = [220.100.100.10]
78      37:36.024 Wellfleet E8-8F-1C 01-80-C2-00-00-00 LLC C S = 42 D = 42 UI
79      37:36.487 **CHEROKEE** Broadcast               ARP C PA = [220.100.100.10]
80      37:37.987 **CHEROKEE** Broadcast          ARP C PA = [220.100.100.10]
81      37:38.024 Wellfleet E8-8F-1C 01-80-C2-00-00-00 LLC C S = 42 D = 42 UI
82      37:39.487 **CHEROKEE** Broadcast           ARP C PA = [220.100.100.10]
83      37:40.023 Wellfleet E8-8F-1C 01-80-C2-00-00-00 LLC C S = 42 D = 42 UI
84      37:40.987 **CHEROKEE** Broadcast           ARP C PA = [220.100.100.10]
85      37:42.023 Wellfleet E8-8F-1C 01-80-C2-00-00-00 LLC C S = 42 D = 42 UI
86      37:42.467 WstDigt—93-73-D7 Broadcast ARP C PA = [220.100.100.53]
87      37:42.487 **CHEROKEE** Broadcast           ARP C PA = [220.100.100.10]
88      37:43.987 **CHEROKEE** Broadcast           ARP C PA = [220.100.100.10]
89      37:44.023 Wellfleet E8-8F-1C 01-80-C2-00-00-00 LLC C S = 42 D = 42 UI
90      37:45.487 **CHEROKEE**       Broadcast           ARP C PA = [220.100.100.10]
91      37:46.023 Wellfleet E8-8F-1C 01-80-C2-00-00-00 LLC C S = 42 D = 42 UI
92      37:46.986 **CHEROKEE** Broadcast           ARP C PA =[220.100.100.10]
93      37:48.085 Wellfleet E8-8F-1C 01-80-C2-00-00-00 LLC C S = 42 D = 42 UI
```

Still additional information can be obtained by the Internet Advisor (IA). Consider the following information also obtained with the IA. This information shows Ethernet vital signs for frames between 688 and 742 and the time when the capture was taken.

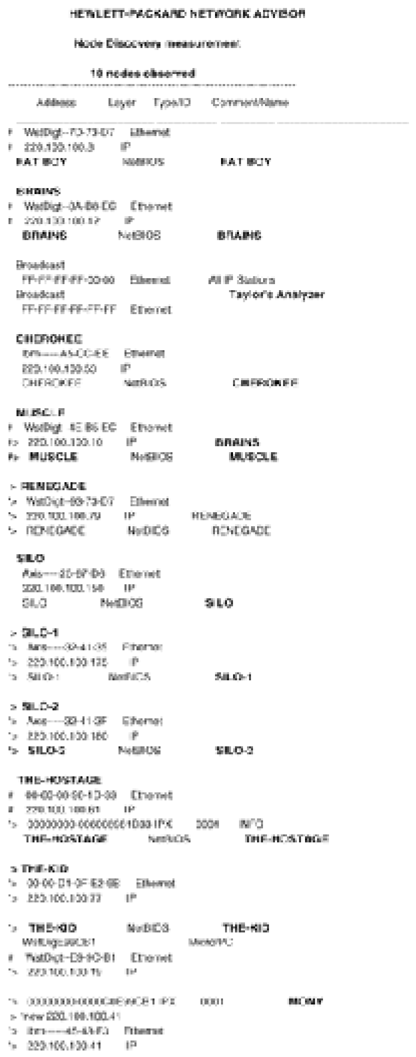
ETHERNET Vital Signs Measurement

Frame Range: 688..742

Threshold Current Average   Peak   Total

Network Counts (Pre-Filter)

Utilization %	40	0.00	0.00	0.08	
Frames	700	0	0	5	54
Local coll	35	0	0	0	0
Late coll	0	0	0	0	0
Remote coll	35	0	0	0	0
Rem late coll	0	0	0	0	0
Bad FCS	0	0	0	0	0
Runt	0	0	0	0	0
Misaligns	0	0	0	0	0



```
            HEWLETT-PACKARD NETWORK ADVISOR

             Node Discovery measurement

                10 nodes observed
------------------------------------------------------------
      Address        Layer   Type/ID    Comment/Name
________________________________________________________________
#  WstDigt--70-73-D7    Ethernet
#  220.100.100.3        IP
 FAT BOY              NetBIOS            FAT BOY

 BRAINS
#  WstDigt--3A-B6-EC    Ethernet
#  220.100.100.17       IP
  BRAINS             NetBIOS           BRAINS

 Broadcast
  FF-FF-FF-FF-00-00    Ethernet         All IP Stations
 Broadcast                                 Taylor's Analyzer
  FF-FF-FF-FF-FF-FF    Ethernet

 CHEROKEE
  IBM----A5-CC-EE    Ethernet
  220.100.100.50       IP
  CHEROKEE            NetBIOS              CHEROKEE

 MUSCLE
#  WstDigt  4E-B6-EC    Ethernet
#>  220.100.100.10       IP                BRAINS
#>  MUSCLE              NetBIOS            MUSCLE

 > RENEGADE
*>  WstDigt--69-73-D7    Ethernet
*>  220.100.100.79       IP           RENEGADE
*>  RENEGADE            NetBIOS          RENEGADE

 SILO
  Ask----20-97-B6    Ethernet
  220.100.100.158      IP
  SILO             NetBIOS              SILO

 > SILO-1
*>  Ask----30-41-35    Ethernet
*>  220.100.100.175      IP
*>  SILO-1            NetBIOS             SILO-1

 > SILO-2
*>  Ask----30-41-3F    Ethernet
*>  220.100.100.180      IP
*>  SILO-2             NetBIOS             SILO-2

 THE-HOSTAGE
#  00-00-00-00-1D-00    Ethernet
#  220.100.100.61       IP
*>  00000000-000000001D00 IPX      0004     INFO
  THE-HOSTAGE           NetBIOS            THE-HOSTAGE

 > THE-KID
*>  00-00-D1-0F-E2-0B    Ethernet
*>  220.100.100.77       IP

*>  THE-KID            NetBIOS            THE-KID
  WstDigt-D9C61                   MeteorPC
#  WstDigt--D9-9C-B1    Ethernet
*>  220.100.100.15       IP

*>  00000000-0000C0E9C61 IPX      0001              MONY
 > new 220.100.100.41
*>  IBM----45-4A-F0    Ethernet
*>  220.100.100.41       IP
```

Figure 18-43
Network node discovery.

Buffer Counts (Post-Filter)

Utilization %	40	0.00	0.00	0.08		
Frames	700	0	0	5	54	
Runts (good FCS)		0	0	0	0	0
Jabbers	0	0	0	0	0	
Jabber (bad FCS)		0	0	0	0	0
Dribble frms	35	0	0	0	0	
Broadcasts	25	0	0	3	16	
Multicasts	2	0	0	2	38	
Buff Overwrites	100	0	0	0	0	

Start Time: Mar 20 98 @ 16:58:54
Sample Time: Mar 20 98 @ 17:00:00

Again, having this information filed for future reference can be invaluable. With this level of information about each node, or major nodes, on the network a clear baseline can be established.

Consider the following information.

ETHERNET Connection Stats

```
Sample Date Time Src. Address Dst. Address Frames Bytes Errors Stn 1 Fr Stn 2 Fr Fr lost

1 Mar 20 98 17:01:43 Network Total 1,64,0,1,0,0
1 Mar 20 98 17:01:43 WellfleetE8-8F-1C 01-80-C2-00-00-00”,1,64,0,1,0
2 Mar 20 98 17:01:53 Network Total 9,858,0,9,0,0
2 Mar 20 98 17:01:53 Wellfleet E8-8F-1C 01-80-C2-00-00-00”,6,384,0,6,0
2 Mar 20 98 17:01:53 WstDigt—93-73-D7 Broadcast”,2,128,0,2,0
2 Mar 20 98 17:01:53 Wellfleet E8-8F-10 Broadcast”,1,346,0,1,0
3 Mar 20 98 17:02:03 Network Total 23,1786,0,23,0,0
3 Mar 20 98 17:02:03 Wellfleet E8-8F-1C 01-80-C2-00-00-00”,11,704,0,11,0
3 Mar 20 98 17:02:03 WstDigt—93-73-D7 Broadcast”,11,736,0,11,0
3 Mar 20 98 17:02:03 Wellfleet E8-8F-10 Broadcast”,1,346,0,1,0
4 Mar 20 98 17:02:13 Network Total 31,2330,0,31,0,0
4 Mar 20 98 17:02:13 Wellfleet E8-8F-1C 01-80-C2-00-00-00”,16,1024,0,16,0
4 Mar 20 98 17:02:13 WstDigt—93-73-D7 Broadcast”,14,960,0,14,0
4 Mar 20 98 17:02:13 Wellfleet E8-8F-10 Broadcast”,1,346,0,1,0
```

Note that the previous information shows details about the Ethernet connection status. With such information in hand, utilization and per-node access percentages can be calculated.

Consider the following information. It presents details about the MS protocol by frame. This is only one of the many protocol decodes the HP IA is capable of performing. The Internet Advisor decodes the frames and makes interpretation of this information easy.

```
*****************************************************************
*****                                  *****
*****     HEWLETT-PACKARD NETWORK ADVISOR            *****
*****                                  *****
*****     Measurement:   MS Windows Stack Decode      *****
*****     Print Type:    Frames 1 to 40            *****
*****     Open Views:    Summary Detailed           *****
*****     Display Mode:  Viewing All Frames         *****
*****     Print Date:    03/20/98                *****
*****     Print Time:    17:14:11                *****
*****************************************************************
```

---

16 46:31.278 **CHEROKEE** 03-00-00-00-00-01 NETB Name query **RENEGADE**

Frame: 16 Time: Mar 20@16:46:31.2788912 Length: 65

******* DETAILED FORMAT *******

```
NetBIOS (NetBEUI)
Header Length    44
Delimiter        EFFF
Command          0A          NAME_QUERY
Optional Data 1  00          Reserved field
  Name Type      00          Unique name type
  Session Number             00       Error: Session number of 0
invalid
Transmit Correlator          0000     Reserved field
Response Correlator          001F
Destination Name             RENEGADE
Source Name   CHEROKEE
```

---

18 46:31.288 00000000-**CHEROKEE** 00000000-Broadcast NETB Find Name
RENEGADE

```
Frame: 18 Time: Mar 20@16:46:31.2883590 Length: 102

         ***** DETAILED FORMAT *****
Novell NetBios (IPX/SPX)
Connection Control Flag        00
          0.......      Non-System Packet
           .0......     Non Send ACK
           ..0.....     Not Attention
            ...0....    Not End of Message
             ....0...   No Rsend Needed
Operation              1         Find Name
Name Type Flag   00
          0.......      Unique Name
           .0......     Name Not Used
        .....0..        Name Not Registered
         ......0.       Name Not Duplicated
         .......0       Name Not Deregistered
Operation              1        Find Name
Name Claim String           RENEGADE
```

```
19 46:31.850 00000000 - CHEROKEE 00000000-Broadcast NETB Find Name
                        RENEGADE

Frame: 19  Time: Mar 20@16:46:31.8507519 Length: 102

          ***** DETAILED FORMAT *****
Novell NetBios (IPX/SPX)
Connection Control Flag          00
           0.......      Non-System Packet
            .0......     Non Send ACK
             ..0.....    Not Attention
              ...0....   Not End of Message
               ....0...  No Rsend Needed
Operation               1          Find Name
Name Type Flag 00
           0.......      Unique Name
           .0......      Name Not Used
           .....0..      Name Not Registered
           ......0.      Name Not Duplicated
           .......0      Name Not Deregistered
Operation               1               Find Name
Name Claim String             RENEGADE
```

```
21 46:32.413 00000000 - CHEROKEE 00000000-Broadcast NETB Find Name
                        RENEGADE
```

```
Frame: 21    Time: Mar 20@16:46:32.4132375 Length: 102

        ***** DETAILED FORMAT *****
Novell NetBios (IPX/SPX)
Connection Control Flag      00
        0.......     Non-System Packet
        .0......     Non Send ACK
        ..0.....     Not Attention
        ...0....     Not End of Message
        ....0...     No Rsend Needed
Operation            1           Find Name
Name Type Flag 00
        0.......     Unique Name
        .0......     Name Not Used
        .....0..     Name Not Registered
        ......0.     Name Not Duplicated
        .......0     Name Not Deregistered
Operation            1           Find Name
Name Claim String              RENEGADE
```

The previous information can be very helpful when troubleshooting network problems. This level of detail provides per-node communication functions.

The HP Internet Advisor provides complete seven-layer decoding and analysis of protocols. Consider the following captured statistics.

Novell Vital Signs Measurement

Frame Range: 186..1759

	Threshold	**Current**	**Average**	**Peak**	**Total**	
Network Util %	10	0.00	2.16	23.81		
IPX Util %	10	0.00	0.00	0.15		
Network Packets		1200	0	37	368	1573
IPX Packets	1000	0	0	13	37	
IPX Packet Size	1000	0	8	151		
Local Tx Rate	1000	0	0	13	37	
Remote Tx Rate	1000	0	0	0	0	
Burst Mode	500	0	0	0	0	
RIP Frames	10	0	0	0	0	
SAP Frames	10	0	0	1	2	
Read Rq Pkts	500	0	0	0	0	
Write Rq Pkts	500	0	0	0	0	

Busy Server %	4	0.00	0.00	0.00		
Buffer Overwrites		100	0	0	0	0

Start Time: Mar 23 98 @ 15:16:58
Sample Time: Mar 23 98 @ 15:18:47

These statistics provide valuable insight into Novell protocol operation in the network. Additional information can also be obtained about IPX/SPX protocol.

As mentioned previously, the IA can decode multiple protocol stacks. Consider the following.

```
*********************************************************************
*****                                          *****
*****      HEWLETT-PACKARD NETWORK ADVISOR          *****
*****                                          *****
*****      Measurement:    Network Stack Decode     *****
*****      Print Type:     All Frames           *****
*****      Open Views:     Summary Detailed       *****
*****      Display Mode:   Viewing All Frames     *****
*****      Print Date:     03/23/98             *****
*****      Print Time:     15:12:47             *****
*********************************************************************
```

---

1 29.464599 **MUSCLE** 03-00-00-00-00-01 NETB Datagram **MUSCLE**->JSPN- RMPTGSBSSDIR

Frame: 1 Time: Mar 23@15:10:29.4645992 Length: 75

******* DETAILED FORMAT *******

NetBIOS

Header Length 44
Delimiter EFFF

```
Command     08             DATAGRAM
Optional Data 1 00            Reserved field
Optional Data 2 0000           Reserved field
Transmit Correlator           0000     Reserved field
Response Correlator           0000     Reserved field
Destination Name              JSPNRMPTGSBSSDIR
Source Name    MUSCLE
  802.2
Destination SAP F0              NetBios
Source SAP      F0              NetBios
Command/Response                 ....-...0   Command
Type  03       Unnumbered
Poll          ...0-....
Modifier       000.-00..       Information
  802.3/Ethernet
Destination address             03-00-00-00-00-01 Group, local
Source address  MUSCLE Individual, global
Length          57
Frame check sequence                  7C-32-0E-3B
>Data size       57
```

```
2 30.339447 WstDigt—93-73-D7 Broadcast ARP C PA = [220.100.100.10]

Frame: 2    Time: Mar 23@15:10:30.3394479 Length: 64

          ***** DETAILED FORMAT *****

ARP/RARP
Hardware       1                    Ethernet
Protocol       08-00                 IP
HW addr length 6
Phys addr length                      4
Operation      1                    ARP Request
Sender HW addr                        00-00-C0-93-73-D7
Sender internet addr                  220.100.100.79
Target HW addr 00-00-00-00-00-00
Target internet addr                 220.100.100.10
  802.3/Ethernet
Destination address                 Broadcast           Broadcast
Source address  WstDigt—             93-73-D7             Individual, global
Type   08-06    ARP
Frame check sequence                BF-A0-DC-C2
>Data size       46
```

```
3 30.339769 MUSCLE WstDigt—93-73-D7 ARP R PA = MUSCLE HA =
0000C04EB5EC
```

Frame: 3 Time: Mar 23@15:10:30.3397692 Length: 64

```
     ***** DETAILED FORMAT *****
ARP/RARP

Hardware        1                 Ethernet
Protocol        08-00              IP
HW addr length 6
Phys addr length              4
Operation       2               ARP Reply
Sender HW addr                   00-00-C0-4E-B5-EC
Sender internet addr            220.100.100.10
Target HW addr 00-00-C0-93-73-D7
Target internet addr            220.100.100.79
  802.3/Ethernet
Destination address             WstDigt—93-73-D7     Individual, global
Source address    MUSCLE                  Individual, global
Type   08-06       ARP
Frame check sequence             F4-93-C6-3E
>Data size        46
```

30.340128 WstDigt—93-73-D7 **MUSCLE** NETB C ID = 32878 Query Name
= **FAT BOY**

Frame: 4 Time: Mar 23@15:10:30.3401281 Length: 96

```
        ***** DETAILED FORMAT *****
NetBIOS

Service Type :       Name Service
Name_Trn_ID          32878
Packet Type :        Name Query Request
Opcode, NM_Flags,        Rcode 01-00    See Bit Fields Below
  Response Flag0.......        Request Packet
  Opcode       .0000...        Query
  Auth. Answer Flag            .....0..        False
  Truncation Flag              ......0.        False
  Recursion Desired Flag       .......1        True
  Recursion Available Flag     0.......        False
  Reserved                     .00.....
  Broadcast Flag               ...0....        False
  Rcode        ....0000
Qdcount        1             No. of Entries In Question
  Section
Ancount        0             No. of Entries In Answer
  Section
Nscount        0             No. of Entries In Authority
  Section
```

```
Arcount         0              No. of Entries In Additional
  Records Section
Name Length     32
Question_Name   FAT BOY     NetBIOS Name
Question_Type   32             General Name Service
  Resource Record
Question_Class  1              Internet Class
  UDP
Source port      137           NETBIOS
Destination port 137           NETBIOS
Length          58
Checksum         A2-72
IP
Version         4
Internet header length          5       (32 bit words)
Precedence       000.-....      Routine
Delay ...0-....  Delay normal
Throughput       ....-0...      Throughput normal
Reliability      ....-.0..      Reliability normal
Total Length     78
Identification   40459
May / Do Not Fragment          .0..-....     Fragmentation allowed
Last / More Fragments          ..0.-....     Last fragment
Offset  0
Time To Live     128
Next Protocol    17            UDP
Checksum         1B-71
Source 220.100.100.79
Destination      MUSCLE
>Data size        58
  802.3/Ethernet
Destination address           MUSCLE           Individual, global
Source address   WstDigt—93-73-D7              Individual, global
Type   08-00    IP
Frame check sequence    B6-8F-A8-0E
>Data size 78
```

```
5 30.341308 MUSCLE WstDigt—93-73-D7   NETB R ID = 32878 Query Name =
          FAT BOY
```

```
Frame: 5     Time: Mar 23@15:10:30.3413080 Length: 108

***** DETAILED FORMAT *****
  NetBIOS
Service Type :       Name Service
Name_Trn_ID          32878
Packet Type :        Positive Name Query Response
Opcode, NM_Flags, Rcode       85-80      See Bit Fields Below
  Response Flag  1.......      Response  Packet
  Opcode         .0000...      Query
  Auth. Answer Flag           .....1..  True
  Truncation Flag            ......0.  False
  Recursion Desired Flag      .......1  True
  Recursion Available Flag             1.......      True
  Reserved         .00.....
  Broadcast Flag             ...0....  False
  Rcode            ....0000
Qdcount          0              No. Of Entries In Question
  Section
Ancount          1              No. Of Entries In Answer
  Section
Nscount          0              No. Of Entries In Authority
  Section
Arcount          0              No. of Entries In Additional
  Records Section
Name Length         32
RR_Name           FAT BOY             NetBIOS Name
RR_Type            32          General Name Service
  Resource Record
RR_Class           1          Internet Class
TTL   0
Rdlength           6
NB_Flags           60-00       See Bit Fields Below
  Group Flag        0.......     Unique Name
  Owner Node Type               .11.....       Reserved Bit(s) Not Zero
  Reserved Bits     ...0000000000000
Sliced Data (Truncated)         Sliced Data
  UDP
Source port         137          NETBIOS
Destination port    137          NETBIOS
Length             70          Partial packet store
Checksum            7D-66       Checksum not checked
  IP
Version            4
Internet header length           5            (32 bit words)
Precedence          000.-....    Routine
Delay   ...0-....  Delay normal
Throughput          ....-0...    Throughput normal
```

```
Reliability         ....-.0..    Reliability normal
Total Length        90           Partial packet store
Identification      63241
May / Do Not Fragment            .0..-....       Fragmentation allowed
Last / More Fragments            ..0.-....       Last fragment
Offset  0
Time To Live       128
Next Protocol      17            UDP
Checksum           C2-66
Source MUSCLE
Destination        220.100.100.79                RENEGADE
>Data size         66
  802.3/Ethernet
Destination address              WstDigt—93-73-D7           Individual, global
Source address          MUSCLE              Individual, global
Type   08-00      IP
>Sliced data
>Data size         86
```

```
6 30.341872 WstDigt—93-73-D7 Broadcast    ARP C PA = [220.100.100.8]
```

Frame: 6   Time: Mar 23@15:10:30.3418728 Length: 64

```
     ***** DETAILED FORMAT*****
ARP/RARP
Hardware           1                 Ethernet
Protocol           08-00             IP
HW addr length     6
Phys addr length                     4
Operation          1                 ARP Request
Sender HW addr                            00-00-C0-93-73-D7
Sender internet addr                      220.100.100.79
Target HW addr 00-00-00-00-00-00
Target internet addr                      220.100.100.8
  802.3/Ethernet
Destination address                       Broadcast           Broadcast
Source address  WstDigt—93-73-D7                 Individual, global
Type  08-06      ARP
Frame check sequence                      35-E7-B6-AA
>Data size         46
```

```
7 32.467583 MUSCLE    Broadcast NETB C ID = 33902 Query Name = JSPN- RMPTGSBSSDIR
```

Frame: 7   Time: Mar 23@15:10:32.4675834 Length: 96

******* DETAILED FORMAT *******

NetBIOS

```
Service Type :     Name Service
Name_Trn_ID    33902
Packet Type :     Name Query Request
Opcode, NM_Flags, Rcode 01-10      See Bit Fields Below
  Response Flag  0.......          Request Packet
  Opcode         .0000...          Query
  Auth. Answer Flag                .....0..     False
  Truncation Flag                  ......0.     False
  Recursion Desired Flag           .......1    True
  Recursion Available Flag         0.......           False
  Reserved           .00.....
  Broadcast Flag                   ...1....        True
  Rcode             ....0000
Qdcount        1                  No. of Entries In Question
  Section
Ancount        0                  No. of Entries In Answer
  Section
Nscount        0                  No. of Entries In Authority
  Section
Arcount        0                  No. of Entries In Additional
  Records Section
Name Length     32
Question_Name JSPNRMPTGSBSSDIR             NetBIOS Name
Question_Type  32                  General Name Service
  Resource Record
Question_Class  1                  Internet Class
  UDP
Source port     137               NETBIOS
Destination port137                NETBIOS
Length          58
Checksum        73-97
  IP
Version         4
```

```
Internet header length     5            (32 bit words)
Precedence                000.-....     Routine
Delay  ...0-....          Delay normal
Throughput                ....-0...     Throughput normal
Reliability               ....-.0..     Reliability normal
Total Length              78
Identification            64009
May / Do Not Fragment     .0..-....     Fragmentation allowed
Last / More Fragments     ..0.-....     Last fragment
Offset  0
Time To Live              128
Next Protocol             17            UDP
Checksum                  BE-C2
Source MUSCLE
Destination               220.100.100.255
>Data size                58
802.3/Ethernet
Destination address                 Broadcast          Broadcast
Source address      MUSCLE   Individual, global
Type  08-00               IP
Frame check sequence      EC-E8-DB-13
>Data size                78
```

```
8 33.218765 MUSCLE Broadcast NETB C ID = 33902 Query Name = JSPNRMPT-
GSBSSDIR

Frame: 8      Time: Mar 23@15:10:33.2187650 Length: 96

     ***** DETAILED FORMAT *****
NetBIOS

Service Type :          Name Service
Name_Trn_ID               33902
Packet Type :           Name Query Request
Opcode, NM_Flags, Rcode  01-10      See Bit Fields Below
  Response Flag          0.......   Request Packet
  Opcode               .0000...   Query
  Auth. Answer Flag                .....0..     False
  Truncation Flag                  ......0.     False
  Recursion Desired Flag           .......1     True
  Recursion Available Flag         0.......         False
  Reserved       .00.....
  Broadcast Flag                   ...1....     True
  Rcode              ....0000
Qdcount              1                  No. of Entries In Question
  Section
Ancount              0                  No. of Entries In Answer
  Section
Nscount              0                  No. of Entries In Authority
```

```
  Section
Arcount              0                    No. of Entries In Additional
  Records Section
Name Length          32
Question_Name          JSPNRMPTGSBSSDIR      NetBIOS Name
Question_Type          32            General Name Service
  Resource Record
Question_Class         1            Internet Class
  UDP
Source port            137          NETBIOS
Destination port       137          NETBIOS
Length                 58
Checksum                 73–97
  IP
Version                4
Internet header length  5             (32 bit words)
Precedence             000.-....      Routine
Delay  ...0-....        Delay normal
Throughput             ....-0...      Throughput normal
Reliability            ....-.0..      Reliability normal
Total Length           78
Identification         64265
May / Do Not Fragment   .0..-....      Fragmentation allowed
Last / More Fragments   ..0.-....      Last fragment
Offset  0
Time To Live           128
Next Protocol          17            UDP
Checksum               BD-C2
Source  MUSCLE
Destination            220.100.100.255
>Data size             58
  802.3/Ethernet
Destination address              Broadcast      Broadcast
Source address  MUSCLE               Individual, global
Type  08-00            IP
Frame check sequence               D1-A1-D2-5B
>Data size             78
```

---

9 33.344719 WstDigt—93-73-D7 Broadcast ARP C PA = [220.100.100.8]

```
Frame: 9       Time: Mar 23@15:10:33.3447195 Length: 64

     ***** DETAILED FORMAT *****
ARP/RARP
Hardware          1              Ethernet
Protocol          08-00          IP
HW addr length    6
Phys addr length                 4
Operation         1              ARP Request
Sender HW addr                   00-00-C0-93-73-D7
Sender internet addr             220.100.100.79
Target HW addr 00-00-00-00-00-00
Target internet addr             220.100.100.8
  802.3/Ethernet
Destination address              Broadcast               Broadcast
Source address    WstDigt—93-73-D7             Individual, global
Type       08-06        ARP
Frame check sequence                     35-E7-B6-AA
>Data size             46
_____________________________________________________________________________
10 33.969839 MUSCLE Broadcast       NETB C ID = 33902 Query Name =
                      JSPNRMPTGSBSSDIR

Frame: 10       Time: Mar 23@15:10:33.9698393 Length: 96

    ***** DETAILED FORMAT *****
NetBIOS

Service Type :       Name Service
Name_Trn_ID          33902
Packet Type :        Name Query Request
Opcode, NM_Flags, Rcode       01–10        See Bit Fields Below
  Response Flag      0.......       Request Packet
  Opcode             .0000...       Query
  Auth. Answer Flag                 .....0..       False
  Truncation Flag                   ......0.       False
  Recursion Desired Flag            .......1       True
  Recursion Available Flag          0.......       False
  Reserved           .00.....
  Broadcast Flag                    ...1....       True
  Rcode              ....0000
Qdcount          1               No. Of Entries In Question
  Section
Ancount          0               No. Of Entries In Answer
  Section
Nscount          0               No. Of Entries In Authority
  Section
Arcount          0               No. of Entries In Additional
```

```
  Records Section
Name Length      32
Question_Name         JSPNRMPTGSBSSDIR       NetBIOS Name
Question_Type     32              General Name Service
  Resource Record
Question_Class    1               Internet Class
  UDP
Source port       137             NETBIOS
Destination port  137             NETBIOS
Length           58
Checksum        73–97
  IP
Version          4
Internet header length            5         (32 bit words)
Precedence       000.-....        Routine
Delay  ...0-....  Delay normal
Throughput        ....-0...        Throughput normal
Reliability       ....-.0..        Reliability normal
Total Length      78
Identification    64521
May / Do Not Fragment             .0..-....    Fragmentation allowed
Last / More Fragments             ..0.-....      Last fragment
Offset  0
Time To Live           128
Next Protocol          17          UDP
Checksum               BC-C2
Source  MUSCLE
Destination            220.100.100.255
>Data size             58
  802.3/Ethernet
Destination address     Broadcast      Broadcast
Source address          MUSCLE  Individual, global
Type   08-00            IP
Frame check sequence                B1-95-93-E7
>Data size             78
```

The previous data capture with the Internet Advisor provides a wealth of information. I captured 10 frames of this sequence and noticed the amount of detailed provided. NetBIOS, ARP, and RARP protocols are presented. Both destination and source addresses are shown. In addition, data size and type of frame are indicated. This level of tracing is valuable to repositories of information for baseline setting and future planning.

Another insight into network traffic is to know what nodes are consuming the majority of the network's bandwidth. Consider the following information.

ETHERNET Top Talkers

Node	**Frames Xmt**	**Bytes Xmt**	**Frames Rcv**	**Bytes Rcv**
BRAINS	1042	213602	1696	1637371
THE-HOSTAGE	989	1150866	502	60320
Axis——32-41-35	710	690040	341	34497
MUSCLE	498	87757	474	155264
WstDigt—93-73-D7	312	39480	439	358026
220.100.100.8	299	27109	474	280046
Axis——32-41-3F	290	321964	144	14717
WstDigE99CB1	162	28951	132	12120
00-00-D1-0F-E2-9B	73	18028	77	11554
Ibm———45-43-F3	27	5389	28	4564
Broadcast	0	0	65	9613
03-00-00-00-00-01	0	0	29	5029
01-00-5E-00-01-18	0	0	1	65

The “top talkers” is a data capture of nodes talking on the network at a given instance. This is another good aspect of information that the Internet Advisor is capable of gathering.

TCP/IP ETHERNET Commentator

Commentating on: TCP/IP

```
TCP: Close Connection       [Normal] Mar 23@14:11:56.7437798
  220.100.100.79            <—> 220.100.100.8
   220.100.100.8
  Port: 1034   NETBIOS 139
  Tx Packets: 4   3
  Low Window: 0             0
  Retrans: 0
  Connection Duration: 0:00:00.0088922
  Frame Number(s): 82
TCP: Close Connection       [Normal] Mar 23@14:12:01.7474972
  220.100.100.79            <—> 220.100.100.19
  Port: 1035   NETBIOS 139
  Tx Packets: 4   3
```

```
   Low Window: 0               0
   Retrans: 0     0
   Connection Duration: 0:00:00.0047942
   Frame Number(s): 94
TCP: Close Connection          [Normal] Mar 23@14:12:01.760128
   220.100.100.79              <—> 220.100.100.175 SILO
   Port: 1036     NETBIOS 139
   Tx Packets: 4     4
   Low Window: 0               0
   Retrans: 0     0
   Connection Duration: 0:00:00.0114625
   Frame Number(s): 107
TCP: Close Connection          [Normal] Mar 23@14:12:01.7635736
   220.100.100.79              <—> 220.100.100.12
    BRAINS
   Port: 1037     NETBIOS 139
   Tx Packets: 4     3
   Low Window: 0               0
   Retrans: 0     0
   Connection Duration: 0:00:00.0036442
   Frame Number(s): 113
TCP: Close Connection          [Normal] Mar 23@14:12:06.7530549
   220.100.100.79              <—> 220.100.100.61
    THE-HOSTAGE
   Port: 1038     NETBIOS 139
   Tx Packets: 4     3
   Low Window: 0               0
   Retrans: 0     0
```

```
Connection Duration: 0:00:00.0033536
   Frame Number(s): 123
TCP: Close Connection           [Normal] Mar 23@14:12:06.7564388
   220.100.100.77               <—> 220.100.100.79
   Port: NETBIOS 139            1039
   Tx Packets: 4     5
   Low Window: 0                0
   Retrans: 0     0
   Connection Duration: 0:00:53.1002132
   Frame Number(s): 130
TCP: Close Connection           [Normal] Mar 23@14:12:11.7688309
   220.100.100.79               <—> 220.100.100.180
   Port: 1040     NETBIOS 139
   Tx Packets: 4     4
   Low Window: 0                0
   Retrans: 0     0
   Connection Duration: 0:00:00.0113425
   Frame Number(s): 140
TCP: Close Connection           [Normal] Mar 23@14:13:29.096155
   220.100.100.8     <—>
   220.100.100.12
   220.100.100.8     BRAINS
   Port: 1041     NETBIOS 139
   Tx Packets: 11               8
   Low Window: 0                0
   Retrans: 0     0
   Connection Duration: 0:00:02.3622054
   Frame Number(s): 217
```

Low TTL	1	0	0	0	0	
IP Packet Size	18000	0	0	1084		
SNMP Get/Set Pkts		10	0	0	0	0
SNMP Trap Pkts		10	0	0	0	0
DNS Packets	10	0	0	0	0	
ARP Packets	10	0	0	6	38	
Low Window	5	0	0	0	0	
Reset Connections		5	0	0	0	0
Routing Packets	50	0	0	0	0	
Buffer Overwrites		100	0	0	0	0

Start Time: Mar 23 98 @ 14:08:47

Sample Time: Mar 23 98 @ 14:29:13

The information provided in this capture can be used to fine-tune the network and plan for additional network traffic loads.

Many more network *captures* can be obtained with the Internet Advisor. For me, the Internet Advisor is one of the most useful network tools I have ever used. You can obtain more information about the Internet Advisor at www.hp.com or

Hewlett-Packard
5070 Centinneal Blvd.
Colorado Springs, CO 80919

18.21 Network Tape Drive (Sony)

Most networks have multiple devices operating on them. The network I designed and built is no exception. During the planning stage of this network, I determined that a tape drive would be the best solution for backup purposes. Because of the capability of the Sony AIT tape drive, I selected it to be the backup tool for the entire network. Consider Fig. 18-44.

Figure 18-44 shows the tape drive connected to *THE-KID*. This system has an SCSI interface and is also connected to the network. THE-KID is running the NT workstation; consequently it has the tape backup software built into it and requires no further software to perform backups. This particular tape drive is capable of 50 GB of storage. This tape drive uses tape with a chip on it and onboard RAM, which enhances access speed.

I recommend contacting Sony for further information. In addition, I suggest that you consider this type of setup to provide backup storage for your network. Sony can be reached at www.sony.com or

Sony Electronics
3300 Zanker
San Jose, CA 95134

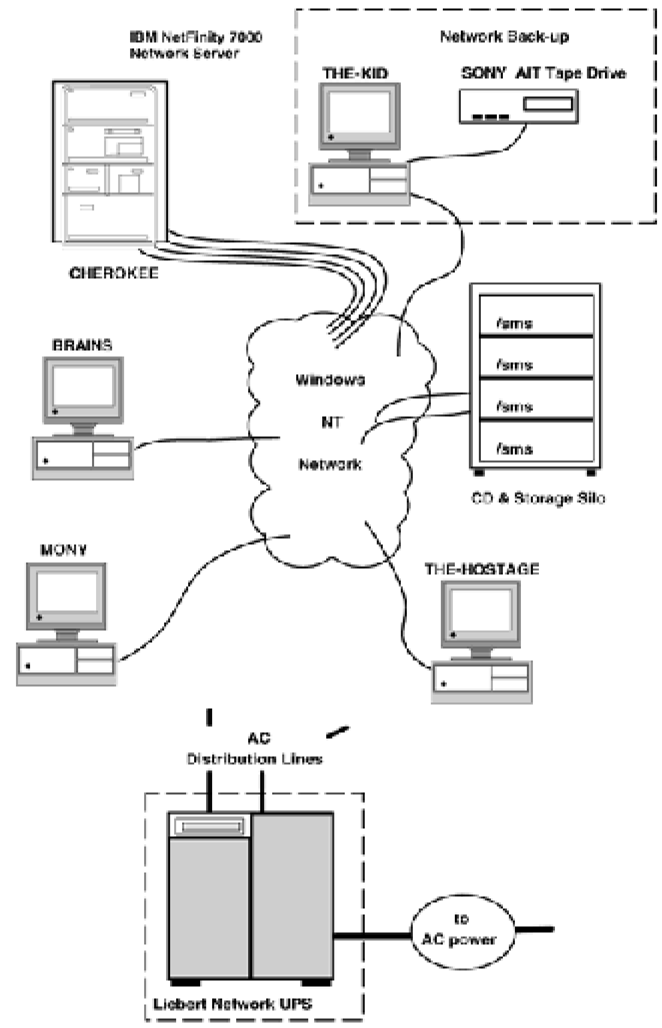


Figure 18-44
Network backup.

18.22 Power Protection (Liebert)

This network involves a lot of equipment. Rather than attempt to use isolated pieces of power protection, I determined that the best approach would be to use an uninterruptible power supply (UPS). I selected Liebert because of their reputation, quality, and product reliability. During the conversation phase with Liebert, prior to determining which UPS is best, I discovered another benefit of Liebert—they have done an excellent job of marrying people and technology. At every turn, my conversations with Liebert personnel provided examples of how to run a company, interact with customers, and build products.

I determined that my network needs could be met well with a 15-kVA UPS. You will need to examine your site information to help you determine what size of UPS you need.

Conceptual Use of the Network UPS

The Liebert UPS I used in this network appears conceptually as shown in Fig. 18-45.

This figure shows the complete network environment. It also shows the Liebert UPS. This UPS provides coverage for the entire network. Consider some of the following specifications for the UPS protecting the network.

Liebert has not only state-of-the-art power protection equipment and qualified personnel but also a wealth of information assimilated to assist current and potential customers, in every way.

Network UPS Details

Additional information about the UPS used in this network includes examination of the LCD display on the front. The panel menu is easy to use and appears as shown in Fig. 18-46.

This interface to the UPS is significant because the design of the menus and the information provided through them is easily understandable by those who are familiar with UPS systems and electrical information.

18.23 Computer Cables and Accessories (Belkin)

Today many computers that operate with laser printers, plotters, and the like require special cables. Cables, gender changes, and other ancillary devices are critical in the connectivity of computers and peripheral devices. This network required considerable cables and accessories.

Because of the significance of these components, I selected Belkin products. I used the following components in this network:

IBM/PC Hayes Modem Gold Premium DB25 F/M, 6 ft
Hayes Modem Gold DB25 F/M, 10 ft
Super VGA Monitor Extension Cable M/M, 10 ft
Serial DB25 M/F, 10 ft

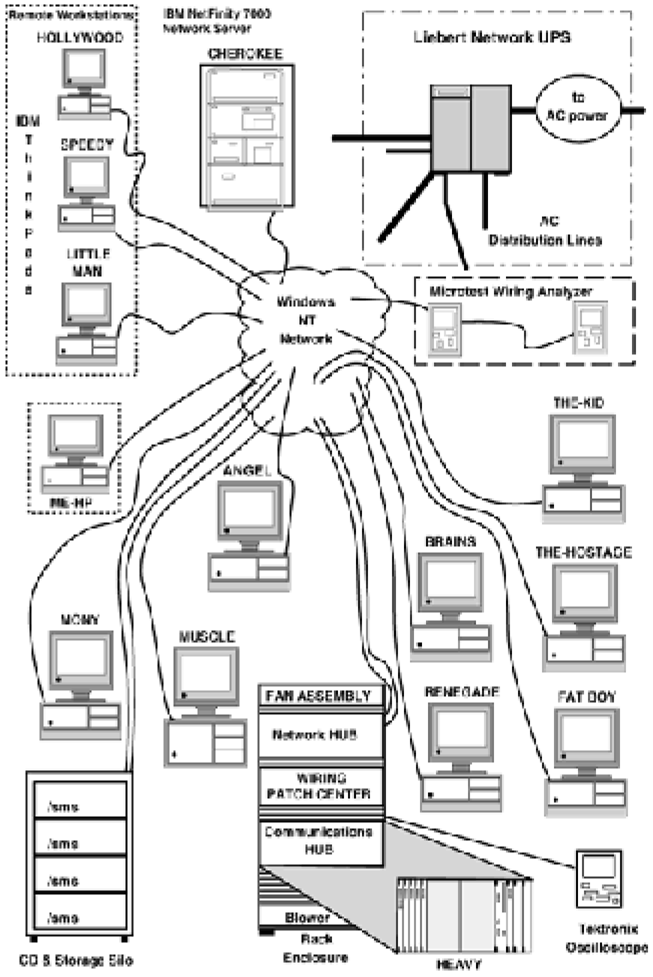


Figure 18-45
Network power protection.

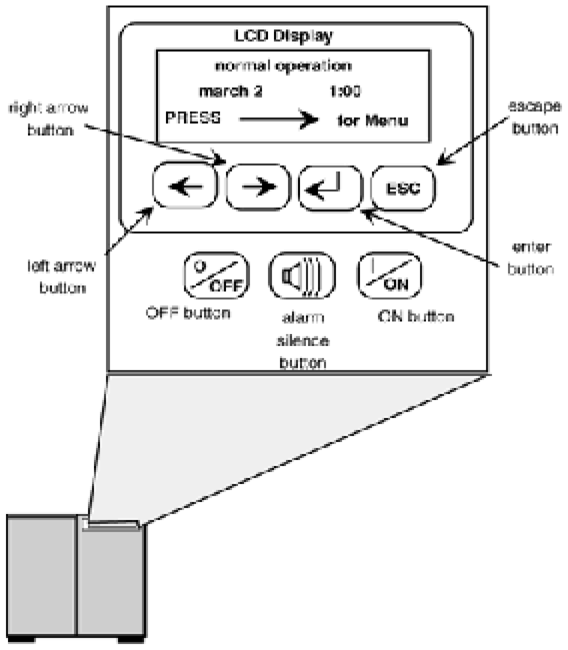


Figure 18-46
Control panel highlighted view.

IEEE 1284 A-B Cable Centronics 36P, 35 ft
IEEE 1284 A-B Cable Centronics 36P, 20 ft
IEEE 1284 A-B Cable Centronics 36P, 20 ft
Gender Changer DB9-F/DB9-F
Gender Changer DB15-M/DB15-M
Gender Changer DB25P/DB25P
SCSI 2 Interface Cable DB50P-M/DB50P-M, 12 ft
Micro DB68P Terminator (Active SCSI)
SCSI-III Differential Terminator DB68P, single-ended

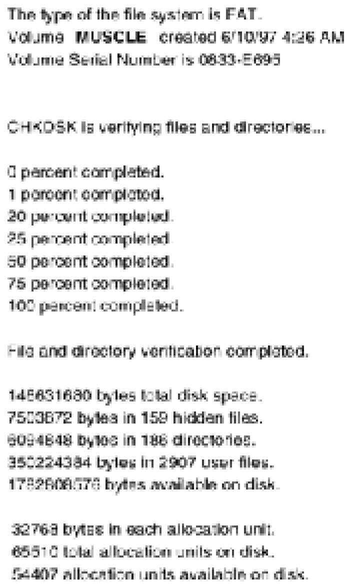
These cables and accessories were critical in installation of the computers and peripherals used in the network. Belkin is probably the most reputable company in the industry today providing cables and accessories. I recommend that you contact them for additional information, at www.belkin.com or

Belkin Components
501 West Walnut Street
Compton, CA 90220

18.24 Software Troubleshooting Tools (Microsoft)

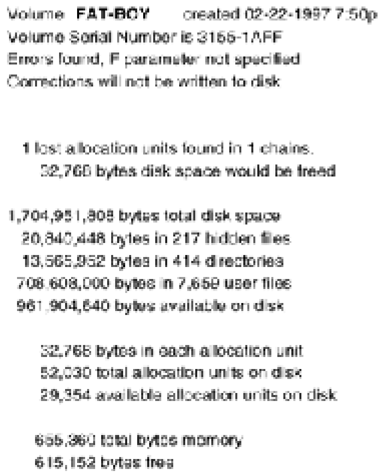
Windows 95 and Windows NT have commands available that can provide valuable information in the troubleshooting of computers and networks. Consider Fig. 18-47.

Figure 18-47 is a diagnostic command used on MUSCLE. The check-disk command reveals the information shown in the figure. If any corrupt files are discovered, a notation will be made and pre sented in the report that the command is capable of generating. Now consider Fig. 18-48.



```
The type of the file system is FAT.
Volume  MUSCLE  created 6/10/97 4:26 AM
Volume Serial Number is 0833-E695


CHKDSK is verifying files and directories...

0 percent completed.
1 percent completed.
20 percent completed.
25 percent completed.
50 percent completed.
75 percent completed.
100 percent completed.

File and directory verification completed.

145631680 bytes total disk space.
7503872 bytes in 159 hidden files.
6094848 bytes in 186 directories.
350224384 bytes in 2907 user files.
1782808576 bytes available on disk.

 32768 bytes in each allocation unit.
 65510 total allocation units on disk.
 54407 allocation units available on disk.
```

Figure 18-47
Diagnostic information for MUSCLE.



```
Volume  FAT-BOY      created 02-22-1997 7:50p
Volume Serial Number is 3166-1AFF
Errors found, F parameter not specified
Corrections will not be written to disk


  1 lost allocation units found in 1 chains.
     32,768 bytes disk space would be freed

1,704,951,808 bytes total disk space
   20,840,448 bytes in 217 hidden files
   13,565,952 bytes in 414 directories
  708,608,000 bytes in 7,659 user files
  961,904,640 bytes available on disk

     32,768 bytes in each allocation unit
     52,030 total allocation units on disk
     29,354 available allocation units on disk

    655,360 total bytes memory
    615,152 bytes free
```

Figure 18-48
Windows diagnostics for FAT-BOY.

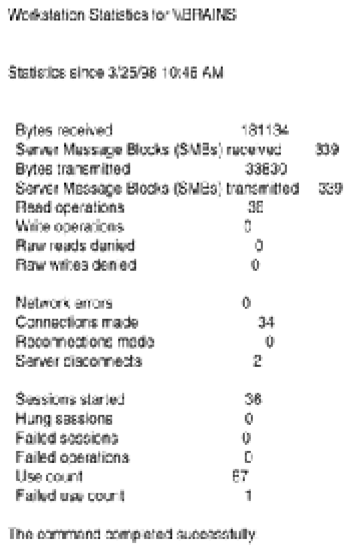
This figure shows the results of the scanned disk function that is a part of the Windows program. It is used for disk maintenance. Notice that FAT-BOY has 1 lost allocation unit found in 1 chain. If this lost allocation unit were corrected, an additional 32,768 bytes would be freed for use. Now consider Fig. 18-49.

Figure 18-49 shows the Windows NT NETstat statistics command issued against `\\BRAINS.` The result is shown in the number of bytes received from other workstations, the number of server message blocks received and transmitted, the number of connections made, and other information. Now consider Fig. 18-50.

Figure 18-50 shows the NETstat server command issued against `\\MUSCLE.` Note those workstations connected.

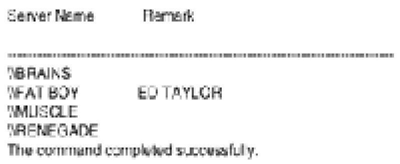
Figure 18-51 shows the NET SHARE command issued against `\\MUSCLE.` Note the *shares* available on the system. This identifies those shares that can be accessed by users throughout the network.

A host of other commands can be used at the command line within Windows NT and Windows-based computers. Software commands can be very helpful in diagnosing network problems. The trick is to understand the information available from these commands. Many additional commands are available in Windows NT and Windows. Take time to review the documentation with the software and explore this aspect of system support.



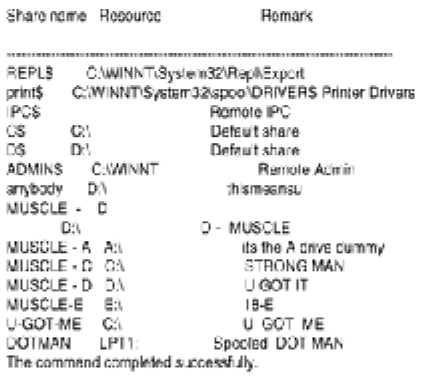
```
Workstation Statistics for \\BRAINS


Statistics since 3/25/98 10:46 AM


 Bytes received                        181134
 Server Message Blocks (SMBs) received       309
 Bytes transmitted                      33630
 Server Message Blocks (SMBs) transmitted    339
 Read operations                        36
 Write operations                      0
 Raw reads denied                        0
 Raw writes denied                      0

 Network errors                        0
 Connections made                        34
 Reconnections made                       0
 Server disconnects                     2

 Sessions started                      36
 Hung sessions                         0
 Failed sessions                       0
 Failed operations                     0
 Use count                           67
 Failed use count                      1

The command completed successfully.
```

Figure 18-49
Windows NT NETstat command.



```
Server Name          Remark

-------------------------------------------------------------------------------
\\BRAINS
\\FAT BOY            ED TAYLOR
\\MUSCLE
\\RENEGADE
The command completed successfully.
```

Figure 18-50
Windows NT NET command.



```
Share name   Resource                    Remark

-------------------------------------------------------------------
REPL$       C:\WINNT\System32\Repl\Export
print$      C:\WINNT\System32\spool\DRIVERS Printer Drivers
IPC$                              Remote IPC
C$          C:\                   Default share
D$          D:\                   Default share
ADMIN$      C:\WINNT                       Remote Admin
anybody     D:\                      thismeansu
MUSCLE  -   D
         D:\                     D -  MUSCLE
MUSCLE - A   A:\                        its the A drive dummy
MUSCLE - C   C:\                        STRONG MAN
MUSCLE - D   D:\                        U GOT IT
MUSCLE-E     E:\                        18-E
U-GOT-ME     C:\                        U  GOT  ME
DOTMAN      LPT1:              Spooled  DOT MAN
The command completed successfully.
```

Figure 18-51
Windows NT NET SHARE command.

18.25 Miscellaneous Devices and Tools

When you get into the design of your network, you may realize what I did; some things you will need have to be looked for extensively or created.

1. *External CD-ROM.* In this network there was a need for an external CD-ROM that could connect via SCSI connection to the USRobotics hub, laptops, and Hewlett-Packard Internet Advisor. Sometimes finding "things" to make the network run is like the proverbial witch hunt. I selected a Sony CD-ROM, model PRD-650WN. Whether this particular device will be available as you read this will be a different story. However, should this specific device not be available, I suppose some vendor will make an SCSI attach CD-ROM available. This device works well. The technology is genuine Sony and Adaptec for the SCSI PCMCIA card. This device is a must-have for network installers and administrators.

2. *Wire testers.* Another device you will need is at least one cable tester. You need to be able to check ac cables, data cables of all sorts, RJ-45 wire, RJ-11 wire, RJ-11 wall connectors, and so forth. *Do not* assume that a particular component that is new is good. I have seen perfectly "new" equipment come right out of a vendor's box bad (defective). No equipment or any type of cable in this network was discovered to be "bad," but that does not mean you should not test cables and connection points. I selected Fluke meters to provide me with ac test ability. This same equipment can be used to obtain amperage readings, cable test, and so on. I ordered two devices to specifically check cable: one was for RJ-45 and the other for RJ-11 wire.

3. *Breakout AC test cable.* I make my own extension cords. Many ask why. If you saw them you would know the reason. Any single one of them could be used to supply voltage and current to multiple dryers, ovens, or other high-amperage devices. Electricity demands respect; you can give it or it will be taken. It does not matter whether my breakout ac test wire is UL-listed; they work. What I recommend is this: Buy some 10-gauge SO-type wire with two conductors with ground. Then connect a regular 110-V three-prong connector on one end. On the other end put one or two pair(s) of regular electrical outlets in a metal box with enclosed top. Carefully, take about 6 to 8 in of the outer sheath off the conductors. Three conductors should be exposed with insulation. Tape these ends where the cuts were made and presto! Now you can literally clamp an amp(lifier) probe on around the hot wire without going into any danger areas and get the amperage pull off the line. Am I recommending that you do this? Absolutely not. I'm telling you what and how I did it. This level of information assisted me in the design phase of ground floor of the network. I had to go back and draw four clean lines to the breaker box to accommodate the new equipment.

Again, you better make it your business to know the electrical part of your network. If you do not have the training and background for this, then get somebody who does. You cannot afford to leave this base uncovered.

19
Domain Name Usage

The original motive for the development of the domain system was growth in the Internet, and the following were considerations:

1. Host name-address mappings were maintained by the network information center (NIC) in a single file (`HOSTS.TXT`) which was FTPed by all hosts (RFC 952, RFC 953).

2. The bandwidth consumed in distributing a new version by this scheme is proportional to the square of the number of hosts in the network, and even when multiple levels of FTP are used, the outgoing FTP load on the NIC host is considerable. Explosive growth in the number of hosts didn't lend itself to ease of distribution.

3. The network population was also changing in character.

4. The timeshared hosts that made up the original ARPAnet were being replaced with local networks of workstations.

5. Local organizations were administering their own names and addresses, but had to wait for the NIC to change `HOSTS.TXT` to make changes visible to the Internet at large. Organizations also wanted some local structure on the namespace.

6. The applications on the Internet were getting more sophisticated and creating a need for general-purpose name service.

7. The result was several ideas about namespaces and their management (IEN-116, RFCs 799, 819, 830). Proposals varied, but a common thread was the idea of a hierarchical namespace, with the hierarchy roughly corresponding to organizational structure, and names using "." as the character to mark the boundary between hierarchy levels. A design using a distributed database and generalized resources was described in RFCs 882 and 883. On the basis of experience with several implementations, the system evolved into the scheme described in this memo.

The terms *domain* or *domain name* are used in many contexts beyond the DNS explained in this text. Often, the latter term refers to a name with structure indicated by dots, but in no relation to the DNS.

19.1 DNS Design Goals

The following design goals of the DNS influence its structure:

1. The primary goal is a consistent namespace which will be used for referring to resources. To avoid the problems caused by ad hoc encodings, names should not be required to contain network identifiers, addresses, routes, or similar information as part of the name.

2. The sheer size of the database and frequency of updates suggest that it must be maintained in a distributed manner, with local caching to improve performance. Approaches that attempt to collect a consistent copy of the entire database will become increasingly expensive and difficult, and hence should be avoided. The same principle holds for the structure of the namespace, and in particular mechanisms for creating and deleting names; these should also be distributed.

3. Where there are tradeoffs between the cost of acquiring data, the speed of updates, and the accuracy of caches, the source of the data should control the tradeoff.

4. The costs of implementing such a facility dictate that it be generally useful, and not restricted to a single application. We should be able to use names to retrieve host addresses, mailbox data, and other as-yet undetermined information. All data associated with a name are tagged with a type, and queries can be limited to a single type.

5. Because the purpose is to use the naming wisely, with dissimilar networks and applications, we provide the ability to use the same namespace with different protocol families or management. For example, host address formats differ between protocols, although all protocols have the notion of address. The DNS tags all data with a class as well as the type, so that we can allow parallel use of different formats for data of type address.

6. We want name server transactions to be independent of the communications system that carries them. Some systems may wish to use datagrams for queries and responses, and only establish virtual circuits for transactions that need the reliability (e.g., database updates, long transactions); other systems will use virtual circuits exclusively. The system should be useful across a wide spectrum of host capabilities. Both personal computers and large timeshared hosts should be able to use the system, although perhaps in different ways.

19.2 Assumptions about DNS Usage

The organization of the domain system derives from some assumptions about the needs and usage patterns of its user community and is designed to avoid many of the complicated problems found in general-purpose database systems. Assumptions here include the following:

1. The size of the total database will initially be proportional to the number of hosts using the system, but will eventually grow to be proportional to the number of users on those hosts as mailboxes and other information are added to the domain system.

2. Most of the data in the system will change very slowly (e.g., mailbox bindings, host addresses), but that the system should be able to deal with subsets that change more rapidly (on the order of seconds or minutes).

3. The administrative boundaries used to distribute responsibility for the database will usually correspond to organizations that have one or more hosts. Each organization that has responsibility for a particular set of domains will provide redundant name servers, either on the organization’s own hosts or on other hosts that the organization arranges to use.

4. Clients of the domain system should be able to identify trusted name servers they prefer to use before accepting referrals to name servers outside of this “trusted” set.

5. Access to information is more critical than instantaneous updates or guarantees of consistency. Hence the update process allows updates to percolate out through the users of the domain system rather than guaranteeing that all copies are simultaneously updated. When updates are unavailable as a result of network or host failure, the usual course is to believe old information while continuing efforts to update it. The general model is that copies are distributed with timeouts for refreshing. The distributor sets the timeout value, and the recipient of the distribution is responsible for performing the refresh. In special situations, very short intervals can be specified, or the owner can prohibit copies.

6. In any system that has a distributed database, a particular name server may be presented with a query that can be answered only by some other server. The two general approaches to dealing with this problem are *recursive*, meaning that the first server pursues the query for the client at another server, and *iterative*, indicating that the server refers the client to another server and lets the client pursue the query. Both approaches have advantages and disadvantages, but the iterative approach is preferred for the datagram style of access. The domain system requires implementation of the iterative approach, but allows the recursive approach as an option.

7. The domain system assumes that all data originate in master files scattered through the hosts that use the domain system. These master files are updated by local system administrators. Master files are text files that are read by a local name server, and hence become available through the name servers to users of the domain system. The user programs access name servers through standard programs called *resolvers.*

8. The standard format of master files allows them to be exchanged between hosts (via FTP, mail, or some other mechanism); this facility is useful when an organization wants a domain but doesn't want to support a name server. The organization can maintain the master files locally using a text editor, transfer them to a foreign host which runs a name server, and then arrange with the system administrator of the name server to get the files loaded.

9. Each host's name servers and resolvers are configured by a local system administrator (RFC 1033). For a name server, these configuration data include the identity of local master files and instructions on which nonlocal master files are to be loaded from foreign servers. The name server uses the master files or copies to load its zones. For resolvers, the configuration data identify the name servers which should be the primary sources of information.

10. The domain system defines procedures for accessing the data and for referrals to other name servers. The domain system also defines procedures for caching retrieved data and for periodic refreshing of data defined by the system administrator. *System administrators* provide the definition of zone boundaries, master files of data, updates to master files, and statements of the refresh policies desired. The *domain system* provides standard formats for resource data, standard methods for querying the database, and standard methods for name servers to refresh local data from foreign name servers.

19.3 Elements of DNS

The DNS has three major components: domain namespace and resource records, name servers, and resolvers.

1. *Domain namespace and resource records.* These are specifications for a tree-structured namespace and data associated with the names. Conceptually, each node and leaf of the domain namespace tree names a set of information, and query operations are attempts to extract specific types of information from a particular set. A query names the domain name of interest and describes the type of resource information that is desired. For example, the Internet uses some of its domain names to identify hosts; queries for address resources return Internet host addresses.

2. *Name servers.* These are server programs which hold information about the domain tree's structure and set information. A name server may cache structure or set information about any part of the domain tree, but in general a particular name server has complete information about a subset of the domain space, and pointers to other name servers that can be used to lead to information from any part of the domain tree. Name servers know the parts of the domain tree for which they have complete information; a name server is said to be an authority for these parts of the namespace. Authoritative information is organized into units called *zones,* and these zones can be automatically distributed to the name servers which provide redundant service for the data in a zone.

3. *Resolvers.* These are programs that extract information from name servers in response to client requests. Resolvers must be able to access at least one name server and use that name server's information to answer a query directly, or pursue the query using referrals to other name servers. A resolver will typically be a system routine that is directly accessible to user programs; hence no protocol is necessary between the resolver and the user program.

These three components roughly correspond to the three layers or views of the domain system. Consider a detailed explanation here that correlates to this.

• From the user's point of view, the domain system is accessed through a simple procedure or OS call to a local resolver.

• The domain space consists of a single tree, and the user can request information from any section of the tree.

• From the resolver's point of view, the domain system is composed of an unknown number of name servers. Each name server has one or more pieces of the whole domain tree's data, but the resolver views each of these databases as essentially static.

• From a name server's point of view, the domain system consists of separate sets of local information called *zones*. The name server has local copies of some of the zones. The name server must periodically refresh its zones from master copies in local files or foreign name servers. The name server must concurrently process queries that arrive from resolvers.

In the interests of performance, implementations may couple these functions. For example, a resolver on the same machine as a name server might share a database consisting of the zones managed by the name server and the cache managed by the resolver.

19.4 Domain Namespace and Resource Records

The domain namespace is a tree structure. Each node and leaf on the tree corresponds to a resource set (which may be empty). The domain system makes no distinctions between the uses of the interior nodes and leaves, and this memo uses the term *node* to refer to both. Each node has a label, which is zero to 63 octets in length. Brother nodes may not have the same label, although the same label can be used for nodes which are not brothers. One label is reserved, and that is the null (i.e., zero length) label used for the root.

The domain name of a node is the list of the labels on the path from the node to the root of the tree. By convention, the labels that compose a domain name are printed or read left to right, from the most specific (lowest, farthest from the root) to the least specific (highest, closest to the root).

Internally, programs that manipulate domain names should represent them as sequences of labels, where each label is a length octet followed by an octet string. Because all domain names end at the root, which has a null string for a label, these internal representations can use a length byte of zero to terminate a domain name.

By convention, domain names can be stored with arbitrary case, but domain name comparisons for all present domain functions are done in a case-insensitive manner, assuming an ASCII character set, and a high-order zero bit. This means that you are free to create a node with label "A" or a node with label "a," but not both as brothers; you could refer to either using "a" or "A." When you receive a domain name or label, you should preserve its case. The rationale for this choice is that we may someday need to add full binary domain names for new services; existing services would not be changed.

When a user needs to type a domain name, the length of each label is omitted and the labels are separated by dots ("."). Since a complete domain name ends with the root label, this leads to a printed form which ends in a dot. We use this property to distinguish between (1) a character string which represents a complete domain name (often called *absolute*), for example, `joejones.ISI.EDU`; and (2) a character string that represents the starting labels of a domain name which is incomplete, and should be completed by local software using knowledge of the local domain (often called *relative*). For example, `joejones` used in the `ISI.EDU domain.`

Relative names are either taken relative to a well-known origin, or to a list of domains used as a search list. Relative names appear mostly at the user interface, where their interpretation varies from implementation to implementation, and in master files, where they are relative to a single-origin domain name. The most common interpretation uses the root "." as either the single origin or as one of the members of the search list, so a multilabel relative name is often one where the trailing dot has been omitted to save typing.

To simplify implementations, the total number of octets that represent a domain name (i.e., the sum of all label octets and label lengths) is limited to 255. A domain is identified by a domain name, and consists of that part of the domain namespace that is at or below the domain name which specifies the domain. A domain is a subdomain of another domain if it is contained within that domain. This relationship can be tested by seeing if the subdomain's name ends with the containing domain's name. For example, `A.B.C.D` is a subdomain of `B.C.D, C.D, D,` and "."

DNS technical specifications do not mandate a particular tree structure or rules for selecting labels; its goal is to be as general as possible, so that it can be used to build arbitrary applications. In particular, the system was designed so that the namespace did not have to be organized along the lines of network boundaries, name servers, and so on. The rationale for this is not that the namespace should have no implied semantics, but rather that the choice of implied semantics should be left open to be used for the problem at hand, and that different parts of the tree can have different implied semantics. For example, the `IN-ADDR.ARPA` domain is organized and distributed by network and host address because its role is to translate from network or host numbers to names; NetBIOS domains (RFC 1001, RFC 1002) are flat because that is appropriate for that application.

However, there are some guidelines that apply to the "normal" parts of the namespace used for hosts, mailboxes, and so forth, that will make the namespace more uniform, provide for growth, and minimize problems as software is converted from the older host table. The political decisions about the top levels of the tree originated in RFC 920. Current policy for the top levels is discussed in RFC 1032. MILNET conversion issues are covered in RFC 1031.

Lower domains which will eventually be broken into multiple zones should provide branching at the top of the domain so that the eventual decomposition can be done without renaming. Node labels which use special characters, leading digits, or other features are likely to break older software which depends on more restrictive choices.

Before the DNS can be used to hold naming information for some kind of object, two needs must be met: (1) a convention for mapping between object names and domain names (this describes how information about an object is accessed); and (2) resource record (RR) types and data formats for describing the object.

These rules can be quite simple or fairly complex. Very often, the designer must take into account existing formats and plan for upward compatibility for existing usage. Multiple mappings or levels of mapping may be required. For hosts, the mapping depends on the existing syntax for host names which is a subset of the usual text representation for domain names, together with RR formats for describing host addresses, etc. Because we need a reliable inverse mapping from address to host name, a special mapping for addresses into the `IN-ADDR.ARPA` domain is also defined.

For mailboxes, the mapping is slightly more complex. The usual mail address <local-part>@<mail-domain> is mapped into a domain name by converting <local-part> into a single label (regardless of the dots it contains), converting <mail-domain> into a domain name using the usual text format for domain names (dots denote label breaks), and concatenating the two to form a single domain name. Thus the mailbox `HOSTMASTER@SRI-NIC.ARPA` is represented as a domain name by `HOSTMASTER.SRI-NIC.ARPA.` An appreciation for the reasons behind this design also must take into account the scheme for mail exchanges (RFC 974).

A typical user is not concerned with defining these rules, but should understand that they usually are the result of numerous compromises between desires for upward compatibility with old usage, interactions between different object definitions, and the inevitable urge to add new features when defining the rules. The way the DNS is used to support some object is often more crucial than the restrictions inherent in the DNS.

Figure 19-1 shows a part of the current domain namespace, and is used in many examples in this RFC. Note that the tree is a very small subset of the actual namespace.

In this example, the root domain has three immediate subdomains: MIL, EDU, and ARPA. The `LCS.MIT.EDU` domain has one immediate subdomain named `XX.LCS.MIT.EDU.` All the leaves are also domains.

19.5 DNS Name Syntax

The DNS specifications attempt to be as general as possible in the rules for constructing domain names. The idea is that the name of any existing object can be expressed as a domain name with minimal changes. However, when assigning a domain name for an object, the prudent user will select a name which satisfies both the rules of the domain system and any existing rules for the object, whether these rules are pub lished or implied by existing programs. For example, when naming a mail domain, the user should satisfy both the rules of this memo and those in RFC 822. When creating a new host name, the old rules for `HOSTS.TXT` should be followed. This avoids problems when old software is converted to use domain names.

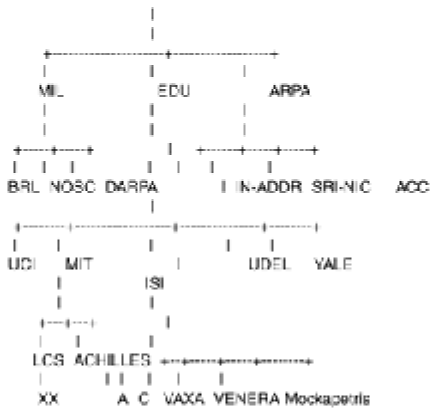


Figure 19-1
Part of the current domain namespace.

The following syntax will result in fewer problems with many applications that use domain names (e.g., mail, TELNET):

<domain> :: = <subdomain> | "
<subdomain> :: = <label> | <subdomain> "."<label>
<label> :: = <letter> [[<ldh-str>] <let-dig>]
<ldh-str> :: = <let-dig-hyp> | <let-dig-hyp> <ldh-str>
<let-dig-hyp> :: = <let-dig> | "-"
<let-dig> :: = <letter> | <digit>
<letter> :: = any one of the 52 alphabetic characters A through Z in upper case and a through z in lower case
<digit> :: = any one of the ten digits 0 through 9

While upper- and lowercase letters are allowed in domain names, no significance is attached to the case; that is, two names with the same spelling but different case are to be treated as if they are identical.

The labels must follow the rules for ARPAnet host names. They must start with a letter, end with a letter or digit, and have as interior characters only letters, digits, and hyphen. There are also some restrictions on the length. Labels must be 63 characters or less.

For example, the following strings identify hosts in the Internet:

A.ISI.EDU XX.LCS.MIT.EDU SRI-NIC.ARPA

A domain name identifies a node. Each node has a set of resource information, which may be empty. The set of resource information associated with a particular name is composed of separate RRs. The order of RRs in a set is not significant, and need not be preserved by name servers, resolvers, or other parts of the DNS.

When we talk about a specific RR, we assume that it has the following: *owner*—which is the domain name where the RR is found; and *type*—which is an encoded 16-bit value that specifies the type of the resource in this resource record. Types refer to abstract resources.

19.6 DNS Queries

Queries are messages which may be sent to a name server to provoke a response. In the Internet, queries are carried in UDP datagrams or over TCP connections. The response by the name server either answers the question posed in the query, refers the requester to another set of name servers, or signals some error condition.

In general, the user does not generate queries directly, but instead makes a request to a resolver which, in turn, sends one or more queries to name servers and deals with the error conditions and referrals that may result. Of course, the possible questions which can be asked in a query do shape the kind of service a resolver can provide.

DNS queries and responses are carried in a standard message format. The message format has a header containing a number of fixed fields which are always present, and four sections which carry query parameters and RRs.

The most important field in the header is a 4-bit field called an *opcode,* which separates different queries. Of the possible 16 values, one (standard query) is part of the official protocol, two (inverse query and status query) are options, one (completion) is obsolete, and the rest are unassigned.

The four sections are (1) *question*—carries the query name and other query parameters; (2) *answer*—carries RRs which directly answer the query; (3) *authority*—carries RRs which describe other authoritative servers (may optionally carry the SOA RR for the authoritative data in the answer section); and (4) *additional*—carries RRs which may be helpful in using the RRs in the other sections.

A *standard* query specifies a target domain name (`QNAME`), query type (`QTYPE`), and query class (`QCLASS`) and asks for RRs which match. This type of query makes up such a vast majority of DNS queries that we use the term *query* to mean standard query unless otherwise specified. The `QTYPE` and `QCLASS` fields are each 16 bits long, and are a superset of defined types and classes.

The `QTYPE` field may contain

`<any type>`	Matches just that type. (e.g., A, PTR)
`AXFR`	Special zone transfer QTYPE
`MAILB`	Matches all mailbox related RRs (e.g. MB and MG)
`*`	Matches all RR types

The `QCLASS` field may contain

`<any class>`	Matches just that class (e.g., IN, CH).
	Matches all RR classes

The query domain name, `QTYPE`, and `QCLASS`, uses the name server to look for matching RRs. In addition to relevant records, the name server may return RRs that point toward a name server that has the desired information or RRs that are expected to be useful in interpreting the relevant RRs. For example, a name server that doesn’t have the requested information may know a name server that does; a name server that returns a domain name in a relevant RR may also return the RR that binds that domain name to an address.

For example, a mailer tying to send mail to `joejones@ISI.EDU` might ask the resolver for mail information about `ISI.EDU`, resulting in a query for `QNAME = ISI.EDU, QTYPE = MX, QCLASS = IN`. The response’s answer section would be

ISI.EDU.	MX	10	VENERA.ISI.EDU.
	MX	10	VAXA.ISI.EDU.

An additional section might be

VAXA.ISI.EDU. A	10.2.0.27	
	A	128.9.0.33
VENERA.ISI.EDU.	A	10.1.0.52
	A	128.9.0.32

Because the server assumes that if the requester wants mail exchange information, it will probably want the addresses of the mail exchanges soon afterward. The `QCLASS = *` construct requires special interpretation regarding authority. Since a particular name server may not know all of the classes available in the domain system, it can never know if it is authoritative for all classes. Hence responses to `QCLASS = *` queries can never be authoritative.

19.7 DNS Name Servers

Name servers are the repositories of information that make up the domain database. The database is divided up into sections called zones, which are distributed among the name servers. While name servers can have several optional functions and sources of data, the essential task of a name server is to answer queries using data in its zones. By design, name servers can answer queries in a simple manner; the response can always be generated using only local data, and either contains the answer to the question or a referral to other name servers “closer” to the desired information.

A given zone will be available from several name servers to insure its availability in spite of host or communication link failure. By administrative fiat, we require every zone to be available on at least two servers, and many zones have more redundancy than that.

A given name server will typically support one or more zones, but this gives it authoritative information about only a small section of the domain tree. It may also have some cached nonauthoritative data about other parts of the tree. The name server marks its responses to queries so that the requester can tell whether the response comes from authoritative data.

19.7.1 DNS Database Zone Division

The domain database is partitioned in one of two ways: by class, and by “cuts” made in the name space between nodes. The class partition is simple. The database for any class is organized, delegated, and maintained separately from all other classes. Since, by convention, the namespaces are the same for all classes, the separate classes can be thought of as an array of parallel namespace trees. Note that the data attached to nodes will be different for these different parallel classes. The most common reasons for creating a new class are the necessity for a new data format for existing types or a desire for a separately managed version of the existing namespace. Within a class, cuts in the namespace can be made between any two adjacent nodes. After all cuts are made, each group of connected namespace is a separate zone. The zone is said to be authoritative for all names in the connected region. Note that the cuts in the namespace may be in different places for different classes or the name servers may be different, for example. This means that every zone has at least one node, and hence domain name, for which it is authoritative, and all the nodes in a particular zone are connected. Given the tree structure, every zone has a highest node which is closer to the root than any other node in the zone. The name of this node is often used to identify the zone.

It would be possible, although not particularly useful, to partition the namespace so that each domain name was in a separate zone or so that all nodes were in a single zone. Instead, the database is partitioned at points where a particular organization wants to take over control of a subtree. Once an organization controls its own zone it can unilaterally change the data in the zone, grow new tree sections connected to the zone, delete existing nodes, or delegate new subzones under its zone.

If the organization has substructure, it may want to make further internal partitions to achieve nested delegations of namespace control. In some cases, such divisions are made purely to make database maintenance more convenient.

The data that describe a zone have four major parts: (1) authoritative data for all nodes within the zone; (2) data that define the top node of the zone (can be thought of as part of the authoritative data); (3) data that describe delegated subzones, that is, cuts around the bottom of the zone; and (4) data that allow access to name servers for subzones, sometimes called “glue” data. All of these data are expressed in the form of RRs, so a zone can be completely described in terms of a set of RRs. Whole zones can be transferred between name servers by transferring the RRs, either carried in a series of messages or by FTPing a master file which is a textual representation. The authoritative data for a zone is simply all of the RRs attached to all of the nodes from the top node of the zone down to leaf nodes or nodes above cuts around the bottom edge of the zone.

Although logically part of the authoritative data, the RRs that describe the top node of the zone are especially important to the zone’s management. These RRs are of two types: name server RRs that list, one per RR, all the servers for the zone; and a single SOA RR that describes zone-management parameters. The RRs that describe cuts around the bottom of the zone are NS RRs that name the servers for the subzones. Since the cuts are between nodes, these RRs are *not* part of the authoritative data of the zone, and should be exactly the same as the corresponding RRs in the top node of the subzone. Since name servers are always associated with zone boundaries, NS RRs are only found at nodes which are the top node of some zone.

The data that makes up a zone, NS RRs, are found at the top node of the zone (and are authoritative) and at cuts around the bottom of the zone (where they are not authoritative), but never in between. One of the goals of the zone structure is that any zone have all the data required to set up communications with the name servers for any subzones; that is, parent zones have all the information needed to access servers for their children zones. The NS RRs that name the servers for subzones are often not enough for this task since they name the servers, but do not give their addresses. In particular, if the name of the name server is itself in the subzone, we could be faced with the situation where the NS RRs tell us that in order to learn a name server’s address, we should contact the server using the address we wish to learn. To fix this problem, a zone contains “glue” RRs which are not part of the authoritative data, and are address RRs for the servers. These RRs are only necessary if the name server’s name is “below” the cut, and are only used as part of a referral response.

19.7.2 DNS Administration Considerations

When some organization wants to control its own domain, the first step is to identify the proper parent zone, and get the parent zone’s owners to agree to the delegation of control. While there are no particular technical constraints dealing with where in the tree this can be done, there are some administrative groupings discussed in RFC 1032 which deal with top-level organization, and middle-level zones are free to create their own rules. For example, one university might choose to use a single zone, while another might choose to organize by subzones dedicated to individual departments or schools. RFC 1033 catalogs available DNS software and discusses administration procedures.

Once the proper name for the new subzone is selected, the new owners should be required to demonstrate redundant name server support. Note that there is no requirement that the servers for a zone reside in a host which has a name in that domain. In many cases, a zone will be more accessible to the Internet at large if its servers are widely distributed rather than being within the physical facilities controlled by the same organization that manages the zone. For example, in the current DNS, one of the name servers for the United Kingdom, or U.K. domain, is found in the United States. This allows U.S. hosts to obtain U.K. data without using limited transatlantic bandwidth.

As the last installation step, the delegation NS RRs and glue RRs necessary to make the delegation effective should be added to the parent zone. The administrators of both zones should ensure that the NS and glue RRs which mark both sides of the cut are consistent and remain so.

The principal activity of name servers is to answer standard queries. Both the query and its response are carried in a standard message format which is described in RFC 1035. The query contains a `QTYPE, QCLASS`, and `QNAME`, which describe the types and classes of desired information and the name of interest. The way that the name server answers the query depends on whether it is operating in recursive mode:

• The simplest mode for the *server* is nonrecursive, since it can answer queries using only local information: the response contains an error, the answer, or a referral to some other server "closer" to the answer. All name servers must implement nonrecursive queries.

• The simplest mode for the *client* is recursive, since in this mode the name server acts in the role of a resolver and returns either an error or the answer, but never referrals. This service is optional in a name server, and the name server may also choose to restrict the clients which can use recursive mode.

Recursive service is helpful in several situations: (1) a relatively simple requester that lacks the ability to use anything other than a direct answer to the question, (2) a request that needs to cross protocol or other boundaries and can be sent to a server which can act as intermediary, or (3) a network where we want to concentrate the cache rather than having a separate cache for each client. *Nonrecursive* service is appropriate if the requester is capable of pursuing referrals and interested in information which will aid future requests. The use of recursive mode is limited to cases where both the client and the name server agree to its use. The agreement is negotiated through the use of 2 bits in query and response messages:

1. The recursion available (RA) bit is set or cleared by a name server in all responses. The bit is true if the name server is willing to provide recursive service for the client, regardless of whether the client requested recursive service. That is, RA signals availability rather than use.

2. Queries contain a bit called recursion desired (RD). This bit specifies whether the requester wants recursive service for this query. Clients may request recursive service from any name server, although they should depend on receiving it only from servers which have previously sent an RA, or servers which have agreed to provide service through private agreement or some other means outside of the DNS protocol.

The recursive mode occurs when a query with RD set arrives at a server which is willing to provide recursive service; the client can verify that recursive mode was used by checking that both RA and RD are set in the reply. Note that the name server should never perform recursive service unless asked via RD, since this interferes with trouble- shooting of name servers and their databases.

If recursive service is requested and available, the recursive response to a query will be one of the following: (1) the answer to the query, possibly prefaced by one or more `CNAME` RRs that specify aliases encountered on the way to an answer; (2) a name error indicating that the name does not exist—this may include `CNAME` RRs that indi cate that the original query name was an alias for a name which does not exist; or (3) a temporary error indication.

If recursive service is not requested or is not available, the nonrecursive response will be one of the following: (1) an authoritative name error indicating that the name does not exist; (2) a temporary error indication; or (3) some combination of (*a*) RRs that answer the question, together with an indication whether the data come from a zone or are cached, (*b*) a referral to name servers which have zones which are closer ancestors to the name than the server sending the reply, or (*c*) RRs that the name server thinks will prove useful to the requester.

19.8 DNS Resolvers

Resolvers are programs that interface user programs to domain name servers. In the simplest case, a resolver receives a request from a user program (e.g., mail programs, TELNET, FTP) in the form of a subroutine call, system call, or other call, and returns the desired information in a form compatible with the local host's data formats.

The resolver is located on the same machine as the program that requests the resolver's services, but it may need to consult name servers on other hosts. Because a resolver may need to consult several name servers, or may have the requested information in a local cache, the amount of time that a resolver will take to complete can vary quite a bit, from milliseconds to several seconds.

One important goal of the resolver is to eliminate network delay and name server load from most requests by answering them from its cache of prior results. It follows that caches which are shared by multiple processes, users, machines, and so on are more efficient than nonshared caches.

The client-resolver interface to the resolver is influenced by the local host's conventions, but the typical resolver-client interface has three functions:

1. *Host name-host address translation.* This function is often defined to mimic a previous `HOSTS.TXT`-based function. Given a character string, the caller wants one or more 32-bit IP addresses. Under the DNS, it translates into a request for type A RRs. Since the DNS does not preserve the order of RRs, this function may choose to sort the returned addresses or select the "best" address if the service returns only one choice to the client. Note that a multiple address return is recommended, but a single address may be the only way to emulate prior `HOSTS.TXT` services.

2. *Host address-host name translation.* This function will often follow the form of previous functions. Given a 32-bit IP address, the caller wants a character string. The octets of the IP address are reversed, used as name components, and suffixed with `IN-ADDR.ARPA`. A type of PTR query is used to get the RR with the primary name of the host. For example, a request for the host name corresponding to IP address `1.2.3.4` looks for PTR RRs for domain name `4.3.2.1.IN-ADDR.ARPA`.

3. *General lookup function.* This function retrieves arbitrary information from the DNS, and has no counterpart in previous systems. The caller supplies a `QNAME`, `QTYPE`, and `QCLASS`, and wants all the matching RRs. This function will often use the DNS format for all RR data instead of the local host's, and returns all RR content (e.g., TTL) instead of a processed form with local quoting conventions.

When the resolver performs the indicated function, it usually has one of the following results to pass back to the client:

• One or more RRs giving the requested data. In this case the resolver returns the answer in the appropriate format.

• A name error (NE). This happens when the referenced name does not exist. For example, a user may have mistyped a host name.

• A data-not-found error. This happens when the referenced name exists, but data of the appropriate type does not. For example, a host address function applied to a mailbox name would return this error since the name exists, but no address RR is present.

The functions for translating between host names and addresses may combine the name error and data-not-found error conditions into a single type of error return, but the general function should not. One reason for this is that applications may ask first for one type of information about a name followed by a second request to the same name for some other type of information; if the two errors are combined, then useless queries may slow the application.

In addition to its own resources, the resolver may also have shared access to zones maintained by a local name server. This gives the resolver the advantage of more rapid access, but the resolver must be careful to never let cached information override zone data. In this discussion the term *local information* is understood to mean the union of the cache and such shared zones, with the understanding that authoritative data are always used in preference to cached data when both are present.

The following resolver algorithm assumes that all functions have been converted to a general lookup function, and uses the following data structures to represent the state of a request in progress in the resolver:

`SNAME`	The domain name we are searching for.
`STYPE`	The QTYPE of the search request.
`SCLASS`	The `QCLASS` of the search request.
`SLIST`	A structure which describes the name servers and the zone which the resolver is currently trying to query. This structure keeps track of the resolver's current best guess about which name servers hold the desired information; it is updated when arriving information changes the guess. This structure includes the equivalent of a zone name, the known name servers for the zone, the known addresses for the name servers, and history information which can be used to suggest which server is likely to be the best one to try next. The zone name equivalent is a match count of the number of labels from the root down which `SNAME` has in common with the zone being queried; this is used as a measure of how "close" the resolver is to `SNAME`.
`SBELT`	A "safety belt" structure of the same form as `SLIST`, which is initialized from a configuration file, and lists servers which should be used when the resolver doesn't have any local information to guide name server selection. The match count will be –1 to indicate that no labels are known to match.
`CACHE`	A structure which stores the results from previous responses. Since resolvers are responsible for discarding old RRs whose TTL has expired, most implementations convert the interval specified in arriving RRs to some sort of absolute time when the RR is stored in the cache. Instead of counting the TTLs down individually, the resolver just ignores or discards old RRs when it runs across them in the course of a search, or discards them during periodic sweeps to reclaim the memory consumed by old RRs.

19.9 Further Information on DNS

Further information can be obtained from the following sources.

RFC 742	K. Harrenstien, *NAME/FINGER,* RFC 742, Network Information Center, SRI International, Dec. 1977.
RFC 768	J. Postel, *User Datagram Protocol,* RFC 768, USC/Information Sciences Institute, Aug. 1980.
RFC 793	J. Postel, *Transmission Control Protocol,* RFC 793, USC/Information Sciences Institute, Sept. 1981.
RFC 799	D. Mills, *Internet Name Domains,* RFC 799, COMSAT, Sept. 1981. (Suggests introduction of a hierarchy in place of a flat namespace for the Internet.)
RFC 805	J. Postel, *Computer Mail Meeting Notes*, RFC 805, USC/Information Sciences Institute, Feb. 1982.
RFC 810	E. Feinler, K. Harrenstien, Z. Su, and V. White, *DOD Internet Host Table Specification,* RFC 810, Network Information Center, SRI International, March 1982. (Obsolet; see RFC 952).
RFC 811	K. Harrenstien, V. White, and E. Feinler, *Hostnames Server,* RFC 811, Network Information Center, SRI International, March 1982. (Obsolete; see RFC 953).
RFC 812	K. Harrenstien and V. White, *NICNAME/WHOIS,* RFC 812, Network Information Center, SRI International, March 1982.
RFC 819	Z. Su and J. Postel, *The Domain Naming Convention for Internet User Applications,* RFC 819, Network Information Center, SRI International, Aug. 1982. (Early thoughts on the design of the domain system. The current implementation is completely different.)
RFC 821	J. Postel, *Simple Mail Transfer Protocol,* RFC 821, USC/Information Sciences Institute, Aug. 1980.
RFC 830	Z. Su, *A Distributed System for Internet Name Service,* RFC 830, Network Information Center, SRI International, Oct. 1982. (Early thoughts on the design of the domain system. Current implementation is completely different.)
RFC 882	P. Mockapetris, *Domain Names—Concepts and Facilities,* RFC 882, USC/Information Sciences Institute, Nov. 1983.
RFC 883	P. Mockapetris, *Domain Names—Implementation and Specification,* RFC 883, USC/Information Sciences Institute, Nov. 1983.
RFC 920	J. Postel and J. Reynolds, *Domain Requirements,* RFC 920, USC/Information Sciences Institute, Oct. 1984. (This explains the naming scheme for top-level domains.)
RFC 952	K. Harrenstien, M. Stahl, and E. Feinler, *DoD Internet Host Table Specification,* RFC 952, SRI, Oct. 1985. (Specifies the format of `HOSTS.TXT`, the host/address table replaced by the DNS.)

RFC 953 K. Harrenstien, M. Stahl, E. Feinler, *HOSTNAME Server,* RFC 953, SRI, Oct. 1985. (This RFC contains the official specification of the host name server protocol, which is rendered obsolete by the DNS. This TCP-based protocol accesses information stored in the RFC-952 format, and is used to obtain copies of the host table.)

RFC 973 P. Mockapetris, *Domain System Changes and Observations,* RFC 973, USC/Information Sciences Institute, Jan. 1986. (Describes changes to RFC 882 and RFC 883 and reasons for them. Now obsolete.)

RFC 974 C. Partridge, *Mail Routing and the Domain System,* RFC 974, CSNET CIC BBN Labs, Jan. 1986. (Describes the transition from `HOSTS.TXT`-based mail addressing to the more powerful MX system used with the domain system.)

RFC 1001 NetBIOS Working Group, *Protocol Standard for a NetBIOS Service on a TCP/UDP Transport: Concepts and Methods,* RFC 1001, March 1987. (This RFC and RFC 1002 are a preliminary design for NetBIOS on top of TCP/IP which proposes to base NetBIOS name service on top of the DNS.)

RFC 1002 NetBIOS Working Group, *Protocol Standard for a NetBIOS service on a TCP/UDP Transport: Detailed Specifications,* RFC 1002, March 1987.

RFC 1010 J. Reynolds and J. Postel, *Assigned Numbers,* RFC 1010, USC/Information Sciences Institute, May 1987. (Contains socket numbers and mnemonics for host names, operating systems, etc.)

RFC 1031 W. Lazear, *MILNET Name Domain Transition,* RFC 1031, Nov. 1987. (Describes a plan for converting the MILNET to the DNS.)

RFC 1032 M. K. Stahl, *Establishing a Domain—Guidelines for Administrators,* RFC 1032, Nov. 1987. (Describes the registration policies used by the NIC to administer the top-level domains and delegate subzones.)

RFC 1033 M. K. Lottor, *Domain Administrators Operations Guide,* RFC 1033, Nov. 1987. (A cookbook for domain administrators.)

M. Solomon, L. Landweber, and D. Neuhengen, "The CSNET Name Server," *Computer Networks* **6**(3), (July 1982). (This article describes a name service for CSNET which is independent from the DNS and DNS use in the CSNET.)

19.10 Summary

The purpose of this chapter is to provide detailed information to those who are working with network design and need detailed information on the DNS. DNS is complex. Its design was to eliminate much work for updating databases used in the 1980s. This chapter has presented an overview of the DNS.

20
DHCP Configuration and Use

The Dynamic Host Configuration Protocol (DHCP) is a way to pass configuration information to hosts on a TCP/IP network. It is based on the Bootstrap Protocol (BOOTP). DHCP adds the capability of automatic allocation of reusable network addresses and additional configuration options. DHCP functions with BOOTP relay agents.

20.1 Perspective

DHCP provides configuration parameters to Internet hosts. It consists of two components: a protocol for delivering host-specific configuration parameters from a DHCP server to a host and a mechanism for allocation of network addresses to hosts.

DHCP is built on a client/server model, where designated DHCP server hosts allocate network addresses and deliver configuration parameters to dynamically configured hosts. Use of the term *server* here refers to a host providing initialization parameters through DHCP; and the term *client* refers to a host requesting initialization parameters from a DHCP server.

A host should not act as a DHCP server unless explicitly configured to do so by a system administrator. The diversity of hardware and protocol implementations in the Internet would preclude reliable operation if random hosts were allowed to respond to DHCP requests. For example, suppose that IP requires the setting of many parameters within the protocol implementation software. Because IP can be used on many dissimilar kinds of network hardware, values for those parameters cannot be guessed or assumed to have correct defaults. Also, distributed address allocation schemes depend on a polling or defense mechanism for discovery of addresses that are already in use. IP hosts may not always be able to defend their network addresses, so that such a distributed address allocation scheme cannot be guaranteed to avoid allocation of duplicate network addresses.

20.1.1 Address Allocation

DHCP supports three mechanisms for IP address allocation. In *automatic allocation* DHCP assigns a permanent IP address to a client. *Dynamic allocation* means that DHCP assigns an IP address to a client for a limited period of time (or until the client explicitly relinquishes the address). In *manual allocation* a client's IP address is assigned by the network administrator, and DHCP is used simply to convey the assigned address to the client. A particular network will use one or more of these mechanisms, depending on the policies of the network administrator.

Dynamic allocation is the only one of the three mechanisms that allows automatic reuse of an address that is no longer needed by the client to which it was assigned. Thus, dynamic allocation is particularly useful for assigning an address to a client that will be connected to the network only temporarily or for sharing a limited pool of IP addresses among a group of clients that do not need permanent IP addresses. Dynamic allocation may also be a good choice for assigning an IP address to a new client being permanently connected to a network where IP addresses are so scarce that it is important to reclaim them when old clients are retired.

Manual allocation allows DHCP to be used to eliminate the error-prone process of manually configuring hosts with IP addresses in environments where (for whatever reasons) it is desirable to manage IP address assignment outside of the DHCP mechanisms.

20.1.2 DHCP Message Format

DHCP messages format is based on the format of BOOTP messages, to capture the BOOTP relay agent behavior described as part of the BOOTP specification and to allow interoperability of existing BOOTP clients with DHCP servers. Using BOOTP relay agents eliminates the necessity of having a DHCP server on each physical network segment.

20.1.3 Recent DHCP Additions

DHCP message type, `DHCPINFORM`, has been a recent addition to the protocol specification. Also, the classing mechanism for identifying DHCP clients to DHCP servers has been extended to include vendor classes. The minimum lease time restriction has been removed. Finally, many editorial changes have been made to clarify the text as a result of experience gained in DHCP interoperability tests.

20.1.4 DHCP Information

Several Internet protocols and related mechanisms that address some parts of the dynamic host configuration problem. Some of these include the *Reverse Address Resolution Protocol* (RARP), which explicitly addresses the problem of network address discovery and includes an automatic IP address assignment mechanism; the *Trivial File Transfer Protocol* (TFTP), which provides for transport of a boot image from a boot server; and the *Internet Control Message Protocol* (ICMP), which provides for informing hosts of additional routers via ICMP redirect messages. ICMP also can provide subnet mask information by way of the ICMP mask request message. Additional information can be obtained by the ICMP information request message. Network hosts can locate routers through the ICMP router discovery mechanism.

BOOTP is a transport mechanism for a collection of configuration information. The Network Information Protocol (NIP), used by Athena at MIT, is a distributed mechanism for dynamic IP address assignment. The *Resource Location Protocol* (RLP) provides for location of higher-level services. Sun Microsystems' diskless workstations use a boot procedure that employs RARP, TFTP, and an RPC mechanism called *bootparams*. These *bootparams* deliver configuration information and operating system code to diskless network hosts. Some Sun (Sun Microsystems) networks also use DRARP and an autoinstallation mechanism to automate the configuration of new hosts in an existing, functional network.

In other related work, the path minimum transmission unit (MTU) discovery algorithm can determine the MTU of an arbitrary Internet path. The Address Resolution Protocol (ARP) has been proposed as a transport protocol for resource location and selection.

20.1.5 Other DHCP Considerations

DHCP was designed to supply DHCP clients with the configuration parameters defined in the host requirement RFCs. After obtaining parameters via DHCP, a DHCP client should be able to exchange packets with any other host in the Internet.

Not all of these parameters are required for a newly initialized client. A client and server may negotiate transmission of those parameters required by the client or specific to a particular subnet.

DHCP allows but does not require the configuration of client parameters not directly related to the IP protocol. DHCP also does not address registration of newly configured clients with the domain name system (DNS). DHCP original design intent is not intended to configure routers.

20.2 Terminology

binding A binding is a collection of configuration parameters, including at least an IP address, associated with or bound to a DHCP client. Bindings are managed by DHCP servers.

BOOTP relay agent A BOOTP relay agent is an Internet host or router that passes DHCP messages between DHCP clients and DHCP servers. DHCP is designed to use the same relay agent behavior as specified in the BOOTP protocol specification.

DHCP client A DHCP client is an Internet host using DHCP to obtain configuration parameters such as a network address.

DHCP server A DHCP server is an Internet or Internet host that returns configuration parameters to DHCP clients.

The original *design intent* of DHCP includes the following considerations:

• DHCP should be a mechanism rather than a policy. DHCP must allow local system administrators control over configuration parameters where desired; this means that local system administrators should be able to enforce local policies concerning allocation and access to local resources where desired.

• Clients should require no manual configuration. Each client should be able to discover appropriate local configuration parameters without user intervention and incorporate those parameters into its own configuration.

• Networks should require no manual configuration for individual clients. Under normal circumstances, a network manager should not have to manually enter any per-client configuration parameters.

• DHCP should not require a server on each subnet. To allow for scale and economy, DHCP must work across routers or through the intervention of BOOTP relay agents.

• A DHCP client must be prepared to receive multiple responses to a request for configuration parameters. Some installations may include multiple, overlapping DHCP servers to enhance reliability and increase performance.

• DHCP must coexist with statically configured, nonparticipating hosts and with existing network protocol implementations.

• DHCP must work with the BOOTP relay agent behavior as described by RFCs 951 and 1542.

• DHCP must provide service to existing BOOTP clients.

DHCP requirements include the following, specific to the transmission of network-layer parameters:

• Guarantee that any network address will not be in use by more than one DHCP client at a time.

• Retain DHCP client configuration across DHCP client reboot. A DHCP client should, whenever possible, be assigned the same configuration parameters (e.g., network address) in response to each request.

• Retain DHCP client configuration across server reboots.

• A DHCP client should be assigned the same configuration parameters despite restarts of the DHCP mechanism.

• Allow automated assignment of configuration parameters to new clients to avoid hand configuration for new clients.

• Support fixed or permanent allocation of configuration parameters to specific clients.

20.3 DHCP Protocol

From the client’s point of view, DHCP is an extension of the BOOTP mechanism. This behavior allows existing BOOTP clients to interoperate with DHCP servers without requiring any change to the clients’ initialization software. RFC 1542 details the interactions between BOOTP and DHCP clients and servers.

20.3.1 DHCP Message Format

The DHCP message format shown in Fig. 20-1 describes each of the fields in the DHCP message. The numbers in parentheses indicate the size of each field in octets. There are two primary differences between DHCP and BOOTP: (1) DHCP defines mechanisms through which clients can be assigned a network address for a finite lease, allowing for serial reassignment of network addresses to different clients; and (2) DHCP provides the mechanism for a client to acquire all the IP configuration parameters that it needs in order to operate. DHCP introduces a small change in terminology intended to clarify the meaning of one of the fields. What was at one time referred to as the *vendor extensions* field in BOOTP is now referred to as the *options* field in DHCP. *Tagged* data items that were used inside the BOOTP *vendor extensions* field, which were formerly referred to as *vendor extensions,* are now termed *options*.

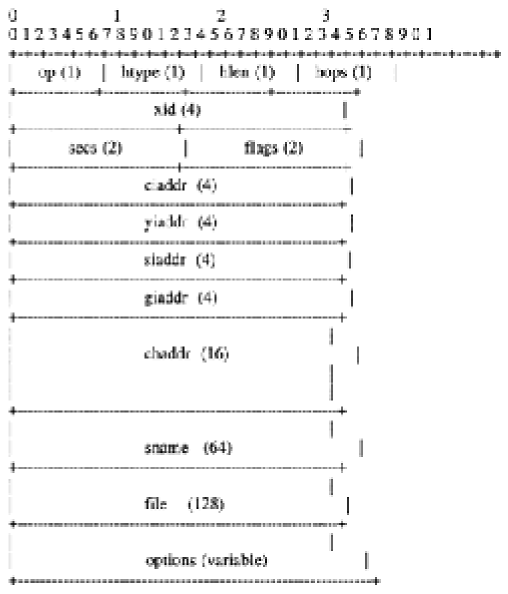


Figure 20-1
DHCP message format.

DHCP defines a new *client identifier* option that is used to pass an explicit client identifier to a DHCP server. This change eliminates the overloading of the `chaddr` field in BOOTP messages, where `chaddr` is used both as a hardware address for transmission of BOOTP reply messages and as a client identifier. The `client identifier` is an opaque key, not to be interpreted by the server. It may contain a hardware address, identical to the contents of the `chaddr` field, or it may contain another type of identifier, such as a DNS name. The `client identifier` chosen by a DHCP client *must* be unique to that client within the subnet to which the client is attached. If a client uses a `client identifier` in one message, it *must* use that same identifier in all subsequent messages, to ensure that all servers correctly identify the client.

DHCP clarifies the interpretation of the `siaddr` field as the address of the server to use in the next step of the client' s bootstrap process. A DHCP server can return its own address in the `siaddr` field if the server is prepared to supply the next bootstrap service; for example, the delivery of an operating system executable image. A DHCP server always returns its own address in the `server identifier` option.

20.3.2 DHCP Message Field Explanation

Consider the explanation for fields in a DHCP message shown in Table 20.1.

The *options* field is now variable length. A DHCP client must be prepared to receive DHCP messages with an *options* field of at least length 312 octets. This requirement implies that a DHCP client must be prepared to receive a message of up to 576 octets, the minimum IP datagram size an IP host must accept. It is possible, however, for DHCP clients to negotiate the use of larger DHCP messages through the `maximum DHCP message size` option. The options field may be further extended into the *file* and *sname* fields.

Table 20-1 Fields in a DHCP Message

Field	Octets	Description
`op`	1	Message op code/message type. 1 = `BOOTREQUEST, 2 = BOOTREPLY`
`htype`	1	Hardware address type.
`hlen`	1	Hardware address length.
`hops`	1	Client sets to zero, optionally used by relay agents when booting via a relay agent.
`xid`	4	Transaction ID, a random number chosen by the client, used by the client and server to associate messages and responses between a client and a server.
`secs`	2	Filled in by client, seconds elapsed since client began address acquisition or renewal process.
`flags`	2	Flags
`ciaddr`	4	Client IP address; filled in only if client is in `BOUND, RENEW`, or `REBINDING` state and can respond to ARP requests.
`yiaddr`	4	Your (client) IP address.
`siaddr`	4	IP address of next server to use in bootstrap; returned in `DHCPOFFER, DHCPACK` by server.
`giaddr`	4	Relay agent IP address, used in booting via a relay agent.
`chaddr`	16	Client hardware address
`sname`	64	Optional server host name, null terminated string.
`file`	128	Boot filename, null terminated string; "generic" name or null in `DHCPDISCOVER`, fully qualified directory pathname in `DHCPOFFER`.
`options var`		Optional parameters field.

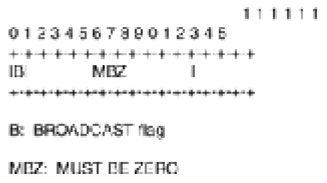


Figure 20-2
DHCP *flags* field format.

When DHCP is used in initial configuration (prior to the client's TCP/IP software complete configuration), DHCP requires mental agility on behalf of the user. TCP/IP software installed should accept and *forward* any IP packets to the IP layer delivered to the client's hardware address before the IP address is configured. DHCP servers and BOOTP relay agents may not be able to deliver DHCP messages to clients that cannot accept hardware unicast datagrams before the TCP/IP software is configured. So, be aware of this when you find yourself in this scenario.

In order to work around some clients that cannot accept IP *unicast* datagrams before the TCP/IP software is configured DHCP uses the *flags* field. Here, the leftmost bit is defined as the BROADCAST (B) flag. See Fig. 20-2.

20.4 DHCP Configuration Parameters Repository

The first service provided by DHCP is to provide persistent storage of network parameters for network clients. The model of DHCP persistent storage is as follows. The DHCP service stores a key-value entry for each client (a unique identifier such as an IP subnet number and a unique identifier within the subnet), and the value contains the configuration parameters for the client.

For example, in this case the key might be the `IP-subnet-number` and `hardware-address`. Alternately, the key might be the pair (`IP-subnet-number, hostname`), thus allowing the server to assign parameters intelligently to a DHCP client moved to a different subnet or that has had a changed hardware addresses. The protocol defines that the key will be `IP-subnet-number, hardware-address` unless the client explicitly supplies an identifier using the `client identifier` option. A client can query the DHCP service to retrieve its configuration parameters. The client interface to the configuration parameters repository consists of protocol messages to request configuration parameters and responses from the server carrying the configuration parameters.

20.5 Network Address Dynamic Allocation

The second service provided by DHCP is the allocation of temporary or permanent network (IP) addresses to clients. The basic mechanism for the dynamic allocation of network addresses is simple: a client requests the use of an address for some period of time. The allocation mechanism (the collection of DHCP servers) guarantees not to reallocate that address within the requested time and attempts to return the same network address each time the client requests an address. In this document, the period over which a network address is allocated to a client is referred to as a *lease*. The client may extend its lease with subsequent requests. It may issue a message to release the address back to the server when the client no longer needs the address. The client may ask for a permanent assignment by asking for an infinite lease. Even when assigning *permanent* addresses a server may choose to give out lengthy but noninfinite leases to allow detection of the fact that the client has been retired.

Some environments will require reassignment of network addresses because of exhaustion of available addresses. In such environments, the allocation mechanism will reuse addresses whose *lease* has expired. The server should use whatever information is available in the configuration information repository to choose an address to reuse. For example, the server may choose the least recently assigned address. As a consistency check, the allocating server should probe the reused address before allocating the address, such as with an ICMP echo request, and the client should probe the newly received address with ARP.

20.6 Client/Server Protocol

DHCP uses the BOOTP message format shown in Fig. 20-3. The `op` field of each DHCP message sent from a client to a server contains a `BOOTREQUEST`. The `BOOTREPLY` is used in the `op` field of each DHCP message sent from a server to a client.

The first four octets of the *options* field of the DHCP message contain decimal values 99, 130, 83 and 99. The remainder of the *options* field consists of a list of tagged parameters that are called *options*. All `vendor extensions` are also DHCP options. For additional information, RFC 1533 provides a complete set of options defined for use with DHCP.

Some options included in the RFC are presented here. One particular option, the DHCP message-type option, must be included in each DHCP message. This option defines the *type* of the DHCP message. Additional options may be allowed, required, or not allowed, depending on the DHCP message type.

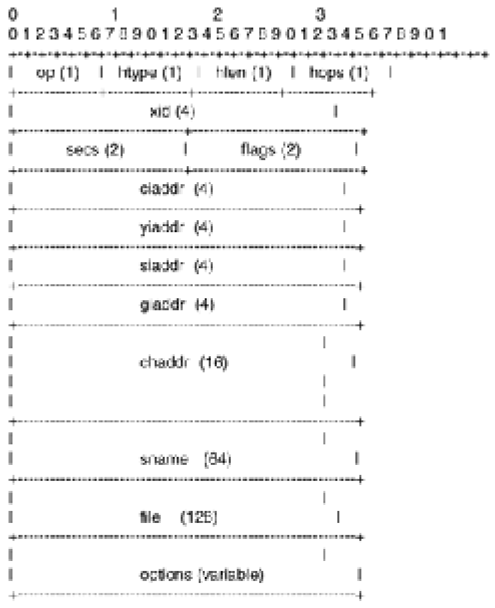


Figure 20-3
BOOTP message format.

DHCP messages and meanings are as follows:

`DHCPDISCOVER`	Client broadcast to locate available servers.
`DHCPOFFER`	Server to client in response to DHCPDISCOVER with offer of configuration parameters.
`DHCPREQUEST`	Client message to servers either requesting offered parameters from one server and implicitly declining offers from all others or confirming correctness of previously allocated address after. For example, a system reboot or extending the lease on a particular network address.
`DHCPACK`	Server to client with configuration parameters including committed network address.
`DHCPNAK`	Server to client indicating client’s notion of network address is incorrect (e.g., client has moved to new subnet) or client’s *lease* as expired.
`DHCPDECLINE`	Client to server indicating network address is already in use.

`DHCPRELEASE`	Client to server relinquishing network address and cancelling remaining lease.
`DHCPINFORM`	Client to server, asking only for local configuration parameters; client already has externally configured network address.

20.6.1 DHCP Message Timeline

Figure 20-4 is an example of the timeline concept between a DHCP client and DHCP server when new address allocation is performed.

20.6.2 DHCP Client Usage

A client should use DHCP to reacquire or verify its IP address and network parameters whenever the local network parameters may have changed; such as at system boot time or after a disconnection from the local network, as the local network configuration may change without the client's or user's knowledge.

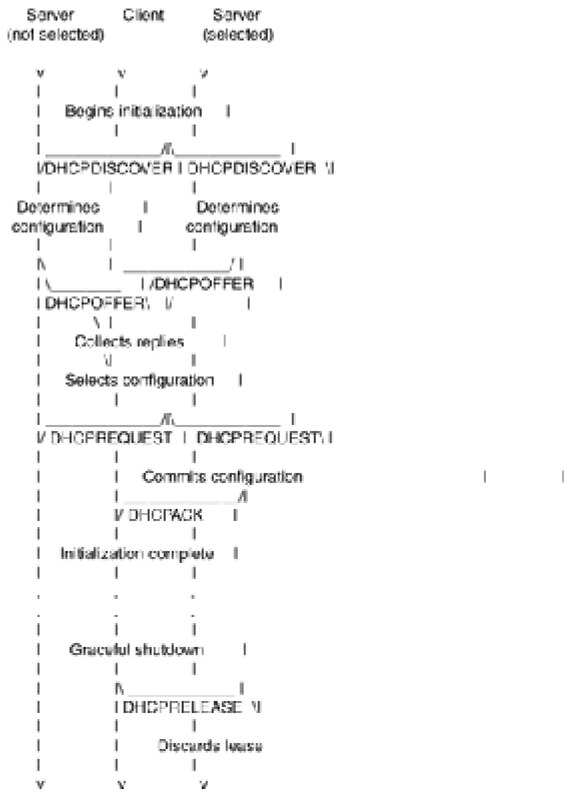


Figure 20-4
DHCP message timeline.

If a client has knowledge of a previous network address and is unable to contact a local DHCP server, the client may continue to use the previous network address until the lease for that address expires. If the lease expires before the client can contact a DHCP server, the client must immediately discontinue use of the previous network address and may inform local users of the problem.

20.6.3. DHCP Client/server Protocol Specification

It is important to understand the DHCP client/server specification. This section makes basic assumptions to provide helpful insights. The first assumption is a DHCP server has a block of network addresses it can satisfy requests for new addresses. Another assumption is that each server maintains a database of allocated addresses and *leases* in its local permanent storage.

Constructing and Sending DHCP Messages

DHCP clients and servers construct DHCP messages by filling in fields in the fixed-format section of the message and appending tagged data items in the variable-length option area. The options area includes first a 4-octet "magic cookie," followed by the options. The last option must always be the *end* option.

DHCP uses UDP as its transport protocol. Its messages from client to server are sent to the DHCP server port 67. DHCP messages are sent from a server to a client on the DHCP client port 68. A server with multiple network address (multihomed host) can use any of its network addresses in outgoing DHCP messages.

The *server identifier* field is used both to identify a DHCP server in a DHCP message and as a destination address from clients to servers. A server with multiple network addresses *must* be prepared to accept any of its network addresses as identifying that server in a DHCP message. To accommodate potentially incomplete network connectivity a server is required to choose an address as a *server identifier*. This address is reachable from the client. For example, if the DHCP server and the DHCP client are connected to the same subnet, the server should select the IP address that the server is using for communication on that subnet as the `server identifier.` If the server is using multiple IP addresses on that subnet, any such address may be used. If the server has received a message through a DHCP relay agent, the server should choose an address from the interface on which the message was received as the `server identifier.` DHCP clients are required to use the IP address provided in the `server identifier` option for any unicast requests to the DHCP server.

DHCP messages broadcast by a client before that client obtains its IP address must have the source address field in the IP header set to 0.

If the `giaddr` field in a DHCP message from a client is nonzero, the server sends any return messages to the *DHCP server* port on the BOOTP relay agent whose address appears in `giaddr.` If the `giaddr` field is zero and the `ciaddr` field is nonzero, then the server unicasts `DHCPOFFER` and `DHCPACK` messages to the address in `ciaddr.` If `giaddr` is zero and `ciaddr` is zero, and the broadcast bit is set, then the server broadcasts `DHCPOFFER` and `DHCPACK` messages to `0xffffffff`. If the broadcast bit is not set and `giaddr` is zero and `ciaddr` is zero, then the server unicasts `DHCPOFFER` and `DHCPACK` messages to the client's hardware address and `yiaddr` address. In all cases, when `giaddr` is zero, the server broadcasts any `DHCPNAK` messages to `0xffffffff`.

If the options in a DHCP message extend into the `sname` and `file` fields, the *option overload* option is required to appear in the `options` field, with value 1, 2, or 3. If the `option overload` option is present in the `options` field, the options in the `options` field *must* be terminated by an `end option`, and *may* contain one or more `pad` options to fill the `options` field. The options in the `sname` and `file` fields are required to begin with the first octet of the field, are also required to be terminated by an `end option`, and be followed by `pad` options to fill the remainder of the field. Any individual option in the `options`, `sname`, and `file` fields are required to be entirely contained in that field. The options in the `options` field must be interpreted first, so that any `option overload` options may be interpreted. The `file` field must be interpreted next, followed by the `sname` field.

The values to be passed in an *option* tag may be too long to fit in the 255 octets available to a single option. Options may appear only once, unless otherwise specified in the options document. The client concatenates the values of multiple instances of the same option into a single parameter list for configuration.

DHCP clients are responsible for all message retransmission. The client *must* adopt a retransmission strategy that incorporates a randomized exponential backoff algorithm to determine the delay between retransmissions. The delay between retransmissions should be chosen to allow sufficient time for replies from the server to be delivered on the basis of the characteristics of the internetwork between the client and the server.

For example, in a 10-Mbit/s Ethernet network the delay before the first retransmission should be 4 s randomized by the value of a uniform random number chosen from the range –1 to +1. Clients with clocks that provide resolution granularity of less than 1 s may choose a noninteger randomization value. The delay before the next retransmission should be 8 s randomized by the value of a uniform number chosen from the range –1 to +1. The retransmission delay should be doubled with subsequent retransmissions up to a maximum of 64 s. The client can provide an indication of retransmission attempts to the user as an indication of the progress of the configuration process.

The `xid` field is used by the client to match incoming DHCP messages with pending requests. A DHCP client must choose `xid`s in such a way as to minimize the chance of using an `xid` identical to one used by another client. For example, a client may choose a different, random initial `xid` each time the client is rebooted, and subsequently use sequential `xid`s until the next reboot. Selecting a new `xid` for each retransmission is an implementation decision. A client may choose to reuse the same `xid` or select a new `xid` for each retransmitted message.

Normally, DHCP servers and BOOTP relay agents attempt to deliver `DHCPOFFER, DHCPACK,` and `DHCPNAK` messages directly to the client using unicast delivery. The IP destination address is set to the DHCP `yiaddr` address and the link-layer destination address is set to the DHCP *chaddr* address. Unfortunately, some client implementations are unable to receive such unicast IP datagrams until the implementation has been configured with a valid IP address.

A client that cannot receive unicast IP datagrams until its protocol software has been configured with an IP address should set the `BROADCAST` bit in the flags field to 1 in any `DHCPDISCOVER` or `DHCPREQUEST` messages that the client sends. The `BROADCAST` bit will provide a hint to the DHCP server and BOOTP relay agent to broadcast any messages to the client on the client’s subnet. A client that can receive unicast IP datagrams before its protocol software has been configured should clear the `BROADCAST` bit to 0. The BOOTP clarifications document discusses the ramifications of the use of the `BROADCAST` bit.

A server or relay agent sending or relaying a DHCP message directly to a DHCP client should examine the `BROADCAST` bit in the *flags* field. If this bit is set to 1, the DHCP message should be sent as an IP broadcast using an IP broadcast address (preferably `0xffffffff`) as the IP destination address and the link-layer broadcast address as the link-layer destination address. If the `BROADCAST` bit is cleared to 0, the message should be sent as an IP unicast to the IP address specified in the `yiaddr` field and the link-layer address specified in the `chaddr` field. If unicasting is not possible, the message can be sent as an IP broadcast using an IP broadcast address (preferably `0xffffffff`) as the IP destination address and the link-layer broadcast address as the link-layer destination address.

DHCP Server Administrative Controls

DHCP servers are not required to respond to every `DHCPDISCOVER` and `DHCPREQUEST` message they receive; an example would be for a network administrator to retain control over the clients attached to the network configuration of DHCP servers to respond only to clients that have been previously registered through some external mechanism. The DHCP specification describes only the interactions between clients and servers when the clients and servers choose to interact; it is beyond the scope of the DHCP specification to describe all the administrative controls that system administrators might want to use. Specific DHCP server implementations may incorporate any controls or policies desired by a network administrator.

In some environments, a DHCP server will have to consider the values of the vendor class options included in `DHCPDISCOVER` or `DHCPREQUEST` messages when determining the correct parameters for a particular client.

A DHCP server needs to use some unique identifier to associate a client with its lease. The client may choose to explicitly provide the identifier through the `client identifier` option. If the client supplies a `client identifier`, the client *must* use the same `client identifier` in all subsequent messages, and the server must use that identifier to identify the client. If the client does not provide a `client identifier` option, the server *must* use the contents of the `chaddr` field to identify the client. It is important for a DHCP client to use an identifier unique within the subnet to which the client is attached in the `client identifier` option. Use of `chaddr` as the client's unique identifier may cause unexpected results, as that identifier may be associated with a hardware interface that could be moved to a new client. Some sites may choose to use a manufacturer's serial number as the `client identifier,` to avoid unexpected changes in a client's network address due to transfer of hardware interfaces among computers. Sites may also choose to use a DNS name as the `client identifier,` causing address leases to be associated with the DNS name rather than a specific hardware box.

DHCP clients are free to use any strategy in selecting a DHCP server among those from which the client receives a `DHCPOFFER` message. The client implementation of DHCP should provide a mechanism for the user to select directly the `vendor class identifier` values.

20.6.4 DHCP Server Function

A DHCP server processes incoming DHCP messages from a client according to the current state of the binding for that client. The following messages can be received by a DHCP server from a client: `DHCPDISCOVER, DHCPREQUEST, DHCPDECLINE, DHCPRELEASE`, and `DHCPINFORM`. Tables 20.2 and 20.3 show the correlation between use of the fields and options in a DHCP message by a server. After this correlation information is provided, a description of the DHCP server action for each possible incoming message is presented.

Table 20-2 Fields Used by DHCP Servers

Field	DHCPOFFER	DHCPACK	DHCPNAK
op	BOOTREPLY	BOOTREPLY	BOOTREPLY
htype			
hlen	Hardware address length (in octets)		
hops	0	0	0
xid	xid from client DHCPDISCOVER message	xid from client DHCPREQUEST message	xid from client DHCPREQUEST message
'secs'	0	0	0
ciaddr	0	ciaddr from DHCPREQUEST or 0	0
yiaddr	IP address offered to client	IP address assigned to client	0
siaddr	IP address of next boostrap server	IP address of next boostrap server	0
flags	flags from client DHCPDISCOVER message	flags from client DHCPREQUEST message	flags from client DHCPREQUEST message
giaddr	giaddr from client DHCPDISCOVER message	giaddr from client DHCPREQUEST message	giaddr from client DHCPREQUEST message
chaddr	chaddr from client DHCPREQUEST message	chaddr from client HCPDISCOVER message	chaddr from client DHCPREQUEST message
sname	Server host name or options	Server host name or options	(unused)
file	client boot file name or options	Client boot file name or options	(unused)

DHCPDISCOVER Message

When a server receives a `DHCPDISCOVER` message from a client, the server chooses a network address for the requesting client. If no address is available, the server may choose to report the problem to the system administrator. If an address is available, the new address should be selected as follows:

The client’s current address as recorded in the client’s current binding, ELSE

Table 20-3 Options Used by DHCP Servers

Option	**DHCPOFFER**	**DHCPACK**	**DHCPNAK**
requested IP address	MUST NOT	MUST NOT	MUST NOT
IP address lease time	MUST	MUST	MUST NOT
		DHCPREQUEST)	
		MUST NOT	
		DHCPINFORM	
		Use	
file/sname fields	MAY	MAY	MUST NOT DHCP
message type	DHCPOFFER	DHCPACK	DHCPNAK
Parameter request list	MUST NOT	MUST NOT	MUST
NOT Message	SHOULD	SHOULD	SHOULD
Client identifier	MUST NOT	MUST NOT	MAY
Vendor class identifier	MAY	MAY	MAY
Server identifier	MUST	MUST	MUST
Maximum message size	MUST NOT	MUST NOT	MUST
All others	MAY	MAY	MUST NOT

The client's previous address as recorded in the client's binding, if that address is in the server's pool of available addresses and not already allocated, ELSE

The address requested in the `Requested IP Address` option, if that address is valid and not already allocated, ELSE

A new address allocated from the server's pool of available addresses; the address is selected based on the subnet from which the message was received (if `giaddr` is 0) or on the address of the relay agent that forwarded the message (`giaddr` when not 0).

A server can assign an address other than the one requested or refuse to allocate an address to a particular client even though free addresses are available. In some network architectures (Internets with more than one IP subnet assigned to a physical network segment), the DHCP client should perhaps be assigned an address from a different subnet than the address recorded in `giaddr.` Hence, DHCP does not require that the client be assigned an address from the subnet in `giaddr.` However, a server is free to choose some other subnet.

While not required for correct operation of DHCP, the server should not reuse the selected network address before the client responds to the server's `DHCPOFFER` message. The server may choose to record the address as offered to the client.

The server must also choose an expiration time for the lease, as follows:

> IF the client has not requested a specific lease in the `DHCPDISCOVER` message and the client already has an assigned network address, the server returns the lease expiration time previously assigned to that address (note that the client must explicitly request a specific lease to extend the expiration time on a previously assigned address), ELSE

> IF the client has not requested a specific lease in the `DHCPDISCOVER` message and the client does not have an assigned network address, the server assigns a locally configured default lease time, ELSE

IF the client has requested a specific lease in the DHCPDISCOVER message (regardless of whether the client has an assigned network address), the server may choose either to return the requested lease (if the lease is acceptable to local policy) or select another lease.

Fields and options used by DHCP servers are listed in Tables 20.2 and 20.3, respectively.

Once the network address and lease have been determined, the server constructs a DHCPOFFER message with the offered configuration parameters. It is important for all DHCP servers to return the same parameters (with the possible exception of a newly allocated network address) to ensure predictable client behavior regardless of which server the client selects. The configuration parameters must be selected by applying the following rules in the order given below. The network administrator is responsible for configuring multiple DHCP servers to ensure uniform responses from those servers. The server *must* return to the client (1) the client’s network address, as determined by the rules given earlier in this section; (2) the expiration time for the client’s lease, as determined by the rules given earlier in this section; and (3) parameters requested by the client, according to the following rules:

• IF the server has been explicitly configured with a default value for the parameter, the server *must* include that value in an appropriate option in the option field, ELSE

• IF the server recognizes the parameter as a parameter defined in the *host requirements document* (HRD), the server *must* include the default value for that parameter as given in the HRD in an appropriate option in the option field, ELSE

• The server *must not* return a value for that parameter, the server *must* supply as many of the requested parameters as possible and must omit any parameters it cannot provide.

In addition

• Any parameters from the existing binding that differ from the HRD defaults.

• Any parameters specific to this client (as identified by the contents of chaddr or client identifier in the DHCPDISCOVER or DHCPREQUEST message), for example, as configured by the network administrator.

• Any parameters specific to this client’s class (as identified by the contents of the vendor class identifier option in the DHCPDISCOVER or DHCPREQUEST message), for example, as configured by the network administrator. The parameters must be identified by an exact match between the client’s vendor class identifiers and the client’s classes identified in the server.

• Parameters with nondefault values on the client’s subnet.

The server *may* choose to return the vendor class identifier used to determine the parameters in the DHCPOFFER message to assist the client in selecting which DHCPOFFER to accept. The server inserts the xid field from the DHCPDISCOVER message into the xid field of the DHCPOFFER message and sends the DHCPOFFER message to the requesting client.

DHCPREQUEST Message

A `DHCPREQUEST` message may come from a client responding to a `DHCPOFFER` message from a server, from a client verifying a previously allocated IP address, or from a client extending the lease on a network address. If the `DHCPREQUEST` message contains a `server identifier` option, the message is in response to a `DHCPOFFER` message. Otherwise, the message is a request to verify or extend an existing lease. If the client uses a `client identifier` in a `DHCPREQUEST` message, it must use that same `client identifier` in all subsequent messages. If the client includes a list of requested parameters in a `DHCPDISCOVER` message, it *must* include that list in all subsequent messages.

Any configuration parameters in the `DHCPACK` message should *not* conflict with those in the earlier `DHCPOFFER` message to which the client is responding. The client should use the parameters in the `DHCPACK` message for configuration.

Clients send `DHCPREQUEST` messages such as the following in the states they are in:

DHCPREQUEST Generated During SELECTING

The client inserts the address of the selected server in `server identifier, ciaddr` must be zero, `requested IP address` must be filled in with the `yiaddr` value from the chosen `DHCPOFFER`.

The client may choose to collect several `DHCPOFFER` messages and select the best. A client indicates its selection by identifying the offering server in the `DHCPREQUEST` message. If the client receives no acceptable offers, the client may choose to try another `DHCPDISCOVER` message. Therefore, the servers may not receive a specific `DHCPREQUEST` from which they can decide whether or not the client has accepted the offer. Because the servers have not committed any network address assignments on the basis of a `DHCPOFFER`, servers are free to reuse offered network addresses in response to subsequent requests. As an implementation detail servers should not reuse offered addresses and may use an implementation-specific timeout mechanism to decide when to reuse an offered address.

DHCPREQUEST Generated During INIT-REBOOT

`Server identifier` must not be filled in, `requested IP address` option must be filled in with client’s notion of its previously assigned address. The `ciaddr` must be zero. The client is seeking to verify a previously allocated, cached configuration. The server should send a `DHCPNAK` message to the client if the `requested IP address` is incorrect, or is on the wrong network.

Determining whether a client in the `INIT-REBOOT` state is on the correct network is done by examining the contents of `giaddr,` the `requested IP address` option, and a database lookup. If the DHCP server detects that the client is on the wrong net, then the server should send a `DHCPNAK` message to the client.

If the network is correct, then the DHCP server should check whether the client’s notion of its IP address is correct. If not, then the server should send a `DHCPNAK` message to the client. If the DHCP server has no record of this client, then it must remain silent, and can output a warning to the network administrator. This behavior is necessary for peaceful coexistence of noncommunicating DHCP servers on the same wire.

If `giaddr` is 0 × 0 in the `DHCPREQUEST` message, the client is on the same subnet as the server. The server must broadcast the `DHCPNAK` message to the `0xffffffff` broadcast address because the client may not have a correct network address or subnet mask, and the client may not be answering ARP requests.

If `giaddr` is set in the `DHCPREQUEST` message, the client is on a different subnet. The server *must* set the broadcast bit in the `DHCPNAK`, so that the relay agent will broadcast the `DHCPNAK` to the client, because the client may not have a correct network address or subnet mask, and the client may not be answering ARP requests.

DHCPREQUEST Generated During RENEWING A `server identifier` must not be filled in, `requested IP address` option must not be filled in, `ciaddr` must be filled in with client's IP address. In this situation, the client is completely configured, and is trying to extend its lease. This message will be unicast, so no relay agents will be involved in its transmission. Because `giaddr` is therefore not filled in, the DHCP server will trust the value in `ciaddr,` and use it when replying to the client.

A client may choose to renew or extend its lease prior to T1. The server may choose not to extend the *lease* but should return a `DHCPACK` message regardless.

DHCPREQUEST Generated During REBINDING

The `server identifier` must not be filled in, `requested IP address` option must not be filled in, `ciaddr` must be filled in with client's IP address. In this situation, the client is completely configured, and is trying to extend its lease. This message must be broadcast to the `0xffffffff` IP broadcast address. The DHCP server should check `ciaddr` for correctness before replying to the `DHCPREQUEST`.

The `DHCPREQUEST` from a REBINDING client is intended to accommodate sites that have multiple DHCP servers and a mechanism for maintaining consistency among leases managed by multiple servers. A DHCP server may extend a client's lease only if it has local administrative authority to do so.

DHCPDECLINE Message

If the server receives a `DHCPDECLINE` message, the client has discovered through some other means that the suggested network address is already in use. The server *must* mark the network address as not available and should notify the local system administrator of a possible configuration problem.

DHCPRELEASE Message

On receipt of a `DHCPRELEASE` message, the server marks the network address as not allocated. The server should retain a record of the client's initialization parameters for possible reuse in response to subsequent requests from the client.

DHCPINFORM Message

The server responds to a `DHCPINFORM` message by sending a `DHCPACK` message directly to the address given in the `ciaddr` field of the `DHCPINFORM` message. The server must not send a lease expiration time to the client and should not fill in `yiaddr`.

Client Messages

Consider the following example of differences between messages from clients in various states:

______________________________ | | INIT-REBOOT | SELECTING | RENEWING
| REBINDING | __________________________ | broad/unicast | broadcast| broadcast |
unicast | broadcast || server-ip | MUST NOT ||MUST NOT |MUST NOT |MUST NOT |
| requested-ip | MUST ||MUST | MUST NOT | MUST NOT | | ciaddr | zero | zero |
| IP address | IP address | ____

20.6.5 DHCP Client Function

A DHCP client can receive the following messages from a server: `DHCPOFFER, DHCPACK,` and `DHCPNAK`.

The remainder of this section describes the action of the DHCP client for each possible incoming message. The client begins in `INIT` state and forms a `DHCPDISCOVER` message. The client should wait a random time between 1 and 10 s to desynchronize the use of DHCP at start up. The client sets `ciaddr` to 0 × 00000000. The client may request specific parameters by including the `parameter request list` option. The client may suggest a network address and/or lease time by including the `requested IP address` and `IP address lease time` options. The client must include its hardware address in the `chaddr` field, if necessary for delivery of DHCP reply messages. The client can include a different unique identifier in the `client identifier` option. If the client included a list of requested parameters in a `DHCPDISCOVER` message, it must include the list in all subsequent messages.

The client generates and records a random transaction identifier and inserts that identifier into the `xid` field. The client records its own local time for later use in computing the lease expiration. The client then broadcasts the `DHCPDISCOVER` on the local hardware broadcast address to the `0xffffffff` IP broadcast address and `DHCP server` UDP port.

If the `xid` of an arriving `DHCPOFFER` message does not match the `xid` of the most recent `DHCPDISCOVER` message, the `DHCPOFFER` message must be silently discarded. Any arriving `DHCPACK` messages must be silently discarded.

The client collects `DHCPOFFER` messages over a period of time, selects one `DHCPOFFER` message from the incoming `DHCPOFFER` messages, and extracts the server address from the `server identifier` option in the `DHCPOFFER` message. The time over which the client collects messages and the mechanism used to select one `DHCPOFFER` are implementation-dependent.

If parameters are acceptable, the client records the address of the server that supplied the parameters from the `server identifier` field and sends that address in the `server identifier` field of a `DHCPREQUEST` broadcast message. Once the `DHCPACK` message from the server arrives, the client is initialized and moves to BOUND state. The `DHCPREQUEST` message contains the same `xid` as the `DHCPOFFER` message. The client records the lease expiration time as the sum of the time at which the original request was sent and the duration of the lease from the `DHCPACK` message. The client should perform a check on the suggested address to ensure that the address is not already in use. For example, if the client is on a network that supports ARP, the client may issue an ARP request for the suggested request. When broadcasting an ARP request for the suggested address, the client must fill in its own hardware address as the sender's hardware address, and 0 as the sender's IP address, to avoid confusing ARP caches in other hosts on the same subnet. If the network address appears to be in use, the client *must* send a `DHCPDECLINE` message to the server. The client *should* broadcast an ARP reply to announce the client's new IP address and clear any outdated ARP cache entries in hosts on the client's subnet.

1. *Initialization with a known network address*. The client begins in `INIT-REBOOT` state and sends a `DHCPREQUEST` message. The client must insert its known network address as a `requested IP address` option in the `DHCPREQUEST` message. The client may request specific configuration parameters by including the `parameter request list` option. The client generates and records a random transaction identifier and inserts that identifier into the `xid` field. The client records its own local time for later use in computing the lease expiration. The client must not include a `server identifier` in the `DHCPREQUEST` message. The client then broadcasts the `DHCPREQUEST` on the local hardware broadcast address to the `DHCP server` UDP port. Once a `DHCPACK` message with an XID field matching that in the client's `DHCPREQUEST` message arrives from any server, the client is initialized and moves to BOUND state. The client records the lease expiration time as the sum of the time at which the `DHCPREQUEST` message was sent and the duration of the lease from the `DHCPACK` message.

2. *Initialization with external assigned network addresses*. The client sends a DHCPINFORM message. The client may request specific configuration parameters by including the parameter request list option. The client generates and records a random transaction identifier and inserts that identifier into the xid field. The client places its own network address in the ciaddr field. The client should not request lease time parameters. The client then unicasts the DHCPINFORM to the DHCP server if it knows the server's address; otherwise it broadcasts the message to the limited (all 1s) broadcast address. DHCPINFORM messages must be directed to the DHCP server UDP port. Once a DHCPACK message with an xid field matching that in the client's DHCPINFORM message arrives from any server, the client is initialized.

3. *Use of broadcast and unicast*. The DHCP client broadcasts DHCPDISCOVER, DHCPREQUEST, and DHCPINFORM messages, unless the client knows the address of a DHCP server. The client unicasts DHCPRELEASE messages to the server. Because the client is declining the use of the IP address supplied by the server, the client broadcasts DHCPDECLINE messages. When the DHCP client knows the address of a DHCP server, in either INIT or REBOOTING state, the client may use that address in the DHCPDISCOVER or DHCPREQUEST rather than the IP broadcast address. The client may also use unicast to send DHCPINFORM messages to a known DHCP server. If the client receives no response to DHCP messages sent to the IP address of a known DHCP server, the DHCP client reverts to using the IP broadcast address.

4. *Reacquisition and expiration*. The client maintains two times, T1 and T2, that specify the times at which the client tries to extend its lease on its network address. T1 is the time at which the client enters the RENEWING state and attempts to contact the server that originally issued the client's network address. T2 is the time at which the client enters the REBINDING state and attempts to contact any server. T1 must be earlier than T2, which, in turn, *must* be earlier than the time at which the client's lease will expire.

To avoid the need for synchronized clocks, T1 and T2 are expressed in options as relative times. At time T1 the client moves to RENEWING state and sends (via unicast) a DHCPREQUEST message to the server to extend its lease. The client sets the ciaddr field in the DHCPREQUEST to its current network address. The client records the local time at which the DHCPREQUEST message is sent for computation of the lease expiration time. The client *must not* include a server identifier in the DHCPREQUEST message.

Any DHCPACK messages that arrive with an xid that does not match the xid of the client's DHCPREQUEST message are silently discarded. When the client receives a DHCPACK from the server, the client computes the lease expiration time as the sum of the time at which the client sent the DHCPREQUEST message and the duration of the lease in the DHCPACK message. The client has successfully reacquired its network address, returns to BOUND state, and may continue network processing.

If no DHCPACK arrives before time T2, the client moves to REBINDING state and sends (via broadcast) a DHCPREQUEST message to extend its lease. The client sets the ciaddr field in the DHCPREQUEST to its current network address. The client must not include a server identifier in the DHCPREQUEST message.

Times T1 and T2 are configurable by the server through options. T1 defaults to (0.5 * duration_of_lease). T2 defaults to (0.875 * duration_of_lease). Times T1 and T2 should be chosen with some random "fuzz" around a fixed value, to avoid synchronization of client reacquisition.

A client can choose to renew or extend its lease prior to T1. The server can choose to extend the client's lease according to policy set by the network administrator. The server *should* return T1 and T2, and their values *should* be adjusted from their original values to take account of the time remaining on the lease.

In both RENEWING and REBINDING states, if the client receives no response to its `DHCPREQUEST` message, the client *should* wait one-half of the remaining time until T2 (in RENEWING state) and one-half of the remaining lease time (in REBINDING state), down to a minimum of 60 s, before retransmitting the `DHCPREQUEST` message.

If the lease expires before the client receives a `DHCPACK`, the client moves to `INIT` state, *must* immediately stop any other network processing, and requests network initialization parameters as if the client were uninitialized. If the client then receives a `DHCPACK` allocating that client its previous network address, the client should continue network processing. If the client is given a new network address, it must not continue using the previous network address and should notify the local users of the problem.

DHCPRELEASE Message

If the client no longer requires use of its assigned network address, the client sends a `DHCPRELEASE` message to the server. Note that the correct operation of DHCP does not depend on the transmission of `DHCPRELEASE` messages.

20.7 Summary

This chapter presents information for those who require detail in the planning and design of TCP/IP networks. The information can also be used in detailed planning of Windows NT networks as well.

21
Remote Procedure Call

Open Network Computing (ONC) acknowledges Remote Procedure Call version 2 as of this writing. RPC is explained in this chapter first in general overview form, then in more detail, with some actual program examples and insights.

21.1 RPC and XDR: An Overview

RPC is a protocol. Technically speaking, it can operate over TCP or UDP as a transport mechanism. This is important because sometimes illustrations show RPC on either TDP or UDP. Applications use RPC to call a routine, thus executing like a client and making a call against a server on a remote host. This type of programming application represents a high-level, peer-oriented relationship between an application and an RPC server. Consequently, this means that these applications are portable to the extent that RPC is implemented.

EXternal Data Representation (XDR) protocol is within RPC. XDR data description language can be used to define data types when heterogeneous hosts are integrated. Having the capability to overcome the inherent characteristics of different architectures lends RPC and XDR a robust solution for distributed application communication. This language permits parameter requests to be made against a file of an unlike type. In short, XDR permits data-type definition in the form of parameters and transmission of these encoded parameters.

XDR provides data transparency by way of encoding (or encapsulating) data at the application layer so that lower layers and hardware do not have to perform any conversions. A powerful aspect of XDR is automatic data conversion performed via declaration statements and the XDR compiler. The XDR compiler generates required XDR calls, thus making the operation less manual in nature.

RPC implements a *portmapper* (similar to the Sun Microsystems's Portmapper), which starts on RPC server initialization. When RPC services start, the operating system assigns a port number to each service. These services inform the portmapper of these port numbers, program numbers, and other information required by the portmapper to know how to match a service with a requester.

Client applications issue a service request to a portmapper. The portmapper, in turn, identifies the requested service and returns the appropriate parameters to the requesting client application. In other words, the portmapper is similar in function to a manager knowing what services are available and their specific addressable locations.

The portmapper can be used in a broadcast scenario. For example, a requesting RPC call can broadcast a call to all hosts on a network. Applicable portmappers report back to the information sought after by the client. Hence, the term *remote procedure call* (RPC).

21.2 A Perspective on RPC and NFS

Network File Service (NFS) is a product of Sun Microsytems. It permits users to execute files without knowing the location of these files. They may be local or remote with respect to the user. Users can create, read, or remove a directory. Files themselves can be written to or deleted. NFS provides a distributed file system that permits a user to capitalize on access capabilities beyond their local file system.

NFS uses Remote Procedure Call (RPC). NFS uses RPC to make execution of a routine on a remote server possible. The idea behind NFS is to have one copy of it on a server that all users on a network can access, so that software (and updates) can be installed on one server and not on multiple hosts in a networked environment. NFS is based on a client/server model. However, with NFS a single NFS server can function to serve the request of many client requests.

Concentration in this chapter is on the specification of version 2 of the message protocol used in Remote Procedure Call. The message protocol iteself is specified with the eXternal Data Representation (XDR) language. I assume that the reader is somewhat familiar with XDR.

I have included explanations of commonly used RPC terms, clients, servers, calls, replies, services, programs, procedures, and versions. Each remote procedure call has two sides to it. One side is an active client side that makes the call to a server which sends back a reply. A *network service* is a collection of one or more remote programs. A *remote program* implements one or more remote procedures. These procedures and their parameters and results are documented in the specific pro gram's protocol specification. A server may support more than one version of a remote program in order to be compatible with changing protocols.

For example, a network file service may be composed of two programs. One program may deal with high-level applications such as file system access control and locking. The other may deal with low-level file input and output and have procedures such as *read* and *write.* A client of the network file service would call the procedures associated with the two programs of the service on behalf of the client.

The terms *client* and *server* in this context apply to a particular transaction, that is, a particular host or piece of software (which could be a process or program) that could operate in both roles at different times. For example, a program that supplies remote execution service could also be a client of a network file service.

21.3 The RPC Model

RPC protocol is based on the remote procedure call model, which is similar to the local procedure call model, in which the caller places arguments to a procedure in some well-specified location and which transfers control to the procedure, and eventually regains control. Next, the results of the procedure are extracted from the specified location and the caller continues execution.

The remote procedure call model is where one thread of control logically winds through two processes: the caller's process and a server's process. The caller process first sends a call message to the server process and waits (blocks) for a reply message. The call message includes the procedure's parameters, and the reply message includes the procedure's operational results. Once the reply message is received, the results of the procedure are extracted, and the caller's execution is resumed. On the server side, a process remains dormant, awaiting the arrival of a call message. When one arrives, the server process extracts the procedure's parameters, computes the results, sends a reply message, and then awaits the next call message.

In this model, only one of the two processes is active at any given time. However, this model is given only as an example. The ONC RPC protocol imposes no restrictions on the current model implemented, and other models are possible. For example, an implementation may have RPC calls operate asynchronously, enabling the client to do useful work while waiting for the reply from the server. Another option is to have the server create a separate task to process an incoming call, so that the original server can be free to receive other requests.

Remote procedure calls differ from local procedure calls by the following:

1. *Error handling.* The failure of the remote server or network must be handled when using remote procedure calls.

2. *Global variables and side effects.* Since the server does not have access to the client’s address space, hidden arguments cannot be passed as global variables or returned as side effects.

3. *Performance.* Remote procedures usually operate one or more orders of magnitude slower than local procedure calls.

4. *Authentication.* Remote procedure calls can be transported over unsecured networks, authentication may be necessary. Authentication prevents one entity from faking its own identity, pretending to be another (different) entity.

Even though there are tools designed to automatically generate client and server libraries for a given service, protocols must still be designed carefully. Careful attention to the details of each system can prevent major problems during the implemetation phase.

21.4 RPC Transports and Semantics

The RPC protocol can be implemented on several different transport protocols. The RPC protocol itself does not differentiate as to how a message is passed from one process to another. It is concerned only with the specification and interpretation of messages. Applications may wish to obtain information about the transport layer through an interface not specified in this document. For example, the transport protocol may impose a restriction on the maximum size of RPC messages, or it may be stream-oriented like TCP, with no size limit. The client and server must agree on their transport protocol choices.

RPC does not try to implement any kind of reliability. Hence, it is important that the application programmer be aware of this because an application may need to recognize the type of transport protocol underneath RPC. If it knows it is running on top of a reliable transport such as TCP, and that most of the work is already done for it. On the other hand, if it is running on top of UDP, it must implement its own timeout, retransmission, and duplicate detection policies because the RPC protocol does not provide these services.

Because of its transport independence, the RPC protocol does not attach specific semantics to the remote procedures or their execution requirements. Semantics can be inferred from (but should be explicitly specified by) the underlying transport protocol. For example, consider RPC running on top of an unreliable transport such as UDP. If an application retransmits RPC call messages after timeouts and does not receive a reply, the application cannot infer anything about the number of times the procedure was executed. If the application does receive a reply, then it can infer that the procedure was executed at least once.

A server may wish to retain previous information such as granted requests from a client and not regrant them in order to ensure some degree of at most one batch of execution semantics. A server can do this by taking advantage of the transaction ID that is packaged with every RPC message. The main use of this transaction ID is when the client RPC entity matches replies to calls. However, a client application may choose to reuse its previous transaction ID when retransmitting a call. The server may choose to remember this ID after executing a call and not execute calls with the same ID in order to achieve some degree of success in executing at most one-time-type semantics. The server is not allowed to examine this ID in any other way except as a test for equality.

On the other hand if RPC uses TCP the application can infer from a reply message that the procedure was executed exactly once, but if it receives no reply message, it cannot assume that the remote procedure was not executed. Even if a connection-oriented transport protocol like TCP is used, an application still needs timeouts and reconnection to handle server crashes and other system glitches.

There are other possibilities for transports besides datagram- or connection-oriented protocols. For example, a request-reply protocol such as VMTransport Protocol (VMTP) is perhaps a natural transport for RPC. RPC uses both TCP and UDP transport protocols.

The notion of binding a particular client to a particular service and transport parameters is *not* part of this particular RPC protocol specification. Binding is important and required but left up to a higher-level operating software.

One could think of the RPC protocol as the *jump-subroutine instruction* (JSR), as it has been called in eras past! The loader (binder) makes JSR useful, and the loader itself uses JSR to accomplish its task.

The RPC protocol provides the fields necessary for a client to identify itself to a service and vice versa in each call and reply message. Security and access control mechanisms can be built on top of this message authentication if desired. Several different authentication protocols can be implemented and/or supported. A required field in the RPC header indicates which protocol is being used.

Authentication parameters are opaque and open-ended to the rest of the RPC protocol. The *flavor* of a credential or verifier refers to the value of the `flavor` field in the `opaque_auth` structure. Flavor numbers like RPC program numbers are also administered centrally. It is possible however, to request new flavor numbers be assigned by applying through electronic mail to *rpc@sun.com;* or one can perform more research into the nature of requesting new flavor numbers. Credentials and verifiers are represented as variable-length opaque data (referring to the body field in the `opaque_auth` structure).

Null authentication is mandatory—it must be available in all implementations. System authentication is typically the program designer's choice. Many applications use this style of authentication, and availability of this flavor in an implementation will enhance interoperability.

Often calls must be made where the client does not care about its identity or the server does not care who the client is. In this case the flavor of the RPC message's credential, verifier, and reply verifier is `AUTH_NONE.` Opaque data associated with `AUTH_NONE` are undefined. It is standard consideration for most applications that the length of the opaque data be zero.

21.5 RPC Protocol Requirements

RPC protocol must supply the following: (1) unique specification of a procedure to be called, (2) provisions for matching response messages to request messages, and (3) provisions for authenticating the caller to service and vice versa.

Besides these requirements, features that detect the following are worth supporting. Protocol rollover errors, systems-specific implementation bugs, user error, and network administration seem to necessitate the following: (1) RPC protocol mismatches, (2) remote program protocol version mismatches, (3) protocol errors (e.g., a misspecification of a procedure's parameters), (4) reasons why remote authentication failed, and (5) any other reasons why the desired procedure was not called.

21.6 RPC Programs and Procedures

The RPC call message has three unsigned integer fields: the remote program number, remote program version number, and remote procedure number. These uniquely identify the procedure to be called. Program numbers are administered by a central authority which is handled via email: *rpc@sun.com*. Once program designers have a program number, they can implement their remote program. Generally, the first implementation has the version number 1. By having a version field of the call message identifying which protocol version the caller is using, it is easier to understand, troubleshoot, and interpret. Version numbers enable support for new and old protocols through the same server process.

The procedure number identifies the procedure to be called. These numbers are documented in the specific program's protocol specification. For example, a file service's protocol specification may state that its procedure number 5 is *read* and procedure number 12 is *write*.

Many times remote program protocols can change over the timespan of several versions, and so can the actual RPC message protocol. Hence, the call message also carries the RPC version number, which is always equal to two reflecting versions 2 RPC as presented here.

The *reply message* to a *request message* has enough information to distinguish the following error conditions: (1) the remote implementation of RPC does not support protocol version 2—the lowest and highest supported RPC version numbers are returned, (2) the remote program is not available on the remote system, (3) the remote program does not support the requested version number—the lowest and highest supported remote program version numbers are returned, (4) the requested procedure number does not exist, and (5) the server perceives the parameters to the remote procedure as garbage.

21.6.1 RPC Authentication

Provisions for authentication of caller to servers and vice versa are part of the RPC protocol. The call message has two authentication fields: the credential and the verifier. The reply message has one authentication field: the response verifier. The RPC protocol specification defines all three fields to be the following opaque type in the XDR language. Consider the following:

```
enum auth_flavor {
AUTH_NONE       =  0,
AUTH_SYS        =  1,
AUTH_SHORT      =  2
/* and more to be defined */
};
struct opaque_auth {
auth_flavor flavor;
opaque body<400>;
};
```

Any `opaque_auth` structure is an `auth_flavor` enumeration followed by up to 400 bytes which are opaque to (uninterpreted by) the RPC protocol implementation. Interpretation and semantics of the data contained within the authentication fields are specified by individual, independent authentication protocol specifications. If authentication parameters are rejected, the reply message contains information stating why they were rejected.

21.6.2 RPC Program Number Assignment

RPC program numbers are given out in groups of hexadecimal 20000000 (decimal 536,870,912) according to the following list:

0	`1fffffff`	defined by *rpc@sun.com*
20000000	`3fffffff`	defined by user
40000000	`5fffffff`	transient
60000000	`7fffffff`	reserved
80000000	`9fffffff`	reserved
a0000000	`bfffffff`	reserved
c0000000	`dfffffff`	reserved
e0000000	`ffffffff`	reserved

The first group is a range of numbers administered by *rpc@sun.com* and should be identical for *all* sites. The second range is for applications peculiar to a particular site. This range is intended primarily for debugging new programs. Program designers can request blocks of RPC program numbers in the first range. All correspondence regarding RPC numbers is between Sun and program developers and/or companies owning or developing the program. The third group is for applications that generate program numbers dynamically. The final groups are reserved for future use and should not be used.

21.7 Functionality of the RPC Protocol

The original design intent of the RPC protocol is for calling remote procedures. Typically, each call message is matched with a reply message. However, the protocol itself is a message-passing protocol with which other (non–procedure call) protocols can be implemented.

1. *RPC batch.* RPC batch is useful for sending a large sequence of call messages to a server. RPC batch typically uses reliable byte stream protocols for its transport, such as the TCP transport mechanism. With RPC batch the *client* never waits for a reply from the server, and the server does not send replies to RPC batch calls. A sequence of RPC batch calls is usually terminated by a legitimate remote procedure call operation. The purpose for this is to clear the link and get positive acknowledgment.

2. *Broadcast remote procedure calls.* In broadcast protocols, the client sends a broadcast call to the network and waits for numerous replies. This requires the use of packet-based protocols (UDP) as the transport protocol mechanism. Servers that support broadcast protocols generally respond only when the call is successfully processed and are silent in the face of errors; however, one should be aware that this varies by application.

RPC broadcast principles also apply to multicasting; thus, an RPC request can be sent to a multicast address.

21.8 RPC Message Protocol

The focus in this section is to define RPC message protocol in the XDR data description language.

```
enum msg_type {
CALL = 0,
  REPLY = 1
};
```

A reply to a call message can take on only one of two forms: the message was either accepted or rejected.

```
enum reply_stat {
MSG_ACCEPTED = 0,
MSG_DENIED = 1
  };
```

Given that a call message was accepted, the following is the status of an attempt to call a remote procedure:

```
enum accept_stat {
    SUCCESS = 0, /* RPC executed successfully
    */ PROG_UNAVAIL = 1, /* remote hasn’t exported program      */
PROG_MISMATCH = 2, /* remote can’t support version #      */
PROC_UNAVAIL
=  3, /* program can’t support procedure      */ GARBAGE_ARGS = 4, /* procedure
can’t decode params      */ SYSTEM_ERR = 5 /* errors like memory allocation
    failure
    */
    };
    Reasons why a call message was rejected:
    enum reject_stat {
    RPC_MISMATCH = 0, /* RPC version number ! =  2
    */ AUTH_ERROR = 1 /* remote can’t authenticate caller
    */
    };
    Why authentication failed:
    enum auth_stat {
    AUTH_OK = 0, /* success
    */
    /*
    * failed at remote end
    */ AUTH_BADCRED      =  1, /* bad credential (seal broken)      */
AUTH_REJECTEDCRED = 2, /* client must begin new session
    */ AUTH_BADVERF      =  3, /* bad verifier (seal broken)      */
AUTH_REJECTEDVERF = 4, /* verifier expired or replayed
    */ AUTH_TOOWEAK      =  5, /* rejected for security reasons
    */
    /*
    * failed locally
    */ AUTH_INVALIDRESP = 6, /* bogus response verifier      */
AUTH_FAILED      =  7 /* reason unknown      */
};
```

The *RPC message* can be described as follows. All messages start with a transaction identifier, `xid`, followed by a two-armed discriminated union. The union’s discriminant is a `msg_type` that switches to one of the two types of the message. The `xid` of a REPLY message always matches that of the initiating CALL message. Note that the `xid` field is only used for clients matching reply messages with call messages or for servers detecting retransmissions; the service side cannot treat this `id` as any type of sequence number.

```
struct rpc_msg {
        unsigned int xid;
        union switch (msg_type mtype) {
        case CALL:
        call_body cbody;
        case REPLY:
        reply_body rbody;
        } body;
        };
```

Body of an RPC call:

In version 2 of the RPC protocol specification, `rpcvers` must be equal to 2. The fields `prog, vers`, and `proc` specify the remote program, its version number, and the procedure within the remote program to be called. After these fields are two authentication parameters: `cred` (authentication credential) and `verf` (authentication verifier). The two authentication parameters are followed by the parameters to the remote procedure, which are specified by the specific program protocol.

The purpose of the authentication verifier is to validate the authentication credential. Note that these two items are historically separate, but are always used together as one logical entity.

```
struct call_body {
unsigned int rpcvers;
/* must be equal to two (2)
*/ unsigned int prog;
unsigned int vers;
unsigned int proc;
opaque_auth cred;
opaque_auth verf;
/* procedure specific parameters start here */
};
```

Body of a reply to an RPC call:

```
union reply_body switch (reply_stat stat) {
case MSG_ACCEPTED:
        accepted_reply areply;
case MSG_DENIED:
        rejected_reply rreply;
} reply;
```

Reply to an RPC call that was accepted by the server:

There could be an error even though the call was accepted. The first field is an authentication verifier that the server generates in order to validate itself to the client. It is followed by a union whose discriminant is an `enum accept_stat.` The `SUCCESS` arm of the union is protocol-specific. The `PROG_UNAVAIL, PROC_UNAVAIL, GARBAGE_ARGS,` and `SYSTEM_ERR` arms of the union are void. The `PROG_MISMATCH` arm specifies the lowest and highest version numbers of the remote program supported by the server.

```
struct accepted_reply {
opaque_auth verf;
       union switch (accept_stat stat) {
       case SUCCESS:
       opaque results[0];
       /*
       * procedure-specific results start here
       */
       case PROG_MISMATCH:
       struct {
       unsigned int low;
       unsigned int high;
       } mismatch_info;
       default:
       /*
       * Void. Cases include PROG_UNAVAIL, PROC_UNAVAIL,
       * GARBAGE_ARGS, and SYSTEM_ERR.
       */
       void;
       } reply_data;
};
```

Reply to an RPC call that was rejected by the server:

The call can be rejected for one of two reasons: the server either is not running a compatible version of the RPC protocol (`RPC_MISMATCH`) or rejects the identity of the caller (`AUTH_ERROR`). In case of an RPC version mismatch, the server returns the lowest and highest supported RPC version numbers. In case of invalid authentication, failure status is returned.

```
union rejected_reply switch (reject_stat stat) {
case RPC_MISMATCH:
       struct {
       unsigned int low;
       unsigned int high;
       } mismatch_info;
case AUTH_ERROR:
       auth_stat stat;
};
```

21.9 RPC Record Marking Standard

When RPC messages are passed on top of a byte-stream transport protocol such as TCP, it is necessary to delimit one message from another in order to detect and possibly recover from protocol errors. This is called *record marking* (RM). One RPC message fits into one RM record.

A *record* is composed of one or more record fragments. A *record fragment* is a 4-byte header followed by 0 to (2**31)–1 bytes of fragment data. The bytes encode an unsigned binary number; as with XDR integers, the byte order is from highest to lowest. The number encodes two values: a boolean, which indicates whether the fragment is the last fragment of the record (bit value 1 implies that the fragment is the last fragment); and a 31-bit unsigned binary value, which is the length in bytes of the fragment's data. The boolean value is the highest-order bit of the header; the length is the 31 low-order bits. This record specification is *not* in XDR standard form.

21.10 RPC Language

Just as there was a need to describe the XDR data types in a formal language, there is also a need to describe the procedures that operate on these XDR data types in a formal language as well. The RPC language is an extension to the XDR language, with the addition of *program, procedure,* and *version* declarations. The following example is used to describe the essence of the language.

21.10.1 RPC Language Example Service

Here is an example of the specification of a simple PING (Packet Internet Groper) program:

```
program PING_PROG {
/*
* Latest version
*/
version PING_VERS_PINGBACK {
void
    PINGPROC_NULL(void) = 0;
/*
    * Ping the client, return the round-trip time
    * (in microseconds). Returns –1 if the operation * timed out.
*/
int
PINGPROC_PINGBACK(void) = 1;
} = 2;
/*
    * Original version
*/
version PING_VERS_ORIG {
    void
PINGPROC_NULL(void) = 0;
} = 1;
} = 1;
const PING_VERS = 2;     /* latest version */
```

The first version described is `PING_VERS_PINGBACK` with two procedures: `PINGPROC_NULL` and `PINGPROC_PINGBACK. PINGPROC_NULL` takes no arguments and returns no results, but it is useful for computing round-trip times from the client to the server and back again. By convention, procedure 0 of any RPC protocol should have the same semantics, and never require any kind of authentication. The second procedure is used for the client to have the server do a reverse PING operation back to the client, and it returns the amount of time (in microseconds) that the operation used. The next version, `PING_VERS_ORIG,` is the original version of the protocol and it does not contain `PINGPROC_PINGBACK` procedure. It is useful for compatibility with old client programs, and as this program matures, it may be dropped from the protocol entirely.

21.10.2 RPC Language Specification

The RPC language is identical to the XDR language defined in RFC 1014, except for the added definition I am providing of the `program-def` described here:

```
program-def:
"program"identifier "{"
    version-def
    version-def *
"}"" = "constant" ";"
    version-def:
"version"identifier "{"
    procedure-def
    procedure-def *
"}" " = "constant" ";"
procedure-def:
type-specifier identifier "("type-specifier
(","type-specifier )* ")" " = " "constant" ";"
```

21.10.3 RPC Syntax Reference Information

The following keywords are added and cannot be used as identifiers: *program* and *version.*

• A *version* name cannot occur more than once within the scope of a program definition; nor can a version number occur more than once within the scope of a program definition.

• A procedure name cannot occur more than once within the scope of a version definition; nor can a procedure number occur more than once within the scope of version definition.

• Program identifiers are in the same namespace as constant and type identifiers.

• Only unsigned constants can be assigned to programs, versions, and procedures.

21.10.4 RPC System Authentication Method

The client may wish to identify itself, for example, as it is identified on a UNIX system. The flavor of the client credential is `AUTH_SYS.` The opaque data constituting the credential encodes the following structure:

```
struct authsys_parms {
unsigned int stamp;
string machinename<255>;
unsigned int uid;
unsigned int gid;
unsigned int gids<16>;
};
```

The `stamp` is an arbitrary ID which the caller machine may generate. The `machinename` is the name of the caller's machine. The `UID` is the caller's effective `user ID.` The `GID` is the caller's effective `group ID.` GIDs are a counted array of groups which contain the caller as a member. The verifier accompanying the credential should have `AUTH_NONE` flavor value (defined above). Note that this credential is unique only within a particular domain of machine names, `UIDs` and `GIDs.`

The flavor value of the verifier received in the reply message from the server may be `AUTH_NONE` or `AUTH_SHORT.` In the case of `AUTH_SHORT,` the bytes of the reply verifier's string encode an opaque structure. This new opaque structure may now be passed to the server instead of the original `AUTH_SYS` flavor credential. The server may keep a cache which maps shorthand opaque structures (passed back by way of an `AUTH_SHORT`-style reply verifier) to the original credentials of the caller. The caller can save network bandwidth and server CPU cycles by using the shorthand credential.

The server may remove the shorthand opaque structure at any time. If this happens, the RPC message will be rejected because of an authentication error. The reason for the failure will be `AUTH_REJECTEDCRED.` At this point, the client may wish to try the original `AUTH_SYS` style of credential.

Use of this particular way of authentication does not guarantee any security for the users or providers of a service. Authentication provided by this scheme can be considered legitimate only when applications using this scheme and the network can be secured externally and privileged transport addresses are used for the communicating endpoints (an example of this is the use of privileged TCP/UDP ports in UNIX systems—note that not all systems enforce privileged transport address mechanisms).

22 Integrating SNA and Frame Relay

This chapter discusses devices which serve as end stations (DTEs) on a public or private frame-relay network (e.g., provided by a common carrier or PTT. Because of the scope of the topic at hand, this chapter forgoes explanation regarding the behavior of those stations that are considered part of the frame-relay network (DCEs) other than to explain situations in which the DTE must react.

The frame-relay network provides a number of virtual circuits that form the basis for connections between stations attached to the same frame-relay network. The resulting set of interconnected devices forms a private frame-relay group which may be either fully interconnected with a complete "mesh" of virtual circuits, or only partially interconnected. In either case, each virtual circuit is uniquely identified at each frame-relay interface by a data-link connection identifier (DLCI). In most circumstances, DLCIs have strictly local significance at each frame-relay interface.

22.1 Frame Format

All protocols must encapsulate their packets within a Q.922 Annex A frame. Additionally, frames contain information necessary to identify the protocol carried within the protocol data unit (PDU), thus allowing the receiver to properly process the incoming packet. The format shall be as shown in Fig. 22-1.

The Q.922 addresses, as presently defined, contain 2 octets and a 10-bit DLCI. In some networks Q.922 addresses may optionally be increased to 3 or 4 octets. The control field is the Q.922 control field.

```
+-------------------------+
|   flag (7E haxadecimal)  |
+-------------------------+
|      Q.922 Address*      |
+--                    --+
|                        |
+-------------------------+
| Control (UI = 0x03)      |
+-------------------------+
| Optional Pad      (0x00) |
+-------------------------+
| NLPID                   |
+-------------------------+
|           .            |
|           .            |
|           .            |
|          Data            |
|           .            |
|           .            |
+-------------------------+
|  Frame Check Sequence     |
+--         .          --+
|      (two octets)       |
+-------------------------+
|  flag (7E hexadecimal)   |
+-------------------------+
```

Figure 22-1
Q.922 frame format.

The UI (0x03) value is used unless it is negotiated otherwise. The use of `XID` (`0xAF` or `0xBF`) is permitted and is discussed later. The `pad` field is used to align the remainder of the frame to a 2-octet boundary. There may be zero or one `pad` octet within the `pad` field and, if present, must have a value of zero.

The network-level protocol ID (NLPID) field is administered by ISO and CCITT. It contains values for many different protocols including IP, CLNP, and IEEE *Subnetwork Access Protocol* (SNAP). This field tells the receiver what encapsulation or what protocol follows. Values for this field are defined in the ISO/IEC TR 9577 standard specification. An NLPID value of 0x00 is defined within ISO/IEC TR 9577 standard specification as the *null network layer* or *inactive set.* Since it cannot be distinguished from a `pad` field, and because it has no significance within the context of this encapsulation scheme, an NLPID value of 0x00 is invalid under the frame-relay encapsulation. Some of the more commonly used NLPID values are listed in Sec. 2.11.

There is no commonly implemented minimum/maximum frame size for frame relay. A network must, however, support at least a 262-octet maximum. Generally, the maximum will be greater than or equal to 1600 octets, but each frame-relay provider will specify an appropriate value for its network. A frame-relay DTE, therefore, must allow the maximum acceptable frame size to be configurable.

The minimum frame size allowed for frame relay is 5 octets between the opening and closing flags assuming a 2-octet Q.922 address field. This minimum increases to 6 octets for 3-octet Q.922 address and 7 octets for the 4-octet Q.922 address format.

22.2 Frame Relay and SNA Interconnection Issues

Two basic types of data packets travel within the frame-relay network: routed packets and bridged packets. These packets have distinct formats and therefore, must contain an indicator that the destination may use to correctly interpret the contents of the frame. This indicator is embedded within the NLPID and SNAP header information.

For those protocols that do not have an NLPID already assigned, it is necessary to provide a mechanism to allow easy protocol identification. There is an NLPID value defined indicating the presence of a SNAP header.

A SNAP header is of the form shown in Fig. 22-2.

All stations must be able to accept and properly interpret both the NLPID encapsulation and the SNAP header encapsulation for a routed packet. The 3-octet organizationally unique identifier (OUI) identifies an organization which administers the meaning of the protocol identifier (PID) which follows. Together they identify a distinct protocol. Note that OUI 0x00-00-00 specifies that the following PID is an Ethertype.

22.3 Routed Frames in Frame Relay

Some protocols will have an assigned NLPID, but because the NLPID numbering space is so limited, not all protocols have specific NLPID values assigned to them. When packets of such protocols are routed over frame-relay networks, they are sent using the NLPID 0x80 (which indicates a SNAP follows) followed by SNAP. If the protocol has an Ethertype assigned, the OUI is 0x00-00-00 (which indicates that an Ethertype follows), and PID is the Ethertype of the protocol in use.

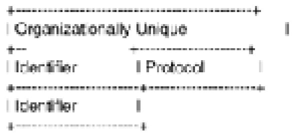


Figure 22-2
SNAP header.

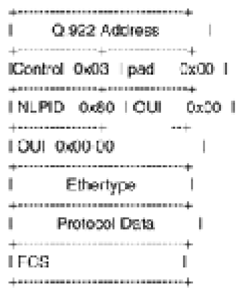


Figure 22-3
Format of routed frames
with Ethertypes.

There will be one `pad` octet to align the protocol data on a 2-octet boundary as shown in Fig. 22-3.

In the few cases when a protocol has an assigned NLPID, 48 bits can be saved using the format shown in Fig. 22-4.

The NLPID encapsulation does not require a `pad` octet for alignment, so none is permitted. In the case of ISO protocols, the NLPID is considered to be the first octet of the protocol data. It is unnecessary to repeat the NLPID in this case. The single octet serves both as the demultiplexing value and as part of the protocol data (refer to Sec. 22.9 for more details). Other protocols, such as IP, have an NLPID defined (0xCC), but it is not part of the protocol itself. See Fig. 22-5.

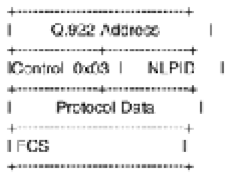


Figure 22-4
Format of routed NLPID protocol.

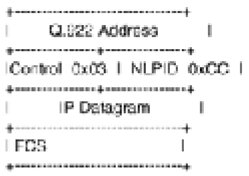


Figure 22-5
Format of routed IP datagram.

22.4 Bridged Frames in Frame-Relay Networks

The second type of frame-relay traffic is bridged packets. These packets are encapsulated using the NLPID value of 0x80 indicating SNAP. As with other SNAP-encapsulated protocols, there will be one `pad` octet to align the data portion of the encapsulated frame. The SNAP header which follows the NLPID identifies the format of the bridged packet. The OUI value used for this encapsulation is the 802.1 organization code 0x00-80-C2. The PID portion of the SNAP header (the 2 bytes immediately following the OUI) specifies the form of the MAC header, which immediately follows the SNAP header. Additionally, the PID indicates whether the original FCS is preserved within the bridged frame.

The IEEE 802.1 organization has reserved the values listed in Table 22-1 for use with frame relay.

In addition, the PID value 0x00-0E, when used with OUI 0x00-80-C2, identifies bridged protocol data units (BPDUs) as defined by 802.1(d) or 802.1(g).

A packet bridged over frame relay will, therefore, have one of the formats shown in Figs. 22-6 to 22-11.

Table 22-1 PID Values for OUI 0x00-80-C2

With preserved FCS	Without preserved FCS	Media
0x00-01	0x00-07	802.3/Ethernet
0x00-02	0x00-08	802.4
0x00-03	0x00-09	802.5
0x00-04	0x00-0A	FDDI
0x00-0B	802.6	

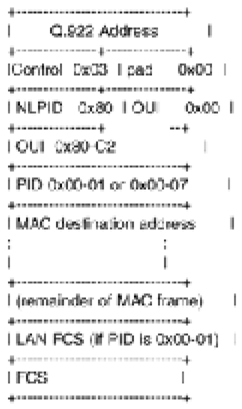


Figure 22-6
Format of bridged Ethernet/802.3 frame.

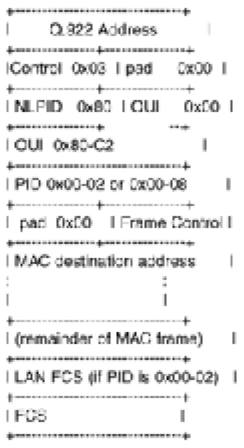


Figure 22-7
Format of bridged 802.4 frame.

```
+-------------------------------+
|        Q.922 Address          |
+---------------+---------------+
|Control  0x03  | pad     0x00  |
+---------------+---------------+
| NLPID   0x80  | OUI     0x00  |
+---------------+            ---+
| OUI  0x80-C2                  |
+-------------------------------+
| PID    0x00-03 or 0x00-09     |
+---------------+---------------+
| pad     0x00  | Frame Control |
+---------------+---------------+
| MAC destination address       |
:                           :
|                           |
+-------------------------------+
| (remainder of MAC frame)      |
+-------------------------------+
| LAN FCS (if PID is 0x00-03)   |
|                           |
+-------------------------------+
| FCS                           |
+-------------------------------+
```

Figure 22-8
Format of bridged 802.5 frame.

```
+-------------------------------+
|        Q.922 Address          |
+---------------+---------------+
|Control  0x03  | pad     0x00  |
+---------------+---------------+
| NLPID   0x80  | OUI     0x00  |
+---------------+            ---+
| OUI  0x80-C2                  |
+-------------------------------+
| PID 0x00-04 or 0x00-0A        |
+---------------+---------------+
| pad     0x00  | Frame Control |
+---------------+---------------+
| MAC destination address       |
:                           :
|                           |
+-------------------------------+
| (remainder of MAC frame)      |
+-------------------------------+
| LAN FCS (if PID is 0x00-04)   |
|                           |
+-------------------------------+
| FCS                           |
+-------------------------------+
```

Figure 22-9
Format of bridged FDDI frame.

```
+-------------------------------+
|        Q.922 Address          |
+---------------+---------------+
| Control 0x03  | pad     0x00  |
+---------------+---------------+
| NLPID   0x80  | OUI     0x00  |
+---------------+            ---+
| OUI  0x80-C2                  |
+-------------------------------+
|        PID  0x00-06           |
+---------------+---------------+ -------
|  Reserved     |     BEtag     | Common
+---------------+---------------+  PDU
|           BAsize              | Header
+-------------------------------+ -------
| MAC destination address       |
:                           :
|                           |
+-------------------------------+
| (remainder of MAC frame)      |
+-------------------------------+
|                           |
+-   Common PDU Trailer        -+
|                           |
+-------------------------------+
| FCS                           |
+-------------------------------+
```

Figure 22-10
Format of bridged 802.6 frame.

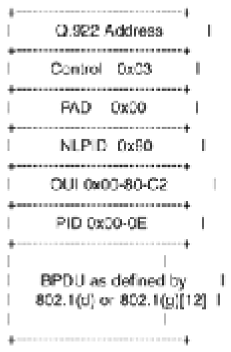


Figure 22-11
Format of BPDU frame.

In bridge 802.6 PDUs, there is only one choice for the PID value, since the presence of a CRC-32 is indicated by the CIB bit in the header of the MAC frame.

The Common Protocol Data unit (CPDU) header and trailer are conveyed to allow pipelining at the egress bridge to an 802.6 subnetwork. Specifically, the CPDU header contains the `BAsize` field, which contains the length of the PDU. If this field is not available to the egress 802.6 bridge, then that bridge cannot begin to transmit the segmented PDU until it has received the entire PDU, calculated the length, and inserted the length into the `BAsize` field. If the field is available, the egress 802.6 bridge can extract the length from the `BAsize` field of the common PDU header, insert it into the corresponding field of the first segment, and immediately transmit the segment onto the 802.6 subnetwork. Thus, the bridge can begin transmitting the 802.6 PDU before it has received the complete PDU. The common PDU header and trailer of the encapsulated frame should not be simply copied to the outgoing 802.6 subnetwork because the encapsulated `BEtag` value may conflict with the previous `BEtag` value transmitted by that bridge.

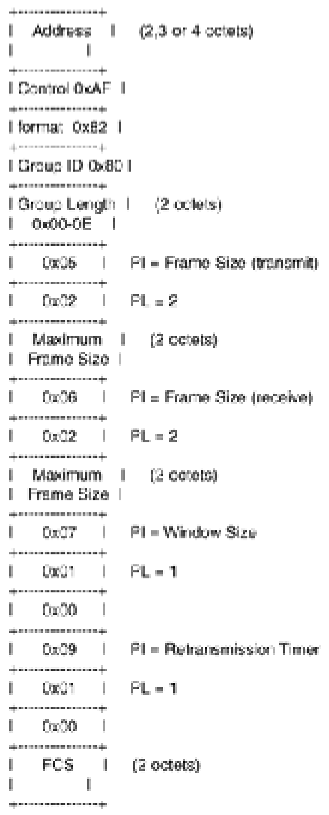
22.5 Data-Link-Layer Parameter Negotiation

Frame-relay stations may choose to support the eXchange Identification (XID) as specified in the Q.922 specification. This XID exchange allows the following parameters to be negotiated at the initialization of a frame-relay circuit: maximum frame size N201, retransmission timer T200, and the maximum number of outstanding Information (I) frames K.

A station may indicate its unwillingness to support acknowledged-mode multiple-frame operation by specifying a value of zero for the maximum window size, K. If this exchange is not used, these values must be statically configured by mutual agreement of data-link connection (DLC) endpoints, or must be defaulted to the values specified in the Q.922 document. Consider the following example of these values:

N201	260 octets
K	3 for a 16-kbit/s link, 7 for a 64-kbit/s link, 32 for a 384-kbit/s link, 40 for a ≥ 1.536-Mbit/s link
T200	1.5 s (see Q.922 for further details)

If a station supporting XID receives an XID frame, it shall respond with an XID response. In processing an XID, if the remote maximum frame size is smaller than the local maximum, the local system shall reduce the maximum size it uses over this DLC to the remotely specified value. Note that this shall be done before generating a response XID.



```
+---------------+
|   Address   |    (2,3 or 4 octets)
|             |
+---------------+
| Control 0xAF  |
+---------------+
| format  0x82  |
+---------------+
| Group ID 0x80 |
+---------------+
| Group Length  |    (2 octets)
|   0x00-0E     |
+---------------+
|    0x05     |    PI = Frame Size (transmit)
+---------------+
|    0x02     |    PL = 2
+---------------+
|   Maximum   |    (2 octets)
|  Frame Size |
+---------------+
|    0x06     |    PI = Frame Size (receive)
+---------------+
|    0x02     |    PL = 2
+---------------+
|   Maximum   |    (2 octets)
|  Frame Size |
+---------------+
|    0x07     |    PI = Window Size
+---------------+
|    0x01     |    PL = 1
+---------------+
|    0x00     |
+---------------+
|    0x09     |    PI = Retransmission Timer
+---------------+
|    0x01     |    PL = 1
+---------------+
|    0x00     |
+---------------+
|    FCS      |   (2 octets)
|             |
+---------------+
```

Figure 22-12
Nonuse of acknowledged-mode
multiple-frame operation.

Figure 22-12 illustrates the use of XID to specify nonuse of acknowledged-mode multiple-frame operation.

22.6 Frame-Relay Fragmentation Issues

Fragmentation allows the exchange of packets that are greater than the maximum frame size supported by the underlying network. In the case of frame relay, the network may support a maximum frame size as small as 262 octets. Because of this small maximum size, it is recommended, but not required, to support fragmentation and reassembly.

Unlike IP fragmentation procedures, the scope of the frame-relay fragmentation procedure is limited to the boundary (or DTEs) of the frame-relay network. The general format of fragmented packets is the same as that of any other encapsulated protocol. The most significant difference is that the fragmented packet will contain the encapsulation header. Thus, a packet is first encapsulated (with the exception of the address and control fields) as defined above. Large packets are then broken up into frames appropriate for the given frame-relay network and are encapsulated using the frame-relay fragmentation format. In this way, a station receiving fragments may reassemble them and then put the reassembled packet through the same processing path as a packet that had not been fragmented.

Frame-relay fragments are encapsulated using the SNAP format with an OUI of 0x00-80-C2 and a PID of 0x00-0D. Individual fragments will, therefore, have the format shown in Fig. 22-13.

The sequence field is a 2-octet identifier that is incremented every time a new complete message is fragmented. It allows detection of lost frames and is set to a random value at initialization. The reserved field is 4 bits long and is not currently defined. It must be set to 0. The final bit is a 1-bit field set to 1 on the last fragment and set to 0 for all other f ragments. The offset field is an 11-bit value representing the logical offset of this fragment in bytes divided by 32. The first fragment must have an offset of zero. Figure 22-14 shows how a large IP datagram is fragmented over frame relay. In this example, the complete datagram is fragmented into two frame-relay frames.

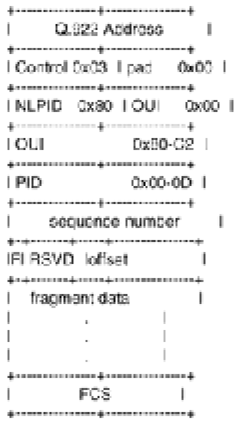


Figure 22-13
Frame-relay fragment
encapsulation.

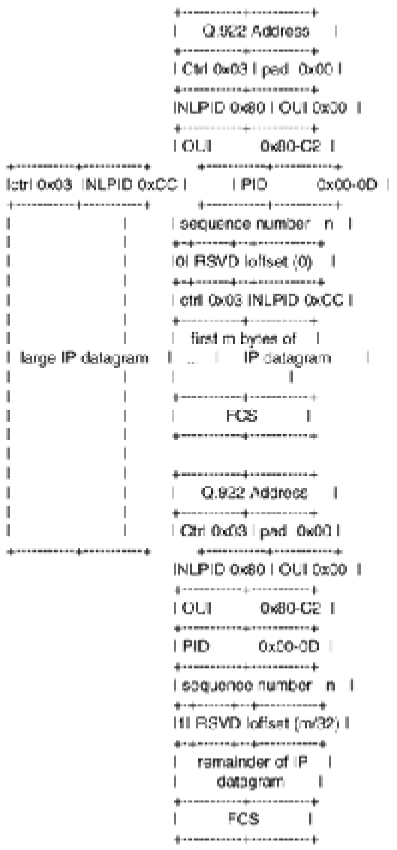


Figure 22-14
Frame-relay fragmentation example.

Fragments must be sent in order starting with a zero offset and ending with the final fragment. These fragments must not be interrupted with other packets or information intended for the same DLC. An end station must be able to reassemble up to 2000 octets and is suggested to support up to 8000 octet reassembly. If a fragment is corrupted or missing at any time during this reassembly process, the entire message is dropped. The upper-layer protocol is responsible for any retransmission in this case. There is no reassembly timer, nor is one needed. This is because the frame-relay service is required to deliver frames in order.

This fragmentation algorithm is not intended to reliably handle all possible failure conditions. As with IP fragmentation, there is a slight possibility of reassembly error and delivery of an erroneous packet. Inclusion of a higher-layer checksum greatly reduces this risk.

22.7 Frame-Relay Address Resolution

In certain situations, a frame-relay station may wish to dynamically resolve a protocol address. Address resolution may be accomplished using the standard Address Resolution Protocol (ARP) encapsulated within a SNAP-encoded frame-relay packet as shown in Fig. 22-15.

In Fig. 22-15 the ARP packet has the following format and values:

Data:

ar$hrd	16 bits	Hardware type
ar$pro	16 bits	Protocol type
ar$hln	8 bits	Octet length of hardware address (n)
ar$pln	8 bits	Octet length of protocol address (m)
ar$op	16 bits	Operation code (request or reply)
ar$sha	noctets	source hardware address
ar$spa	moctets	source protocol address
ar$tha	noctets	target hardware address
ar$tpa	moctets	target protocol address

ar$hrd - assigned to frame relay is 15 decimal
(0x000F)

ar$pro - see assigned numbers for protocol ID number for
the protocol using ARP. (IP is 0x0800).

ar$hln - length in bytes of the address field (2, 3, or 4)

ar$pln - protocol address length is dependent on the
protocol (ar$pro) (for IP ar$pln is 4).

ar$op - 1 for request and 2 for reply.

ar$sha - Q.922 source hardware address, with C/R, FECN,
BECN, and DE set to zero.

ar$tha - Q.922 target hardware address, with C/R, FECN,
BECN, and DE set to zero.

Because DLCIs within most frame-relay networks have only local significance, an end station will not have a specific DLCI assigned to itself. Therefore, such a station does not have an address to put into the ARP request or reply. Fortunately, the frame-relay network does provide a method for obtaining the correct DLCIs. The solution proposed for the locally addressed frame-relay network shown in Fig. 22-16 will work equally well for a network where DLCIs have global significance. The DLCI carried within the frame-relay header is modified as it traverses the network. When the packet arrives at its destination, the DLCI has been set to the value that, from the standpoint of the receiving station, corresponds to the sending station. For example, in Fig. 22-16, if station A were to send a message to station B, it would place DLCI 50 in the frame-relay header. When station B received this mes sage, however, the DLCI would have been modified by the network and would appear to B as DLCI 70.

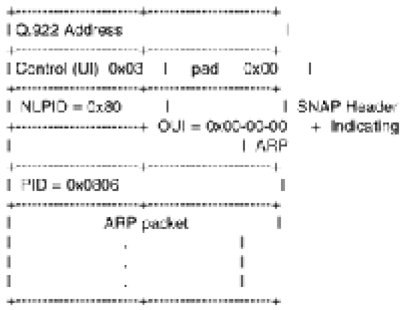


Figure 22-15
Address resolution

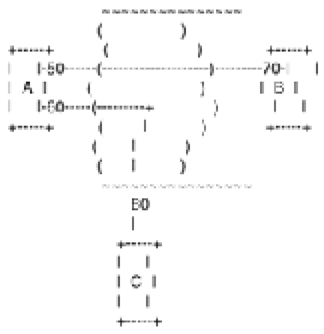


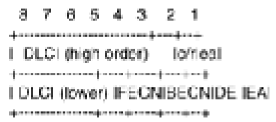
Figure 22-16
Conceptual view of a frame-relay network.

Lines between stations represent data-link connections (DLCs). The numbers indicate the local DLCI associated with each connection.

DLCI to Q.922 Address Table for the conceptual view frame-relay network

DLCI (decimal)	Q.922 address (hex)
50	0x0C21
60	0x0CC1
70	0x1061
80	0x1401

If you know about frame relay, you should understand the correlation between DLCI and Q.922 address. The translation between the DLCI and the Q.922 address is based on a 2-byte address length using the Q.922 encoding format. The format is:



```
 8  7  6  5  4  3   2  1
+-------------------+----+--
| DLCI (high order)    |c/r|ea|
+----------+----+----+----+
| DLCI (lower) |FECN|BECN|DE |EA|
+----------+----+----+----+
```

For ARP and its variants, the `FECN, BECN, C/R,` and `DE` bits are assumed to be 0.

When an ARP message reaches a destination, all hardware addresses will be invalid. The address found in the frame header will, however, be correct. Although it does violate the purity of layering, frame relay may use the address in the header as the sender hardware address. It should also be noted that the target hardware address, in both ARP request and reply, will also be invalid. This should not cause problems since ARP does not rely on these fields, and in fact, an implementation may zero-fill or ignore the target hardware address field entirely.

As an example of how this address replacement scheme may work, refer to Fig. 22-16. If station A (protocol address pA) wished to resolve the address of station B (protocol address pB), it would format an ARP request with the following values:

ARP request from A

ar$op	1 (request)
ar$sha	unknown
ar$spa	pA
ar$tha	undefined
ar$tpa	pB

Because station A will not have a source address associated with it, the source hardware address field is not valid. Therefore, when the ARP packet is received, it must extract the correct address from the frame-relay header and place it in the source hardware address field. In this way, the ARP request from A will become

ARP request from A as modified by B

ar$op 1	(request)
ar$sha	0x1061 (DLCI 70) from frame relay header
ar$spa	pA
ar$tha	undefined
ar$tpa	pB

Station B's ARP will then be able to store station A's protocol address and Q.922 address association correctly. Next, station B will form a reply message. Many implementations simply place the source addresses from the ARP request into the target addresses and then fill in the source addresses with its addresses. In this case, the ARP response would be

ARP response from B

ar$op 2	(response)
ar$sha	unknown
ar$spa	pB
ar$tha	0x1061 (DLCI 70)
ar$tpa	pA

Again, the source hardware address is unknown and, when the request is received, station A will extract the address from the frame-relay header and place it in the source hardware address field. Therefore, the response will become

ARP response from B as modified by A

ar$op 2	(response)
ar$sha	0x0C21 (DLCI 50)
ar$spa	pB
ar$tha	0x1061 (DLCI 70)
ar$tpa	pA

Station A will now correctly recognize station B having protocol address pB associated with Q.922 address 0x0C21 (DLCI 50). Reverse ARP (RARP) will work in exactly the same way. Still using the conceptual view of a frame-relay network as an example, consider station C as an address server; the following RARP exchanges will occur:

RARP request from A	**RARP request as modified by C**
ar$op 3 (RARP request)	ar$op 3 (RARP request)
ar$sha unknown	ar$sha 0x1401 (DLCI 80)
ar$spa undefined	ar$spa undefined
ar$tha 0x0CC1 (DLCI 60)	ar$tha 0x0CC1 (DLCI 60)
ar$tpa	pC ar$tpa pC

Station C will then look up the protocol address corresponding to Q.922 address 0x1401 (DLCI 80) and send the RARP response.

RARP response from C	**RARP response as modified by A**
ar$op 4 (RARP response)	ar$op 4 (RARP response)
ar$sha unknown	ar$sha 0x0CC1 (DLCI 60)
ar$spa pC	ar$spa pC
ar$tha 0x1401 (DLCI 80)	ar$tha 0x1401 (DLCI 80)
ar$tpa pA	ar$tpa pA

This means that the frame-relay interface must intervene only in the processing of incoming packets. In the absence of suitable multicast, ARP may still be implemented. To do this, the end station simply sends a copy of the ARP request through each relevant DLC, thereby simulating a broadcast.

The use of multicast addresses in a frame-relay environment is presently under study by frame-relay providers. At such time that the issues surrounding multicasting are resolved, multicast addressing may become useful in sending ARP requests and other “broadcast” messages.

Because of the inefficiencies of broadcasting in a frame-relay environment, a new address-resolution variation was developed. It is called *inverse ARP* and describes a method for resolving a protocol address when the hardware address is already known. In the frame-relay case, the known hardware address is the DLCI. Using inverse ARP for frame relay follows the same pattern as ARP and RARP use; that is, the source hardware address is inserted at the receiving station.

In our example, station A may use inverse ARP to discover the protocol address of the station associated with its DLCI 50. The inverse ARP request would be as follows:

InARP Request from A (DLCI 50)

ar$op   8   (InARP request)
ar$sha unknown
ar$spa pA
ar$tha 0x0C21 (DLCI 50)
ar$tpa unknown

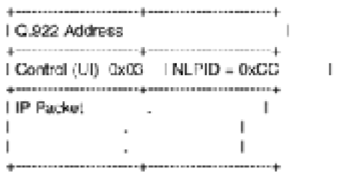
When station B receives this packet, it will modify the source hardware address with the Q.922 address from the frame-relay header. This way, the `InARP` (inverse ARP) request from A will become

ar$op   8   (InARP request)
ar$sha 0x1061
ar$spa pA
ar$tha 0x0C21
ar$tpa unknown

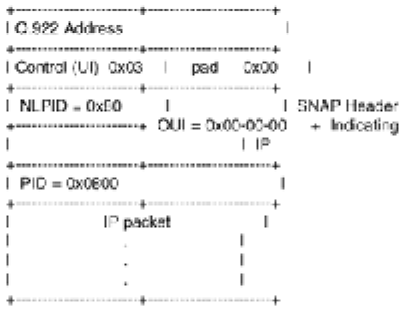
Station B will format an `InARP` response and send it to station A as it would for any ARP message.

22.8 IP over Frame Relay

Internet Protocol (IP) datagrams sent over a frame-relay network conform to the encapsulation described previously. Within this context, IP could be encapsulated in two different ways (see Fig. 22-17).



(a)



(b)

Figure 22-17
NLPID value indicating (*a*) IP and
(*b*) SNAP.

Although both encapsulations shown in Fig. 22-17 are supported under the given definitions, it is advantageous to select only one method as the appropriate mechanism for encapsulating IP data. Therefore, IP data will be encapsulated using the NLPID value of 0xCC indicating IP as shown in option 1 (Fig. 22-17*a*). This (option 1) is more efficient in transmission (48 fewer bits), and is consistent with the encapsulation of IP in X.25.

22.9 Other Protocols over Frame Relay

As with IP encapsulation, there are alternate ways to transmit various protocols within the scope of this definition. To eliminate the conflicts, the SNAP encapsulation is used only if no NLPID value is defined for the given protocol.

As an example of how this works, ISO CLNP has an NLPID defined (0x81). Therefore, the NLPID field will indicate ISO CLNP and the data packet will follow immediately. The frame would be as shown in Fig. 22-18.

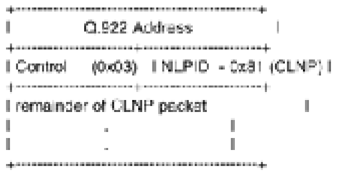


Figure 22-18
CLNP data packet identified
"by NLPID.

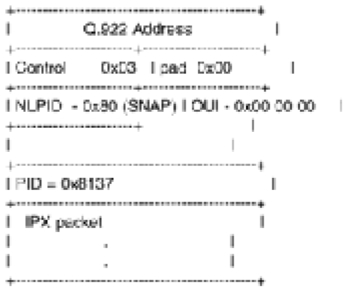


Figure 22-19
Encapsulation of IPX packet
by SNAP header.

In this example, the NLPID is used to identify the data packet as CLNP. It is also considered part of the CLNP packet and as such, the NLPID should not be removed before being sent to the upper layers for processing. The NLPID is not duplicated. Other protocols, such as IPX, do not have an NLPID value defined. As mentioned above, IPX would be encapsulated using the SNAP header. In this case, the frame would appear as shown in Fig. 22-19.

22.10 A Bridge Model for Frame Relay

The model for bridging in a frame-relay network is identical to the model for remote bridging as described in IEEE P802.1g, *Remote MAC Bridging*, and supports the concept of virtual ports. Remote bridges with LAN ports receive and transmit MAC frames to and from the LANS to which they are attached. They may also receive and transmit MAC frames through virtual ports to and from other remote bridges. A virtual port may represent an abstraction of a remote bridge's point of access to one, two, or more other remote bridges.

Remote bridges are statically configured as members of a remote bridge group by management. All members of a remote bridge group are connected by one or more virtual ports. The set of remote MAC bridges in a remote bridge group provides actual or "potential" MAC-layer interconnection between a set of LANs and other remote bridge groups to which the remote bridges attach.

In a frame-relay network there must be a full mesh of frame-relay VCs between bridges of a remote bridge group. If the frame-relay network is not full-mesh, then the bridge network must be divided into multiple remote bridge groups.

The frame-relay VCs that interconnect the bridges of a remote bridge group may be combined or used individually to form one or more virtual bridge ports. This gives flexibility to treat the frame-relay interface either as a single virtual bridge port, with all VCs in a group, or as a collection of bridge ports (individual or grouped VCs).

When a single virtual bridge port provides the interconnectivity for all bridges of a given remote bridge group (i.e., all VCs are combined into a single virtual port), the standard spanning-tree algorithm may be used to determine the state of the virtual port. When more than one virtual port is configured within a given remote bridge group, then an "extended" spanning-tree algorithm is required. Such an extended algorithm is defined in IEEE 802.1g. The operation of this algorithm is such that a virtual port is put into backup only if there is a loop in the network external to the remote bridge group.

The simplest bridge configuration for a frame-relay network is the LAN view where all VCs are combined into a single virtual port.

Frames, such as BPDUs, which would be broadcast on a LAN, must be flooded to each VC (or multicast if the service is developed for frame-relay services). Flooding is performed by sending the packet to each relevant DLC associated with the frame-relay interface. The VCs in this environment are generally invisible to the bridge. Specifically, the bridge sends a flooded frame to the frame-relay interface and does not "see" that the frame is being forwarded to each VC individually. If all participating bridges are fully connected (full-mesh), the standard spanning-tree algorithm will suffice in this configuration.

Typically LAN bridges learn which interface a particular end station may be reached on by associating a MAC address with a bridge port. In a frame-relay network configured for the LAN-like single bridge port (or any set of VCs grouped together to form a single bridge port), however, the bridge must associate a MAC address not only with a bridge port but also with a connection identifier. For frame-relay networks, this connection identifier is a DLCI. It is unreasonable and perhaps impossible to require bridges to statically configure an association of every possible destination MAC address with a DLC. Therefore, frame-relay LAN-modeled bridges must provide a mechanism to allow the frame-relay bridge port to dynamically learn the associations. To accomplish this dynamic learning, the receiving frame-relay interface must know to look into the bridged packet to gather the appropriate information.

A second frame-relay bridging approach, the point-to-point view, treats each frame-relay VC as a separate bridge port. Flooding and forwarding packets are significantly less complicated using the point-to-point approach because each bridge port has only one destination. There is no need to perform artificial flooding or to associate DLCIs with destination MAC addresses. Depending on the interconnection of the VCs, an extended spanning-tree algorithm may be required to permit all virtual ports to remain active as long as there are no true loops in the topology external to the remote bridge group.

It is also possible to combine the LAN view and the point-to-point view on a single frame-relay interface. To do this, certain VCs are combined to form a single virtual bridge port while other VCs are independent bridge ports.

Figure 22-20 illustrates the different possible bridging configurations. The dashed lines represent virtual circuits.

Since there is less than a full mesh of VCs between the bridges in this example, the network must be divided into more than one remote bridge group. A reasonable configuration is to have bridges A, B, and C in one group and bridges A and D in a second.

Configuration of the first bridge group combines the VCs interconnection the three bridges (A, B, and C) into a single virtual port. This is an example of the LAN view configuration. The second group would also be a single virtual port which simply connects bridges A and D. In this configuration the standard spanning-tree algorithm is sufficient to detect loops.

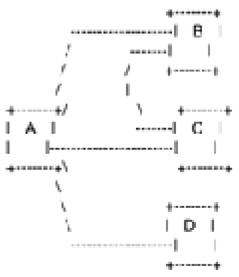


Figure 22-20
Four bridging configurations.

An alternative configuration has three individual virtual ports in the first group corresponding to the VCs interconnecting bridges A, B, and C. Since the application of the standard spanning-tree algorithm to this configuration would detect a loop in the topology, an extended spanning-tree algorithm would have to be used in order for all virtual ports to be kept active. Note that the second group would still consist of a single virtual port and the standard spanning-tree algorithm could be used in this group. Using the same drawing, one could construct a remote bridge scenario with three bridge groups. This would be an example of the point-to-point case. Here, the VC connecting A and B, the VC connecting A and C, and the VC connecting A and D are all bridge groups with a single virtual port.

22.11 NLPIDs and PIDs

Some commonly used NLPIDs are

0x00	Null network layer or inactive set (not used with frame relay)
0x80	SNAP
0x81	ISO CLNP
0x82	ISO ESIS
0x83	ISO ISIS
0xCC	Internet IP

Some PIDs of OUI 00-80-C2 are as follows:

With preserved FCS	Without preserved FCS	Media
0x00-01	0x00-07	802.3/Ethernet
0x00-02	0x00-08	802.4
0x00-03	0x00-09	802.5
0x00-04	0x00-0A	FDDI
	0x00-0B	802.6
	0x00-0D	Fragments
	0x00-0E	BPDUs as defined by 802.1(d) or 802.1(g)[12].

22.12 Connection-Oriented Procedures

The network-level protocol ID (NLPID) field is administered by ISO and CCITT. It contains values for many different protocols including IP, CLNP (ISO 8473) CCITT Q.933, and ISO 8208. The scheme's flexibility consists in the identification of multiple alternatives to identify different protocols used by any of the following and over frame-relay networks: (1) end-to-end systems, (2) LAN-to-LAN bridge and routers, or (3) a combination of these (1 and 2).

Figure 22-21 summarizes a generic encapsulation technique over frame-relay networks.

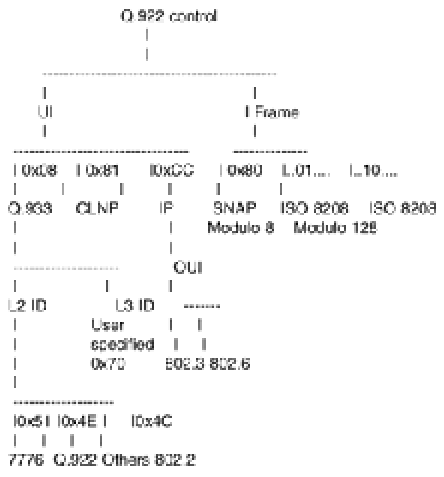


Figure 22-21
A generic encapsulation technique.

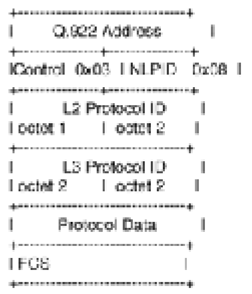


Figure 22-22
Format of other protocols
using Q.933 NLPID.

For those protocols which do not have an NLPID assigned or do not have a SNAP encapsulation, the NLPID value of 0x08, indicating CCITT recommendation Q.933, should be used. The four octets following the NLPID include both layer 2 and layer 3 protocol identification. The code points for most protocols are currently defined in ANSI T1.617, on the low-layer compatibility information element. There is also an escape for defining nonstandard protocols. See Figs. 22-22 and 22-23.

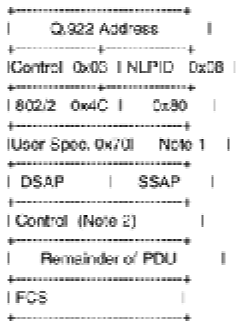


Figure 22-23
ISO 8802/2 with user-specified
layer 3.

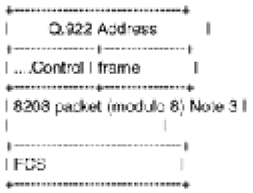


Figure 22-24
Format of ISO 8208
frame modulo 8.

Encapsulations using I frame (layer 2) is as follows. The Q.922 I frame is used to support layer-3 protocols which require the acknowledged-mode data-link layer (e.g., ISO 8208). The `C/R` bit (T1.618 address) will be used for command and response indications. See Figs. 22-24 and 22-25.

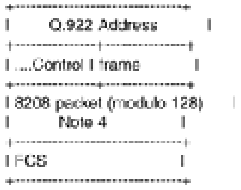


Figure 22-25
Format of ISO 8208 frame
modulo 128.

23
Integrating Frame Relay into SNA

SNA networks are constantly being migrated toward frame relay as a data-link-layer protocol. This shift in a data-link protocol use with SNA has been attributed to the fact that frame relay is gaining market share in the WAN environment because of its operating speed and reliability and that SNA is primarily a WAN protocol and implemented as such. Hence, migration of SNA over frame relay seems to be a natural progression.

It is not enough for applications and systems to be designed to accommodate such transfer; the conduit between points must be able to provide such data transfer. Frame relay is a typical choice for diverse networks, particularly those that use SNA as their backbone.

Frame relay provides real-time communication between end users. This is possible because frame relay serves as an interface into public and/or private networks. Frame-relay networks pass frames from origin to destination without intermediate nodes performing packet assembly/disassembly. Frame relay is also considered a protocol. A frame-relay control protocol is also defined, explaining how users make service requests in the network.

Frame relay is designed to support data in bursts and provide high speeds. Because of the nature of SNA networks, this makes it an ideal candidate. It is *not* a store-and-forward—based technology; rather it is a bidirectional conversational method of communication. Most frame-relay standards are concentrated at layers 1 and 2; however, standards do define the mechanism for upper-layer protocols to "hook" into frame relay. For example, this means that frame relay operates between users, that is, between origin and destination *networks.* Figure 23-1 best conveys this concept of frame-relay operation.

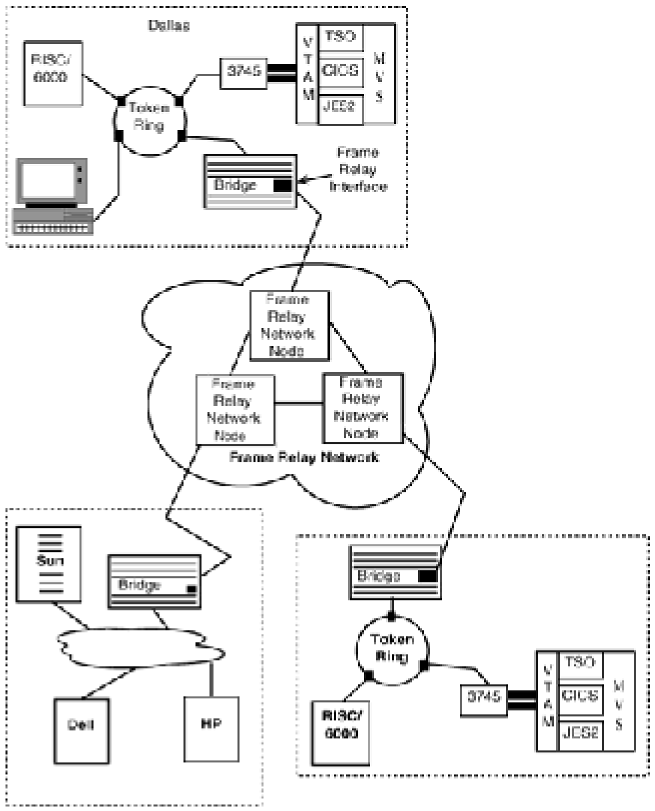


Figure 23-1
Conceptual view of frame relay.

Three sites using frame relay as an interface are shown in Fig. 23-1. These sites are dispersed throughout the frame-relay network. Additionally, it is used to connect multiple geographically different networks. This example shows each location implementing frame relay. Each physical location uses frame relay for an interface into what is considered a frame-relay network.

Frame relay has been defined by at least three entities: ANSI, the ITU (formerly called the CCITT), and the Local Management Interface (LMI) group. ANSI has a list of specifications. These standards generally have the ISDN name within them. The ITU has its list of recommendations. The ITU has two groups of standards that relate to frame relay: the I and Q groups. Last, LMI standards were created by Cisco Systems, Digital Equipment Corporation, Northern Telecom, Inc., and StrataCom, Inc. These four vendors collectively created standards that parallel ANSI and the ITU. This chapter also presents additional information related to frame relay and may help you in your migration of SNA into frame relay.

23.1 Standards Affecting Frame Relay

Before exploring various aspects of frame relay, this list of references may help if further information is needed to supplement that provided here in this chapter. Many different standards play a role in frame relay. The following list will help in pursuing additional information for frame relay.

ANSI	
T1.601	*Basic Access Interface*
T1.602	*ISDN Data Link Layer Signalling Specifications*
T1S1	*/90-75 Frame Relay Bearer Service, Architectural Framework Description*
T1S1	*/90-214 Core Aspects of Frame Protocol for Use with Frame Relay Bearer Service*
T1S1	*/90-213 Signalling Specification for Frame Relay Bearer Service*
T1.606	*ISDN Architectural Framework*
T1.607	*Digital Subscriber Signalling Service*
T1.617	*Standards Concerning Customer Interface*
T1.618	*Standards Concerning Customer Interface*
T1S1	*/90-051R2 Carrier to Customer Interface*

ITU Recommendations	
I.122	*Residual Error Rate*
I.233.1	*Frame Relay Bearer Services*
I.370	*Congestion Management for the ISDN Frame Relay Bearer Services*
I.372	*Frame Relaying Bearer Service Network-to-Network Interface Requirements*

I.555	*Internetworking between FMBS and Other Services*
Q.922	*ISDN Data Link Layer Specification for Frame Mode Basic Call Control*
Q.933	*DSS1 Signaling Specification for Frame Mode Basic Call Control*
G.703	*Physical and Electrical Characteristics of Hierarchical Digital Interfaces*
G.704	*Synchronous Frame Structures Used at Primary and Secondary Hierarchical Levels*
E.164	*Numbering Plan for the ISDN Era*
V.36	*Modems for Synchronous Data Transmission Using 60–108 kHz Group Band Circuits*
X.121	*International Numbering Plan for Public Data Networks*
X.76	*Network-to-Network Interface between Public Data Networking Providing the Frame Relay Data Transmission Service*

23.2 Frame-Relay Specifications and Concepts

Functionally, it helps to understand the correlation between frame relay and the specifications that undergrid it. Consider Fig. 23-2.

This figure shows the relation of the connections and the interfaces in a frame-relay network. Note that the user is shown as connecting to both public and private frame-relay network interfaces. This is the case because of the variable ways to connect to a frame-relay carrier.

This is another point that should be pointed out early: most frame-relay networks actually exist outside companies. What I mean here is that most frame-relay networks are implemented by service providers, and these serve as the technology of the backbone that links together different network components, in different locations. Hence, frame-relay networks actually require frame-relay equipment on the customer premises and at the location in which the service provider provides access to their backbone network.

The net effect of frame relay is that it is ideally suited for SNA because SNA is intended primarily for WANs and distributed over wide geographic areas.

23.3 Three Frame Relay-SNA Implementations

Functionally, frame relay and SNA interoperate well together. Frame relay operates at speeds from 56 kbits/s to T1/E1 circuit speeds. It can be used in private networks such as large corporate intranets that can span multiple time zones or continents or to perform the carrier backbone functions in the public sector.

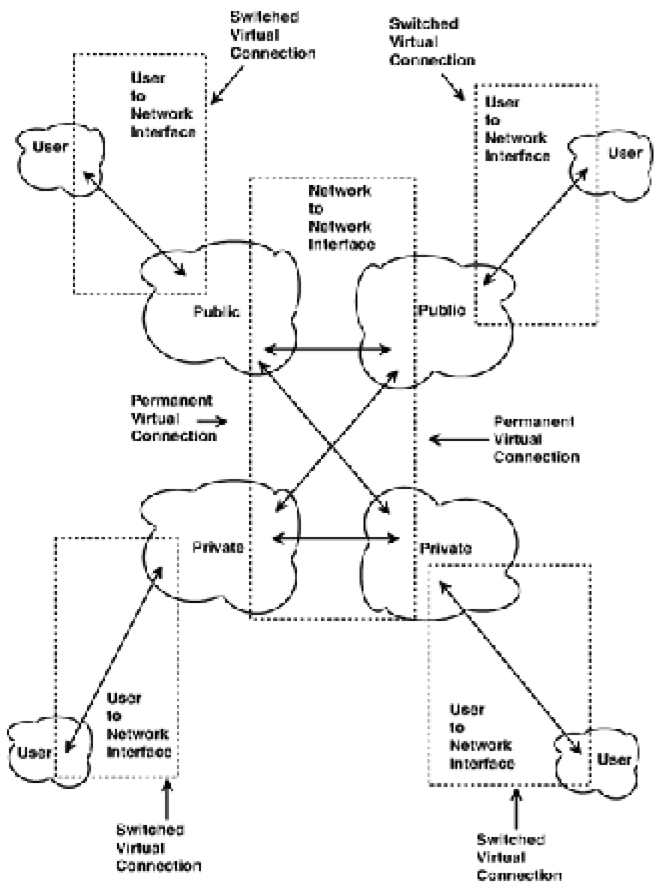

Switched Virtual Connection
Switched Virtual Connection
User to Network Interface
User
User to Network Interface
User
Network to Network Interface
Public
Public
Permanent Virtual Connection
Permanent Virtual Connection
Private
Private
User to Network Interface
User
User to Network Interface
User
Switched Virtual Connection
Switched Virtual Connection

Figure 23-2
Frame-relay connections and interfaces.

For the most part, frame-relay networks can provide a guaranteed bandwidth for each virtual circuit used (virtual circuits are discussed in further detail later); hence, the net effect of this is that no one single circuit can hog all the bandwidth. Granted exceptions do exist, and even in frame-relay there are no 100 percent guarantees, but it is a technology with measurable performance points.

Frame relay is sometimes called *fast packet* because error correction is performed by end stations in the network. This is in contrast to those nodes that would perform error checking in each hop within a network.

Characteristically, frame relay is connection-oriented at a data-link level. In this respect it is similar to Token-Ring technology. Don’t misinterpret what I am saying; frame relay and Token Ring do not operate the same way, but they share the characteristic of being connection-oriented at the data-link layer. To understand this more clearly, consider Ethernet, which is not connection-oriented, but is a data-link-level protocol.

Categorically, frame-relay networks fall into one of three types: (1) public, (2) private, and (3) hybrid. In a *public* frame-relay network, the frame-relay equipment is provided by a long-distance carrier; in this instance, a user merely needs the customer premise equipment to connect into this type of network. In a *private* frame-relay network, leased lines are maintained by a given entity and frame-relay equipment is exploited throughout the organization to create the frame-relay backbone. A *hybrid* frame-relay network is a mix between leased and switched connections and privately owned and maintained and publicly owned and maintained equipment.

Frame-relay implementation 1. Frame relay itself actually implements several standards and protocols at different layers and locations within a frame-relay network. Before examining these standards and protocols more thoroughly, consider Fig. 23-3, which shows an SNA implementation of frame relay. Specifically, it shows a private SNA—frame-relay network. Note that the NCPs and communication controllers are the SNA components that interact within the frame-relay cloud. Attached to the communication controllers are data terminal equipment (DTE) and SNA nodes. This figure shows the ability for interaction with the VTAM host node by any connected communication controller attached to the frame-relay cloud. In Fig. 23-3 the private frame-relay network is actually switched and uses typical telecommunication carrier lines. The functionality of these nodes throughout this network is such as to make interoperability among all nodes in the network with the host node possible. Another characteristic of this network presented in Fig. 23-3 is its inherent nature. Specifically, this network can carry multiple upper-layer protocols as applications require, can carry steady application data to and from users throughout the network, and can even accommodate batch jobs if remote use merits such interaction. The nature of frame relay is such that the virtual creation between each user-destination link can provide a wide range of bandwidth to the entire network while making its implementation economical.

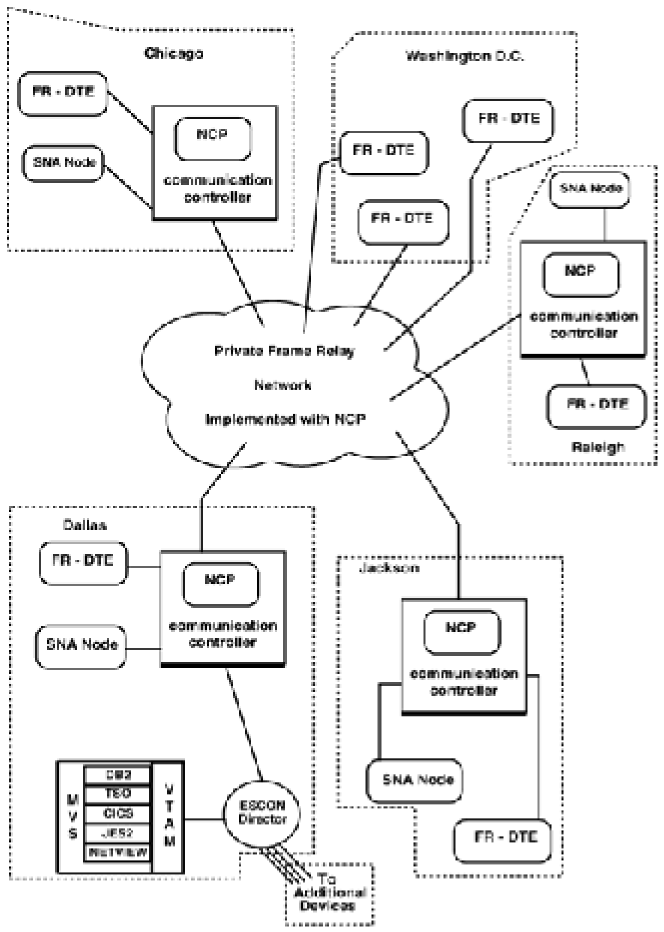


Figure 23-3
Private SNA–frame-relay network.

Frame-relay implementation 2. Frame relay and SNA can also be integrated as shown in Fig. 23-4.

Note that Fig. 23-4 differs from Fig. 23-3 in that the former shows pure SNA integrated over a private frame-relay network. Figure 23-3 shows a hybrid SNA environment with both SNA and nonnative SNA equipment alike. One advantage of this blueprint over the previous one is its manageability. In a pure SNA network, managing devices (both hardware and software) is straightforward because of the nature of their operation.

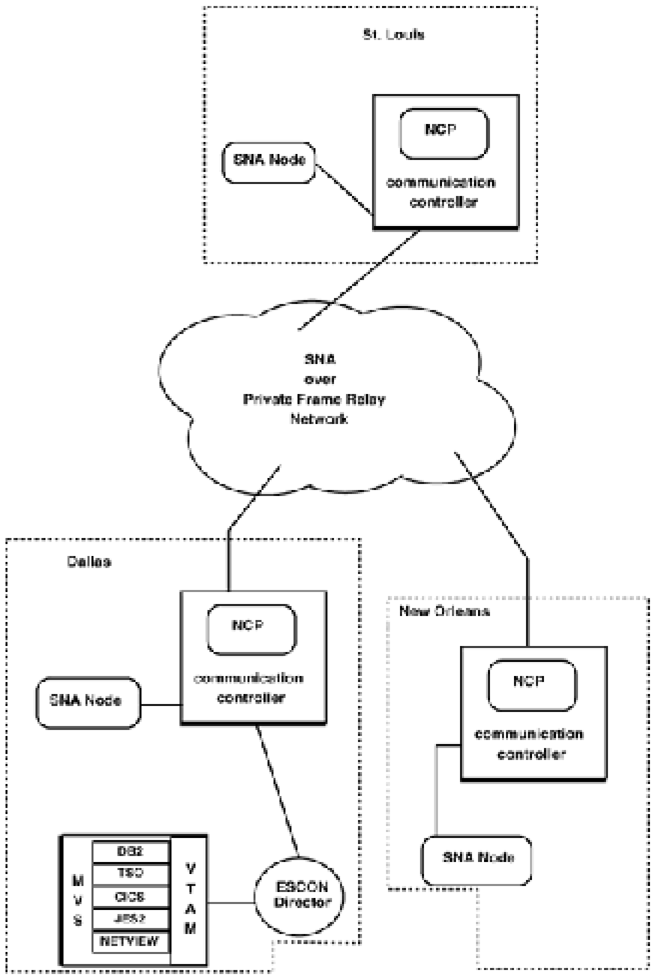


Figure 23-4
SNA integrated over private frame-relay network.

Frame-relay implementation 3. Another implementation of frame relay and SNA is shown in Fig. 23-5; note the public frame-relay network with the SNA network routed through it. This example is typical of many installations in which SNA is routed through frame relay. The network can easily be segmented into two components: the private part (all the SNA components) and the public part (the frame-relay network that is actually part of the long-distance carrier service). This implementation of frame relay integrated into SNA provides the best of both worlds. Management of the SNA network can occur within the confines of all the components that are SNA. On the other hand, management of the frame-relay network is the responsibility of the carrier service.

These are examples of how frame relay is integrated into SNA networks. There are other variations of these. In fact, because of the power behind SNA–frame-relay technology, a potentially unlimited number of implementation scenarios exists.

23.4 SNA Frame-Relay Principles

Frame relay operates on multiple principles, including virtual links, permanent virtual connections, and the data-link connection identifier.

23.4.1 The Virtual Link

A basic frame-relay principle is a virtual connection. The concept alone is interesting because the *virtual connection* may differ with respect to its lifetime, or the time it exists. But, in practical respects, the connection is permanent in the sense that it *remains* as long as necessary. Figure 23-6 shows a basic frame-relay example.

Figure 23-6 shows three hosts connected to a frame-relay node, physical links connecting each host to the node, memory inside the frame-relay device, and a connection mapping table within the frame-relay node. Host number 1's physical link supports two permanent virtual connections (PVCs). However, the physical links of hosts 2 and 3 support one PVC each.

The connection mapping table is at the heart of frame-relay operations within the frame-relay node shown in Fig. 23-2. The connections made are dynamic and based on requests from the incoming data stream. The connection mapping table is responsible for matching the request of source to the destination; this constitutes a route, or in frame-relay lingo, a virtual connection.

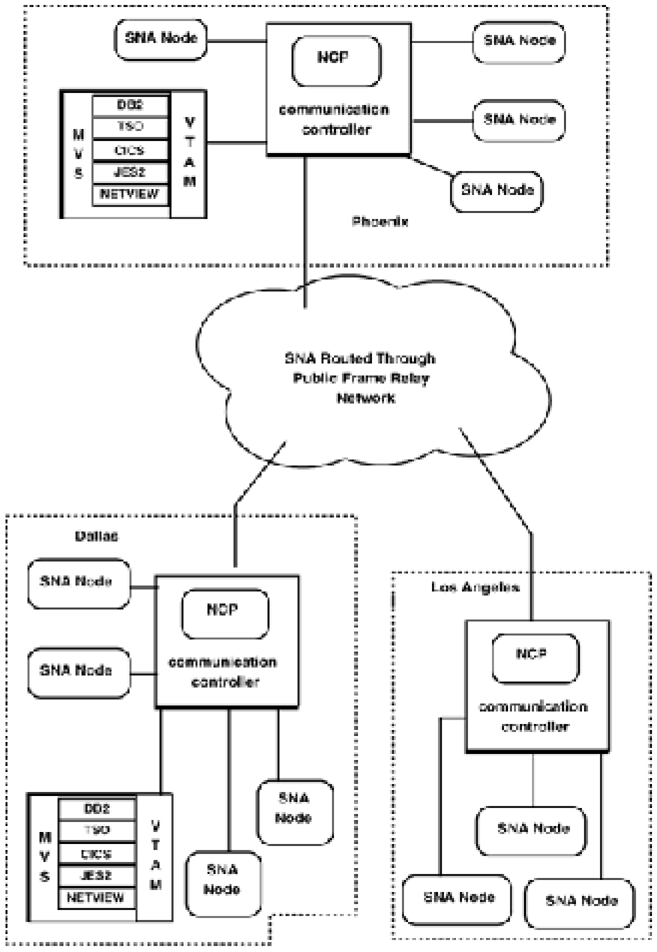


Figure 23-5
SNA routed through public frame-relay network.

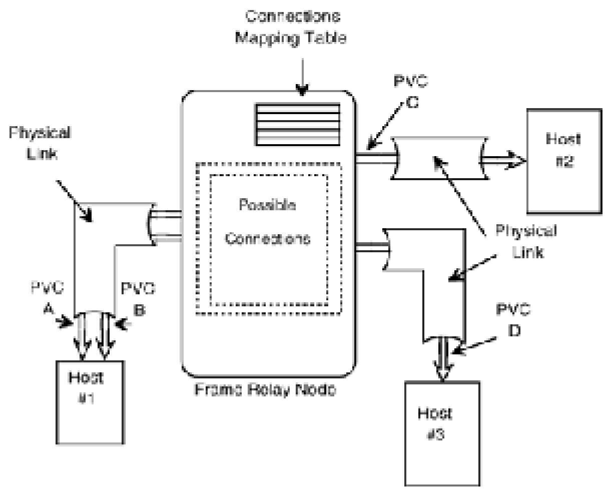


Figure 23-6
Enhanced view of a frame-relay node.

Figure 23-7 is similar to Fig. 23-6, but Fig. 23-7 shows the frames and a highlighted view of the mapping table.

Note that Fig. 23-7 shows PVC A 1 and PVC B 5 originating at host 1. Figure 23-7 also shows these connecting to PVC C 3 and PVC D 7, respectively. To be more precise, mapping between A 1 and C 3 and B 5 and D 7 is performed inside the frame-relay node.

23.4.2 Data-Link Connection Identifier

The inbound frame from host 1 via PVC A 1 contains a *data-link connection identifier* (DCLI), which is the local address in frame relay. The DCLI address is relevant in reference to the particular link. It identifies the frame and its type. This means that the same DCLI value could be used at both ends of the frame-relay network where each host connects via a particular link. Differentiating the DCLI is the physical link that the frame traverses and the host. Hence, more than one identifying factor is used for frame identification.

In the connection mapping table each entry includes the following information: node ID, link ID, and DCLI

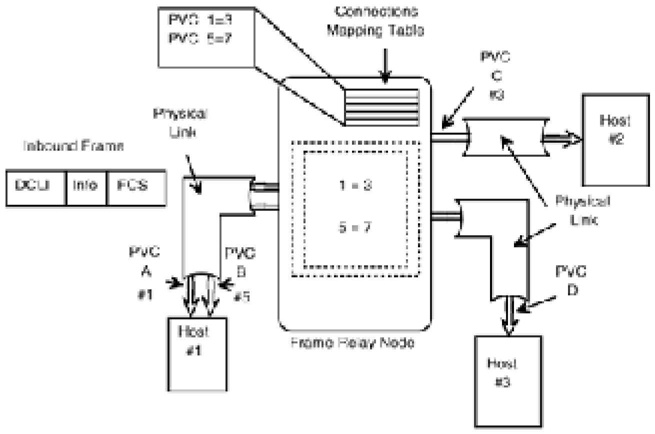


Figure 23-7
Highlighted view of the mapping table.

23.4.3 Frame-Relay Costs

Aside from the hardware and software required to implement a frame-relay connection, additional fees are incurred, such as the *commitment information rate* (CIR), which is the bandwidth available from one end to another; the *port access rate,* which refers to access into the frame-relay network; and the *network access charge,* which reflects the costs of the line connecting a given site to the access point in the frame-relay network.

Understanding these aspects of frame relay is important. For example, if data are transmitted in bursts, one must know what the burst data rate is. Another factor to understand that relates to this is the normal or average throughput required on a daily basis.

23.5 Frame-Relay Frame Components

Frame-relay frame components are shown in Fig. 23-8. Maximum frame size is 8250 bytes, and the minimum is generally considered 262 bytes.

Figure 23-8 is based on the CCITT I.441 recommendation. There are variations of this structure, primarily those with different methods of implementing addressing. A significant point to note about frame relay is that it utilizes the same standards as those of ISDN.

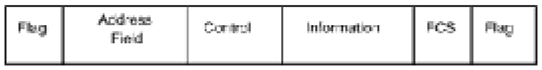


Figure 23-8
CCITT I.441 frame-relay frame format.

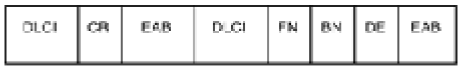


Figure 23-9
Highlighted view of the address field.

The flag field indicates the beginning of the frame, and the address field typically consists of the components shown in Fig. 23-9.

In Fig. 23-9 the `DLCI` fields identify a logical channel connection in a physical channel or port, thus identifying a predetermined destination (simply stated, it identifies the connection). The `CR` field contains a bit indicating a command response. The `EAB` field contains a bit set at either `1` or `0`. The field indicates extended addressing. The `FN` field is sometimes referred to as the forward explicit congestion notification (`FECN`). The bit in this field indicates whether congestion was encountered during the transfer from origin to destination. The `BN` field indicates that congestion was encountered on the return path in which congestion was encountered. The `DE` field is the *discard-eligibility field,* which indicates whether the frame can be disposed of during transfer if congestion is encountered. A bit setting of `1` means s a discard eligibility, whereas a bit setting of `0` indicates a higher setting for the frame and should not be discarded.

23.6 Frame-Relay Virtual Circuits Integrated into SNA

Different frame types of virtual circuits have been defined for use with frame relay. As a result, this technology is recommended for use with multimedia. In a sense these circuits represent different definable services, including switched virtual circuit (SVC), permanent virtual circuit (PVC), and multicast virtual circuit (MVC).

1. *Switched virtual circuit (SVC).* This type of circuit is similar to telephone usage. When the circuit is needed, a request is made. When the circuit is not needed, the circuit is not used. Information is passed from origin to destination to set up the call and to bring it down. Some information provided in the call setup phase includes bandwidth allocation parameters, quality-of-service parameters, and virtual channel identifiers.

2. *Permanent virtual circuit (PVC).* This type of connection is considered point-to-point. It could be thought of as a leased line in that it is dedicated. This type of circuit is used for long periods of time. Commands are still used to both set up and bring down the call. The difference between the PVC and a SVC is duration.

3. *Multicast virtual circuit (MVC).* MVCs are best described as being a connection between groups of users, through which individual users can use SVC connections as well as PVC connections. Technically, this type of connection is considered permanent and has, to date, generally been considered a local management interface (LMI) extension.

23.7 Frame-Relay Access Devices Integrated into SNA

Different devices can be used to connect devices into a frame-relay environment. Some of those are examined here.

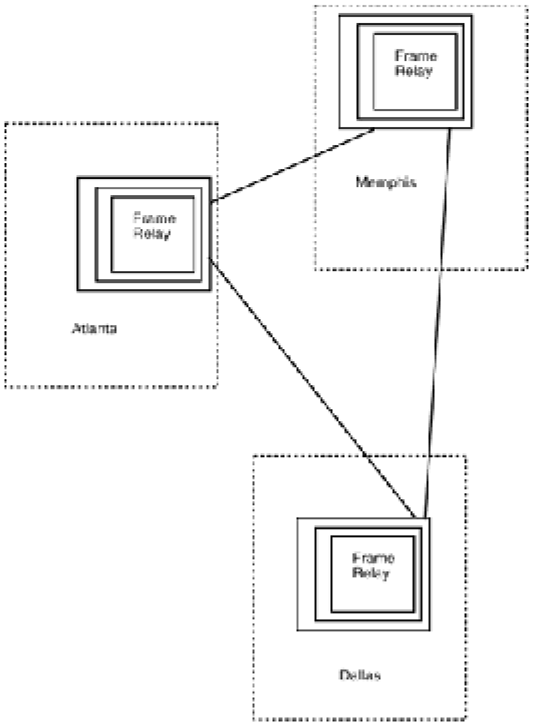


Figure 23-10
Network backbone components.

23.7.1 Switches

Frame-relay networks can be accessed via different types of devices. For example, switches similar to those accommodating X.25 provide a way to access frame-relay networks. However, these switches are typically implemented in the sense of creating a backbone. Figure 23-10 depicts an example of this type device implemented in three environments.

Figure 23-10 shows a network backbone made up of three components: switches in Memphis, Dallas, and Atlanta.

23.7.2 Network Device

A more focused view of the Memphis-Dallas-Atlanta equipment is represented in Fig. 23-11.

Figure 23-11 shows a network device (specifically a bridge) connecting a Token Ring and Ethernet network into the frame-relay environment.

23.7.3 Frame-Relay Access Device

A *frame-relay access device* (FRAD) is a particular piece of equipment that typically connotes capabilities including packet assembly/disas sembly, speeds of DS0, T1, or fractional T1. Most FRADs can handle multiple protocols and focus network traffic into a centralized managed facility, as shown in Fig. 23-12.

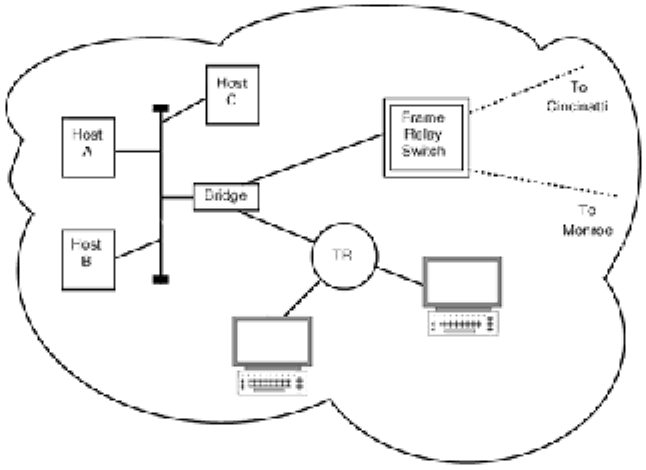


Figure 23-11
Focused view of FRAD equipment.

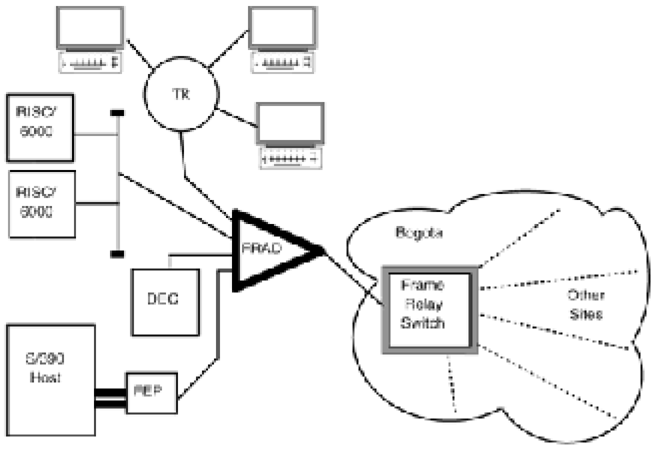


Figure 23-12
Highlighted view of an FRAD.

FRADs can be the best component to concentrate multiple devices into a single unit.

Other devices may be used in connecting a variety of resources into a frame-relay network. Many vendors offer frame-relay support as an additional function. One such example is IBM and their network control program (NCP), which operates on a front-end controller (FEC).

23.8 Consumer Tips

During my work and research with frame relay I encountered some information that is helpful the consumers purchasing frame-relay equipment. Some of those tips are included in this section.

1. Frame-relay standards are still being defined and implemented by vendors differently.

2. When considering a frame-relay device, ascertain what it does; specifically, determine whether it supports DCLI support; header bit support, and `FCS, FECN, BECN,` and `DE` bits within the frame. Also determine if congestion control is performed to standards, and if so, which one(s). Then determine whether multiple protocols are supported.

3. Find out how a given device supports `FECN, BECN,` and `DE` bits. How do they function in respect to congestion management? For example, one may say that `DE` is supported, but what does this mean? Does it mean the bit is read, or can it be set? The latter is important.

4. Is link management supported? If so, what type is it?

5. Does the device support the transparent mode?

6. Is the device switch-oriented, or is it primarily an access device used to provide access into a frame-relay network?

This is only a brief, basic list meant to help those beginning with frame relay. A more exhaustive list can be deduced from reference sources included in an earlier work (Taylor, *The McGraw-Hill Interworking Handbook*, p. 138).

23.9 ITU Q931 Standards Affecting Frame Relay

CALL PROCEEDING and OVERLAP SENDING are effected if the latter is used following either (1) the receipt by the network of a SENDING COMPLETE indication which the network understands or (2) analysis by the network that all call information necessary to effect call establishment has been received.

If a network can determine that access to the requested services and supplementary service is authorized and available, the network shall (1) send a CALL PROCEEDING message to the user, (2) stop timer T302, and (3) enter the OUTGOING CALL PROCEEDING state.

Similarly, if the network determines that a requested service or supplementary service is not authorized or not available, the network shall initiate call clearing in accordance with Sec. 5.3 (of ITU Q931) with one of the following causes: (1) bearer capability not authorized, (2) bearer capability not presently available, (3) service or option not available or unspecified, or (4) bearer service not implemented.

The CALL PROCEEDING message is sent to indicate that the requested call establishment has been initiated, and no more call establishment information will be accepted. If a supplementary service is not authorized or is not available, the procedure to be used is defined in the supplementary service control procedures.

On receiving the CALL PROCEEDING message, the user shall enter the OUTGOING-CALL PROCEEDING state. If the calling user employs timer T304, the user shall stop timer T304 when the CALL PROCEEDING message is received. If the calling user employs timer T304, then, on expiry of T304, the user shall initiate call clearing covered with cause 102, “recovery on time expiry.”

An alerting or connect indication received from the called party will stop timer T302 and cause an ALERTING or CONNECT message, respectively, to be sent to the calling user. No CALL PROCEEDING message shall be sent by the network. If, for symmetry purposes, the calling user employs timer T304, the user shall stop timer T304 on receiving the ALERTING or CONNECT message.

Network Functions at the Expiration of Timer T302

1. Initiate call clearing with cause “invalid number format” (incomplete number) sent to the calling user and with cause “recovery on timer expiry” sent toward the called user, if the network determines that the call information is definitely incomplete.

2. Send a CALL PROCEEDING message and enter the OUTGOING-CALL PROCEEDING state.

Called User Clearing

If the SETUP message has been delivered on a point-to-point data link and a RELEASE COMPLETE or DISCONNECT message is received before a CONNECT message has been received, the network performs these functions: (1) stop timer T303, T304, T310, or T301 (if running); and (2) continue to clear the user and clear the call to the calling user with the cause received in the RELEASE COMPLETE or DISCONNECT message.

If the timer T312 has expired and the network receives a DISCONNECT message from the called user after receiving a SETUP ACKNOWLEDGE, CALL PROCEEDING, or ALERTING message, but prior to receiving a CONNECT message, the network shall continue to clear the user. The network shall stop timer T304 (if running) for this user.

If the SETUP ACKNOWLEDGE or PROCEEDING messages have been received, the cause sent to the calling user shall be a cause received from the called user, giving preference to (in order of priority) “user busy,” “call rejected,” or any other appropriate cause sent by a called user.

If the SETUP ACKNOWLEDGE or CALL PROCEEDING messages have been received, the cause sent to the calling user shall be a cause received from the called user, giving preference to (in order of priority) #17 “user busy,” #21 “call rejected,” or any other appropriate cause sent by a called user.

On receipt of a RESUME message, the network enters the RESUME REQUEST state. After a positive validation of the call identity that relates to the suspended call containing a valid identity that relates to a currently suspended call, the network shall send a RESUME ACKNOWLEDGE message to the user, release the call identity, stop timer T307, and enter the ACTIVE state. The RESUME ACKNOWLEDGE message shall specify the B channel reserved for the call by the network by means of the channel identification element, coded “B channel is indicated, no alternative is acceptable.”

General Information Element Errors

The general-information element error procedures may also apply to information elements in code sets other than 0. In that case, the diagnostics in the cause information element may indicate information elements other than those in codeset 0 by applying the locking or nonlocking shift procedures.

Information Element out of Sequence

A variable-length information element which has a code value lower than that of the variable-length information element preceding it shall be considered as an out-of-sequence information element.

If the network or user receives a message containing an out-of-sequence information element, it may ignore this information element and continue to process the message. If this information is mandatory, and the network or user chooses to ignore this out-of-sequence information element, then the error-handling procedure for missing mandatory information elements is employed. If the ignored information element is nonmandatory, the receiver continues to process the message.

Some implementations may choose to process all the information elements received in a message regardless of the order in which they are placed.

Duplicated Information Elements

If an information element is repeated in a message in which repetition of the information element is not permitted, only the contents of information element appearing first shall be handled and all subsequent repetitions of the information element shall be ignored. When repetition of information elements is permitted, only the contents of permitted information elements shall be handled. If the limit on repetition of information elements is exceeded, the contents of information elements appearing first up to the limit of repetitions shall be handled and all subsequent repetitions of the information element shall be ignored.

Mandatory Information Element Errors

When a message other than SETUP, DISCONNECT, RELEASE, or RELEASE COMPLETE is received which has one or more mandatory information elements missing, no action should be taken on the message and no state change should occur. A STATUS message is then returned with the cause “mandatory information element is missing.”

When a SETUP or RELEASE message is received which has one or more mandatory information elements missing, a RELEASE COMPLETE message with the cause “mandatory information element is missing” shall be returned.

When a DISCONNECT message is received with the cause information element missing, the actions taken shall be the same as if a DISCONNECT message with the cause “normal, unspecified” were received with the exception that the RELEASE message sent on the local interface contains the cause “mandatory information element is missing.”

When a RELEASE COMPLETE message is received with a cause information element missing, it will be assumed that a RELEASE COMPLETE message was received with the cause “normal, unspecified.”

The sending or receipt of the STATUS message in such a situation will not directly affect the call state of either the sender or receiver. The side having received the STATUS message shall inspect the cause information element. If the STATUS message contains the cause “message type nonexistent or not implemented,” timer T322 shall continue to time for an explicit response to the STATUS ENQUIRY message. If a STATUS message is received that contains cause 30, “response to STATUS ENQUIRY,” timer T322 shall be stopped and the appropriate action taken, on the basis of the information in that STATUS message, relative to the current state of the receiver. If timer T322 expires and a STATUS message with cause 97, “message type nonexistent or not implemented,” was received, the appropriate action shall be taken, according to the information in that STATUS message, relative to the current call state of the receiver.

23.10 Excerpts and Highlights of ITU Recommendation G.703

The characteristics of interfaces at nonhierarchical bit rates, except $n \times 64$ kbit/s interfaces conveyed by 1544- or 2048-kbit/s interfaces, are specified in the respective equipment recommendations. The jitter specifications presented below are intended to be imposed at international interconnection points.

The interfaces described later in this section correspond to the ports T (output port) and T″ (input port) as recommended for interconnection in CCIR Recommendation AC/9 with reference to Report AH/9 of CCIR Study Group 9. (This report defines points T and T″.)

For signals with bit rates of $n \times 64$ kbit/s ($n = 2$ to 31) which are routed through multiplexing equipment specified for the 2048-kbit/s hierarchy, the interface shall have the same electrical physical characteristics as those for the 2048-kbit/s interface. For signals with bit rates of $n \times 64$ kbit/s ($n = 2$ to 23) which are routed through multiplexing equipment specified for the 1544-kbit/s hierarchy, the interface shall have the same electrical physical characteristics as those for the 1544-kbit/s interface.

In both directions of transmission, three signals can be carried across the interface: 64-kbit/s information signal, 64-kHz timing signal, and 8-kHz timing signal.

The 64-kbit/s information signal and the 64-kHz timing signal are mandatory. However, although an 8-kHz timing must be generated by the controlling equipment (e.g., PCM multiplex or time slot access equipment), it should not be mandatory for the subordinate equipment on the other side of the interface to either utilize the 8-kHz timing signal from the controlling equipment or to supply an 8-kHz timing signal. The detection of an upstream fault can be transmitted across the 64-kbit/s interface by transmitting an ALARM INDICATION signal toward the subordinate equipment. The interface should be bit-sequence-independent at 64 kbit/s.

23.10.1 Centralized Clock Interface

The term *centralized clock* is used to describe an interface in which, for both directions of transmission of the information signal, the associated timing signals are supplied from a centralized clock, which may be derived for example from certain incoming line signals.

23.10.2 Contradirectional Interface

The term *contradirectional* is used to describe an interface across which the timing signals associated with both directions of transmission are directed toward the subordinate equipment. The return loss at the input ports should have the minimum values listed in Table 23-1.

To provide nominal immunity against interference, input ports are required to meet the following requirements. A nominal aggregate signal, encoded as a 64-kbit/s codirectional signal, and having a pulse shape as defined in the pulse mask, shall have added to it interfering signal with the same pulse shape as the wanted signal. The interfering signal should have a bit rate within the limits specified in this recommendation but should not be synchronous with the wanted signal. The interfering signal shall be combined with the wanted signal in a combining network, with an overall zero loss in the signal path and with the nominal impedance 120 Ω to give a signal to interference ratio of 20 dB. The binary content of the interfering signal should comply with Recommendation 0.152 ($2^{11} - 1$ bit period). No errors shall result when the combined signal, attenuated by up to the maximum specified interconnecting cable loss, is applied to the input port.

Table 23-1 Overvoltage Protection Requirement

Frequency range, kHz	**Return loss, dB**
4–13	12 dB
13–256	18
256–384	14

If the symmetrical pair is screened, the screen shall be connected to the earth at the output port, and provision shall be made for connecting the screen of the symmetrical pair to earth, if required, at the input port.

23.10.3 Specifications at the input ports

The digital signals presented at the input ports should be as defined above but modified by the characteristics of the interconnecting pairs. The attenuation of these pairs at a frequency of 32 kHz should be in the range 0 ˅20to ˅203 ˅20dB. This attenuation should take into account any losses incurred by the presence of a digital distribution frame between the equipments.

The return loss at the input ports should have the minimum values shown in Table 23-2.

To provide nominal immunity against interference, input ports are required to meet the following requirement. A nominal aggregate signal, encoded as a 64-kbit/s contradirectional signal, and having a pulse shape as defined in the pulse mask, shall have added to it an interfering signal with the same pulse shape as the wanted signal. The interfering signal should have a bit rate within the limits specified in this Recommendation but should not be synchronous with the wanted signal. The interfering signal shall be combined with the wanted signal in a combining network with an overall zero loss in the signal path and with the nominal impedance 120 Ω to give a signal to interference ratio of 20 dB. The binary content of the interfering signal should comply with Recommendation 0.152 (2^{11} – 1 bit period). No errors shall result when the combined signal, attenuated by up to the maximum specified interconnecting cable loss, is applied to the input port.

Table 23-2 Overvoltage Protection Requirement

Frequency range, kHz		
Data signal	**Composite timing signal**	**Return loss, dB**
1.6–3.2	3.2–6.4	12
3.2–64	6.4–128	18
64–96	128–192	14

The return loss specification for both the data signal and the composite timing signal input ports. If the symmetrical pairs are screened, the screens shall be connected to the earth at the output port, and provision shall be made for connecting the screens of the symmetrical pairs to earth, if required, at the input port.

Interface at 1544 kbits/s

Interconnection of 1554-kbit/s signals for transmission purposes is accomplished at a digital distribution frame. The signal shall have a bit rate of 1544 kbit/s ±50 parts per million (ppm). One symmetrical pair shall be used for each direction of transmission. Test load impedance shall be 100 Ω, resistive. An AMI (bipolar) code or B8ZS code shall be used. Connecting line systems require suitable signal content to guarantee adequate timing information. This can be accomplished either by use of B8ZS code, scrambling, or by permitting not more than 15 spaces between successive marks and having an average mark density of at least 1 in 8. The shape for an isolated pulse measured at the distribution frame shall fall within the mask and meet the other requirements stated in ITU Standard G.703. For pulse shapes within the mask, the peak undershoot should not exceed 40 percent of the peak pulse (mark).

The voltage within a time slot containing a zero (space) shall be no greater than either the value produced in that time slot by other pulses (marks) ±0.1 of the peak pulse (mark) amplitude, whichever is greater in magnitude.

Interface at 6312 kbits/s

Interconnection of 6312-kbit/s signals for transmission purposes is accomplished at a digital distribution frame. The signal shall have a bit rate of 6312 kbit/s ±30 ppm. One symmetrical pair of characteristic impedance of 110 Ω, or one coaxial pair of characteristic impedance of 75 Ω shall be used for each direction of transmission. Test load impedance shall be 110 Ω resistive or 75 Ω resistive as appropriate.

A pseudo-ternary code shall be used as indicated in greater details presented forth in the ITU G.703 standard. The shape for an isolated pulse measured at the distribution frame shall fall within that which is set forth in the tables presented in ITU G.703.

The voltage within a time slot containing a zero (space) shall be no greater than either the value produced in that time slot by other pulses or ±0.1 of the peak pulse (mark) amplitude, whichever is greater in magnitude.

Interface at 32,064 kbits/s

Interconnection of 32,064 kbit/s for signals for transmission purposes is accomplished at a digital distribution frame. The signal shall have a bit rate of 32,064 kbit/s ±10 ppm. One coaxial pair shall be used for each direction of transmission. The test load impedance shall be 75 Ω ±5 percent resistive, and the test method shall be direct. A scrambled AMI code shall be used for this particular interface speed.

The voltage within a time slot containing a zero (space) shall be no greater than either the value produced in that time slot by other pulses (marks) or ±0.1 of the peak pulse (mark) amplitude, whichever is greater in magnitude.

For an all-1s pattern transmitted, the power measured in a 3-kHz bandwidth at the point where the signal arrives at the distribution frame shall be as follows:

16,032 kHz: +5 dBm to +12 dBm

32,064 kHz: at least 20 dB below the power at 16,032 kHz

The connectors and coaxial cable pairs in the distribution frame shall be 75 Ω ±5 percent.

Interface at 44,736 kbits/s

Interconnection of 44,736-kbit/s signals for transmission purposes is accomplished at a digital distribution frame. The signal shall have a bit rate of 44,736 kbit/s ±20 ppm. The signal shall have a frame structure consistent with Recommendation G.752. Specifically, it shall contain the frame alignment bits F_0, F_{11}, F_{12} and the multiframe alignment bits M_1 to M_7, as defined in that ITU standard document.

One coaxial pair shall be used for each direction of transmission. Test load impedance shall be 75 Ω ±5 percent resistive, and the test method shall be direct. The B3ZS code shall be used. The transmitted pulses have a nominal 50 percent duty cycle. The shape for an isolated pulse measured at the point where the signal arrives at the distribution frame falls within guidelines set forth in the G.703 document. The voltage within a time slot containing a zero (space) shall be no greater than either the value produced in that time slot by other pulses (marks) or ±0.05 of the peak pulse (mark) amplitude, whichever is greater in magnitude.

For an all-1s pattern transmitted, the power measured in a 3-kHz bandwidth at the point where the signal arrives at the distribution frame shall be as follows:

22,368 kHz: –1.8 to +5.7 dBm

44,736 kHz: at least 20 dB below the power at 22,368 kHz

The digital signal presented at the input port shall be defined above but modified by the characteristic of the interconnecting pair. The attenuation of this pair shall be assumed to follow an *f* law, and the loss at a frequency of 1024 kHz shall be in the range of 0 to 6 dB. This attenuation should take into account any losses incurred by the presence of a digital distribution frame between the equipments. For the jitter to be tolerated at the input port, refer to ITU Standard G.823. The return loss at the input port should have the minimum values listed in Table 23-3.

Table 23-3 Overvoltage Protection Requirement

Frequency range, kHz	Return loss, dB
51–102	12
102–2048	18
2048–3072	14

To ensure adequate immunity against signal reflections that can arise at the interface as a result of impedance irregularities at digital distribution frames and at digital output ports, input ports are required to meet the following requirement. A nominal aggregate signal, encoded into HDB3 and having a pulse shape as defined in the pulse mask, shall have added to it an interfering signal with the same pulse shape as the wanted signal. The interfering signal should have a bit rate within the limits specified in the ITU recommendation but should not be synchronous with the wanted signal. The interfering signal shall be combined with the wanted signal in a combining network, with an overall zero loss in the signal path and with the nominal impedance 75 Ω (in the case of coaxial pair interface) or 120 Ω (in the case of symmetrical pair interface), to give a signal to interference ratio of 18 dB. The binary content of the interfering signal should comply with Recommendation 0.151 (2^{15}–1 bit period). No errors shall result when the combined signal, attenuated by up to the maximum specified interconnecting cable loss, is applied to the input port. A receiver implementation providing an adaptive rather than a fixed threshold is considered to be more robust against reflections and should therefore be preferred.

The digital signal presented at the input port shall be as defined above but modified by the characteristic of the interconnecting pair. The attenuation of this pair shall be assumed to follow an *f* law, and the loss at a frequency of 4224 kHz shall be in the range 0 to 6 dB. This attenuation should take into account any losses incurred by the presence of a digital distribution frame between the equipments.

The return loss at the input port should have the minimum values listed in Table 23-4.

To ensure adequate immunity against signal reflections that can arise at the interface as a result of impedance irregularities at digital distribution frames and at digital output ports, input ports are required to meet the following requirement. A nominal aggregate signal, encoded into HDB3 and having a pulse shape as defined in the pulse mask, shall have added to it an interfering signal with the same pulse shape as the wanted signal. The interfering signal should have a bit rate within the limits specified in this recommendation but should not be synchronous with the wanted signal. The interfering signal shall be combined with the wanted signal in a combining network, with an overall zero loss in the signal path and with the nominal impedance 75 Ω to give a signal to interference ratio of 20 dB. The binary content of the interfering signal should comply with Recommendation 0.151 (2^{15} – 1 bit period). No errors shall result when the combined signal, attenuated by up to the maximum specified interconnecting cable loss, is applied to the input port.

Table 23-4 Overvoltage Protection Requirement

Frequency range, kHz	Return loss, dB
211–422	12
422–8,448	18
8,448–12,762	13

Interface at 34,368 kbits/s

At this interface the bit rate is 34,368 kbit/s ±20 ppm and the code is HDB3.

The digital signal presented at the input port shall be as defined above but modified by the characteristics of the interconnecting pair. The attenuation of this cable shall be assumed to follow approximately an *f* law, and the loss at a frequency of 17,184 kHz shall be in the range 0 to 12 dB.

The return loss at the input port should have the minimum values shown in Table 23-5.

To ensure adequate immunity against signal reflections that can arise at the interface as a result of impedance irregularities at digital distribution frames and at digital output ports, input ports are required to meet the following requirement. A nominal aggregate signal, encoded into HDB3 and having a pulse shape as defined in the pulse mask, shall have added to it an interfering signal with the same pulse shape as the wanted signal. The interfering signal should have a bit rate within limits specified in the ITU G.703 Recommendation but should not be synchronous with the wanted signal. The interfering signal shall be combined with the wanted signal in a combining network, with an overall zero loss in the signal path and with the nominal impedance 75 Ω to give a signal to interference ratio of 20 dB. The binary content of the interfering signal should comply with Recommendation 0.151 (2^{23} – 1 bit period). No errors shall result when the combined signal, attenuated by up to the maximum specified interconnecting cable loss, is applied to the input port.

Table 23-5 Overvoltage Protection Requirement

Frequency range, kHz	Return loss, dB
860–1,720	12
1,720–34,368	18
34,368–51,550	14

ITU G.703 Recommendation is complete in its presentation of the information as it relates to the excerpts of the important components here. I recommend that you obtain this ITU specification and the others listed throughout this chapter to enable you to work with frame relay in SNA.

23.11 Additional Standards Affecting Frame Relay

Many different standards play a role in frame relay. The following will help in pursuing additional information for frame relay:

I.122	Q.920	1.320	X.25
I.233	Q.921	1.320	X.31
I.130	Q.922	1.430	X.134
I.441	Q.930	1.431	X.213
I.450	Q.931	1.462	X.300
I.451			

23.12 Summary

Frame-relay principles were explained, including the concept of virtual links, the data-link connection indentifier, and the associated costs of frame relay beyond supporting hardware and software. Frame-relay frame components were presented and the fields that constitute a frame briefly explained.

The concept of virtual circuits was explained. The switched virtual circuit (SVC) is similar to a telephone, a permanent virtual circuit (PVC) is similar to a leased-line arrangement, and a multicast virtual circuit (MVC) is where a group of users can be reached through one multicast.

Access devices used in frame-relay networks were discussed. The basic functions of a switch network-related device, and frame-relay access device were explained. Different implementations may use one or more of these devices.

A brief list of consumer tips was included for the consumer who is new to frame relay. A list of reference sources was included to provide direction to additional information.

24
Frame-Relay Interoperability Standards

Frame relay relies on multiple standards to achieve its functionality. One such standard that operates around and in conjunction with frame relay is the ITU X.25 recommendation. It is a packet-switching technology that dates back to the mid to late 1960s and early 1970s. Around that time, many public network service providers were in somewhat of a dilemma. A networking technology was beginning, but no interface for connecting computers or other equipment was available. Hence, X.25 technology is rooted in that need for an interface into the data network.

The CCITT drafted its first specification for X.25 in 1974. Since that time, the ITU specifications have been updated every 4 years. They have been published, and are now available on the open market in their entirety on CD-ROM. Demystifying X.25 is easy; it merely involves a definition that explains how data are to be exchanged between data terminal equipment (DCE) and data circuit-terminating equipment (DCE).

The remainder of this chapter explores a number of aspects of X.25 technology, and one section (24.9.3) includes a list of references for those individuals needing further information.

24.1 Frame Relay and Switching Technology

Different types of switching technology are available. Three basic types are presented here.

24.1.1 Packet Switch

Fundamental to packet-switched networks is the fact that data traversing the network are divided into packets and moved from origin to destination. Each packet contains user data and control and identification information. One premise of packet-switching technology is that since packets are relatively small and each contains control and identification information, a computer port or line is occupied only when packets are moving through them. As a result, this means that port(s) and line(s) are available for others to use. Another premise on which packet-switching technology is based is that a packet-switched network has multiple nodes in the network, and hence uses multiple routes to reach any given destination from any given origin. Figure 24-1 is a conceptual view of a packet-switched network.

The packet-switched devices shown in Fig. 24-1 are also linked together in multiple ways, providing multiple paths from any given point to any other given point. Three examples of moving packets through the network are shown. First, host 1 sends data to host 2, host 3 sends data to host 4, and host 2 sends data to host 5.

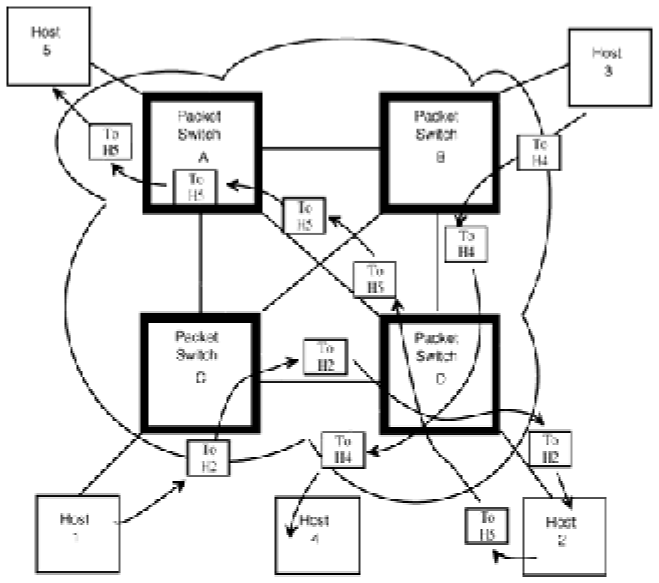


Figure 24-1
Conceptual view of packet switching.

The packet-switch node D has multiple traffic paths traversing it. This is possible because in a practical sense it is merely routing relatively small packets from one point to another.

24.1.2 Circuit Switch

A circuit-switch–based network is best described by analogy with, for example, the telephone network. For instance, two users may occupy a point-to-point line as long as they like (reasonably, of course). Circuit switching has its own characteristics. If a host utilizes such a service to communicate with another host, then numerous issues are the responsibility of the transmitting and receiving host.

Issues such as error checking, protocols in use, and flow control are the responsibility of the sending and receiving hosts. Figure 24-2 depicts an example of two hosts using a switched circuit.

Figure 24-2 shows a corporation and another business, respectively. This example shows these companies' systems communicating via circuit switching. Here, each Customer Information Communication Subsystem (CICS) (an IBM program) passes data to its respective transaction program. This scenario shows one modem in each site; however, in reality multiple modems may exist, thus providing redundancy. Moreover, multiple sites such as these may exist in the circuit-switched network.

24.1.3 Message Switch

Message switching is best described as a store-and-forward network. This type of network environment is used primarily for data. Figure 24-3 shows an example of a message-switched system with a corporation having a chain of stores covering a large geographic area.

Figure 24-3 shows four physical locations, each with a dedicated host used in this message-switched network. Note that each host in Jackson, Atlanta, Detroit, and Orlando have multiple disks attached to the hosts involved in the entire message-switched network.

Each location has satellite drugstores that connect to the host that is part of the backbone network. These typical examples include a *bus* network located in Jackson, the Token-Ring network in Atlanta, the *FDDI* network serving as the backbone network for the Orlando area, and two hosts serving Detroit. Each network or host connects to a central system in the cities shown. These hosts located in the four major cities are dedicated and make up the backbone of the message-switched network. However, this does not have to be the case. Examine Fig. 24-3 again; the backbone host could easily be located at the satellite offices.

The network works as shown in Fig. 24-3. Host 2 has a message for data destined for host 5 just outside Atlanta. Note that when the data leave host 2, they are stored in host A and then forwarded to host B and routed to host 5. Note that all hosts have multiple disk drives connected to them because they serve as the central repository for customer archive files.

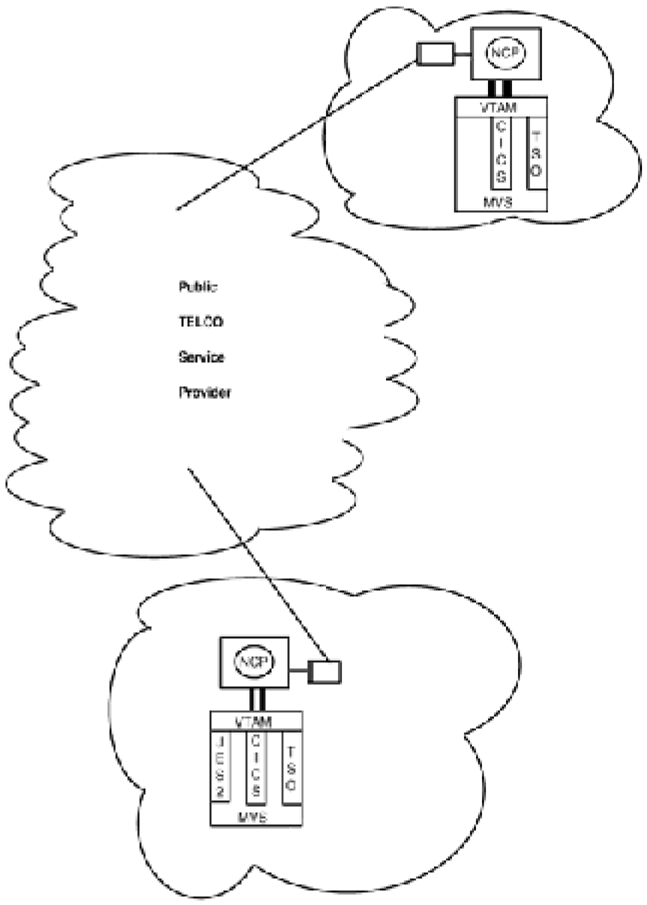


Figure 24-2
Conceptual view of a circuit-switching network.

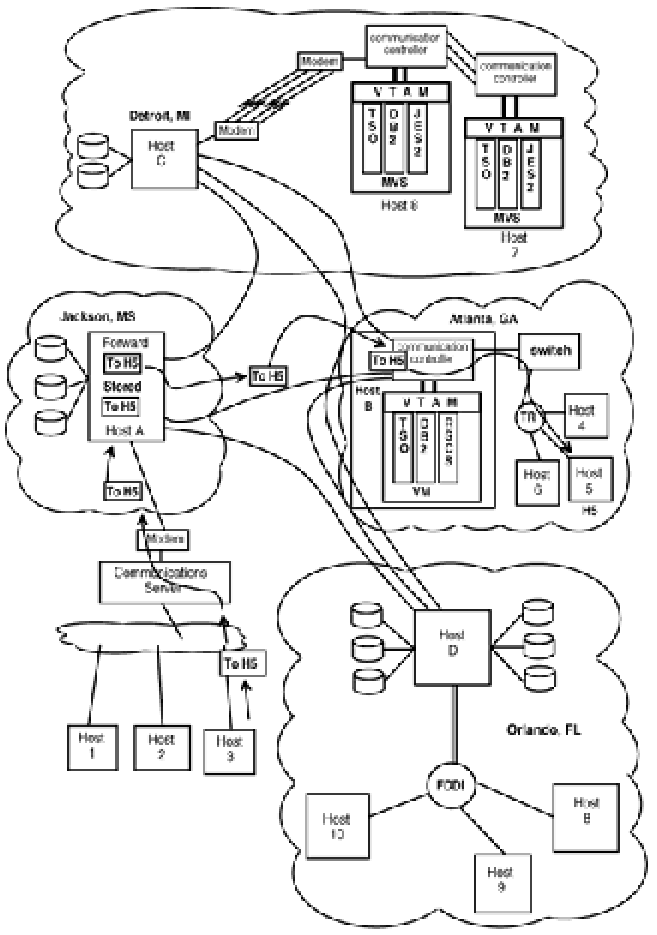


Figure 24-3
Conceptual view of a message-switching network.

The store-and-forward notion as in this network means that the message examines the packet for the destination address and then routes the packet onto the appropriate link for the destination. Put simply, in Fig. 24-3, computers (hosts) A, B, C, and D perform the *switching* func tion. The amount of time elapsed in the store-and-forward network for moving a data packet differs with respect to a number of variables. The amount of load, size of hosts, and other aspects factor into the elapsed-time equation. In some cases, data packets may be stored on disk and forwarded at a later time.

Each of these types of networks are complex and have their own nuances of how they operate, but efforts here are focused on packet-switching technology.

24.2 Layer Analysis: Highlights of X.25 Layers

X.25 layers do not directly correlate with the OSI model. Interestingly, the X.25 layers predate the OSI model in terms of structure by some 4 to 6 years. These layers and their functions are explained here.

X.25 layers make up the lower three layers in a protocol stack. Different methods of technology are supported at each layer. Figure 24-4 shows an example of X.25.

In Fig. 24-4 note that multiple interfaces are supported at the physical layer. At layer 2 there are two operating definitions, and at layer 3, packet-layer protocol is supported.

24.2.1 Physical Layer

This layer is where electrical transmission occurs and where a host interfaces with media in some way, whether the medium is hard or soft. Data at this level are represented by voltages which image binary 1s and 0s. According to Fig. 24-4, four voltages are defined.

1. *X.21.* X.21 is a CCITT recommendation used by X.25. X.21 defines the interface for synchronous operation between a DCE and DTE used in public data networks. This recommendation specifies more than just physical-layer interface standards; for example, it defines the numbers of pins on the interface and the action that each pin performs. X.25 can use this physical-layer recommendation. According to the CCITT 1992 recommendations which I use as a reference, X.21 is defined for a 15-pin connector and supports data-transfer rates of up to 9.6 kbits/s, operating in unbalanced mode. In balanced mode the recommendation calls for data rates of up to 64 kbits/s. The major point of this specification is to ensure synchronous operations.

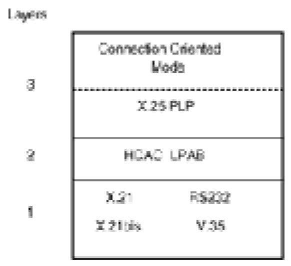


Figure 24-4
X.25 layers.

2. *X.21bis.* X.21bis is similar to the v.24 recommendation. X.21 defines leased-line operation between a DTE and a packet-switched network. It particularly states the interface parameters to perform synchronous operations with the V series modems. The v.24 recommendation is a widely implemented standard implemented among modems. The recommendation itself defines data transfer, flow control, and timing with other interface circuits. The importance of the standard is that it defines DTE and DCE interaction and functions. From the viewpoint of the CCITT, X.21bis was scheduled to be phased out and replaced by X.21. However, as mentioned earlier, we live in a de facto world. X.21bis utilizes the standards set forth for v.24. These standards include data transfer, control, and timing through its circuits.

3. *RS-232 and V.35.* X.25 can also use the RS-232 interface. The EIA RS-232 standard does not specify the physical connector; rather, it specifies the electrical, procedural, and functional characteristics of the interface.

The v.35 specification calls for higher throughput than does RS-232. It can be used for duplex operation over leased lines. The CCITT recommendations details the meaning of interfaces.

24.2.2 Data-Link Layer

Basic to the data-link layer is its purpose, which is to transfer frames from a DTE to a DCE error-free across a link. Another function of the data link is flow control and monitoring for errors and/or link failures. X.25 employs two techniques at the data-link layer: High-Level Data Link Control (HDLC) and Link Access Procedure (LAP). The latter is a subset of HDLC. Both are presented here to show their similarities and differences.

High-Level Data-Link Control (HDLC)

HDLC is a data-link-layer protocol used by X.25. HDLC supports half- and full-duplex transmission. It also supports switched, nonswitched, point-to-point, and multipoint communication links. A node using HDLC can be categorized into one of the following: (1) *primary station*—this category defines nodes in

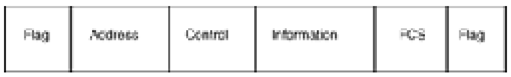


Figure 24-5
HDLC frame structure.

control of the data link, (2) *secondary station*—nodes in this category require a primary station for operation (in a sense they are dependent on the primary station), and (3) *peer station*—This is classified as a *peer* station because it can function as a primary- or secondary-type node.

HDLC is a bit-oriented protocol and is robust because it has a level of transparency; this means that either ASCII or EBCDIC data can be in the data field and are insignificant for transmission purposes. Figure 24-5 is an example of an HDLC frame.

The beginning flag field in the HDLC frame is set to binary 0 1 1 1 1 1 1 0. The same is true for the ending flag of the HDLC frame. Nodes attached to the data link monitor the data link examining the flags as they pass. The purpose of this is the need to supply continuously transmitted frames to keep the link active. The next field in the frame, the *address field,* contains the primary and secondary node addresses. Each node has a unique address and is located here during frame transmission.

The *control field* is next in the HDLC frame structure. This field contains commands, responses, and sequence numbers to maintain frame data flow between primary and secondary nodes. There are three different control frame formats: information, supervisory, and unnumbered. The *information* field holds user data. The *supervisory* control frame contains control information such as request for retransmission of frames, acknowledgment of frame receipts, and other control information such as requesting temporary suspension of frame transmission. The unnumbered frame performs functions such as initialization of the link, disconnection of the link, and other link-related functions. The information field exists if the information format is used in the control field for user data. If so, it contains user data. The frame-check sequence performs error checking to determine whether an error has occurred between the sending and receiving nodes. The ending flag is the same as the beginning flag field.

Link Access Protocol Balanced (LAPB)

LAPB is considered a subset of HDLC and is prevalent in X.25 implementations. Other subsets of HDLC exist, but the focus here is on LAPB because of its widespread use with X.25. The basic difference between LAPB and HDLC is that the former considers DTEs and DCEs as peers, meaning that either can initiate communications with the other. LAPB can operate in this manner because it is balanced and hence considers each node capable of initiating the link.

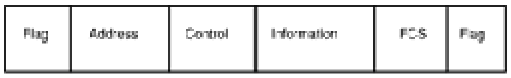


Figure 24-6
LAPB frame structure.

Figure 24-6 shows the frame structure of LAPB. Note that it is the same as for HDLC.

HDLC and LAPB frames differ in terms of the structure and contents of the control field. In either case the X.25 packet and data from higher layers are carried in the information field of both HDLC and LAPB frames.

24.2.3 Network Layer

At the network layer data from the upper layers are inserted into the X.25 packet. This is achieved because the network layer is actually split into two sublayers: (1) the connection-oriented sublayer, which is closest to the transport layer; and the X.25 *Packet-Layer Protocol* (PLP) sublayer. It is at this layer where the connection-oriented data are mapped to the PLP sublayer and then moved via the X.25 structure. A simple view of an X.25 format can be divided into the header and user data portions as shown in Fig. 24-7.

Figure 24-7 is a simple view of a packet structure containing data. Numerous types of packets are used in X.25. For example, different packet types are used for call setup, flow control, restart, data and interrupt, and for diagnostic and registration purposes.

Figure 24-8 shows an example of a packet used for call setup.

The *general format identifier* indicates information such as whether modulo 8 or 128 is used, whether user data are contained or control data, and whether the end-to-end acknowledgment bit is set to binary 1 or 0. This is where the Q bit (which distinguishes between different types of data) or the L bit (which distinguishes the addressing scheme) is set.

The *logical channel group number* and *logical channel number* indicate the user's session identification. The *packet-type identifier* indicates whether the packet is a call setup, reset, data transfer, or other type of packet. If the packet does contain data and more are to follow, then an *M* bit is set to indicate that more data will follow.

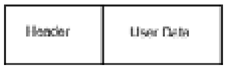


Figure 24-7
X.25 packet structure.

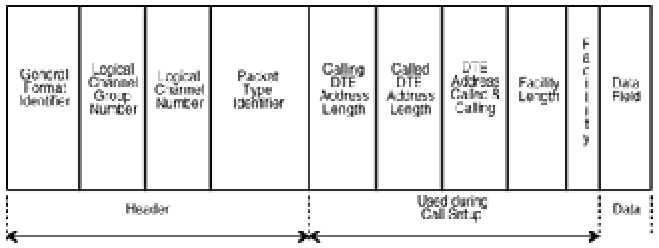

General Format Identifier
Logical Channel Group Number
Logical Channel Number
Packet Type Identifier
Calling DTE Address Length
Called DTE Address Length
DTE Address Called & Calling
Facility Length
Facility
Data Field
Header
Used during Call Setup
Data

Figure 24-8
Example of an X.25 call setup packet.

The calling DTE address length, called DTE address length, and the called DTE address that is being called and is calling indicate the DTE addresses of the respective entities placing and receiving these calls. Also, the first two fields mentioned here indicate the length of these addresses, respectively.

The *facility length field* is used to specify the length of the facility field because multiple facilities can be requested. Certain DTE-specified facilities may be put in this field or facilities offered by a public data network, for example. A few specific examples will clarify this point. A facility exists for a user to obtain information about the charges related to the call. Another example is call redirection; in this instance a call can be forwarded to another DTE. This is similar to the call-forwarding feature available through some telephone service providers. Regardless of the facility, it is placed in the facility field of the packet. The *data field* is user data from upper layers.

After a call is set up, the calling DTE address length, called DTE address length, DTE address (called and calling), along with the facility length field and the facility field, are no longer needed. The header and the data portion are used.

Generally, approximately eight groups of packets are identifiable: (1) call setup, (2) call clearing, (3) data, (4) diagnostic, (5) interrupt, (6) flow control, (7) reset, and (8) restart.

24.3 Terminology Used with Frame Relay and X.25

X.25 has fairly specialized terms that need to be understood to effectively work with this technology. Some of the more popular terms are listed in this section.

call A request initiated by a user to establish a connection with a target DTE.

control packet This packet contains information that is transferred between DTEs. Examples of information in these packets include reset, clear, and call.

data packet That packet by which data are sent through a network. X.25 supports varying sizes of packets.

DCE Data circuit-terminating equipment. From a user's perspective, this is the end of the network link. It could be a modem, or it could be an interface card within a node.

DSE Data-switching exchange. This term generally refers to logical switching that occurs within a node, most notably a computer.

DTE Data terminal equipment. Generically, this could be anything that uses an X.25 interface. Typically, it is a terminal or host.

interrupt packet This packet is small and has a low priority for transmission through the network. It carries information required for an interrupt.

LCN Logical channel number. This refers to a specific DTE connection within an X.25 network.

link level Reference to line control or the link protocol that transfers frames from one network to another.

logical channel Reference to a virtual circuit that is mapped to a physical link.

packet That unit which carries data through the network between source to destination DTEs.

packet level The level at which the protocol performs a mulitplexing function of the physical channel, thus creating multiple logical channels.

PAD Packet assembler/diassembler. This is hardware and/or software that performs the function of formatting incoming data into X.25 packets. In other words, it converts X.25 packets into other communication protocols.

permanent virtual circuit A relationship between a logical channel on one port and another on a different port in another part of a network.

physical level The lowest level referred to within a network. It is the level that provides the electrical interface into a network (however, an interface can support optical fiber, thus producing an optical signal). This is the layer where interface standards such as RS-232, V.35, and others operate.

qualified data packet In this type of packet the Q bit is set to the ON position. This lets the DTEs know that different types of data are contained in the packet and can be used any way the DTEs require.

signaling terminal exchange This is an interface that supports DTEs between different networks using the X.75 CCITT recommendation.

switched virtual circuit A circuit set up by a calling user that sends a special type of packet *called* a call packet.

24.4 X.25 Concepts

X.25 networking utilizes numerous concepts. Some of those are explained in this section because of their prevalence in the networking arena.

24.4.1 Channels and Circuits

These two terms are often confusing, but a simple example may demystify this. Consider Fig. 24-9.

Figure 24-9 shows two hosts connected to the X.25 network. Note that the logical channels are identified with each host and the modem in use. This means that logical channels refer to this local connection. Note the line indicating a connection from one host to the other. This refers to the virtual channel. The channel is virtual because it may traverse a number of different hosts within the network before reaching its destination; therefore, these connections within the X.25 *cloud* are considered virtual.

24.4.2 PAD Types and Functions

A packet assembler/disassembler (PAD) can be defined as either hardware or software that performs the function of putting non-X.25 data into X.25 packets. Three different PADs are presented here.

X.3

The CCITT X.3 recommendation calls for characters to be put into packets to pass through the X.25 network. It also calls for the virtual call setup, the clear and reset functions, as well as other required actions. In short, this recommendation is responsible for the appropriate parameters in call setup between a non-X.25 host and a target DTE.

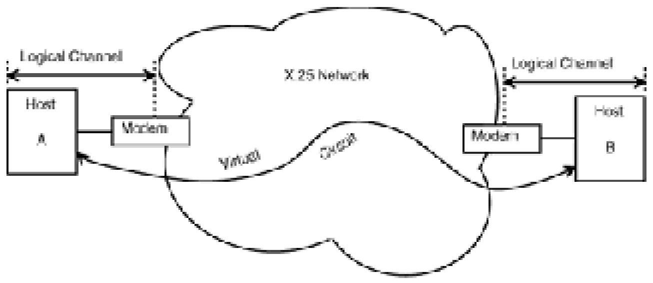


Figure 24-9
Conceptual view of channels and circuits.

X.28

This CCITT recommendation specifies the data flow between the non-X.25 device and the X.3 PAD. The X.28 parameters specify use of X or V series specifications with modems. The basic purpose of this PAD is to transmit the correct call request packet.

X.29

The CCITT recommendation for X.29 specifies control messages used during the communication between the calling DTE and the called DTE. It specifically explains support for information exchange during any phase of the connection.

Figure 24-10 best represents a conceptual view of these three PADs.

24.4.3 X.25 and the SNA Environment

IBM's System Network Architecture (SNA) environment supports connectivity with X.25 networks. This functionality is achieved by a program called *network packet switched interface* (NPSI), which runs on what is termed a *front-end processor* (FEP) or *communication controller.* This program permits bidirectional connections between SNA and an X.25 environment. Figure 24-11 illustrates these connections.

Figure 24-11 shows multiple X.25 networks communicating and a connection into the SNA network via NPSI on the FEP.

24.4.4 X.25 Interfaces

Communication within X.25 networks occurs via *logical channels,* in X.25 lingo. As defined previously, this is a reference to a virtual circuit mapped to a physical link. From a user's viewpoint, this is a dedicated physical circuit. Communication between the caller and the called is a function performed through one of the following *interfaces.*

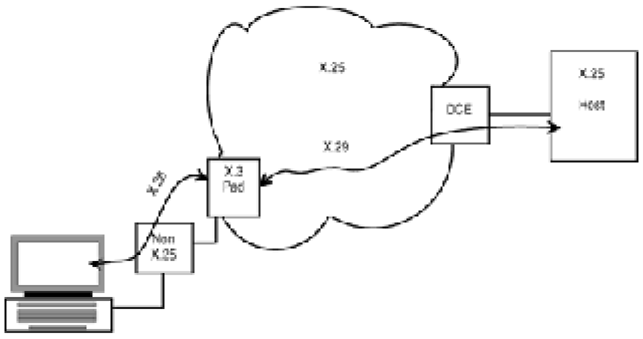


Figure 24-10
Conceptual view of X.3, X.28, and X.29 pads.

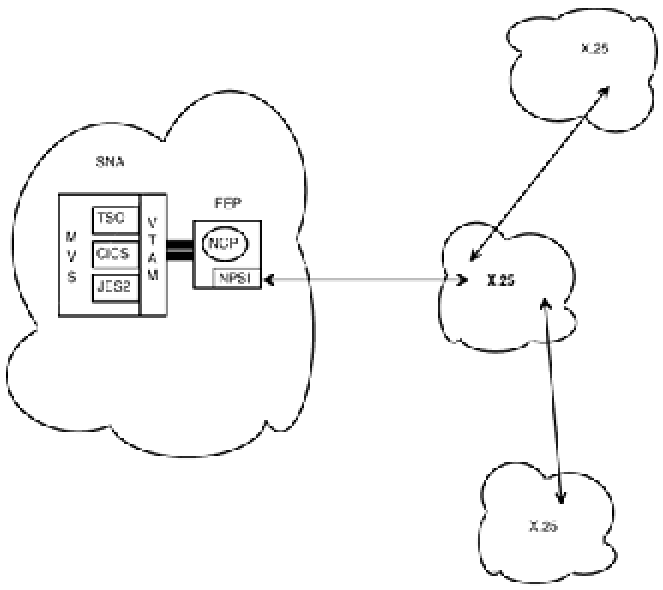


Figure 24-11
Conceptual view of NPSI.

1. *Permanent virtual circuit.* This type of interface provides a service similar to what is called a dedicated line in the telephone arena. In this scenario a user(s) has been defined to the network and the logical channel number and part of the header of the X.25 packet are predetermined. This type of circuit is always considered “up” and available for use.

2. *Virtual circuit.* A virtual circuit is the opposite of a permanent virtual circuit in that it requires that call setup and bringdown functions be performed for each call. Logical channel numbers (LCNs) are thereby distributed dynamically. Using the telephone analogy, this is similar to placing a telephone call; that is, the telephone does not go into the off-hook position until a user is ready to enter the number of the target party. Likewise, when the conversation is over, the telephone receiver is placed in the on-hook position.

3. *Fast select.* Fast select permits a call request packet to contain user data up to 128 bytes. In turn, the called DTE can respond to the calling DTE and also carry user data in the packet. If the call is accepted and transmitted, then continuation of the call takes on the characteristics of a switched virtual call.

4. *Fast select/immediate clear function.* This type of call is similar to the fast select in that it contains user data. In short, a packet is sent to a target point and once the return packet is received, the virtual call is brought down. This is in contrast to the fast select alone, which is similar to the virtual call.

24.5 Frame Relay and X.25 Information

Some ITU recommendations are listed here:

X.21	X.25-3	X.25-A
X.21bis	X.25-4	X.25-D1
X.25-1	X.25-5	X.25-D2
X.25-2	X.25-6	X.25-D3
		X.25-I

24.6 Frame Relay and ISDN Recommendations

Frame relay and ISDN work together in many ways in the marketplace today. ISDN is a comprehensive topic that includes a vast amount of standards, protocols, and information contributed by a number of standards-making bodies.

24.7 ISDN in Contrast to Frame Relay

You may have wondered about ISDN in contrast to frame relay if your background is not related to this technology. The technology is not new in that it was just invented a few years ago, but rather its implementation and use generally keep it one step removed from most except those who work directly with it.

24.7.1 Working Definition

ISDN is the acronym for *Integrated Services Digital Network.* Practically speaking, it has more to do with functionality from the perspective of regional telephone companies and service provider implementations than with an isolated implementation. In fact, ISDN *users* realize its benefits and are typically not involved in its “implementation.” This point will be clarified later.

For example, many telephone companies (in the United States and other countries) have implemented the ISDN technology in central offices, private digital systems, and private branch exchanges (PBXs). This is significant because the technology is based on digital, not analog, signals. Not only are digital signals fundamental to ISDN founda tions, but ISDN can also support voice, data, video, electronic mail, and numerous other services integrated together make it a powerful technology on which to build. The result is the potential for services offered through the telephone network that were not possible previously. Figure 24-12 is a conceptual view of the idea behind frame-relay and ISDN implementations.

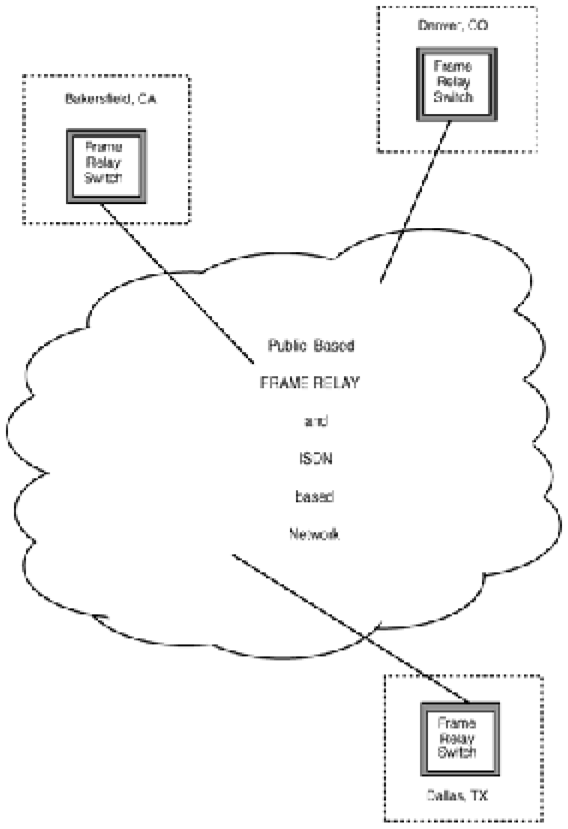


Figure 24-12
Conceptual view of a frame-relay network.

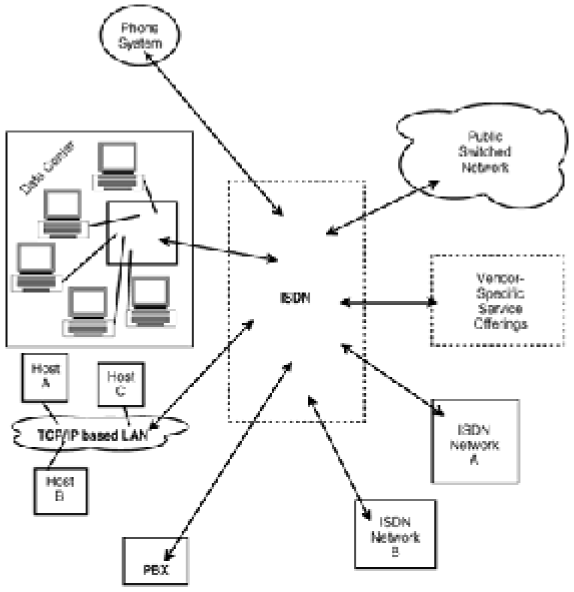


Figure 24-13
A user's conceptual view of an ISDN network.

ISDN realization requires more than telephone system implementation. In fact, ISDN requires the calling party, the network, and the destination party to implement ISDN technology. If these three conditions are not met, some ISDN capabilities are not possible.

Another good conceptual view of the ISDN network as viewed by ordinary users is shown in Fig. 24-13.

Figure 24-13 shows multiple types of services and scenarios connected to an ISDN backbone network. Not all these may communicate with each other; this is simply a representation of how such a scenario could appear to a user viewing the network conceptually.

24.7.2 ITU Recommendations

Frame relay is a protocol, and so is ISDN. ISDN operation, standards, and recommendations have been defined by the ITU. The I series of ITU recommendations covers a broad range of ISDN technology and is

Table 24-1 ITU Recommendations for ISDN Technology

ITU number	Function
I.110	Contains general information such as terminology used, the structure of the I series of recommendations, and basic capabilities of ISDN
I.200	Specifies ISDN service capabilities such as circuit, packet, and a variety of digital services
I.3000	Includes principles, protocols, and architecture
I.400	Includes specifications of user interfaces and network-layer functions
I.500	Includes interface standards, principles for internetworking ISDN networks, and related topics
I.600	Includes ISDN maintenance recommendations such as subscriber access, basic access, and primary rate access

divided into groups of 100. Table 24-1 categorically describes the ITU recommendations.

These recommendations, along with many others, detail ISDN according to the ITU and can be obtained from sources that provide standards and recommendations from entities such as the ITU.

24.8 ISDN Channels

ISDN has basic concepts that are applicable in the majority of implementations. This section explains terms and concepts on which ISDN is built, including the components used in a given ISDN implementation.

The term *channel* is used frequently in ISDN to convey the meaning of service provided. Channels are an integral part of ISN technology, and there are different types of channels.

Channels in general are entities through which physical or logical data, voice, video, or other *information* travels. This is important because in ISDN, different types of channels are defined. A definable characteristic about channels is that they can be identified as being digital or analog. Either way, they carry signals from one entity to another.

The remainder of this section explains the different types of channels available with ISDN.

24.8.1 The D Channel

In ISDN the D channel is used to convey user signaling messages. This type of channel uses *out-of-band signaling,* meaning that network-related signals are carried on a separate channel from user data. These signals transmitted over the D channel convey the characteristics of the service on behalf of the user. The term *out of band* arose because the network signal is out of band with the user signal.

The protocol used on the D-channel defines logical connection between terminal equipment (TE) and the local exchange (LE) via local loop termination equipment. In order to use this arrangement, customer premise equipment (CPE) that performs switching functions is required. Figure 24-14 is a conceptual view of this idea.

The essence of understanding the D channel is knowing that it uses out-of-band signaling but carries user data. It operates at approxi mately 16 or 64 kbits/s and is used by user equipment to transmit requests and messages within the network. In summary, the D channel provides signaling service between a user and the network and provides packet-mode data transfer.

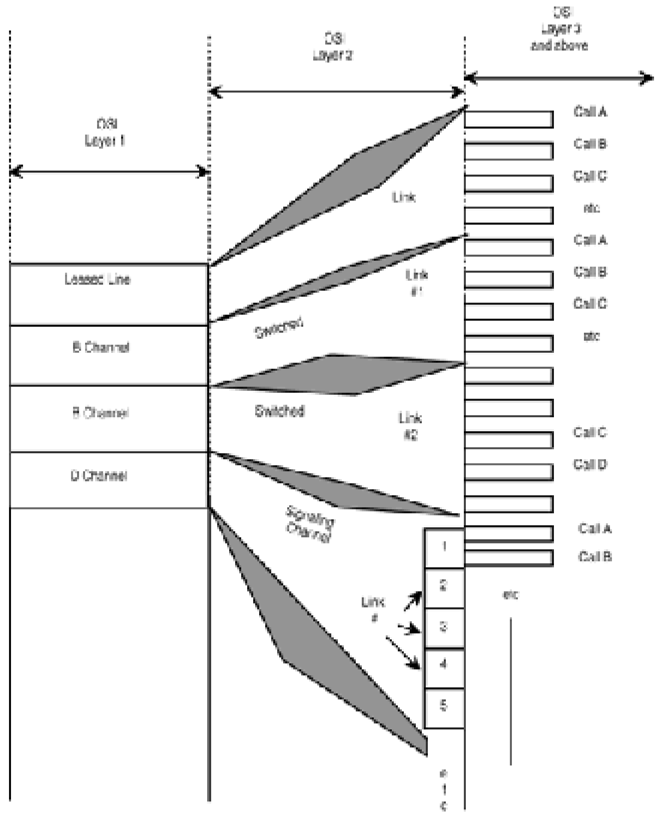


Figure 24-14
Conceptual view of ISDN B and D channels.

24.8.2 The B Channel

The B channel carries voice, data, video, and data. This channel functions at a constant 64 kbits/s and can be used for packet- and circuit-switching applications. The difference between the two is packet-switching applications utilize a logical connection through a network and no dedicated facilities exist. This is generally referred to as a *store-and-forward* method of data transfer. Circuit switching differs because its switching technology is based on devices that are connected via some resource for the extent of the call (or communication instance).

B and D channels work in harmony. The D channel is used to transfer requests for services that are delivered on a B channel. Figure 24-14 also shows an example of B and D channel operation.

24.8.3 H Channel

There are numerous types of H channels. The basic differences between them is the services they offer. As a general rule, H channels have a considerably higher transfer rate than do B channels. These channels effectively meet the needs of real-time videoconferencing, digital-quality audio, and other services requiring a much higher bandwidth.

The basic H channel, H_0, consists of one channel that can provide rates of 384 kbits/s. H_{11} channels can support throughput rates of approximately 1536 Mbits/s, and the H_{12} channel sustains rates of up to approximately 1920 Mbits/s. This type of channel is most suitable for a trunk where subdivision can be implemented to maximize the effective bandwidth.

24.9 Signaling System 7

Signaling System 7 (SS7) is a standard being implemented by regional and long-distance telephone companies today. Its relationship to ISDN is important. Without SS7 implementation in the telephone systems, ISDN is somewhat inhibited. SS7 and ISDN are interrelated in many ways. The fundamental reason for this is that SS7 is digital by design and its link capacity outstrips that of SS6.

SS7 is a protocol and method for networks of switching entities to communicate with one another. Although ISDN can be implemented without SS7, the implementation of SS7 and ISDN together can be more comprehensive in scope.

24.9.1 Characteristics

SS7 is a complex standard. The ITU has a series of recommended standards about the signaling system. Some highlights of SS7 characteristics are

• It can accommodate digital communications in networks using digital channels.

• It can also operate with analog communication channels.

• It can be used domestically in the United States or internationally.

• It is a layered architecture.

• Speeds are 56 and 64 kbits/s.

• In regard to information transfer, it is considered reliable because it ensures sequential movement of signals through a network and provides a mechanism to prevent loss or duplicate signals.

• It can operate in point-to-point network implementations and can be used with satellite communications.

• Its method of handling routing and delivery of control messages is also considered reliable.

• It has built-in management and maintenance capabilities, and also a method for call control.

SS7 CCITT recommendations define packet-switched network functions but do not restrict implementation to specific hardware. According to ITU recommendations, two signaling points are defined: a *signaling point*, a point in the network capable of handling control information; and a *signaling transfer point*, which is identified as an entity where routing of messages can be achieved.

24.9.2 Protocol Components

SS7 protocol can be categorized into three groups: a signaling connection control part (SCCP), an application, and a message transfer part (MTP).

The SCCP protocol specifies five classes of network service:

0. Unsequenced connectionless
1. Sequenced connectionless
2. Connection-oriented
3. Flow-control connection-oriented
4. Flow-control connection-oriented with error recovery

The SCCP supports OSI addressing capabilities and in a sense functions to deliver messages intended for a specific user once the message reaches the destination signaling point. The SCCP works with the MTP to achieve OSI network layer support.

The MTP protocol has multiple layers of signaling. The first level corresponds to the physical layer of the OSI model. It functions as the signaling data link. The second level of the protocol operates at the data-link layer of the OSI model. It is, by design, a bit-oriented protocol; consequently, it is robust in its capabilities. The third level corresponds roughly to the network layer of the OSI model. It is concerned with routing and link management.

SS7 application protocol correlates with upper layers in the OSI reference model. The structure at this layer is divided into a telephone user part which focuses on signaling processes required for voice communications. The data user part is related to requirements for circuit-oriented networks and has been usurped by the ISDN user part.

24.9.3 Additional Information

The CCITT and ANSI standards for SS7 are comprehensive in scope. The following list is only a partial one, but provides an excellent starting point for those who need additional information. A more complete list can be obtained by standard source suppliers who maintain lists of standards-making bodies such as ANSI and the ITU.

• ANSI T1.110, *SS7 Overview*

• ANSI T1.111, *Overview of the Message Transfer Part*

• ANSI T1.113, *ISDN User Part*

• ITU Q.700, *SS7 Overview*

• ITU Q.701, *Message Transfer Part*

• ITU Q.702, *Signaling Data Link*

• ITU Q.711–Q.716, *Signaling Connection Control Part*

• ITU Q.730, *ISDN Supplementary Services*

• ITU Q.761–Q.766, *ISDN User Part*

24.10 ISDN-User Interface

Two ISDN-user interfaces are explained in this section: basic rate interface and the primary rate interface.

24.10.1 Basic Rate Interface

This is one way to access ISDN. This interface consists of two B channels and one D channel according to the ITU recommendation I.430. Hence, it is sometimes referred to as the“2B + D interface.” No specific protocol specifications restrictions are applicable here. This interface utilizes circuit-switched transparent “pipes” to effect a connection between two end users by way of the ISDN network. Examples of where this interface is implemented would be PBXs, individual terminals, videoconference units, personal computers, and workstations.

24.10.2 Primary Rate Interface

This is also a way to access ISDN. This interface calls for one of the following implementations: 23 B channels and 1 D channel, 24 B channels and 1 D channel, or 30 B channels and 1 D channel.

These originate from the ITU I.431 recommendation. In some ways this is basically the bandwidth equivalent to T-1s used in the United States. This interface calls for point-to-point, serial, synchronous communications. A fundamental difference between the primary rate interface and the basic rate interface is that the primary rate interface can support H channels as well as B channels. As a result, bandwidth is greatly enhanced.

These two interfaces are used by a business, personal user, or otherwise to connect them directly to the local telephone company’s central office. In large scenarios the basic rate interface is used to connect individual users to the organizations PBX, and in turn the PBX is connected to the local telephone company central office. However, this could be achieved by connecting this latter scenario to an interchange carrier via a broadband interface. The connection is made via an ISDN interface board, ISDN controller, or an external terminal adapter, and the connection is physically established.

24.11 Practical Use of ISDN

Like many technologies used in networks that may not reside inside a user’s workplace environment, ISDN is somewhat abstract. ISDN applications can be understood more easily than the internal operations for those who are not technically adept. This section hones in on practical services that are based directly or indirectly on ISDN technology.

24.11.1 Automatic Number Identification

Automatic number identification (ANI) (also known as *caller ID*) is a service that provides the individual being called the telephone number of the caller prior to the party answering the call. This service is beneficial for a company who prides itself in customer service.

Just the telephone number and a well-designed and implemented database can be a powerful tool for a company to better serve their customers. The following examples are based on the ANI service and an updated database; consider the implications:

• Identification of the customer: name, address, phone number(s), and other information pertinent to serving the customer.

• Identification of the customer’s language preference. Our diverse culture today is a blend of individuals who speak different languages. This is important information for companies who have international customers and those who do not speak English. With a good database and ANI, a caller speaking a language other than English can be routed immediately to a service representative who can speak their native language. This in itself communicates to the customer that the company cares enough about its customers to have individuals who can communicate in a variety of languages.

• Ascertaining callers’ telephone numbers who terminate a call (hang up) before a representative can respond because they have been on hold for an inordinate amount of time. Consequently, a representative can return the customer’s call.

• Identifying callers by their phone numbers against a database in a particular category. For example, assume a customer purchases large quantities of *widgets* and has had a long-standing relationship with a particular sales representative because the representative understands the needs of the caller. In this case a caller and that caller’s files can be directed to the appropriate representative.

• A customer’s preferred method of payment can be determined according to their prior payment history; therefore, asking the customer to repeat the same numbers and information again can be avoided except to verify appropriate information.

These ANI-based services are available today for large system computing to personal computer systems. Regardless of the implementation, the results of such a service generally make a positive impression on a customer.

24.11.2 Other ISDN Technologies

1. *Electronic library interconnections.* With the help of ISDN and supporting technology today, it is possible for libraries in geographically remote parts of the United States to exchange information electronically that would have been impractical a decade ago.

2. *Electronic manual access.* Many companies such as Philips, Apple Computer, and Microsoft have circulated, and others are beginning to circulate, manuals in electronic media and distribute them to databases through ISDN networks. These manuals span topics such as hardware maintenance, software changes, and a variety of other topics.

3. *Image retrieval.* A major advantage to ISDN and supporting systems is the ability to transmit images, full-motion video, text, graphics, data, and prerecorded video information. This is particularly significant in the realm of medicine. Now CAT (computerized axial tomography) scans can be moved in entirety from participating ISDN customers in the medical community.

Other practical everyday uses of ISDN and explanations to the consumers who utilize its capabilities are not generally presented in technical form. Suffice it to say that ISDN is growing in terms of vendor products and telephone system implementations, and is even coming close to the houses we live in.

Companies such as AT&T, Sprint, and MCI are implementing ISDN and variations thereof. The end result will be services via networks that a widespread customer base in time will have access to.

24.12 Summary

Frame relay and ISDN work parallel in many ways, and some aspects of these two technologies overlap. Actual ISDN technology is removed from most users. The services of ISDN are the result of the implementation. ISDN services provide capabilities for moving voice, data, live video, text, imaging, and variations of multimedia technology from one location to another in real time.

ITU and ANSI have numerous recommendations and proposed standards related to ISDN and supporting peripheral equipment. These standards are lengthy; my references on the topic consume many linear feet of bookshelf space. A list of CCITT and ANSI specifications was provided for the reader who needs additional information on the topic.

ISDN channels were explained. The B, D, and H channels were explained in light of their features and differences. SS7 characteristics and protocol components were also presented; and references for additional information were also provided.

ISDN basic rate interface and primary rate interfaces were explained. A brief explanation on how interfaces are used was also presented.

A section on the practical uses of ISDN was presented to help the reader understand some of the services that are based on ISDN technology, directly or indirectly.

Appendixes

Appendix A
SNA and TCP/IP Bibliography

Abbatiello, Judy, and Ray Sarch, eds., 1987, *Telec Communications & Data Communications Factbook,* New York, N.Y.: Data Communications; Ramsey, N.J.: CCMI/McGraw-Hill.

Apple Computer, Inc., 1991, *Planning and Managing AppleTalk Networks,* Menlo Park, Calif.: Addison-Wesley.

Apple Computer, Inc., 1992, *Technical Introduction to the Macintosh Family,* 2d ed., Menlo Park, Calif.: Addison-Wesley.

Ashley, Ruth, and Judi N. Fernandez, 1984, *Job Control Language,* New York, N.Y.: Wiley.

Aspray, William, 1990, *John Von Neumann and the Origins of Modern Computing,* Cambridge, Mass.: MIT Press.

ATM Forum, The, 1993, *ATM User-Network Interface Specification,* Englewood Cliffs, N.J.: Prentice-Hall.

Bach, Maurice J., 1986, *The Design of The UNIX Operating System,* Englewood Cliffs, N.J.: Prentice-Hall.

Baggott, Jim, 1992, *The Meaning of Quantum Theory,* New York, N.Y.:, Oxford University Press.

Bashe, Charles J., Lyle R. Johnson, John H. Palmer, and Emerson W. Pugh, 1986, *IBM's Early Computers,* Cambridge, Mass.: MIT Press.

Berson, Alex, 1990, *APPC Introduction to LU6.2,* New York, N.Y.: McGraw-Hill.

Black, Uyless, 1989, *Data Networks Concepts, Theory, and Practice,* Englewood Cliffs, N.J.: Prentice-Hall.

Black, Uyless, 1991, *The V Series Recommendations Protocols for Data Communications over the Telephone Network,* New York, N.Y.: McGraw-Hill.

Black, Uyless, 1991, *The X Series Recommendations Protocols for Data Communications Networks,* New York, N.Y.: McGraw-Hill.

Black, Uyless, 1992, *TCP/IP and Related Protocols,* New York, N.Y.: McGraw-Hill.

Blyth, W. John, and Mary M. Blyth, 1990, *Telecommunications: Concepts, Development, and Management,* Mission Hills, Calif.: Glencoe/McGraw-Hill.

Bohl, Marilyn, 1971, *Information Processing,* 3d ed., Chicago: Science Research Associates.

Bradbeer, Robin, Peter De Bono, and Peter Laurie, 1982, *The Beginner's Guide to Computers,* Reading, Mass.: Addison-Wesley.

Brookshear, J. Glenn, 1988, *Computer Science: An Overview,* Menlo Park, Calif.: Benjamin/Cummings.

Bryant, David, 1971, *Physics,* London, U.K.: Hodder & Stoughton.

Campbell, Joe, 1984, *The RS-232 Solution,* Alameda, Calif.: SYBEX.

Campbell, Joe, 1987, *C Programmer's Guide to Serial Communications*, Carmel, Ind.: Howard W. Sams.

Chorafas, Dimitris N., 1989, *Local Area Network Reference,* New York, N.Y.: McGraw-Hill.

Comer, Douglas, 1988, *Internetworking with TCP/IP Principles, Protocols, and Architecture,* Englewood Cliffs, N.J.: Prentice-Hall.

Comer, Douglas E., 1991, *Internetworking with TCP/IP, Vol. I: Principles, Protocols, and Architecture,* Englewood Cliffs, N.J.: Prentice-Hall.

Comer, Douglas E., and David L. Stevens, 1991, *Internetworking with TCP/IP, Vol. II: Design, Implementation, and Internals,* Englewood Cliffs, N.J.: Prentice-Hall.

Dayton, Robert L., 1991, *Telecommunications: The Transmission of Information,* New York, N.Y.: McGraw-Hill.

Dern, Daniel P., 1994, *The Internet Guide for New Users,* New York, N.Y.: McGraw-Hill.

Digital Equipment Corp., 1991, *DECnet Digital Network Architecture (Phase V): Network Routing Layer Functional Specification*, EK-DNA03-FS-001, Maynard, Mass.: Digital Equipment Corp.

Digital Equipment Corp., 1993, *DECnet/OSI for OpenVMS: Introduction and Planning,* AA-PNHTB-TE, Maynard, Mass.: Digital Equipment Corp.

Digital Equipment Corp., 1993, *OpenVMS DCL Dictionary: A-M,* AA-PV5LA-TK, Maynard, Mass.: Digital Equipment Corp.

Digital Equipment Corp., 1993, *OpenVMS DCL Dictionary: N-Z,* AA-PV5LA-TK, Maynard, Mass.: Digital Equipment Corp.

Digital Equipment Corp., 1993, *OpenVMS Glossary,* AA-PV5UA-TK, Maynard, Mass.: Digital Equipment Corp.

Digital Equipment Corp., 1993, *OpenVMS Software Overview,* AA-PVXHA-TE, Maynard, Mass.: Digital Equipment Corp.

Edmunds, John J., 1992, *SAA/LU 6.2 Distributed Networks and Applications,* New York, N.Y.: McGraw-Hill.

Feit, Sidnie, 1993, *TCP/IP Architecture, Protocols, and Implementation,* New York, N.Y.:, McGraw-Hill.

Forney, James S., 1989, *MS-DOS Beyond 640K Working with Extended and Expanded Memory,* Blue Ridge Summit, Pa.: Windcrest Books.

Forney, James S., 1992, *DOS Beyond 640K,* 2d ed., Blue Ridge Summit, Pa.: Windcrest/McGraw-Hill.

Fortier, Paul J., 1989, *Handbook of LAN Technology,* New York, N.Y.: Intertext Publications/Multiscience Press.

Gasman, Lawrence, 1994, *Broadband Networking,* New York, N.Y.: Van Nostrand-Reinhold.

Graubart-Cervone, H. Frank, 1994, *VSE/ESA JCL Utilities, Power, and VSAM,* New York, N.Y.: McGraw-Hill.

Groff, James R., and Paul N. Weinbert, 1983, *Understanding UNIX: A Conceptual Guide,* Carmel, Ind.: Que.

Hecht, Jeff, 1990, *Understanding Fiber Optics,* Carmel, Ind.: Howard W. Sams.

Hewlett-Packard Company, 1992, *HP OpenView SNMP Agent Administrator's Reference,* J2322-90002, Ft. Collins, Colo.: Hewlett-Packard.

Hewlett-Packard Company, 1992, *HP OpenView SNMP Management Platform Administrator's Reference,* J2313-90001, Ft. Collins, Colo.: Hewlett-Packard.

Hewlett-Packard Company, 1992, *HP OpenView Windows User's Guide,* J2316-90000, Ft. Collins, Colo.: Hewlett-Packard.

Hewlett-Packard Company, 1992, *Using HP-UX: Hp 9000 Workstations,* B2910-90001, Ft. Collins, Colo.: Hewlett-Packard.

Hewlett-Packard Company, 1991, *Using the X Window System*, B1171-90037, Ft. Collins, Colo.: Hewlett-Packard.

IBM Corp., 1980, *IBM Virtual Machine Facility: Terminal User's Guide,* GC20-1810-9, Poughkeepsie, N.Y.: IBM.

IBM Corp., 1983, *IBM System/370 Extended Architecture: Principles of Operation,* SA22-7085-0, Research Triangle Park, N.C.: IBM.

IBM Corp., 1985, *IBM 3270 Information Display System: 3274 Control Unit Description and Programmer's Guide,* GA23-0061-2, Research Triangle Park, N.C.: IBM.

IBM Corp., 1986, *JES3 Introduction,* GC23-0039-2, Poughkeepsie, N.Y.: IBM.

IBM Corp., 1987, *IBM System/370: Principles of Operation,* GA22-7000-10, Poughkeepsie, N.Y.: IBM.

IBM Corp., 1988, *IBM Enterprise Systems Architecture/370: Principles of Operation,* SA22-7200-0, Poughkeepsie, N.Y.: IBM.

IBM Corp., 1988, *3270 Information Display System: Introduction,* GA27-2739-22, Research Triangle Park, N.C.: IBM.

IBM Corp., 1989, *MVS/ESA Operations: System Commands Reference Summary,* GX22-0013-1, Poughkeepsie, N.Y.: IBM.

IBM Corp., 1990, *VM/ESA and Related Products: Overview,* GG24-3610-00, Poughkeepsie, N.Y.: IBM.

IBM Corp., 1990, *Enterprise Systems Architecture/390: Principles of Operation,* SA22-7201-00, Poughkeepsie, N.Y.: IBM.

IBM Corp., 1990, *Enterprise System/9000 Models 120, 130, 150, and 170: Introducing the System,* GA24-4186-00, Endicott, N.Y.: IBM.

IBM Corp., 1990, *IBM 3172 Interconnect Controller: Presentation Guide,* White Plains, N.Y.: IBM.

IBM Corp., 1990, *IBM VSE/ESA: System Control Statements,* SC33-6513-00, Mechanicsburg, Pa.: IBM.

IBM Corp., 1990, *IBM VSE/POWER: Networking,* SC33-6573-00, Mechanicsburg, Pa.: IBM.

IBM Corp., 1990, *MVS/ESA SP Version 4 Technical Presentation Guide,* GG24-3594-00, Poughkeepsie, N.Y.: IBM.

IBM Corp., 1990, *Virtual Machine/Enterprise Systems Architecture,* GC24-5441, Endicott, N.Y.: IBM.

IBM Corp., 1991, *Dictionary of Computing,* SC20-1699-8, Poughkeepsie, N.Y.: IBM.

IBM Corp., 1991, *Enterprise Systems Architecture/390 ESCON I/O Interface: Physical Layer,* SA23-0394-00, Kingston, N.Y.: IBM.

IBM Corp., 1991, *Enterprise Systems Connection,* GA23-0383-01, Kingston, N.Y.: IBM.

IBM Corp., 1991, *Enterprise Systems Connection: ESCON I/O Interface,* SA22-7202-01, Poughkeepsie, N.Y.: IBM.

IBM Corp., 1991, *Enterprise Systems Connection Manager,* GC23-0422-01, Kingston, N.Y.: IBM.

IBM Corp., 1991, *Installation Guidelines for the IBM Token-Ring Network Products,* GG24-3291-02, Research Triangle Park, N.C.: IBM.

IBM Corp., 1991, *NetView: NetView Graphic Monitor Facility Operation,* SC31-6099-1, Research Triangle Park, N.C.: IBM.

IBM Corp., 1991, *Systems Network Architecture: Concepts and Products,* GC30-3072-4, Research Triangle Park, N.C.: IBM.

IBM Corp., 1991, *Systems Network Architecture: Technical Overview,* GC30-3073-3, Research Triangle Park, N.C.: IBM.

IBM Corp., 1991, *Systems Network Architecture: Type 2.1 Node Reference,* Version 1, SC20-3422-2, Research Triangle Park, N.C.: IBM.

IBM Corp., 1991, *3174 Establishment Controller: Functional Description,* GA23-0218-08, Research Triangle Park, N.C.: IBM.

IBM Corp., 1991, *Virtual Machine/Enterprise System Architecture: General Information,* GC24-5550-02, Endicott, N.Y.: IBM.

IBM Corp., 1992, *APPN Architecture and Product Implementations Tutorial,* GG24-3669-01, Research Triangle Park, N.C.: IBM.

IBM Corp., 1992, *ES/9000 Multi-Image Processing, Vol. 1: Presentation and Solutions Guidelines,* GG24-3920-00, Poughkeepsie, N.Y.: IBM.

IBM Corp., 1992, *High Speed Networking Technology: An Introductory Survey,* GG24-3816-00, Raleigh, N.C.: IBM.

IBM Corp., 1992, *IBM Networking Systems: Planning and Reference,* SC31-6191-00, Research Triangle Park, N.C.: IBM.

IBM Corp., 1992, *MVS/ESA and Data in Memory: Performance Studies*, GG24-3698-00, Poughkeepsie, N.Y.: IBM.

IBM Corp., 1992, *MVS/ESA: General Information for MVS/ESA System Product Version 4,* GC28-1600-04, Poughkeepsie, N.Y.: IBM.

IBM Corp., 1992, *Sockets Interface for CICS—Using TCP/IP Version 2 Release 2 for MVS: User's Guide,* GC31-7015-00, Research Triangle Park, N.C.: IBM.

IBM Corp., 1992, *Synchronous Data Link Control: Concepts,* GA27-3093-04, Research Triangle Park, N.C.: IBM.

IBM Corp., 1992, *TCP/IP Version 2 Release 2.1 for MVS: Offload of TCP/IP Processing,* SA31-7033-00, Research Triangle Park, N.C.: IBM.

IBM Corp., 1992, *TCP/IP Version 2 Release 2.1 for MVS: Planning and Customization,* SC31-6085-02, Research Triangle Park, N.C.: IBM.

IBM Corp., 1992, *The IBM 6611 Network Processor*, GG24-3870-00, Raleigh, N.C.: IBM.

IBM Corp., 1992, *3172 Interconnect Controller: Operator's Guide,* GA27-3970-00, Research Triangle Park, N.C.: IBM.

IBM Corp., 1992, *3172 Interconnect Controller: Planning Guide,* GA27-3867-05, Research Triangle Park, N.C.: IBM.

IBM Corp., 1992, *3270 Information Display System: Data Stream Programmer's Reference,* GA23-0059-07, Research Triangle Park, N.C.: IBM.

IBM Corp., 1992, *VM/ESA : CMS Primer,* SC24-5458-02, Endicott, N.Y.: IBM.

IBM Corp., 1992, *VM/ESA Release 2 Overview*, GG24-3860-00, Poughkeepsie, N.Y.: IBM.

IBM Corp., 1992, *VSE/ESA Version 1.3: An Introduction Presentation Foil Master*, GG24-4008-00, Raleigh, N.C.: IBM.

IBM Corp., 1993, *IBM Network Products Implementation Guide,* GG24-3649-01, Raleigh N.C.: IBM.

IBM Corp., 1993, *IBM VSE/Interactive Computing and Control Facility: Primer,* SC33-6561-01, Charlotte, N.C.: IBM.

IBM Corp., 1993, *LAN File Services/ESA: VM Guide and Reference,* SH24-5264-00, Endicott, N.Y.: IBM.

IBM Corp., 1993, *LAN File Services/ESA: MVS Guide and Reference*, SH24-5265-00, Endicott, N.Y.: IBM.

IBM Corp., 1993, *LAN Resource Extension and Services/VM: Guide and Reference,* SC24-5622-01, Endicott, N.Y.: IBM.

IBM Corp., 1993, *MVS/ESA: JES2 Command Reference Summary,* GX22-0017-03, Poughkeepsie, N.Y.: IBM.

IBM Corp., 1993, *MVS/ESA JES2 Commands,* GC23-0084-04, Poughkeepsie, N.Y.: IBM.

IBM Corp., 1993, *MVS/ESA: System Commands*, GC28-1626-05, Poughkeepsie, N.Y.: IBM.

IBM Corp., 1993, *NetView: Command Quick Reference,* SX75-0090-00, Research Triangle Park, N.C.: IBM.

IBM Corp., 1993, *NetView: Installation and Administration,* SC31-7084-00, Research Triangle Park, N.C.: IBM.

IBM Corp., 1993, *System Information Architecture: Formats,* GA27-3136, Research Triangle Park, N.C.: IBM.

IBM Corp., 1993, *System Network Architecture: Architecture Reference, Version 2,* SC30-3422-03, Research Triangle Park, N.C.: IBM.

IBM Corp., 1993, *The Host as a Data Server Using LANRES and Novell NetWare,* GG24-4069-00, Poughkeepsie, N.Y.: IBM.

IBM Corp., 1993, *Virtual Machine/Enterprise Systems Architecture*, SC24-5460-03, Endicott, N.Y.: IBM.

IBM Corp., 1993, *VM/ESA: CMS Command Reference,* SC24-5461-03, Endicott, N.Y.: IBM.

IBM Corp., 1993, *VM/ESA: CP Command and Utility Reference,* SC24-5519-03, Endicott, N.Y.: IBM.

IBM Corp., 1993, *VTAM: Operation,* SC31-6420-00, Research Triangle Park, N.C.: IBM.

IBM Corp., 1993, *VTAM: Resource Definition Reference Version 4 Release 1 for MVS/ESA,* SC31-6427-00, Research Triangle Park, N.C.: IBM.

IBM Corp., 1994, *LAN Resource Extension and Services/VM: General Information,* GC24-5618-03, Endicott, N.Y.: IBM.

IBM Corp., 1994, *LAN Resource Extension and Services/MVS: General Information,* GC24-5625-03, Endicott, N.Y.: IBM.

IBM Corp., 1994, *LAN Resource Extension and Services/MVS: Guide and Reference,* SC24-5623-02, Endicott, N.Y.: IBM.

Jain, Bijendra N., and Ashok K. Agrawala, 1993, *Open Systems Interconnection,* New York, N.Y.: McGraw-Hill.

Kessler, Gary C., 1990, *ISDN*, New York, N.Y.: McGraw-Hill.

Kessler, Gary C., and David A. Train, 1992, *Metropolitan Area Networks Concepts, Standards, and Services,* New York, N.Y.: McGraw-Hill.

Killen, Michael, 1992, *SAA and UNIX IBM's Open Systems Strategy,* New York, N.Y.: McGraw-Hill.

Killen, Michael, 1992, *SAA Managing Distributed Data,* New York, N.Y.: McGraw-Hill.

Kochan, Stephen G., and Patrick H. Wood, 1984, *Exploring the UNIX System,* Indianapolis, Ind.: Hayden Books.

McClain, Gary R., 1991, *Open Systems Interconnection Handbook*, New York, N.Y.: Intertext Publications/Multiscience Press.

Madron, Thomas W., 1988, *Local Area Networks: The Next Generation,* New York, N.Y.: Wiley.

Martin, James, 1989, *Local Area Networks Architectures and Implementations,* Englewood Cliffs, N.J.: Prentice-Hall.

Meijer, Anton, 1987, *Systems Network Architecture: A tutorial,* London, U.K.: Pitman; New York, N.Y.: Wiley.

Merrow, Bill, 1993, *VSE/ESA Performance Management and Fine Tuning,* New York, N.Y.: McGraw-Hill.

Merrow, Bill, 1994, *VSE/ESA Concepts and Facilities,* New York, N.Y.: McGraw-Hill.

Nash, Stephen G., ed., 1990, *A History of Scientific Computing,* New York, N.Y.: ACM Press.

Naugle, Matthew G., 1991, *Local Area Networking,* New York, N.Y.: McGraw-Hill.

Naugle, Matthew, 1994, *Network Protocol Handbook,* New York, N.Y.: McGraw-Hill.

Nemzow, Martin A. W., 1992, *The Ethernet Management Guide: Keeping the Link,* 2d ed., New York, N.Y.: McGraw-Hill.

O'Dell, Peter, 1989, *The Computer Networking Book,* Chapel Hill, N.C.: Ventana Press.

Parker, Sybil P., ed., 1984, *McGraw-Hill Dictionary of Science and Engineering,* New York, N.Y.: McGraw-Hill.

Pugh, Emerson W., Lyle R. Johnson, and John H. Palmer, *IBM's 360 and Early 370 Systems,* Cambridge, Mass.: MIT Press.

Pugh, Emerson W., 1984, *Memories That Shaped an Industry,* Cambridge, Mass.: MIT Press.

Ranade, Jay, and George C. Sackett, 1989, *Introduction to SNA Networking Using VTAM/NCP,* New York, N.Y.: McGraw-Hill.

Rose, Marshall T., 1991, *The Simple Book: An Introduction to Management of TCP/IP-Based Internets,* Englewood Cliffs, N.J.: Prentice-Hall.

Rose, Marshall T., 1990, *The Open Book: A Practical Perspective on OSI,* Englewood Cliffs, N.J.: Prentice-Hall.

Samson, Stephen L., 1990, *MVS Performance Management,* New York, N.Y.: McGraw-Hill.

Savit, Jeffrey, 1993, *VM/CMS Concepts and Facilities,* New York, N.Y.: McGraw-Hill.

Schatt, Stan, 1990, *Understanding Local Area Networks,* 2d ed., Carmel, Ind.: Howard W. Sams.

Schlar, Serman K., 1990, *Inside X.25: A Manager's Guide,* New York, N.Y.: McGraw-Hill.

Seyer, Martin D., 1991, *RS-232 Made Easy: Connecting Computers, Printers, Terminals, and Modems,* Englewood Cliffs, N.J.: Prentice-Hall.

Sidhu, Gursharan S., Richard F. Andrews, and Alan B. Oppenheimer, 1990, *Inside AppleTalk,* 2d ed., Menlo Park, Calif.: Addison-Wesley.

Spohn, Darren L., 1993, *Data Network Design,* New York, N.Y.: McGraw-Hill.

Stallings, William, 1987–1988, *Handbook of Computer-Communications Standards,* Vols. 1–3, New York, N.Y.: Macmillan.

Stallings, William, 1989, *ISDN: An Introduction,* New York, N.Y.: Macmillan.

Stamper, David A., 1986, *Business Data Communications,* Menlo Park, Calif.: Benjamin/Cummings.

Tang, Adrian, and Sophia Scoggins, 1992, *Open Networking with OSI,* Englewood Cliffs, N.J.: Prentice-Hall.

Umar, Amjad, 1993, *Distributed Computing: A Practical Synthesis,* Englewood Cliffs, N.J.: Prentice-Hall.

White, Gene, 1992, *Internetworking and Addressing,* New York, N.Y.: McGraw-Hill.

Zwass, Vladimir, 1981, *Introduction to Computer Science,* New York, N.Y.: Barnes & Noble Books.

Appendix B
SNA and TCP/IP Trademarks

Trademarks are listed alphabetically, followed (in parentheses) by their trademark (T) or registered trademark (R) designations, then proprietor (company, institute, etc.) names (DEC = Digital Equipment Corp,; IBM = International Business Machines Corp.; IEEE = Institute of Electrical and Electronic Engineers; MIT = Massachusetts Institute of Technology).

ACF/VTAM (R: IBM)

ACMS (T: DEC)

AIX (T: IBM)

AIXwindows (T: IBM)

ALL-IN-1 (T: DEC)

Alpha AXP (T: DEC)

APDA (T: Apple Computer, Inc.)

Apollo (R: Apollo Computer, Inc.)

Apple and Apple logo (R,R: Apple Computer, Inc.)

AppleColor (R: Apple Computer, Inc.)

Apple Desktop Bus (T: Apple Computer, Inc.)

AppleShare (T: Apple Computer, Inc.)

AppleTalk (T: Apple Computer, Inc.)

Apple IIGS (R: Apple Computer, Inc.)

AS/400 (R: IBM)

A/UX (R: Apple Computer, Inc.)

AXP and AXP logo (T,T: DEC)

Bookreader (T: DEC)

CDA (T: DEC)

CDD (T: DEC)

CDD/REpository (T: DEC)

CI COHESION (T: DEC)

CICS (R: IBM)

CICS/ESA (R: IBM)

CICS/MVS (R: IBM)

Cisco (R: Cisco Systems, Inc.)

DATABASE 2, DB2 (R,R: IBM)

DEC (T: DEC)

DEC ACCESSWORKS (T: DEC)

DEC GKS (T: DEC)

DEC MAILworks (T: DEC)

DEC PHIGS (T: DEC)

DEC Rdb for Open VMS (T: DEC)

DEC RTR (T: DEC)

DEC VTX (T: DEC)

DEC VUIT (T: DEC)

DECalert (T: DEC)

DECamds (T: DEC)

DECdecision (T: DEC)

DECdesign (T: DEC)

DECdtm (T: DEC)

DECforms (T: DEC)

DECmcc (T: DEC)

DECmessageQ (T: DEC)

DECnet (R: DEC)

DECNIS (T: DEC)

DECperformance Solution (T: DEC)

DECplan (T: DEC)

DECprint (T: DEC)

DECquery (T: DEC)

DECram (T: DEC)

DECscheduler (T: DEC)

DECserver (T: DEC)

DECset (T: DEC)

DECtalk (T: DEC)

DECterm (T: DEC)

DECthreads (T: DEC)

DECtp (T: DEC)

DECtrace (T: DEC)

DECwindows (T: DEC)

DECwrite (T: DEC)

DFSMS, DFSMS.MVS (R,R: IBM)

Digital and Digital logo (T,T: DEC)

DNA (T: DEC)

EDT (T: DEC)

80386, 80386SX, 80486, 80486SX (T,T,T,T: Intel Corp.)

Enterprise System/3090, ES/3090 (R,R: IBM)

Enterprise System/4381, ES/4381 (R,R: IBM)

Enterprise System/9000, ES/9000 (R,R: IBM)

Enterprise Systems Architecture/ 370, ESA/370 (R,R: IBM)

Enterprise Systems Architecture/390, ESA/390 (R,R: IBM)

Enterprise Systems Connection Architecture, ESCON, ESCON XDF (R,R,R: IBM)

EtherCard PLUS (T: Western Digital Corp.)

Etherlink (T: 3Com Corp.)

Ethernet (R: Xerox Corp.)

eXcursion (T: DEC)

Finder (T: Apple Computer, Inc.)

GDDM (R: IBM)

Hardware Configuration Definition (R: IBM)

Hiberbatch (R: IBM)

Hiberspace (R: IBM)

HP (R: Hewlett-Packard Co.)

HSC (T: DEC)

HyperCard (R: Apple Computer, Inc.)

HYPERchannel (R: Network Systems Corp.)

IBM (R: IBM)

IBMLink (R: IBM)

ImageWrite (R: Apple Computer, Inc.)

IMS (R: IBM)

IMS/ESA (R: IBM)

Information Warehouse (R: IBM)

Intel (R: Intel Corp.)

Internetwork Packet eXchange (R: Novell, Inc.)

IPX (R: Novell, Inc.)

KanjiTalk (T: Apple Computer, Inc.)

Kerberos (R: MIT)

LaserWriter (R: Apple Computer, Inc.)

Lat (T: DEC)

LattisNet (R: SynOptics Communications, Inc.)

LinkWorks (T: DEC)

Lisa (R: Apple Computer, Inc.)

MacApp (R: Apple Computer, Inc.)

MacDraw (R: Claris Corp.)

Macintosh (R: Apple Computer, Inc.)

MacPaint (R: Claris Corp.)

MacWorks (R: Apple Computer, Inc.)

MacWrite (R: Claris Corp.)

Madge (T: Madge Networks Ltd.)

Microsoft (T: Microsoft Corp.)

Microsoft C (R: Microsoft Corp.)

Microsoft Windows (R: Microsoft Corp.)

MPW (T: Apple Computer, Inc.)

MSCP (T: DEC)

MS-DOS (R: Microsoft Corp.)

MultiFinder (T: Apple Computer, Inc.)

MVS, MVS/ESA, MVS/SP, MVS/XA (R,R,R,R: IBM)

NAP (R: Automated Network Management, Inc.)

NCP (R: IBM)

NCS (R: Apollo Computer, Inc.)

NETMAP (R: SynOptics Communications, Inc.)

NetView (R: IBM)

NetWare (R: Novell, Inc.)

NuBus (T: Texas Instruments)

OpenEdition (R: IBM)

OpenVMS (T: DEC)

OSF, OSF/Motif (R,R: Open Software Foundation, Inc.)

OS/2 (R: IBM)

PATHWORKS (T: DEC)

PC-AT (T: IBM)

PC-NFS (R: Sun Microsystems, Inc.)

POLYCENTER (T: DEC)

Portmapper 9R: Sun Microsystems, Inc.)

POSIX (T: IEEE)

PostScript (R: Adobe Systems, Inc.)

Presentation Manager (R: IBM)

Processor Resource/Systems Manager, PR/SM (R,R: IBM)

Proprinter (T: IBM)

PSF (R: IBM)

PS/2 (R: IBM)

PS2/2 (T: IBM)

RACF (R: IBM)

Reliable Transaction Router (T: DEC)

RISC System/6000 (T: IBM)

RS6000 (R: IBM)

RT (T: IBM)

rtVAX (T: DEC)

SANE (R: Apple Computer, Inc.)

SDLC (R: IBM)

SNA (R: IBM)

Sun (R: Sun Microsystems, Inc.)

SunOS (R: Sun Microsystems, Inc.)

Switcher (T: Apple Computer, Inc.)

Sysplex Timer (R: IBM)

Systemview (R: IBM)

System/370 (R: IBM)

3090 (R: IBM)

TMSCP (T: DEC)

Trellis (T: DEC)

TURBOchannel (T: DEC)

ULTRIX (T: DEC)

UNIX (R: UNIX System Laboratories, Inc.) (UNIX was originally developed in the 1970s by AT&T and Bell Laboratories)

VAX (R: DEC)

VAX Ada (T: DEC)

VAX APL (T: DEC)

VAX BASIC (T: DEC)

VAX BLISS-32 (T: DEC)

VAX C (T: DEC)

VAX COBOL (T: DEC)

VAX DATATRIEVE (T: DEC)

VAX DBMS (T: DEC)

VAX DIBOL (T: DEC)

VAX DOCUMENT (T: DEC)

VAX DSM (T: DEC)

VAX FORTRAN (T: DEC)

VAX LISP (T: DEC)

VAX MACRO (T: DEC)

VAX Notes (T: DEC)

VAX OPS5 (T: DEC)

VAX Pascal (T: DEC)

VAX RALLY (T: DEC)

VAX RMS (T: DEC)

VAX SCAN (T: DEC)

VAX SQL (T: DEC)

VAX TEAMDATA (T: DEC)

VAXcluster (T: DEC)

VAXELN (T: DEC)

VAXft (T: DEC)

VAXmail (T: DEC)

VAXshare (T: DEC)

VAXsimPLUS (T: DEC)

VAXstation (T: DEC)

VIDA (T: DEC)

VM/ESA (R: IBM)

VM/XA (R: IBM)

VMS (T: DEC)

VTAM (R: IBM)

VT100, VT220, VT330 (T,T,T: DEC)

Windows (T: Microsoft, Inc.)

Word for Windows (T: Microsoft, Inc.)

WPS (T: DEC)

WPS-PLUS (T: DEC)

Xerox (T: Xerox Corp.)

XNS (T: Xerox Corp.)

X/Open (R: X.Open Co., Ltd.)

XUI (T: DEC)

XWindow (R: MIT)

X-Windows (T: MIT)

Appendix C
SNA and TCP/IP Abbreviations

3270	Reference to a 3270 Data Stream Supporting Entity
377	Reference to Remote Job Entry
370/XA	370/eXtended Architecture
5250	Reference to a 5250 Data Stream Supporting Entity
576	The Minimum Datagram Size that *all* Hosts, Including Routers, Must Accommodate
AAA	Autonomous Administrative Area
AAI	Administration Authority Identifier
AAL	ATM Adaptation Layer
AARP	AppleTalk Address Resolution Protocol
ABOM	A-bis Operations and Maintenance
ACB	Access-Control Block (in VTAM); Adapter Control Block (in NCP); Application Control Block
ACCS	Automated Calling Card Service
ACD	Automatic Call Distribution
ACDF	Access-Control Decision Function
ACE	Access-Control List Entry
ACF	Access-Control Field; Advanced Communications Function
ACF/VTAM	Advanced Communications Function for the Virtual Telecommunications Access Method
ACIA	Access Control Inner Areas
ACID	Atomicity, Consistency, Isolation, and Durability
ACK	Positive Acknowledgment
ACL	Access-Control List
ACP	Ancillary Control Process

ACS	Access-Control Store
ACSA	Access-Control Specific Area
ACSE	Association Control Service Element
ACSP	Access-Control Specific Point
ACTLU	Activate Logical Unit
ACTPU	Activate Physical Unit
ACU	Auto Calling Unit
AD	Addendum Document to an OSI Standard
ADF	Adapter Description File
ADMD	Administrative Management Domain
ADP	Adapter Control Block; AppleTalk Data Stream Protocol
ADPCM	Adaptive Differential Pulse-Code Modulation
ADSP	AppleTalk Data Stream Protocol
AE	Application Entity
AEI	Application Entity Invocation
AEP	AppleTalk Echo Protocol
AET	Application Entity Title
AF	Auxiliary Facility
AFI	Authority and Format Identifier
AFP	AppleTalk Filing Protocol
AI	Artificial Intelligence
AIFF	Audio Interchange File Format
AIX	Advanced Interactive eXecutive
ALDC	Adaptive Lossless Data Compression
ALS	Application-Layer Structure
ALU	Application-Layer User
AMI	Alternating Mark Inversion
ANI	Automatic Number Identification
ANR	Automatic Network Routing
ANS	American National Standard
ANSI	American National Standards Institute
AP	Application Process; Argument Pointer
APAR	Authorized Program Analysis Report
APB	Alpha Primary Bootstrap
APD	Avalanche Photodiode
APDU	Application Protocol Data Unit
API	Application Program Interface; Application Programming Interface

APIC	Advanced Programming Interrupt Controller
APLI	ACSE/Presentation Library Interface
APP	Applications Portability Profile
APPC	Advanced Peer-to-Peer Communication; Advanced Program-to-Program Communication
APPL	Application Program
APPN	Advanced Peer-to-Peer Networking
ARC	Automatic Reconfiguration Facility
APT	Application Process Title
ARP	Address Resolution Protocol
ARPA	Advanced Research Projects Agency
ARQ	Automatic Repeat Request
ARS	Automatic Route Selection
AS/400	Application System/400
ASC	Accredited Standards Committee
ASCII	American Standard Code for Information Interchange
ASDC	Abstract Service Definition Convention
ASE	Application Service Element
ASM	Address-Space Manager
ASN	Abstract Syntax Notation
ASN.1	Abstract Syntax Notation One
ASO	Application Service Object
ASP	Abstract Service Primitive; AppleTalk Session Protocol
ASPI	Advanced SCSI Programming Interface
AST	Asynchronous System Trap
ASTLVL	Asynchronous System Trap Level
ASTSR	Asynchronous System Trap Summary Register
ATM	Asynchronous Transfer Mode; Abstract Text Method
ATP	AppleTalk Transaction Protocol
ATS	Abstract Test Suite
AU	Access Unit
AVA	Attribute Value Assertion
B-ISDN	Broadband ISDN
B8ZS	Bipolar 8-Zeros Substitution
BACM	Basic Access Control Model
BAR	Base Address Register
BAS	Basic Activity Subset
BASIC	Beginner’s All-purpose Symbolic Instruction Code

BB	Begin Bracket
BBS	Bulletin Board System
BC	Begin Chain
BCC	Block Check Character
BCS	Basic Combined Subset
BCVT	Basic Class Virtual Terminal
Bellcore	Bell Communications Research, Inc.
BER	Box Event Record(s); Bit Error Rate
BGP	Border Gateway Protocol
BIB	Backward Indicator Bit
BIS	Bracket Initiation Stopped
BISUP	Broadband ISUP
BITS	Building Integrated Timing Systems
BITNET	Because It’s Time Network
BIU	Basic Information Unit
BLU	Basic Link Unit
BMS	Basic Mapping Support
BMU	Basic Measurement Unit
BN	Boundary Node
BNF	Backus-Naur Form
BNN	Boundary Network Node
BOC	Bell Operating Company
BOM	Beginning of Message
BOOTP	Bootstrap Protocol
BPDU	Bridged Protocol Data Unit
BRI	Basic Rate Interface
BSC	Binary Synchronous Communication
BSD	Berkeley Software Division
BSS	Basic Synchronization Subset; Base Station Subsystem
BSSMAP	Base Station Subsystem Mobile Application Part
BTAM	Basic Telecommunications Access Method
BTU	Basic Transmission Unit
CA	Channel Adapter (also Attachment); Certification Authority
CAD	Computer-Aided Design
CAE	Common Applications Environment; Computer-Aided Engineering
CAF	Channel Auxiliary Facility
CAI	Computer-Assisted Instruction

CASE	Common Application Service Elements; Computer-Aided Software Engineering
CATV	Community Antenna Television
CBD	Changeback Declaration
CBEMA	Computer and Business Equipment Manufacturers Association
CCA	Conceptual Communication Area
CCAF	Call Control Agent Function
CCAF+	Call Control Agent Function Plus
CCB	Connection (also Channel) Control Block
CCIRN	Coordinating Committee for Intercontinental Research Networking
CCIS	Common Channel Interoffice Signaling
CCITT	Consultative Committee for International Telephony and Telegraphy
CCO	Context Control Object
CCR	Commitment, Concurrency, and Recovery
CCS	Common Communications Support; Common Channel Signaling; Console Communication Services
CCU	Central (also Communications) Control Unit
CCW	Channel Command Word (in SNA)
CD	Countdown Counter; Committee Draft
CDF	Configuration Data Flow
CDI	Change-Direction Indicator
CDRM	Cross-Domain Resource Manager (in SNA)
CDRSC	Cross-Domain Resource
CDS	Conceptual Data Storage or Store; Central Directory Server
CEBI	Conditional End-Bracket Indicator
CEI	Connection Endpoint Identifier
CEN/ELEC	Committee European de Normalization Electrotechnique
CEP	Connection Endpoint
CEPT	Conference of European Postal and Telecommunications Administrations
CESID	Caller Emergency Service Identification
CF	Control Function
CGI	Common Gateway Interface
CGM	Computer Graphics Metafile
CHILL	CCITT High-Level Language
CI	Computer Interconnect

CICS	Customer Information Control System
CID	Communication (or Connection) Identifier
CIDR	Classless Interdomain Routing
CIGOS	Canadian Interest Group on Open Systems
CIM	Computer-Integrated Manufacturing
CIR	Commitment Information Rate
CIS	Card Information Structure
CLAW	Common Link Access to Workstation (protocol)
CLI	Connectionless Internetworking
CLIST	Command List
CLNP	Connectionless Network Protocol
CLNS	Connectionless Network Service
CLSDST	Close Destination
CLTP	Connectionless Transport Protocol
CLTS	Connectionless Transport Service
CLU	Control Logical Unit
CMC	Communication Management Configuration
CMIP	Common Management Information Protocol
CMIS	Common Management Information Service
CMISE	Common Management Information Service Element
CMOL	CMIP over Logical Link Control
CMOT	CMIP over TCP/IP
CMS	Conversational Monitoring System
CMT	Connection Management
CN	Composite Node
CNM	Communication Network Management
CNMA	Communication Network for Manufacturing Applications
CNMI	Communication Network Management Interface
CNN	Composite Network Node
CNOS	Change Number of Sessions
CNT	Communications Name Table
CO	Central Office
COCF	Connection-Oriented Convergence Function
CODEC	Coder/Decoder
COI	Connection-Oriented Internetworking
COM	Continuation-of-Message DMPDU
CONF	Confirm
CONS	Connection-Oriented Network Service

CORBA	Common Object-Oriented Request Broker Architecture
COS	Class of Service; Corporation for Open Systems
COSM	Class-of-Service Management
COTP	Connection-Oriented Transport Protocol
COTS	Connection-Oriented Transport Service
CP	Control Point; Control Program
CPCB	Control Point (also Program) Control Block
CPDU	Common Protocol Data Unit
CPE	Customer Premises Equipment
CPH	Call Party Handling
CPF	Control Program Facility
CPI	Common Programming Interface
CPI-C	Common Programming Interface with C Language
CPMS	Control Point Management Services
CPU	Central Processing Unit
CRACF	Call-Related Radio Access Control Function
CRC	Cyclical Redundancy Check
CRST	Cluster-Route-Set-Test
CRT	Cathode-Ray Tube
CRV	Call Reference Value
CS	Configuration Services
CSA	Common Service or Storage Area
CSm	Call Segment Model
CS-MUX	Circuit Switching Multiplexer
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
CSP	Communications Scanner Processor
CSN	Card Select Number Register
CSNET	Computer Science Network
CSS	Control, Signaling, and Status Store
CSU	Channel Service Unit
CTC	Channel-to-Channel
CTCA	Channel-to-Channel Adapter
CTCP	Communication and Transport Control Program
CTS	Clear-to-Send; Common Transport Semantics
CUA	Channel Unit Address; Common User Access
CUG	Closed User Group
CURACF	Call Unrelated Service Function

CUT	Control Unit Terminal
CVS	Connection View State
CVT	Communications Vector Table
DACD	Directory Access-Control Domain
DAD	Draft Addendum
DAF	Distributed Architecture Framework; Framework for Distributed Applications; Destination Address Field
DAP	Directory Access Protocol
DAS	Dual Attachment Station; Dynamically Assigned Sockets
DASD	Direct-Access Storage Device
DAT	Dynamic Address Translation
dB	Decibel(s)
DBCS	Double-Byte Character Set
DC	Data Chaining
DCA	Document Content Architecture
DCC	Data Country Code
DCE	Data Communications Equipment; Distributed Computing Environment; Data Circuit-Terminating Equipment; Distributed Computing Environment
DCL	Digital Command Language
DCS	Defined Context Set
DDB	Directory Database Function
DDCMP	Digital Data-Communication Message Protocol
DCSS	Discontiguous Shared Segment
DDIM	Device Driver Initialization Model
DDM	Distributed Data Management
DDN	Defense Data Network
DDName	Data Definition Name
DDP	Datagram Delivery Protocol
DDS	Digital Data Service
DES	Data Encryption Standard
DEV	Device Address Field
DFC	Data Flow Control
DECnet	Digital Equipment Equipment’s Network Architecture
DFI	DSP Format Identifier
DFT	Distributed Function Terminal
DHCP	Dynamic Host Configuration Protocol
DH	DMPDU Header
DIA	Document Interchange Architecture

DIB	Directory Information Base
DIS	Draft International Standard
DISP	Draft International Standardized Profile; Directory Information Shadowing Protocol
DIT	Directory Information Tree
DIU	Distribution Interchange Unit
DL	Distribution List
DLC	Data-Link Control or Connection
DLCEP	Data-Link Connection Endpoint
DLCI	Data-Link Connection Identifier
DLPDU	Data-Link Protocol Data Unit
DLS	Data-Link Service
DLSAP	Data-Link Service Access Point
DLSDU	Data-Link Service Data Unit
DLU	Dependent (also Destination) Logical Unit
DLUR	Dependent Logical Unit Requestor
DLUS	Dependent Logical Unit Server
DM	Data Mark
DMA	Direct Memory Access
DMD	Directory Management Domain
DMI	Digital Multiplexed Interface; Definition of Management Information; Desktop Management Interface
DMO	Domain Management Organization
DMPDU	Derived MAC Protocol Data Unit
DMTF	Desktop Management Task Force
DMUX	Double Multiplexer
DN	Distinguished Name
DNS	Domain Name System or Service
DNHR	Dynamic Nonhierarchical Routing
DoD	U.S. Department of Defense
DOP	Directory Operational Binding Management Protocol
DOS	Disk Operating System
DP	Draft Proposal
DPG	Dedicated Packet Group
dpi	Dots per Inch
DQDB	Distributed Queue Dual Bus
DR	Definite Response or Dynamic Reconfiguration (in SNA)
DRDA	Distributed Relational Database Architecture

DRDS	Dynamic Reconfiguration Data Set
DS	Directory Service(s); Desired State
DS3	Telephony Classification of Leased-Line Speed
DS-*n*	Digital Signaling Level *n*
DSA	Directory Service Agent; Digital Storage Architecture
DSAP	Destination Service Access Point
DSD	Data Structure Definition
DSE	DSA Specific Entries
DSECT	Dummy Control Section
DSL	Digital Subscriber Line
DSP	Directory Service Protocol; Domain-Specific Part
DSPT	Display Station Pass-Through (TELNET)
DSS 1	Digital Subscriber Signaling System Number 1
DSTINIT	Data Services Task Initialization
DSU	Digital Services Unit
DSUN	Distribution Services Unit Name
DT	DMPDU Trailer
DTE	Data Terminal Equipment
DTMF	Dual-Tone Multifrequency
DTR	Data Terminal Ready
DU	Data Unit
DUP	Data User Port
DUA	Directory User Agent
DVD	Digital Videodisk
DVMRP	Distance Vector Multicast Routing Protocol
DVT	Destination Vector Table
E.164	An ATM Address Format Specified by ITU-TS
email	Electronic Mail
EAS	Extended Area Service
EB	End Bracket
EBCDIC	Extended Binary-Coded Decimal Interchange Code
EACK	Extended Acknowledgment
EARN	European Academic Research
ECA	Event Detection Point
ECC	Enhanced Error Checking and Correction
ECH	Echo Canceler with Hybrid
ECMA	European Computer Manufacturers Association
ECO	Echo Control Object

ECSA	Exchange Carriers Standards Association
EDI	Electronic Data Interchange
EDIFACT	EDI For Administration, Commerce and Transport
EDIM	EDI Message
EDIME	EDI Messaging Environment
EDIMS	EDI Messaging System
EDI-MS	EDI Message Store
EDIN	EDI Notification
EDI-UA	EDI-User Agent
EEI	External Environment Interface
EGP	Exterior Gateway Protocol
EIA	Electronic Industries Association
EISA	Extended Industry Standard Architecture
EIT	Encoded Information Type
EN	End Node
ENA	Extended Network Addressing
ENV	European Prestandards
EOF	End of File
EOL	End of Line
EOM	End-of-Message DMPDU
EOR	End of Record
EOT	End-of-Transmission
EP	Emulation Program
ER	Explicit Route; Exception Response
EREP	Environmental Recording Editing and Printing
ERP	Error-Recovery Procedure
ES	End System
ESA	Enterprise Systems Architecture; Enhanced Subarea Addressing
ESCON	Enterprise System Connection
ESF	Extended Superframe Format
ESH	End System Hello
ES-IS	End System Intermediate System
ESS	Electronic Switching System
ESTELLE	Extended State Transition Language
ETB	End-of-Text Block
ETR	Early Token Release
ETX	End of Text

EUnet	European UNIX network
EUUG	European UNIX User’s Group
EXLST	Exit List
EXT	External Trace File
EWOS	European Workshop on Open Systems
FA	Framework Advisory
FADU	File Access Data Unit
FARNET	Federation of American Research Networks
FAS	Frame Alignment Sequence
FAT	File Allocation Table
FC	Frame Control Field
FCC	Federal Communications Commission
FCS	Frame-Check Sequence
FDB	File Descriptor Block
FDCO	Field Definition Control Object
FDDI	Fiber Distributed Data Interface
FDDI-FO	FDDI Follow-On (FDDI)
FDE	Full-Duplex Ethernet
FDFE	Full-Duplex Fast Ethernet
FDL	File Definition Language
FDM	Frequency-Division Multiplexing
FDR	Field Definition Record
FDT	Formal Description Technique
FDX	Full Duplex
FEC	Field Entry Condition; Front-End Controller
FECN	Forward Explicit Congestion Notification
FEE	Field Entry Event
FEI	Field Entry Instruction
FEICO	Field Entry Instruction Control Object
FEIR	Field Entry Instruction Record
FEP	Front-End Processor
FEPCO	Field Entry Pilot Control Object
FEPR	Field Entry Pilot Record
FER	Field Entry Reaction
FFOL	FDDI Follow-on LAN
FH	Frame Header
FID	Format Identification
FIFO	First in, First out

FIPS	Federal Information Processing Standard
FISU	Fill-in Signal Unit
FM	Function Management
FMD	Function Management Data
FMH	Function Management Header
FOD	Office Document Format
FOR	Forward Transfer
FNC	Federal Networking Council
FQPCID	Fully Qualified Procedure Correlation (also Path Control Identifier)
FR	Family of Requirements
FRAD	Frame-Relay Access Device
FRFH	Frame-Relay Frame Header or Handler
FRICC	Federal Research Internet Coordinating Committee
FRMR	Frame Reject
FRSE	Frame-Relay Switching Equipment
FRTE	Frame-Relay Terminal Equipment
FS	Frame Status Field
FSG	SGML Interchange Format
FSM	Finite State Machine
FTAM	File Transfer, Access, and Management
FTP	File Transfer Protocol in TCP/IP
FYI	For Your Information
FX	Foreign Exchange Service
Gbit	Gigabit
Gbits/s	Gigabits per Second
GCS	Group Control System
GDDM	Graphical Data Display Manager
GDMO	Guidelines for the Definition of Managed Objects
GDS	Generalized Data Stream
GFI	General Format Indicator
GFP	Global Functional Plane
GGP	Gateway-to-Gateway Protocol
GMT	Greenwich Mean Time
GPS	Global Positioning System
GOSIP	Government Open Systems Interconnect Protocol
GSA	General Services Administration
GTF	Generalized Trace Facility

GUI	Graphical User Interface
GWNCP	Gateway NCP
GWSSCP	Gateway SSCP
HAL	Hardware Abstraction Layer
HASP	Houston Automatic Spooling Priority
HCD	Hardware Configuration Definition
HCL	Hardware Compatibility List
HCS	Header-Check Sequence
HDB3	High-Density Bipolar—Three zeros
HDLC	High-Level Data-Link Control
HDX	Half-Duplex
HI-SAP	Hybrid Isochronous-MAC Service Access Point
HLR	Home Location Register
HMI	Human-Machine Interface
HMP	Host Monitoring Protocol
H-MUX	Hybrid Multiplexer
HOB	Head of Bus
HPO	High-Performance Option
HPR	High-Performance Routing
HP-SAP	Hybrid packet-MAC Service Access Point
HRC	Hybrid Ring Control
HS	Half-Session
HSC	Hierarchical Storage Controller
HSLN	High-speed Local Network
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
Hz	Hertz (cycles per second)
IA	Internet Advisor
IAB	Internet Architecture Board
IAC	Interapplicaton Communication
IANA	Internet Assigned Number Authority
IADCS	Interactivity Defined Context Set
IAN	Integrated Analog Network
IAP	Inner Administrative Point
IBM	International Business Machines Corporation
IC	Interexchange Carrier
ICA	Integrated Communication Adapter
ICCF	Interactive Computing and Control Facility

ICD	International Code Designator
ICF	Isochronous Convergence Function
ICI	Interface Control Information
ICMP	Internet Control Message Protocol
ICV	Integrity Check Value
ID	Identification or Identifier
IDI	Initial Domain Identifier
IDN	Integrated Digital Network; Interface Definition Notation
IDP	Initial Domain Part; Internetwork Datagram Packet
IDU	Interface Data Unit
IEC	Interexchange Carrier; International Electrotechnical Commission
IEEE	Institute of Electrical and Electronic Engineers
IEN	Internet Engineering Notes
IF	Information Flow
IETF	Internet Engineering Task Force
IESG	Internet Engineering Steering Group
IGM	Information-Gathering Module
IGMP	Internet Group Management Protocol
IGP	Interior Gateway Protocol
IGRP	Internet Gateway Routing Protocol
IIA	Information Interchange Architecture
IHL	Internet Header Length
ILD	Injection Laser Diode
ILU	Independent Logical Unit
IMAC	Isochronous Media Access Control
IMIL	International Managed Information Library
IML	Initial Microcode Load
IMP	Interface Message Processor
IMPDU	Initial MAC Protocol Data Unit
IMR	Intensive-Mode Recording
IMS	Information Management System
IMS/VM	Information Management System/Virtual Storage
IN	Intelligent Network; Interchange Node
IND	Indication
INN	Intermediate Network Node
INTAP	Interoperability Technology Association for Information Processing

IOC	Input/Output Control
IOCP	Input/Output Control Program
IONL	Internal Organization of Network Layer
IOPD	Input/Output Problem Determination
IP	Internet Protocol
IPCS	Interactive Problem Control System
IPng	IP Next Generation
IPv4	IP version 4
IPv6	IP version 6
IPC	Interprocess Communication
IPDS	Intelligent Printer Data Stream
IPI	Initial Protocol Identifier
IPICS	ISP Implementation Conformance Statement
IPL	Initial Program Load(er)
IPM	Interpersonal Message
IPM-UA	Interpersonal Messaging User Agent
IPMS	Interpersonal Messaging System
IPN	Interpersonal Notification
IPR	Isolated Pacing Response
IPX	Internet Packet eXchange
IR	Internet Router
IRN	Intermediate Routing Node
IRQ	Interrupt Request Lines
IRS	Initial Receive Sequence
IRTF	Internet Research Task Force
IS	International Standard
ISA	Industry-Standard Architecture
ISAM	Index-Sequential Access Method
ISC	Intersystem Communications in CICS
ISCF	Intersystems Control Facility
ISDN	Integrated Services Digital Network
ISH	Intermediate System Hello
ISN	Initial Source Number
IS-IS	Intermediate System-to-Intermediate System
ISO	International Standards Organization
ISODE	ISO Development Environment
ISP	International Standard Profile
ISPBX	Integrated Services Private Branch Exchange

ISPF/PDF	Interactive System Productivity Facility/Program Development Facility
ISPSN	Initial Synchronization Point Serial Number
ISR	Intermediate Session Routing
ISS	Initial Send Sequence
ISSI	Interswitching System Interface
ISUP	ISDN User Part
IT	Information Technology
ITC	Independent Telephone Company
ITU	International Telecommunication Union
ITU-TS	International Telecommunication Union—Telecommunication Section
IUCV	Interuser Communication Vehicle
IUT	Implementation under Test
IVDT	Integrated Voice/Data Terminal
IWU	Interworking Unit
IXC	Interexchange Carrier
JCL	Job Control Language
JES	Job Entry Subsystem
JSR	Jump Subroutine
JTC	Joint Technical Committee
JTM	Job Transfer and Manipulation
KA9Q	TCP/IP Implementation for Amateur Radio
kbit	Kilobit
kbits/s	Kilobits per Second
kHz	Kilohertz
km	Kilometer(s)
LAB	Latency Adjustment Buffer; Line Attachment Base
LAN	Local Area Network
LANRES	Local Area Network Resource Extension and Services
LANSUP	LAN Adapter NDIS Support
LAP	Link Access Procedure or Protocol
LAPB	Link Access Procedure (also Protocol) Balanced
LAPD	Link Access Procedures on the D Channel
LAPS	LAN Adapter and Protocol Support
LATA	Local Access and Transport Area
LCF	Log Control Function
LCN	Logical Channel Number
LE	Local Exchange
LEC	Local Exchange Carrier
LED	Light-Emitting Diode
LEN	Low-Entry Networking

LFSI	Local Form Session Identifier
LH	Link Header
LI	Length Indicator
LIB	Line Interface Base
LIC	Line Interface Coupler
LIDB	Line Information Database
LIS	Logical IP Subnet
LIVT	Link Integrity Verification Test
LLAP	LocalTalk Link Access Protocol
LLC	Logical Link Control
LL2	Link Level 2
LME	Layer Management Entity
LMI	Layer (also Local) Management Interface
LOCKD	Lock Manager Daemon
LOTOS	Language of Temporal Ordering Specifications
LPAR	Logical Partitioned Mode
LPD	Line Printer Daemon
LPDA	Link Problem Determination Application or Aid
LPR	Line Printer
LRC	Longitudinal Redundancy Check
LS	Link Station
LSE	Local System Environment
LSL	Link Support (also Services) Layer
LSRR	Loose Source and Record Route
LSS	Low-Speed Scanner
LSSU	Link Status Signal Unit
LT	Local Termination
LU	Logical Unit
LWS	Linear Whitespace
LZW	Lempel-Ziv-Welch (coding)
m	Meter(s)
MAC	Media (or Medium) Access Control
MACE	Macintosh Audio Compression and Expansion
MACF	Multiple Association Control Function

MAN	Metropolitan Area Network
MAP	Manufacturing Automation Protocol
MAU	Media (or Multistation) Access Unit
MBA	MASSBUS Adapter
Mbit	Megabit
Mbits/s	Megabits per Second
MBONE	Multicast BackBONE
MBZ	Must Be Zero
MCA	Microchannel Architecture
MCF	MAC Convergence Function
MCI	Microwave Communications, Inc.; Media Control Interface
MCP	MAC Convergence Protocol
MCR	Monitor Console Routine
MD	Management Domain
MFA	Management Functional Area
MFD	Master File Directory
MFJ	Modified Final Judgment
MFS	Message Formatting Services in IMS
MHP	Message-Handling Package
MHS	Message-Handling Service or System
MHz	Megahertz
MIB	Management Information Base
MIC	Media Interface Connector; Memory in Cassette
MID	Message Identifier
MILNET	Military Network
MIM	Management Information Model
MIME	Multipurpose Internet Mail Extension
MIN	Mobile Identification Number; Multiple Interaction Negotiation
mips	Million Instructions per Second
MIS	Management Information Systems
MIT	Managed Information Tree
MLID	Multiple Link Interface Driver
MMF	Multimode Fiber
MMS	Manufacturing Message Specification or Service
MOSS	Maintenance and Operator Subsystem
MOTIS	Message Oriented Text Interchange System
MOT	Means of Testing

MPAF	Midpage Allocation Field
MPC	Multipath Channel
MPG	Multiple Preferred Guests
MPTN	Multiprotocol Transport Networking
MRO	Multiregion Operation in CICS
ms	Millisecond(s)
MS	Management Service(s); Message Store
MSC	Mobile Switching Center
MSCP	Mass Storage Control Protocol
MSHP	Maintain System History Program
MSL	Maximum Segment Lifetime
MSN	Multiple Systems Networking
MSNF	Multiple Systems Networking Facility
MSS	Metropolitan Area Network (MAN) Switching System; Maximum Segment Size
MST	Multiplexed Slotted and Token Ring
MSU	Management Services Unit
MTA	Message Transfer Agent
MTACP	Magnetic Tape Ancillary Control Process
MTBF	Mean Time between Failures
MTTD	Mean Time of Diagnosis
MTOR	Mean Time of Repair
MTP	Message Transfer Part
MTS	Message Transfer System
MTSE	Message Transfer Service Element
MTU	Maximum (or Minimum) Transfer (also Transmission) Unit
MVC	Multicast Virtual Circuit
MVS	Multiple Virtual Systems or Storage
MVS/XA	Multiple Virtual Storage/eXtended Architecture
MVS/370	Multiple Virtual Storage/370
NAK	Negative Acknowledgment in BSC
NAP	Network Access Provider
NAU	Network Addressable (also Accessible) Unit
NBP	Name-Binding Protocol
NC	Network Connection; Numerical Controller
NCB	Node Control Block
NCCF	Network Communications Control Facility
NCEP	Network Connection Endpoint

NCP	Network Control Program; Network Core Protocol
NCS	Network Computing System
NCTE	Network Channel-Terminating Equipment
NDIS	Network Driver Interface Specification
NetBEUI	NetBIOS Extended User Interface
NetBIOS	Network Basic Input/Output System
NFB	Number of Fragment Blocks
NFS	Network File System (also Server)
NIB	Node Identification (also Initialization) Block
NIC	Network Interface Card (also Center)
NIF	Network Information File
NIP	Network Information Protocol
NISDN	Narrowband ISDN
NIS	Names Information Socket
NIST	National Institute of Standards and Technology
NIUF	North American ISDN Users’ Forum
NJE	Network Job Entry
NLM	NetWare Loadable Module
NLDM	Network Logical Data Manager
NLPID	Network-Level Protocol Identifier
NLS	Natural-Language Software
nm	Nanometer(s)
NM	Network Management
NMP	Network Management Process
NMVT	Network Management Vector Transport
NMS	Network Management Station
NN	Network Node
NNT	NetView-NetView Task
NOC	Network Operations Center
NOF	Node Operator Facility
NPA	Numbering Plan Area
NPAI	Network Protocol Control Information
NPDA	Network Problem Determination Application
NPDU	Network Protocol Data Unit
NPM	NetView Performance Monitor
NPSI	NCP (or Network) Packet Switching Interface
NRF	Network Routing Facility
NRN	Nonreceipt Notification

NRZ	Non–Return to Zero
NRZI	Non-Return-to-Zero Inverted
ns	Nanosecond(s)
NS	Network Service
NSAP	Network Service Access Point
NSDU	Network Service Data Unit
NSF	National Science Foundation; Network Search Function
NTFS	Windows NT File System
NTO	Network Terminal Option
NVFS	Network Virtual File System
NVLAP	National Voluntary Accreditation Program
NVP	Network Virtual Protocol
NVT	Network Virtual Terminal
OAF	Origination Address Field
OAM	Operations, Administration, and Maintenance
OAM&P	Operations Administration, Maintenance, and Provisioning
OC-*n*	Optical Carrier Level *n*
OC3	155 Million Bits per Second over Fiber
OCA	Open Communication Architectures
OCC	Other Common Carrier
ODA	Office (also Open) Document Architecture
ODI	Open Data-Link Interface
ODIF	Office Document Interchange Format
ODINSUP	ODI NSIS Support
ODP	Open Distributed Processing
OIT	Object Identifier Tree
OIW	OSI Implementation Workshop
OLRT	Online Real Time
OLU	Originating Logical Unit
OM	Object Management
ONA	Open Network Architecture
ONC	Open Network Computing
OPNDST	Open Destination
O/R	Originator/Recipient
OS	Operating System
OS/400	Operating System/400 for the AS/400 Computer
OSE	Open Systems Environment
OSF	Open Software Foundation

OSI	Open Systems Interconnection
OSI/CS	OSI Communications Subsystem
OSIE	Open System Interconnection Environment
OSILL	Open Systems Interconnection Lower Layer(s)
OSIUL	Open Systems Interconnection Upper Layer(s)
OSPF	Open Shortest Path First
OSNS	Open Systems Network Services
OUI	Organizationally Unique Identifier
P-MAC	Packet Switched Media Access Control
PA	Prearbitrated
PABX	Private Automatic Branch Exchange
PAD	Packet Assembler/Disassembler
PAF	Prearbitrated Function
PAI	Protocol Address Information
PAM	Pass along Message
PANS	Pretty Amazing New Stuff
PAP	Printer Access Protocol
PARC	Palo Alto Research Center
PBX	Private Branch Exchange
PC	Path Control; Personal Computer
PCCU	Physical Communications Control Unit
PCEP	Presentation Connection Endpoint
PCI	Protocol Control Information; Presentation Context Identifier; Peripheral Component Interconnect bus
PCM	Pulse-Code Modulation
PCO	Points of Control and Observation
PCTR	Protocol Conformance Test Report
PDA	Personal Digital Assistant
PDAD	Proposed Draft Addendum
PDAU	Physical Delivery Access Unit
PDC	Packet Data Channel
PDISP	Proposed Draft International Standard Profile
PDN	Public Data Network
PDS	Partitioned Data Set
PDU	Protocol Data Unit
PDV	Presentation Data Value
PELS	Picture Elements (Pixels)
PEM	Privacy Enhanced Mail

PEP	Partitioned Emulation Program
PETS	Parameterized Executable Test Suite
PFA	Predictive Failure Analysis
PGF	Presentation Graphics Feature
PH	Packet Handler or Handling
PhC	Physical-Layer Connection
PhCEP	Physical Connection Endpoint
PhL	Physical Layer
PhPDU	Physical-Layer Protocol Data Unit
PhS	Physical-Layer Service
PhSAP	Physical-Layer Service Access Point
PhSDU	Physical-Layer Service Data Unit
PHY	Physical Layer
PI	Protocol Interpreter
PICS	Protocol Information Conformance Statement
PID	Protocol Identifier
PIN	Positive-Intrinsic Negative Photodiode
PING	Packet Internet Groper
PIP	Program Initialization Parameters
PITM	Person in the Middle
PIU	Path Information Unit
PIXIT	Protocol Implementation eXtra Information for Testing
PKCS	Public Key Cryptosystems
PLC	Programmable Logic Controller
PLCP	Physical-Layer Convergence Protocol
PLMN	Public Land Mobile Network
PLP	Packet-Layer Protocol
PLS	Primary Link Station; Physical Signaling
PLU	Primary Logical Unit
PM	Protocol Machine
PMD	Physical-Layer Medium-Dependent
POI	Point of Initiation; Program Operator Interface
POP	Point of Presence
POSI	Promoting Conference for OSI
POSIX	Portable Operating System Interface
POTS	Plain Old Telephone Service
POWER	Priority Output Writers and Execution (Processors and Input) Reader

PPDU	Presentation Protocol Data Unit
PPO	Primary Program Operator
PPP	Point-to-Point Protocol
PPSDN	Public Packet Switched Data Network
PRI	Primary Rate Interface
PRMD	Private Management Domain
PR/SM	Processor Resource/Systems Manager
PS	Presentation Services
PSAP	Public Safety Answering Point
PSC	Public Service Commission
PSDN	Packet Switched Data Network
PSID	Product-Set Identification
PSN	Packet Switched Network
PSPDN	Packet Switched Public Data Network
PSTN	Public Switched Telephone Network
PSTN	Public Switched Telephone Network
PTF	Program Temporary Fix
PTLXAU	Public Telex Access Unit
PTN	Public Telephone Network
PTT	Post, Telegraph, and Telephone
PU	Physical Unit
PUC	Public Utility Commission
PUCP	Physical Unit Control Point
PUMS	Physical Unit Management Services
PUP	PARC Universal Packet
PUT	Program Update Tape
PVC	Private (also Permanent) Virtual Circuit
PVN	Private Virtual Network
P1	Protocol 1 (Message Transfer Protocol/MHS/X.400)
P2	Protocol 2 (Interpersonal Messaging MHS/X.400)
P3	Protocol 3 (Submission and Delivery Protocol/MHS/X.400)
P5	Protocol 5 (Teletext Access Protocol)
P7	Protocol 7 (Message Store Access Protocol in X.400)
QA	Queued Arbitrated
QAF	Queued Arbitrated Function
QC	Quiesce Complete
QEC	Quiesce at End of Chain
QMF	Query Management Facility

QOS	Quality of Service
QPSX	Queued Packet and Synchronous Switch
QUIPU	X.500 Conformant Directory Services in ISODE
RA	Recursion Available
RAB	Record Access Block
RACF	Resource Access Control Facility
RAID	Redundant Array of Independent Disks
RAM	Random-Access Memory
RARE	Reseaux Associes pour la Recherche Europeenne; European Association of Research Networks
RARP	Reverse Address Resolution Protocol
RAS	Remote Access Service
RBOC	Regional Bell Operating Company
RD	Routing Domain; Route Redirection; Recursion Desired
RDA	Relative Distinguished Names; Remote Database Access
RDI	Restricted Digital Information
RDN	Relative Distinguished Name
RDP	Reliable Datagram Protocol
RDT	Resource Definition Table
RECFMS	Record Formatted Maintenance Statistics
REJ	Reject
REQ	Request
RESP	Response
RESYNC	Resynchronization
REXX	Restructured eXtended eXecutor
RFC	Request for Comments or Change
RFP	Request for Proposal
RFQ	Request for Price Quotation
RH	Response or Request Header
RIB	Routing Information Base
RIF	Routing Information Field
RIM	Request Initialization Mode
RIP	Router Information Protocol
RIPE	Reseaux IP Europeens; European Continental TCP/IP Network Operated by EUnet
RISC	Reduced Instruction Set Computer
RJE	Remote Job Entry
RLP	Resource Location Protocol

RM	Reference Model; Record Marking
RMS	Record Management Services
rms	Root Mean Square
RMT	Ring Management
RN	Receipt Notification
RNAA	Request Network Address Assignment
RNR	Receiver Not Ready
RODM	Remote Object Data Manager
ROSE	Remote Operations Service Element
RPC	Remote Procedure Call (in OSF/DCE and Elsewhere)
RPL	Request Parameter List; Remote Program Load
RPOA	Recognized Private Operating Agency
RQ	Request Counter
RR	Receiver Ready; Resource Record
RS	Relay System
RSCS	Remote Spooling Communication Subsystem
RSCV	Route Selection Control Vector
RSF	Remote Support Facility
RSP	Response
RSS	Route Selection Services
RTM	Response-Time Monitor
RTMP	Routing Table Maintenance Protocol
RTO	Round-Trip Timeout
RTP	Rapid Transport Protocol; Real-Time Protocol
RTR	Ready to Receive
RTS	Request to Send
RTSE	Reliable Transfer Service Element
RTT	Round-Trip Time
RU	Request or Response Unit
S/390	IBM’s System/390 Hardware Architecture
s	Second(s)
SA	Source Address Field; Subarea; Sequenced Application
SAA	System Applications Architecture; Specific Administrative Area(s)
SAB	Subnetwork-Access Boundary
SABM	Set Asynchronous Mode Balanced
SACF	Single Association Control Function
SACK	Selective Acknowledgment

SAF	SACF Auxiliary Facility
SALI	Source Address Length Indicator
SAM	Security Accounts Manager
SAMBE	Set Asynchronous Mode Balanced Extended
SAO	Single Association Object
SAP	Service Access Point; Service Advertising Protocol
SAPI	Service Access Point Identifier
SAS	Single-Attachment Station
SAS	Statically Assigned Sockets
SASE	Specific Application Service Element
SATF	Shared-Access Transport Facility
SATS	Selected Abstract Test Suite
SAW	Session Awareness (Data)
SBA	Set Buffer Address
SBCS	Session Connection Manager
SBI	Stop Bracket Initiation
SC	Session Connection or Control; Subcommittee
SCB	Session Control Block
SCC	Specialized Common Carrier
SCCP	Signaling Connection Control Part
SCE	System Control Element
SCEP	Session Connection Endpoint
SCIF	Single-Console Image Facility
SCM	Session Connection Manager
SCP	Service Control Point
SCS	System Conformance Statement; SNA Character String (also Stream)
SCSI	Small Computer System Interface
SCTR	System Conformance Test Report
SDH	Synchronous Digital Hierarchy
SDIF	Standard Document Interchange Format
SDL	System Description Language
SDLC	Synchronous Data-Link Control
SDN	Software-Defined Network
SDSE	Shadowed DSA Entries
SDU	Service Data Unit
SE	Session Entity
SG	Study Group

SGFS	Special Group on Functional Standardization
SGML	Standard Generalized Markup Language
SIA	Stable Implementation Agreements
SID	Security ID
SIM	Set Initialization Mode
SIO	Start I/O
SIP	SMDS Interface Protocol
SLIP	Serial Line IP
SLU	Secondary Logical Unit
SM	Session Manager
SMAE	System Management Application Entity
SMASE	Systems Management Application Service Element
SMB	Server Message Block
SMDR	Station Message Detail Recording
SMDS	Switched Multimegabit Data Service
SMF	Single-Mode Fiber; System Management Facility
SMFA	Systems Management Functional Area
SMI	Structure of the OSI Management Information Service
SMIB	Stored Message Information Base
SMP	System Modification Program; Symmetrical Multiprocessing
SMS	Service Management System or Subsystem
SMT	Station Management Standard
SMTP	Simple Mail Transfer Protocol
SNA	System Network Architecture
SNAP	Subnetwork Attachment Point; Subnetwork Access Protocol (IEEE)
SNAcF	Subnetwork Access Function
SNAcP	Subnetwork Access Protocol
SNADS	SNA Distribution Services
SNARE	Subnetwork Address Routing Entity
SNCP	Single-Node Control Point
SNDCP	Subnetwork-Dependent Convergence Protocol
SNI	Subscriber-Network Interface; SNA Network Interconnection or Interface
SNICP	Subnetwork Independent Convergence Protocol
SNMP	Simple Network Management Protocol
SNPA	Subnetwork Point of Attachment
SNRM	Set Normal Response Mode

SOA	Start of Authority
SON	Sent Outside Node
SONET	Synchronous Optical Network
SP	Signaling Point
SPAG	Standards Promotion and Applications Group
SPC	Signaling Point Code
SPCS	Service Point Command Service
SPDU	Session Protocol Data Unit
SPE	Synchronous Payload Envelope
SPF	Shortest Path First
SPI	Subsequent Protocol Identifier
SPM	FDDI to SONET Physical-Layer Mapping Standard
SPS	Sync-Point Services
SPSN	Synchronization Point Serial Number
SQE	Signal Quality Error
SQL	Structured Query Language
SRH	SNARE Request Hello
SRM	System Resource Manager
SRRT	Smoothed Round-Trip Time
SS	Switching System; Session Services; Start-Stop
SS6	Signaling System Number 6
SS7	Signaling System Number 7
SSA	Subschema Specific Area
SSAP	Source (or Session) Service Access Point
SSCP	System Services Control Point
SSDU	Session Service Data Unit
SSM	Single Segment Message DMPDU
SSP	System Support Program
SSRR	Strict Source and Record Route
STA	Spanning Tree Algorithm
ST	Sequenced Terminal
STD	Standard
STM	Synchronous Transfer Mode
STM	Station Management
STM-*n*	Synchronous Transport Module level *n*
STP	Shielded Twisted Pair; Service Transaction Program (in LU6.2); Signal Transfer Point
STS-*n*	Synchronous Transport Signal Level *n*

SUERM	Signal Unit Error Rate Monitor
SUT	System under Test
SVA	Shared Virtual Area
SVC	Switched Virtual Circuit; Supervisor Call
SWS	Silly-Window Syndrome
SYN	Synchronizing Segment; Synchronous Character in IBM’s Bisync Protocol
SYNC	Synchronization
T	Transport
T3	A Designation of Telephony Used over DS3 Speed Lines
TA	Terminal Adapter
TAF	Terminal Access Facility
TAG	Technology Advisory Group
TAP	Trace Analysis Program
TC	Transport Connection or Technical Committee
TCAM	Telecommunications Access Method
TCB	Task (also Transmission) Control Block
TCC	Transmission Control Code
TCEP	Transport Connection Endpoint
TCM	Time-Compression Multiplexing
TCP	Transmission Control Protocol; Total Control Manager
TCP/IP	Transmission Control Protocol/Internet Protocol
TCT	Terminal Control Table in CICS
TDL	Total Data Length
TDM	Time-Division Multiplexing; Topology Database Management
TDMA	Time-Division Multiple Access
TDR	Time-Domain Reflectometer
TE	Terminal Equipment
TELNET	Remote Logon in TCP/IP
TEP	Transport Endpoint
TFTP	Trivial File Transfer Protocol
TG	Transmission Group
TH	Transmission Header
THD	Total Harmonic Distortion
THT	Token-Holding Timer
TIC	Token-Ring Interface Coupler
TIE	Translated Image Environment

TINA	Telecommunications Information Network Architecture
TINA-C	Telecommunication Information Network Architecture Consortium
TI RPC	Transport Independent RPC
TLI	Transport-Layer Interface
TLMAU	Telematic Access Unit
TLV	Type, Length, and Value
TLXAU	Telex Access Unit
TMP	Text Management Protocol
TMS	Time-Multiplexed Switching
TN	TELNET
TOP	Technical and Office Protocol
TOS	Type of Service
TN3270	A Version of TELNET that Implements IBM 3270 Data Stream
TP	Transaction Program or Processing; Transport Protocol
TPDU	Transport Protocol Data Unit
TPF	Transaction-Processing Facility
TP-PMD	Twisted-Pair PMD
TPS	Two-Processor Switch
TPSP	Transaction Processing Service Provider
TPSU	Transaction Processing Service User
TPSUI	TPSU Invocation
TP0	TP Class 0—Simple
TP1	TP Class 1—Basic Error Recovery
TP2	TP Class 2—Multiplexing
TP3	TP Class 3—Error Recovery and Multiplexing
TP4	TP Class 4—Error Detection and Recovery
TR	Technical Report; Token Ring
TRA	Token-Ring Adapter
TRPB	Truncated Reverse-Path Broadcast
TRS	Topology and Routing Services
TRSS	Token-Ring Subsystem
TRT	Token Rotation Timer
TS	Transaction Services; Transport Service
TSAP	Transport Service Access Point
TSC	Transmission Subsystem Controller
TSCF	Target System Control Facility

TSDU	Transport Service Data Unit
TSI	Time-Slot Interchange
TSO	Timesharing Option
TSR	Terminate and Stay Resident (Program)
TSS	Transmission Subsystem
TTCN	Tree and Tabular Combined Notation
TTL	Time to Live
TTP	Timed Token Protocol; Transport Test Platform
TTRT	Target Token Rotation Time
TTY	Teletype
TUP	Telephone User Part
TVX	Valid Transmission Timer (FDDI)
TWX	Teletypewriter Exchange Service
UA	Unnumbered Acknowledgment; User Agent; Unsequenced Application
UART	Universal Asynchronous Receiver and Transmitter
UCB	Unit Control Block
UDI	Unrestricted Digital Information
UDP	User Datagram Protocol
UOW	Unit of Work
UPS	Uninterruptible Power Supply
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
URN	Uniform Resource Name
USB	Universal Serial Bus
User-ASE	User Application Service Element
USS	Unformatted System Services
UT	Unsequenced Terminal; Universal Time
UTC	Universal Time, Coordinated
UTP	Unshielded Twisted Pair (cable)
UUCP	UNIX-to-UNIX Copy Program
VAC	Value-Added Carrier
VAN	Value-Added Network
VAS	Value-Added Service
vBNS	Reference to 155-Mbit/s Deployment of Internet Backbone scheduled for implementation in 1995
VCI	Virtual Channel Identifier (DQDB)
VCNS	VTAM Common Network Services

VDT	Videodisplay Terminal
VESA	Video Electronics Standards Association
VEST	VAX Environment Software Translator
VINES	Virtual Networking system
VIT	VTAM Internal Trace
VLR	Visitor Location Register
VLSI	Very Large Scale Integration
VPI/VCI	Virtual Path Identifier and a Virtual Call Identifier
VM	Virtual Machine
VMD	Virtual Manufacturing Device
VM/SP	Virtual Machine/System Product
VMTP	VM Transport Protocol
VPD	Vital Product Data
VPN	Virtual Private Network
VR	Virtual Route
VRPWS	Virtual Route Pacing Window Size
VS	Virtual Storage
VSAM	Virtual Storage Access Method
VSCS	VM/SNA Console Support
VSE	Virtual Storage Extended
VSE/ESA	Virtual Storage Extended/Enterprise Systems Architecture
VT	Virtual Terminal
VTAM	Virtual Telecommunications Access Method
VTE	Virtual Terminal Environment
VTP	Virtual Terminal Protocol
VTPM	Virtual Terminal Protocol Machine
VTSE	Virtual Terminal Service Element
VVIEF	VAX Vector Instruction Emulation Facility
WACA	Write Access Connection Acceptor
WACI	Write Access Connection Initiator
WAIS	Wide Area Information Server
WAN	Wide Area Network
WAVAR	Write Access Variable
WBC	Wideband Channel
WD	Working Document
WG	Working Group
WNM	Workgroup Node Manager
WP	Working Party

WWW	World Wide Web
X	X-Window System
X.25	An ITU-TX Standard; a Transport-Layer Service
X.400	The ITU-TS Protocol for Electronic Mail
XAPIA	X.400 API Association
XCF	Cross-System Coupling Facility
Xdm	X Display Manager
XDR	eXternal Data Representation
XDS	X/Open Directory Services API
Xds	X Display Server
XI	SNA X.25 Interface
XID	Exchange Identification
XNS	Xerox Network Standard
XRF	Extended Recovery Facility
XTI	X/Open Transport Interface
XUDTS	eXtended Unitdata Service
ZIP	Zone Information Protocol
ZIS	Zone Information Socket

Appendix D
SNA and TCP/IP Terminology

abend Abnormal end of task.

Abstract Syntax Notation One The OSI language used to describe abstract entities or concepts in a machine.

abstract syntax Machine-independent types and values, defined using an ASN.1.

ACB name The name of a microinstruction. A name typically specified on the VTAM APPL definition statement.

accept In a VTAM application program, this means to establish a session with a logical unit in response to a CINIT request.

access-control entry Specifies identifiers and access rights to be granted to or denied holders of the identifiers, default protection for directories, or security alarms.

access-control list A list defining the kinds of access granted or denied to users of an object.

access method A technique for moving data between main storage and input/output devices. Also a technique used in telecommunications.

access method control block (ACB) A control block that links an application program to VSAM or VTAM.

acknowledgment A positive response sent to indicate successful reception of information.

acquire In VTAM, to take over resources that were formerly controlled by an access method in another domain.

activate To initialize a resource.

active Operational. The state of a resource when it has been activated and is operational.

active application An application currently capable of being used by a user.

active window The terminal window where current operations are in the foreground.

ACTLU In SNA, a command used to start a session with a logical unit.

ACTPU In SNA, a command used to start a session with a physical unit.

adapter control block (ACB) In NCP, a control block that contains line control information and the states of I/O operations; a definition in an I/O database describing an adapter or device controller and the I/O interconnect.

adaptive session-level pacing Pacing where session components exchange pacing windows that may vary in size during the course of a session.

address In data communication, this is a designated identifier.

address descriptor record Information that reflects the address of an Apple event.

address mask A bit mask used to select bits from an IP address for subnet addressing.

address resolution Conversion of an IP address into a corresponding physical address, such as Ethernet or Token Ring.

address resolution protocol (ARP) A TCP/IP protocol used to dynamically bind (or map) a high-level IP address to low-level physical hardware addresses. ARP works across single physical networks and is limited to networks that support hardware broadcast.

address space Addresses used to uniquely identify network-accessible units, sessions, adjacent link stations, and links in a node for each network in which the node participates.

addressing In data communication, the way in which a station selects the station to which it is to send data.

adjacent control point A control point directly connected to an APPN, LEN, or composite node.

adjacent NCPs Network control programs connected by subarea links with no intervening NCPs.

adjacent nodes Two nodes connected by at least one path that connects no other node.

adjacent SSCP table A table identifying SSCPs with which the VTAM can enter into a session.

adjacent subareas Subareas connected by one or more links with no intervening subareas.

Advanced Communications Function (ACF) A group of IBM-licensed programs.

Advanced Peer-to-Peer Network (APPN) An upper-layer networking protocol based on peer technology. The fundamental difference between APPN and SNA is that VTAM is not required for session establishment after initial download of parameters.

Advanced Peer-to-Peer Network (APPN) end node A node that can register its local LUs with its network node server. It can also have links to multiple nodes, but can have only one CP-CP session with a network node at a time. CP-CP sessions can never be established between end nodes.

Advanced Peer-to-Peer Network (APPN) interchange node A node that can characteristically be described by its functions, which include controlling network resources, performing CDRM functions in subarea networks, and owning NCPs. This type of node appears to be an APPN-type node to an APPN network, and it appears to be a subarea node to a subarea network. It can reside between an APPN network and a subarea network, thus providing integration of the two.

Advanced Peer-to-Peer Network (APPN) LEN node A type of node that differs from an end node because it supports functions not supported by an end node. For example, an LEN node supports ILU protocols but not CP-CP sessions.

Advanced Peer-to-Peer Network (APPN) network node An APPN network node performs the following functions: (1) distributed directory services, (2) intermediate routing services in an APPN network, (3) network services for specific end nodes, (4) intermediate-session routing, and (5) serving as the management service focal point. The APPN network node cooperates with other network nodes to maintain a network topology database, which is used to select optimal routes for LU-LU sessions based on requested classes of service. An APPN network node can also attach to a subarea network as a peripheral node or to other end nodes.

Advanced Program-to-Program Communication (APPC) A protocol architecture based on T2.1 architecture utilizing LU6.2 to accomplish peer communications. Many references to APPC are simply LU6.2. The basic meaning of APPC is communication between programs: i.e., transactions programs. This type of protocol permits communication between programs written in different languages.

Advanced Research Projects Agency (ARPA) The government agency that funded the ARPAnet. A participant in early development of TCP/IP. It was later renamed DARPA (Defense Advanced Research Projects Agency).

alert In SNA, a message sent to a management subsystem typically using Network Management Vector Transport (NMVT) protocol.

alertable In DECnet architecture, a synchronous alert that delivers data to specific points.

alertsafe According to Digital Equipment documentation, a routine that can be called without risk of triggering an alert while asynchronous delivery of alerts is enabled.

alias file A file that contains a pointer to another file, directory, or volume.

alias name A naming convention sometimes used to refer to a different name which means the same as the alias or refers to another name which may be used as a pointer.

allocate An LU6.2 verb that assigns a session to a conversation.

allocation class According to Digital Equipment documentation, a unique number between 0 and 255 that the system manager assigns to a pair of hosts and to the dual-pathed devices that the hosts make available to other nodes in a VMS cluster.

alpha primary bootstrap According to Digital Equipment documentation, the primary bootstrap program that initializes an AXP system with OpenVMS AXP.

alternate route A route which is not the primary method of moving data from point to point.

American Standard Code for Information Interchange (ASCII) A standard code using a character set based on binary digits (1s and 0s). This character set is the foundation for defining letters, numbers, and specialized control functions.

ampersand (&) When used in the command of most UNIX operating systems, this symbol places the task or tasks in background operation.

Ancillary Control Process A process acting as an interface between software and an I/O driver.

Apple event According to Apple documentation, a high-level function that adheres to the Apple Event Interprocess Messaging Protocol.

Apple event handler According to Apple documentation, a defined function that extracts pertinent data from an Apple event.

Apple Event Interprocess Messaging Protocol A standard defined by Apple Computer, Inc., for communication among applications. According to Apple documentation, high-level events that adhere to this protocol are called *Apple events.*

Apple event parameter According to Apple documentation, a keyword specifying a descriptor record containing data which the target application of an Apple event must use.

Apple file exchange A program that permits file transfer between Apple computers and DOS-based computers.

AppleShare file server Software that permits users to perform tasks in AppleTalk networks.

AppleShare print server A Macintosh computer-software combination that manages network printing.

AppleTalk Data Stream Protocol (ADSP) An AppleTalk protocol that pro vides the capability for maintaining a connection-oriented session between entities on an Internet.

AppleTalk Echo Protocol (AEP) A response-oriented protocol in which a response is sent when a packet is received.

AppleTalk Filing Protocol (AFP) The protocol used between an application and file server. AFP is a client of ASP.

AppleTalk Internet router Router software operating on an Apple computer supporting up to eight AppleTalk networks.

AppleTalk Phase 2 Introduced in June 1989, it extended the capability of AppleTalk Phase 1; for example, it supports Token Ring and provides more efficient routing techniques. It is included in System/7 software.

AppleTalk Session Protocol (ASP) A protocol used to establish and maintain logical connections between a workstation and a server.

AppleTalk Transaction Protocol (ATP) An AppleTalk protocol that functions in many ways like a connection-oriented protocol. For example, it orients the packets into the order in which they need to be received, and if any packets have been lost, automatic retransmission of a packet(s) is performed.

AppleTalk transition A message containing information indicating specific occurrences that relate to the AppleTalk transition queue.

AppleTalk transition queue An operating system queue.

application layer According to ISO OSI model, this is OSI layer 7. It provides application services.

application program Software that provides functions needed by a user. A *user* may be defined as a human, API, transaction program, or other logical entity.

application program interface (API) An addressable point that serves the function of bringing together two or more entities.

application program major node According to IBM documentation, in VTAM, this is a group of application program minor nodes.

application result handler A program designed to perform predefined functions of results generated from applications.

application server A computer used solely to provide processing power for application programs.

application service element (ASE) A definition explaining the capabilities of an application entity.

application service object A subobject of an application entity; it may contain ASE elements or service objects as subobjects.

application transaction program The program built around LU6.2 protocols.

apply A System Modification Program (SMP) command, used in certain SNA environments, whereby programs and/or program fixes can be added to system libraries.

APPN intermediate routing network An APPN network consisting of network nodes and their interconnections.

argument list According to Digital Equipment, a vector of entries representing a procedure parameter list and possibly a function value.

argument pointer According to Digital Equipment, general register 12 on VAX systems. By convention, AP contains the address of the base of the argument list for procedures initiated using CALL instructions.

ARPAnet Served as central backbone for the Internet during the 1970s and most of the 1980s.

assignment statement In Digital Command Language (DCL), the association of a symbol name to use with a character string or numeric value. In Digital Equipment implementations, symbols can define synonyms for system commands or can be used for variables in command procedures.

asynchronous (ASYNC) In data communication, transmission of data without synchronous protocol.

asynchronous system trap A method of identifying a specific event via a software simulated interrupt.

Asynchronous Transfer Mode (ATM) A CCITT standard defining cell relay. This is where data of multiple types of services, such as data, voice, and/or video, are moved in fixed-size cells throughout a network.

association control service element Used to establish and terminate associations between applications.

attenuation A term used with fiber optics. It refers to the reduction of signal loss, measured in decibels.

attribute From a security perspective, an identifier or holder of an identifier. When used in a conversation concerning threads, it specifies detailed properties about the objects to be created.

attributes object A term describing details of the objects to be created.

authentication A process of establishing identity.

authorized path In VTAM under MVS, a facility that enables an application program to specify that a data transfer or related operation be carried out in a privileged and more efficient manner.

automatic logon In SNA, a process by which VTAM automatically creates a session-initiation request to establish a session between two logical units (LUs). The result of this request is the SNA BIND command from the primary logical unit (PLU) to the secondary logical unit (SLU). Hence, an LU-LU session is established if the BIND image is correct and other parameters are accurate.

automatic record locking According to Digital Equipment documentation, a capability provided by Open VMS Record Management Services that allows a user to lock only one record in a specific shared file at any given time.

A/UX toolbox According to Apple documentation, a library that enables a program running under the A/UX operating system to call the Macintosh user interface toolbox and operating system routines.

background process A term used in most UNIX environments for a process that does not require total attention of the system for operation.

backplane interconnect According to Digital Equipment documentation, VAX systems have an internal processor bus that allows I/O device controllers to communicate with main memory and the central processor. These I/O controllers may reside on the same bus as memory and the central processor, or they may be on a separate bus.

BASE Disk In SNA, and specifically the VM operating system, the disk containing text decks and macroinstructions for VTAM, NetView, and VM/SNA console support (VSCS).

base priority Priority that a VM system assigns to a process when it is created. Normally, it comes from the authorization file.

basic logical object An object in a specific logical structure that has no subordinate.

baud The number of times per second that a signal can change on a transmission line.

begin bracket In SNA, the value of the begin-bracket indicator in the request header (RH). It is the first request in the first chain of a bracket. Its value denotes the start of a bracket.

Berkeley broadcast A non standard IP broadcast address using all zeros instead of all ones in the host portion.

best-Effort delivery A description of network technologies that do not provide reliability at link levels.

Bézier curve A curve defined by three outline points, two of which serve as endpoints, also called *control handles;* the third point does not lie on the curve. These points determine the degree of curvature.

Big Endian A format for storage or transmission of binary data in which the most significant byte comes first.

Binary Synchronous Communication (BSC) A telecommunication protocol using a standard set of transmission control characters and control character sequences.

BIND In SNA, a request for session activation between logical units.

BIND command In SNA, the command used to start a session and define the characteristics thereof.

BIND image In SNA, this consists of session parameters used to establish and govern the session between logical units. In VTAM, the BIND image is located in the LOGMODE table in the form of entries—one entry per type of image required to define the LU.

BIND pacing In SNA, BIND pacing can be used to prevent a BIND standoff. This technique is used by the Address Space Manager (ASM).

bitmap Generally speaking, an array of data bits used for graphic images.

bitmap device A device that displays bitmaps.

bitmap font A bitmap font is created made from a matrix of dots.

BIU segment In SNA, data contained within a path information unit (PIU). It consists of either a request/response header (RH) followed by all or part of a request/response unit (RU), or only part of an RU.

blocking AST An AST that can be requested by a process using the block management system services. A blocking AST is delivered to the requesting process when it prevents another process from accessing a resource.

boot server According to Digital Equipment documentation, the management center for a VMScluster system and its major source provider. The boot server provides disk access and downloads satellite nodes.

bootstrap block That part of the index file on a system disk that contains a program which loads the operating system into memory.

boundary function In SNA, protocol support for attached peripheral nodes.

boundary node (BN) A boundary node fundamentally performs the function of transforming network addresses to local addresses.

bracket In SNA, one or more chains of RUs and their responses that are exchanged between session partners.

bracket protocol A data flow control protocol in which session partners exchange data via IBM's SNA bracket protocol.

broadcast IN TCP/IP, a request for logical connection at layers 1 and 2 in the network. Broadcast technology is exemplified by Ethernet.

broadcast search In APPN, the simultaneous search to all network nodes in the form of a request for some type of data.

broadcast storm A situation in a network using broadcast technology where a considerable number of broadcasts are put on the medium at one time.

browse With regard to functions that can be performed on an entity, browse merely permits viewing.

Btrieve According to Novell documentation, this is a complete indexed record management system designed for high-performance data handling.

bucket According to Digital Equipment documentation, a storage structure used for building and processing files. A bucket contains one or more records or record cells. Buckets are the unit of contiguous transfer between Open VMS Record Management Services buffers and the disk.

buffer A temporary storage area used to hold input or output data.

bundle bit The FINDER uses information in a bundle bit (BNDL resource) to associate icons with the file.

button Generally refers to a mouse button. Mice may have two or three buttons. Two-button mice are common in non-X-Windows environments. Three-button mice are common in X-Windows environments and can be used in other environments as well, depending on the mouse vendor.

cache A high-speed storage buffer.

call To invoke a program, routine, or subroutine.

call-connected packet A DTE packet transmission indicating DCE acceptance of the incoming call.

call progress signal Communication from the DCE to the calling DTE indicating the status of a call.

call request packet In X.25 communications, a call supervision packet transmitted by a DTE to ask for a call establishment through the network.

calling The process of transmitting selection signals to establish a connection between data stations.

cancel To terminate.

carrier In data communication, a continuous frequency capable of being modulated. Also a signal.

carrier sense A device (transceiver, interface board, or other entity) capable of detecting a constant frequency.

Carrier Sense Multiple Access with Collision Detection (CSMA/CD) A protocol utilizing equipment capable of detecting a carrier which permits multiple access to a common medium. This protocol also has the ability to detect a collision because this type of technology is broadcast-oriented.

casual connection In SNA, a connection made in subarea networks with PU T5 nodes attached via a boundary function using low-entry networking capabilities.

catalog In SNA, a list of pointers to libraries, files, or data sets.

central directory server According to IBM documentation, an APPN network node that provides a repository for network resource locations.

central processing unit (CPU) The circuitry that executes instructions.

channel Generically speaking, a channel is a path over which data can move. In most IBM documentation, the word *channel* has a specific meaning. For example, IBM supports two types of channels today: parallel and serial.

channel adapter A device used to attach the communication controller to a host channel.

channel-attached In SNA terminology, this indicates a serial or parallel data-link protocol. IBM has other data-link protocols, such as SDLC, ESCON, and Token Ring.

channel link A data-link connection between two devices.

channel unit address (CUA) This term is used in SNA, particularly VTAM.

character-coded An SNA term used in VTAM meaning unformatted.

chooser According to Apple documentation, an accessory that lets a user select shared devices, such as printers and file servers.

CI-only VMScluster configuration According to Digital Equipment documentation, this is a type of VMScluster configuration in which the computer interconnect device is used for most interprocessor communication. In these configurations, a node may be a VAX processor or a hierarchical storage controller (HSC).

CINIT A request sent from an SSCP to a primary logical unit requesting that a BIND command be issued.

circuit switching Connectivity on demand between DTEs and DCEs.

cladding The surrounding part of a cable that protect the core optical fibers. This part is between the fibers and the cable jacket.

class of service (COS) In SNA, class of service defines explicit routes, virtual routes, and priority, and is used to provide a variety of services within the network.

class-of-service database In APPN, this is a database maintained independently by each network node, and optionally by APPN end nodes.

cleanup A term used in SNA referring to how sessions are terminated between LUs. Specifically, it is a network services request that causes a particular LU-LU session to be ended immediately.

clear-indication packet A call supervision packet that data circuit-terminating equipment (DCE) transmits to inform data terminal equipment (DTE) that a call has been cleared.

clear request packet A call supervision packet transmitted by DTE to ask that a call be cleared.

click To press and release a mouse button.

client A term used to connote peer technology. Clients are programs used to initiate something. In TCP/IP clients can be found in TELNET and FTP. In the X-Windowing system, a client is typically a program that conforms to the X protocol and works in conjunction with an X server.

client application In Apple environments, an application using Apple Event Interprocess Messaging Protocol to request a service, such as printing a list of files or spell-checking a list of words.

clocking The use of clock pulses to control synchronization.

closed user group (CUG) A user group that can communicate only with other users in the group.

CLSDST Close destination.

cluster A network of computers in which only one computer has attached file-system disk drives.

cluster controller In SNA, the precursor to the establishment controller. It is a device to which terminals and printers attach. IBM's 3174 cluster controllers appear as a PU2.0.

CMOT The use of Common Management Information Services over TCP/IP. Succinctly, it is the implementation of the OSI network management specification over TCP/IP.

collision An event in which two or more devices simultaneously perform a broadcast on the same medium. This term is used in Ethernet networks, and also in networks where broadcast technology is implemented.

collision detection A term used to define a device that can determine when a simultaneous transmission attempt has been made.

command A request to execute an event. According to Digital Equipment, when this is used in reference to Digital Command Language (DCL), it means an instruction, generally an English-type word, entered by the user at a terminal or included in a command procedure.

command facility A component of IBM's NetView program that is the base for command processors that can monitor, control, automate, and improve the operation of an SNA network.

command-line prompt Generally, this refers to a place on a terminal where commands can be entered. The direction of these commands is contingent on the system or software from which the command line is generated. According to Hewlett-Packard (HP) documentation, this is the prompt that appears on the screen immediately after login. Usually the command-line prompt is either a $ (for Bourne and Korn shells) or a % (for C shells), but it can be modified. A popular modification is to print the current working directory and the history stack number before the $ or %. You can find the command-line prompt by pressing RETURN several times. Everytime you press RETURN, the HP-UX operating systems prints the prompt.

command list (CLIST) In IBM's SNA, this is a list of commands and statements designed to perform a specific function for the user. A variety of CLISTs exists. Examples are NetView, TSO, and REXX.

Common Management Information Protocol (CMIP) The OSI protocol for systems management.

common management information service element The application service element responsible for relaying systems management information.

common parent The lowest-level directory that appears in pathnames of multiple files or directories on the same volume.

communication adapter Generally agreed to be a device (normally an interface) that provides a common point for a device and the data-communication device being used.

communication control unit A communication device that controls transmission of data over lines in a network.

communication controller A control unit whose operations are controlled by one or more programs stored and executed in the unit. In most instances it manages lines and routing of data through a network.

communication identifier (CID) A VTAM key for locating the control blocks that represent a session. This key is created during session establishment and deleted when the session ends.

communication line Deprecated term for telecommunication line.

communication network management (CNM) According to IBM documentation as the term applies to SNA, this is the process of designing, installing, operating, and managing distribution of information and control among users of communication systems.

communication network management (CNM) application program According to IBM documentation, this is a VTAM application that issues and receives formatted management services request units for PUs. NetView is an example of a CNM application program.

communication network management (CNM) interface A common point where applications can move data and commands to the access method. These data and commands are associated with communication system management.

component Generally speaking, this can be hardware or software.

composite LEN node A group of nodes made up of a single type 5 node and subordinate type 4 nodes. To a type 2.1 node, a composite LEN node appears as one LEN node. Examples of a composite LEN node are NCP and VTAM.

composite network node (CNN) A group of nodes made up of a type 5 node and its subordinate type 4 nodes that appear as a single APPN network node to the APPN network.

composite node In IBM networking, specifically SNA with VTAM, a type 5 node and its owned type 4 nodes that collectively appear as a single node to other APPN nodes in an APPN network.

computer interconnect (CI) A fault-tolerant, dual-path bus, with a bandwidth of 70 Mbits/s.

concurrency controls Methods provided by Btrieve to resolve possible conflicts when two applications attempt to update or delete the same records at the same time.

configuration The manner in which the hardware and software of an information-processing system are organized and interconnected.

configuration restart In SNA, the VTAM recovery facility that can be used after a failure or deactivation of a major node, VTAM, or the host processor to restore the domain to its status at the time of the failure or deactivation.

configuration services Network services in a control point that activate, deactivate, and record the status of PUs, links, and link stations.

congestion loss In Digital Equipment's DECnet, a condition in which data packets transmitted over a network are lost when the DECnet for the Open VMS Routing layer is unable to buffer them.

connected To have a physical path from one point to another.

connection In data communications, there are two types of connection: physical and logical. A *physical* connection consists of a tangible path between two or more points. In a *logical* connection two or more endpoints are able to communicate.

connection control block A data structure that is used by ADSP to store state information about the connection end.

connection end The combination of an AppleTalk socket and the ADSP information maintained by the socket client.

connection listening socket A socket that accepts ADSP requests to open connections and passes them on to a socket client.

connection network A representation within an APPN network of a shared-access transport facility (SATF), such as a Token Ring, that allows nodes identifying their connectivity to the SATF by a common virtual routing node to communicate without having individually defined connections to one another.

connection-oriented internetworking A set of subnetworks connected physically and capable of connection-oriented network service.

connection-oriented service A type of service offered in some networks and consisting of three phases: connection establishment, data transfer, and connection release.

connection server A program that accepts an open-connection request passed to it by a connection listener and selects a socket to respond to the request.

connectionless internetworking A set of subnetworks connected physically and capable of providing connectionless network service.

Connectionless Network Protocol (CLNP) A protocol that does not perform retransmissions or error recovery at a transport layer.

connectionless service A network service that delivers data or packets as separate pieces. An example of connectionless service is TCP/IP's Internet Protocol (IP).

connectivity The notion of device communication interchange, even if such devices are diverse.

console communication services (CCS) A term used in SNA environments. CCS acts as an interface between the control program and the VSCS component of VTAM for VM.

contention A term frequently used in SNA. It has multiple meanings, depending on the context. One example relates to sessions. In this case the network-accessible units attempt to initiate the same action at the same time against the other. Another example is contention in LU6.2. Here, it is the attempt to allocate a session by two programs against the other at the same time.

context Information of a process maintained by the process manager.

context dependence When the glyph corresponding to a character can be modified depending on the preceding and following characters.

continue-any mode This specifies whether VTAM is to receive the data in terms of logical records or buffers.

continue-specific mode In IBM's VTAM, the state of a session or APPC conversation permitting its input to satisfy only RECEIVE requests issued in specific mode.

contour A closed loop in a TrueType outline glyph, defined by a group of outline points.

control block A storage area that holds control information.

control logical unit (CLU) A logical unit that resides in a transaction-processing facility (TPF) type 2.1 node. It is used to pass private protocol requests between the TPF type 2.1 node and the logon manager which is a VTAM application program. Communication flow between the CLU and the logon manager enables a logical unit controlled by VTAM to establish a session with TPF.

control panel A place that allows a user to set or "control" a feature of some sort.

control panels folder In Apple documentation, a directory located in the system folder for storing control panels, allowing users to modify their work environment on their Apple or Macintosh computer.

control point (CP) The managing component of a type 2.1 node that manages the resources of that node. In an APPN network node the CP can engage in CP-CP sessions with other APPN nodes.

control program (CP) Normally this type of program performs system-oriented functions such as scheduling and supervising execution of programs of a computer system. In IBM's Account Interactive eXecutive it is that part of the operating system which determines the order in which basic functions should be performed.

control vector A particular structure that is one of a general class of RU substructures basically characterized by a variable length and having a 1-byte key used as an identifier.

controller A device that coordinates and controls the operation of one or more input/output devices.

controlling application program According to IBM documentation, an application program with which a secondary logical unit (other than an application program) is automatically put in session whenever the secondary logical unit is available.

controlling logical unit This can be either an application program or a device-type LU.

conversation A logical connection between two transaction programs using an LU 6.2 session.

conversational monitoring system (CMS) An IBM Virtual Machine (VM) operating system facility that provides interactive timesharing, problem solving, and program development capabilities. CMS operates under control of the control program component of a VM system.

converted command An intermediate form of a character-coded command produced by VTAM through use of an unformatted system services definition table. Converted command format is fixed. The unformatted system services (USS) table must be constructed in such a manner that character-coded commands are converted into the predefined, converted command format.

core Apple event According to Apple documentation, an event that nearly all applications can use to communicate.

cost According to Digital Equipment documentation, a numeric value assigned to a circuit that exists between two adjacent nodes. In the DECnet for Open VMS network, data packets are routed on paths with the lowest cost.

CP capabilities One definition according to IBM documentation is the level of network services provided by the control point (CP) in an APPN end node or network node. Control Program capabilities are exchanged during activation of CP-CP sessions between nodes.

CP-CP session A parallel session between two control points, using LU6.2 protocols and a mode name of CPSVCMG, on which network services requests and replies are exchanged.

CP name The name of a control point (CP), consisting of a network ID qualifier identifying the network to which the CP's node belongs, and a unique name within the scope of that network ID identifying the CP. Each APPN or LEN end node has one CP name assigned to it at system-definition time.

cross-domain In SNA, this term refers to resources in a different domain. Domains, in SNA, relate to ownership.

cross-domain keys In SNA, a pair of cryptographic keys used by a system services control point (SSCP) used to encipher the session cryptography key that is sent to another SSCP.

cross-domain link A subarea link connecting two subareas that are in different domains. A physical link connecting two domains.

cross-domain resource (CDRSC) A term used in SNA referring to a resource (typically software) that resides in another domain, under the control of a different VTAM.

Cross-Domain Resource Manager (CDRM) A term used in SNA referring to the function in the SSCP which controls initiation and termination of cross-domain sessions.

cross-network LU-LU session A reference used in SNA to refer to a session between logical units (LUs) in different networks.

cross-network session A session whose path traverses more than one SNA network.

cryptographic In SNA terminology, this pertains to transformation of data to conceal its meaning.

cryptographic session In SNA, an LU-LU session in which a function management data (FMD) request may be enciphered before it is transmitted and deciphered after it is received.

CSALIMIT Common service area (CSA) buffer-use limit.

currency The previous, current, and next position of a record in a file. There are two types of currency: logical and physical. The *physical* previous, current, and next positions form the physical currency. The *logical* previous, current, and next positions form the logical currency. According to Zac, it means money.

Customer Information Control System (CICS) An IBM software program that supports real-time transactions between remote users and custom-written transactions. It includes facilities for building, using, and maintaining databases.

cut buffer A memory area that holds text which has been deleted from a file.

cycle The complete oscillation of a wave.

CWALL An NCP threshold of buffer availability below which the NCP will accept only high-priority path information units (PIUs).

data channel An IBM term used as a synonym for input/output channel.

data circuit A pair of associated transmit and receive channels providing a means for two-way data communication.

data circuit-terminating equipment (DCE) In a data station, equipment that provides signal conversion and coding between the data terminal equipment (DTE) and the line.

data communication Transfer of data among functional units by means of data transmission according to a protocol. The transmission, reception, and validation of data.

Data Encryption Standard (DES) In computer security, this refers to the National Institute of Standards and Technology (NIST) Data Encryption Standard, adopted by the U.S. government as *Federal Information Processing Standard* (FIPS) Publication 46, which allows only hardware implementations of the data encryption algorithm.

data flow control (DFC) In SNA, a request or response unit (RU) category used for requests and responses exchanged between the DFC layer in one half-session and the DFC layer in the session partner.

data host node A term used in SNA referring to a host dedicated to processing applications that does not control network resources, except for its channel-attached or communication adapter-attached devices.

data link In SNA, a synonym for link.

data-link control (DLC) A set of rules used by nodes at layer 2 within a network. The data link is governed by data-link protocols such as Ethernet or Token Ring.

Data-Link Control (DLC) Protocol Rules used by two nodes at a data-link layer to accomplish an orderly exchange of information. Examples are Ethernet, channel, FDDI, and Token Ring.

data-link layer Layer 2 of the OSI reference model. It synchronizes transmission and handles error correction for a data link.

data-link level The conceptual level of control logic between high-level logic and a data-link protocol that maintains control of the data link.

data network An arrangement of data circuits and switching facilities for establishing connections between data terminal equipment. A term commonly found in X.25 network implementations.

data packet In X.25, a packet used for the transmission of user data on a virtual circuit at the DTE/DCE interface.

data server According to common definition in Apple documentation, an application that acts like an interface between a database extension on a Macintosh computer and a data source, which can be on either the Macintosh computer or on a remote host computer. It can be either a database server program that can provide an interface to a variety of different databases or the data source itself, such as a Macintosh application.

data set The manner in which data, programs, and other representations of information are stored in IBM's MVS operating system environment.

data stream In SNA, multiple data streams are identified. Generally, this is a continuous stream of data elements being transmitted, in character or binary-digit form, using a defined format.

data switching exchange (DSE) Equipment at a single location that provides switching functions, such as circuit switching, message switching, and packet switching.

data terminal equipment (DTE) That part of a data station which constitutes a data source, data link, or both.

data types IN SNA, particularly NetView, these can be alerts, events, or statistics.

data unit In the OSI environment, the smallest unit of a file content meaningful to an FTAM file action.

database A collection of data with a given structure.

datagram delivery protocol In AppleTalk networks, a protocol that provides socket-to-socket delivery of data packets.

DCE clear confirmation packet A call supervision packet that a DCE transmits to confirm that a call has been cleared.

deactivate A term frequently used in SNA environments meaning to take a resource out of service.

deallocate A term used in APPC indicating that an LU6.2 application program interface (API) terminates a conversation and makes the session free for a future conversation.

decipher To convert enciphered data in order to restore the original data.

decrypt In computer security, to decipher or decode. Synonym for decipher.

de facto standard A standard that is the result of some technology that has been developed and used and has achieved some level of popularity.

default SSCP list In VTAM, a list of SSCPs to which a session request can be routed when a Cross-Domain Resource Manager (CDRM) is not specified.

Defense Advanced Research Projects Agency (DARPA) Formerly called *Advanced Research Projects Agency* (ARPA). The government agency that funded research and experimentation with the ARPAnet.

Defense Data Network (DDN) Used loosely to refer to MILNET, ARPAnet, and the TCP/IP protocols they use. More specifically, it is MILNET and associated parts of the connected Internet that connect military installations.

defined context set A set of presentation contexts negotiated between peer presentation entities.

definite response (DR) According to IBM documentation, this is used in SNA and is a protocol that requests the receiver of the request to return a response unconditionally, whether positive or negative, to that request chain.

definition statement In IBM's VTAM program, this statement describes an element of the network. In IBM's NCP, it is a type of instruction that defines a resource.

DELTA disk In SNA, the virtual disk in a VM operating system that contains program temporary fixes (PTFs) that have been installed but not yet merged.

de jure standard A standard set by a body or official consensus.

descent line An imaginary line usually marking the greatest distance below the baseline of the descenders of glyphs in a particular font.

descriptor A data buffer parameter passed for an extended-get or extended-step operation.

descriptor type An identifier for the type of data referred to by the handle in a descriptor record.

desktop folder In Apple environments, a directory located at the root level of each volume. It is used by the FINDER to store information about the icons that appear on the desktop area of the screen. The desktop folder is what the user sees on the screen.

destination logical unit (DLU) The logical unit to which data are to be sent.

device When the term is used in networking scenarios, it is typically used generically for a modem, host, terminal, or other entity.

dial-in In most SNA environments, the notion of inbound traffic toward the host.

dial-out In most networking environments, the notion of outbound capabilities to access resources elsewhere.

Digital Command Language According to Digital Equipment documentation, a command interpreter in the operating system. It provides a means of communication between the user and the operating system.

Digital Data Communication Message Protocol (DDCMP) According to Digital Equipment documentation, the link-level protocol that DEC uses in their network products. DDCMP operates over serial lines, delimits frames by a special character, and includes checksums at the link level. It was relevant to TCP/IP because the original NSFNET used DDCMP over its backbone lines.

Digital Network Architecture According to Digital Equipment, a set of protocols governing the format, control, and sequencing of message-exchange for all Digital network implementations. The protocols are layered, and they define rules for data exchange from the physical-link level up through the user interface level. DNA controls all data that travel throughout a Digital network. DNA also defines standard network management and network generation procedures.

Digital Storage Architecture (DSA) According to Digital Equipment documentation, the specifications from Digital governing the design of and interface to mass storage products. DSA defines the functions to be performed by host computers, controllers, and drives, and specifies how they interact to manage mass storage.

Digital Storage Systems Interconnect A data bus that uses the System Communication Architecture protocols for direct host-to-storage communications. The DSSI cable can extend to 6 m and has a peak bandwidth of 4 MB.

Direct-Access Storage Device (DASD) An IBM term for a disk drive in mainframe environments. A device in which access time is effectively independent of the location of the data.

direct activation According to IBM documentation, the activation of a resource as a result of an activation command specifically naming the resource.

direct deactivation According to IBM documentation, the deactivation of a resource as a result of a deactivation command specifically naming the resource.

directed broadcast address In TCP/IP-based environments, an IP address that specifies all hosts on a specific network. A single copy of a directed broadcast is routed to the specified network where it is broadcast to all machines on that network.

directed locate search A search request sent to a specific destination node known to contain a resource, such as a logical unit, to verify the continued presence of the resource at the destination node and to obtain the node's connectivity information for route calculation.

directed search Synonym for directed locate search.

directory (1) Depending on the environment, this term is used in different ways. For example, in UNIX environments a directory is a listing of files and the files themselves. This definition is generally the case in most environments; however, some vendors contend that a significant difference exists. (2) In IBM's VM/SP environment, a directory is a control program (CP) disk file that defines each virtual machine's normal configuration: the user ID, password, normal and maximum allowable virtual storage, CP command privilege classes allowed, dispatching priority, logical editing symbols to be used, account number, and CP options desired. In APPN, this is a database that lists names of resources and records the CP name of the node where each resource is located. Another definition of a directory is the subdivision of a volume, available in the hierarchical file system. A directory can contain files and other directories.

Directory Access Protocol The protocol used between a directory user agent and a directory system agent.

directory entry An object in the directory information base for modeling information. It can be an object entry or an alias entry.

directory information base A set of directory entries. It contains objects to which the directory provides access and which includes all of the pieces of information that can be read or manipulated using the directory operations.

Directory Information Shadowing Protocol A protocol used for shadowing between two directory service agents in the directory services standard.

directory information tree A tree structure of the directory information base.

directory name Names for directory entries in the directory information base.

Directory Operational Binding Management Protocol A protocol used by directory service agents to activate showing agreement. This allows directory service agents to establish, modify, and terminate operational bindings.

directory service (DS) According to its use in OSI environments, an application service element that translates the symbolic names used by application processes into the complete network addresses used.

directory service agent An application entity that offers the directory services.

Directory Service Protocol The protocol used between two directory system agents.

directory services (DS) A control-point component of an APPN node that maintains knowledge of the location of network resources.

directory user agent An application entity that provides the directory services.

disable In a loose sense this term is similar to *deactivate*.

disabled Pertaining to a state of a processing unit that prevents the occurrence of certain types of interruption.

discarded packet A piece of data, called a *packet*, that is intentionally destroyed.

disconnection Termination of a physical connection.

discontiguous shared segment (DCSS) According to IBM documentation, an area of virtual storage outside the address range of a virtual machine. It can contain read-only data or reentrant code. It connects discontiguous segments to a virtual machine's address space so that programs can be fetched.

discretionary controls Security controls that are applied at the user's option.

distributed computing environment (DCE) An Open Software Foundation (OSF) set of standards for distributed computing. Also, DCE is distributed computing in the general sense of the term.

disjoint network According to IBM documentation, a network of two or more subnetworks with the same network identifier that are indirectly connected, for example, through SNA network interconnection.

disjoint SSCP According to IBM documentation, an SSCP that does not have a direct SSCP-SSCP session with other SSCPs in its network-ID subnetwork.

disk cache A part of RAM that acts as an intermediate buffer when data are read from and written to file systems on secondary storage devices.

display A term generally used to refer to a terminal.

display server As defined in X-Windows system arenas, this is the software that controls the communication between client programs and the display including the keyboard-mouse-screen combination.

distinguished name Name of a directory entry.

distributed directory database According to documentation related to IBM's APPN architecture, the complete listing of all resources in the network as maintained in individual directories scattered throughout an APPN network.

distributed network directory Synonym for distributed directory database.

domain That part of a computer network in which the data-processing resources are under common control. Also a part of the DNS naming hierarchy. Syntactically, a domain name consists of a sequence of names separated by periods.

Domain Name Service (DNS) In TCP/IP environments, a protocol for matching object names and network addresses. It was designed to replace the need to update `/etc/hosts` files of participating entities throughout a network.

domain name system (DNS) The online distributed database system used to map human-readable machine names into IP addresses. DNS servers throughout the connected Internet implement a hierarchical namespace that allows sites freedom in assigning machine names and addresses. DNS also supports separate mappings between mail destinations and IP addresses.

domain operator According to IBM documentation, a person or program that controls operation of resources controlled by one SSCP.

domain search In the context of APPN networking, a search initiated by a network node to all its client APPN end nodes when a search request is received by a network node and that network node does not have any entry in its database for the requested resource.

dotted-decimal notation A phase typically found in TCP/IP network conversations. Specifically, this refers to the addressing scheme of the Internet Protocol (IP). It is the representation of a 32-bit address consisting of four 8-bit numbers written in base 10 with periods separating them.

double-byte character set (DBCS) A set of characters in which each character is represented by 2 bytes. For example, languages such as Japanese, Chinese, and Korean use this method to represent characters.

downline system load According to Digital Equipment documentation, this is a DECnet for Open VMS function that permits an unattended target node to receive an operating system file image or terminal server image from another node.

drag In Apple and X-Windows environments this means to press and hold down a mouse button while moving the mouse on the desktop. Typically, dragging is used with menu selecting, moving, and resizing operations.

drain This term refers to APPC. It means to honor pending allocation requests before deactivating session with a partner logical unit.

drop cable In the IBM cabling system, this cable runs from a faceplate to the distribution panel in a wiring closet. In TCP/IP networking environments where thicknet cable and transceivers are used, it is that cable between the devices' network interface card and the transceiver.

drop folder In Apple environments, a type of folder (holding place) that serves as a private mailbox for individuals. Once someone places a file in a drop folder, only the owner of the drop folder can retrieve it. According to Apple documentation, users can create drop folders by setting the appropriate AppleShare or Macintosh file-sharing access privileges.

DSRLST Direct search list. In SNA this is a message unit that contains a search request sent throughout subarea networks to obtain information about a network resource such as its name and routing and status information.

DTE/DCE interface The physical interface and link access procedures between data terminal equipment (DTE) and data circuit-terminating equipment (DCE).

dump A term used frequently in SNA environments. Typically it means to obtain the contents of some aspect of memory. Specifically, it means to record, at a particular instant, the contents of all or part of one storage device in another storage device. One noted professional in the field, Zelma Gandy, defines a dump as "The contents of memory used for debugging." A *core dump* refers to the extraction of the contents of main memory. A *VTAM dump* is used loosely to refer to reading data areas in IBM's VTAM program offering.

duplex Pertaining to communication in which data can be sent and received at the same time.

dynamic In a generic sense, this means to do something on the fly. A more specific explanation is to perform an operation that does not require a predetermined or fixed time.

dynamic node ID assignment According to Apple documentation, this refers to the AppleTalk addressing scheme that assigns node IDs dynamically, rather than associating a permanent address with each node. Dynamic node ID assignment facilitates addition and removal of nodes from the network by preventing conflicts between old node IDs and new node IDs.

dynamic path update A generic reference meaning the process of changing network path parameters for sending information without regenerating complete configuration tables.

dynamic reconfiguration data set (DRDS) According to IBM documentation, this is a term used in VTAM. It refers to a data set used for storing definition data that can be applied to a generated communication controller configuration at the operator's request or can be used to accomplish dynamic reconfiguration of NCP, local SNA, and packet major nodes. This type of reconfiguration data set can be used to dynamically add PUs and LUs, delete PUs and LUs, and move PUs. It is activated with the VARY DREDS operator command.

dynamic window A window that may change its title or reposition any of the objects within its content area.

echo In data communication, a reflected signal on a communications channel.

Electronic Data Interchange (EDI) A set of standard data formats for electronic information exchange.

element This term has different meanings in different networking environments. In SNA, it means the particular resource within a subarea that is identified by an element address.

element address According to IBM documentation, a value in the element address field of the network address identifying a specific resource within a subarea.

electromagnetic interference (EMI) A type of noise that results when currents are induced in electric conductors.

emulation program (EP) A program that simulates the functions of another program. A generic example could be a 3270 terminal emulation program. Other possibilities also exist.

EN In IBM's APPN architecture manuals, this refers to an APPN end node.

enable To make functional. In a loose sense, this means to activate.

enabled Capable of performing work.

encapsulate Generally agreed on in the internetworking community as surrounding one protocol with another protocol to pass the foreign protocol through the native environment.

encipher According to IBM documentation, this means to scramble data or to convert data to a secret code that masks the meaning of the data to any unauthorized recipient. In VTAM, to convert clear data into enciphered data.

encrypt Synonym for encipher.

encryption The process of transforming (or scrambling) data into an unintelligible form.

end bracket A term specifically used in SNA. It is the value of the end-bracket indicator in the request header (RH) of the first request of the last chain of the bracket. The value denotes the end of the bracket.

end node With reference to use in APPN, a node that can receive packets addressed to it and send packets to other nodes. It cannot route packets from other nodes.

end-node domain That area defined by an end-node control point, attached links, and its local LUs.

end system–intermediate system (ES-IS) An Open Systems term used to refer to a routing exchange protocol that provides an automated means for ISs and ESs on a subnetwork to dynamically determine each other's existence. It also means to permit an IS to inform an ES of a potentially better route toward a destination.

end user Defined by IBM documentation as either a program or a human.

end-user verification LU6.2 identification check of end users by means of identifiers and passwords on the attach function management headers (FMHs).

entry mask According to Digital Equipment documentation, on VAX systems it is a word where the bits represent the registers to be saved when a procedure is called with a CALLS or CALLG instruction, and restored when the procedure executes a RET instruction.

entry point According to IBM documentation, a type 2.0, 2.1, 4, or 5 node that provides distributed network management support. It sends network management data about itself and the resources it controls to a focal point for centralized processing, and it receives and executes focal-point-initiated commands to manage and control its resources.

Ethernet A data-link-level protocol. It (version 2.0) was defined by Digital, Intel, and the Xerox Corporations in 1982. It specified a data rate of 10 Mbits/s, a maximum station distance of 2.8 km, maximum number of stations as 1024, a shielded coaxial cable using baseband signaling, functionality of CSMA/CD, and a best-effort delivery system.

Ethernet meltdown A term used where Ethernet protocol is used as the data-link-layer protocol in a network. It is an event that causes saturation or near saturation on an Ethernet data link. This scenario usually results from illegal or misrouted packets and lasts a short time.

EtherTalk A term used in Apple and Ethernet environments. It is software that enables AppleTalk protocols to run over industry-standard Ethernet technology.

event This term can mean a predefined occurrence in a given network. It has a specific meaning in SNA, where, in NetView, it is a record indicating irregularities of operation in physical elements of a network.

event class According to Apple documentation, an attribute that identifies a group of related Apple events. The event class appears in the message field of the Apple event's event record. In conjunction with the event ID attribute, the event class specifies what action an Apple event performs.

event ID According to Apple documentation, an attribute that identifies a particular Apple event within a group of related Apple events. The event class appears in the WHERE field of the Apple event's event record. In conjunction with the event class attribute, the event ID specifies what action an Apple event performs.

exception An abnormal condition in comparison to what has been predefined as a normal condition.

exception response (ER) According to IBM documentation, a protocol requested in the for-of-response-requested field of a request header that directs the receiver to return a response only if the request is unacceptable as received or cannot be processed.

exception service routine According to Digital Equipment documentation, a routine by which VAX and AXP hardware initially pass control to service an exception. An exception service routine passes control to a general exception dispatcher that attempts to locate a condition handler to further service the exception.

exchange identification (XID) In SNA, this is a specific type of basic link unit used to convey node and link characteristics between adjacent nodes. In the SNA network, XIDs are exchanged between link stations before and during link activation to establish and negotiate link and node characteristics, and after link activation to communicate changes in these characteristics.

EXEC According to IBM documentation, in a VM operating system, a user-written command file that contains CMS commands, other user-written commands, and execution control statements, such as branches.

executable image An image that can be run in a process. When run, an executable image is read from a file for execution in a process.

executive A generic name for the collection of procedures included in the operating system software that provides the basic OS control and monitoring functions.

executive mode According to Digital Equipment documentation, the second most privileged processor access mode. The Open VMS record management services and many of the operating system's system service procedures execute in executive mode.

exit To execute an instruction within a portion of a program in order to terminate the execution of that portion.

exit list (EXLST) According to IBM documentation, in VTAM it is a control block that contains the addresses of routines that receive control when specified events occur during execution.

exit program Synonym for exit routine.

exit routine One of two types of routines: installation exit routes or user exit routes.

expedited flow According to IBM documentation, a data flow designated in the transmission header (TH) used to carry network control, session control, and various data flow control request/response units (RUs). Expedited flow is separate from the normal flow which carries primarily end-user data.

explicit route (ER) According to IBM's documentation, in SNA a series of one or more transmission groups that connect two subarea nodes. It is identified by an origin subarea address, a destination subarea address, an explicit route number, and a reverse explicit route number.

explicit route length The number of transmission groups in an explicit route.

Extended Architecture (XA) According to IBM documentation, an exten sion to System/370 architecture. It takes advantage of changing features such as the addressability of the hardware architecture.

extended attribute block According to Digital Equipment documentation, an Open VMS record management services user data structure that contains additional file attributes beyond those expressed in the file access block, such as boundary types and file-protection information.

Extended Binary-Coded Decimal Interchange Code (EBCDIC) IBM's coded character set of 8-bit characters (256 total) and control functions.

extended network addressing In IBM's traditional subarea networking, this is the addressing system that splits addresses into an 8-bit subarea and a 15-bit element portion. The subarea portion of the address is used to address host processors or communication controllers. The element portion is used to permit processors or controllers to address resources.

extended recovery facility (XRF) In SNA, a facility that provides an alternate subsystem to take over sessions from the failing subsystem.

extended subarea addressing According to IBM documentation, a network addressing system used in a network with more than 255 subareas.

fan-out box A term used in LAN circles that refers to a device that functions like a hub. It provides the capability for multiple connections to make a central connection.

Fiber Distributed Data Interface (FDDI) An IEEE 802–compatible physical and data-link control standard for a 100-Mbit/s fiber ring; an ANSI standard for a LAN using optical fibers.

field-formatted In SNA, this pertains to requests or responses that are encoded into fields, each having a specified format such as binary codes, bit-significant flags, and symbolic names.

file access data unit A subtree of the hierarchical access structure. It is used to specify a location on a file structure.

file attributes Terminology whose origins are in OSI which refer to the properties of a file that do not depend on an FTAM dialog.

File Definition Language (FDL) According to Digital Equipment documentation, this is a special-purpose language used to write specifications for data files. These specifications are written in text files called FDL files; they are then used by Open VMS record management services (RMS) utilities and library routines to create the actual data files.

file directory The OSI equivalent of a directory in a file system.

file filter function According to Apple documentation, a function supplied by your application to determine which files the user can open through a standard file dialog box.

file header According to Digital Equipment documentation, this is a block in the index file describing a file on a Files-11 disk structure. The file header c ontains information needed by the file system to find and use the file. Some of this information is displayed when the DCL command DIRECTORY is entered. There is at least one file header for every file on the disk.

file ID In the Apple environment this is an unchanging number assigned by the file manager to identify a file on a volume. When it establishes a file ID, the file manager records the filename and parent directory ID of the file.

file identifier According to Digital Equipment documentation, this is a 6-byte value used to uniquely identify a file on a Files-11 disk volume. The file number, file sequence number, and relative volume number are contained in the file identifier.

file organization In Digital Equipment networks, this is the particular file structure used as the physical arrangement of the data contained in a file on a mass storage medium. The Open VMS RMS file organizations are sequential, relative, and indexed.

Files-11 A Digital Equipment term used to refer to the name of the structure used by the RSX-111, IAS, and Open VMS operating systems.

Files-11 ancillary control process A Digital Equipment term referring to the interface process that is the file manager for the Files-11 on-disk structure.

Files-11 On-Disk Structure Level 1 According to Digital Equipment documentation, the original Files-11 structure used by IAS, RSX-11M, and RSX-11S for disk volumes. VAX systems support structure level 1 to ensure compatibility among systems. AXP systems do not support structure level 1.

Files-11 On-Disk Structure Level 2 Digital Equipment reference to the second-generation disk file structure supported by the operating system. The Files-11 data structure prepares a volume to receive and store data in a way recognized by the operating system.

file server A generic term used to refer to a computer whose primary task is to control the storage and retrieval of data from hard disks. Any number of other computers can be linked to the file server in order to use it to access data. This means that less storage space is required on the individual computer.

file system specification record A term used in Apple computer environments. It is a record that identifies a stored file or directory by volume reference number, parent directory ID, and name. The file system specification record is the file identification convention adopted by system software version 7.0.

File Transfer, Access, and Management An OSI application protocol standard which allows remote files to be transferred, accessed, and managed.

File Transfer Protocol (FTP) A term found in TCP/IP-based networks. It is a program that runs as a TCP application. It does not move a file from one place to another; rather, it copies a source file to a destination file. Consequently, two files exist unless one is deleted. FTP consists of a client and a server. The FTP client is used to invoke the FTP program. The FTP server is used to serve the request of the client. In normal implementations, FTP uses ports 20 and 21.

file translator A generic term referring to a utility program that converts a file from one computer format to another, such as from Macintosh to DOS. Apple file exchange is a file translator that is supplied with Macintosh system software.

filter A device or program that separates data, signals, or material in accordance with specified criteria.

flow control In SNA, the process of managing data-rate transfer between components within the network. This same concept refers to other networking environments.

focal point According to IBM documentation, in NetView the focal-point domain is the central host domain. It is the central control point for any management services element containing control of the network management data.

font size The size of the glyphs in a font in points, measured from the baseline of one line of text to the baseline of the next line of single-spaced text.

font style How a font (character or number) is represented.

foreground process In the X-Windows system, a process that has the terminal window's attention. This is in contrast to a background process.

foreground process A process currently interacting with the user.

formatted system services A term used in IBM's SNA, specifically VTAM. It is a portion of VTAM that provides certain system services as a result of receiving a field-formatted command, such as an INITIATE or TERMINATE command.

fragment A term used in TCP/IP network environments. One of the pieces that results when an IP router divides an IP datagram into smaller pieces for transmission across a network. Fragments use the same format as datagrams. Fields in the IP header declare whether a datagram is a fragment, and if so, the offset of the fragment in the original datagram. IP software at the receiving end must reassemble fragments into complete datagrams.

frame One definition generally agreed on is a packet as it is transmitted across a serial line. The term originates from character-oriented protocols. According to the meaning in OSI environments, it is a data structure pertaining to a particular area of data. It also consists of slots that can accept values of specific attributes.

frame relay A protocol defined by the CCITT and ANSI that identifies how data frames are switched in higher speeds than X.25, but in packet mode.

frame-relay frame handler (FRFH) A term reflecting a router function that is using the address field in a frame-relay frame.

frame-relay switching equipment (FRSE) A term used in frame-relay environments referring to a device capable of relaying frames to the next device in a frame-relay network en route to a frame-relay terminal equipment (FRTE) destination.

frame-relay switching equipment (FRSE) subport set In frame-relay technology this is the set of primary and, optionally, substitute subports within an FRSE that represent those used for a given segment set.

frame-relay switching equipment (FRSE) support An agreed-on set of NCP frame-relay functions that include the frame-relay frame handler (FRFH) functions, defined by ANSI Standards T1.617 and T1.618.

frame-relay terminal equipment (FRTE) In frame-relay-technology–based networks, a device capable of connecting to a frame-relay network. An FRTE adds a frame header when sending data to the frame-relay network and removes the frame header when receiving data from the frame-relay network.

frequency The rate of signal oscillation that is expressed in hertz.

frequency-division multiplexing (FDM) A technique that provides for division of frequency bandwidth into smaller subbands to provide each user exclusive use of a subband. A method of multiplexing data on a carrier channel based on frequency.

FTAM A term from the OSI networking protocols. It manipulates file transfer, access, and management.

full-duplex (FDX) Synonym for duplex.

full-screen mode Where the contents of an entire terminal screen can be displayed at once.

function management header (FMH) According to IBM documentation, one or more headers, optionally present in the leading request units (RUs) of an RU chain, that allow one LU to select a transaction program or device at the session partner. It also permits other control functions such as changing the destination or characteristics of the data during the session, and transmit between session partners status or user information about the destination.

gateway In internetworking terminology, the agreed-on definition is to perform protocol conversion between dissimilar protocols. This may be upper-layer protocol conversion only, or it may include lower-layer protocol conversion as well; this depends on the vendor offering and the implementation. Examples are TCP/IP-to-SNA gateways, DECnet-to-SNA gateways, and AppleTalk-to-TCP/IP gateways. In SNA, IBM uses the term in multiple ways. One explanation according to IBM documentation is the combination of machines and programs that provide address translation, name translation, and SSCP rerouting between independent SNA networks. In SNA a gateway consists of one "gateway" NCP and at lease one "gateway" VTAM.

gateway-capable host According to IBM documentation, a host node that has a defined NETID and SSCPNAME but does not perform gateway control functions.

gateway NCP According to IBM documentation, an NCP that performs address translation permitting cross-network session traffic. In this sense the gateway NCP connects two or more independent SNA networks.

gateway VTAM According to IBM documentation, an SSCP capable of cross-network session initiation, termination, takedown, and session outage notification.

generalized trace facility (GTF) In SNA, an optional program that records significant system events, such as supervisor calls and start I/O operations, for the purpose of problem determination.

generation In SNA, the process of assembling and link editing definition statements so that resources can be identified to all the necessary programs in a network. This is the origin of the term GEN.

generation definition According to IBM documentation, the definition statement of a resource used in generating a program.

generic unbind Synonym for session deactivation request.

global Affecting the entire file, the entire system, or the entire image, depending on context.

global symbol Agreed on as a symbol defined in a module of a program potentially available for reference by another module. The linker resolves (matches references with definitions) global symbols.

global symbol table In a library, an index of defined global symbols used to access the modules defining the global symbols. The linker also puts global symbol tables into an image.

glyph A distinct visual representation of a character that a display device, such as a monitor or printer, can display.

gold key According to Digital Equipment documentation, the upper left key on VT100 series terminal keypads. This enables alternate keypad functions.

Government Open Systems Interconnect OSI Profile (GOSIP) A federal information-processing standard that specifies a well-defined set of OSI protocols for government communications systems procurement. GOSIP was intended to eliminate the use of TCP/IP protocols on government internets, but clarifications have specified that government agencies can continue to use TCP/IP.

government OSI profiles Functional standards used by the government agencies in their procurement of open-system equipments and software.

graphical data display manager (GDDM) According to IBM documentation, this is used in the NetView performance monitor (NPM), in conjunction with the presentation graphics feature (PGF) to generate online graphs in the NPM graphic subsystem.

gray region Typically the term is used in X-Windows environments or Apple environments to define the desktop or the display area of all active devices, excluding the menu bar on the main screen and the rounded corners on the outermost screens. It is the area in which windows can be moved.

group Generically defined as a set of users in a system.

group control system (GCS) A VM component that provides multiprogramming and shared memory support.

guest A term used in the Apple environment meaning a user who is logged onto an AppleShare file server without a registered user name and password. A guest cannot own a directory or folder.

half-duplex (HD, HDX) Transmission in only one direction at a time.

half-duplex operation Data-link transmission in which data can be in both directions, one way at a time.

half-duplex transmission Data transmission in either direction, one direction at a time.

half-open connection A scenario in which one end connection is established but the other connection end is unreachable or has disposed of its connection information.

half-session According to IBM documentation, a session-layer component consisting of the data flow control and transmission control components. These constitute one end of a session.

hardcopy Generally a paper printout.

hardware address Also called a *hard address.* In Ethernet networks, this is the 48-bit address assigned to the Ethernet network interface card. In Token Ring this is the 12-digit hexadecimal address assigned to the network interface card.

hardware monitor In SNA, a component of NetView. It is called the *network problem determination application* (NPDA). It is used to identify network problems, such as hardware, software, and microcode.

header Control information that precedes user data in a frame or datagram that passes through networks. Specifically, this portion of a message contains control information.

hertz The number of cycles per second.

heartbeat Technically, this is known as *signal quality error* (SQE). It is a voltage in a receiver that can be sent to a controller (interface board) to inform the controller that collision detection is functional.

help balloon A term typically used in Apple environments referring to a rounded-type window containing explanatory information for the user.

help panel Also called a *help menu;* a display of information concerning a particular topic requested.

hierarchical routing From a TCP/IP perspective, this type routing is based on a hierarchical addressing scheme. Most TCP/IP routing is based on a two-level hierarchy in which an IP address is divided into a network portion until the datagram reaches a gateway that can deliver it directly. The concept of subnets introduces additional levels of hierarchical routing.

hierarchy In SNA, this type of networking is considered traditional. Traditional SNA networking requires VTAM for session establishment. The term used in light of networking can be contrasted with peer-to-peer networking, which does not require an intervening component for session establishment.

high-performance option (HPO) According to IBM documentation, an extension to VM/SP. The fundamental purpose of HPO is that it provides performance and operation enhancements for large system environments.

home directory Generally, this concept exists in all hosts. It is the place where a user originates their operations in any system. For example, in UNIX the `.profile` file contains the beginning place for a user. This `.profile` is customized for users, and the beginning place, or home base, can differ. In Digital Equipment's VMS environment the same is true but different terms are used, as with IBM's MVS, VM, and VSE operating systems as well.

hop In APPN, a portion of a route that has no intermediate nodes. It consists of only a single transmission group connecting adjacent nodes. One definition is the moving of a packet through a router.

hop count A measure of distance between two points in the Internet. Each hop count corresponds to one router separating a source from a destination (e.g., a hop count of 3 indicates that three routers separate a source from a destination). This term is generally used in TCP/IP networks as a measure of distance between two points in an internet. A hop count of *n* means that *n* routers separate the source and the destination.

host master key In SNA, deprecated term for mastery cryptography key.

host node According to Digital Equipment documentation on DECnet for Open VMS network, this is a node that provides services for another node. For the VAX Packetnet System Interface, this is a node that accesses a packet-switching data network by means of an X.25 multihost connector node. It is also referred to as the node that makes a device available to other nodes in a VMScluster configuration. A host node can be either a processor that adds the device to the mass storage control protocol server database or a hierarchical storage controller server. According to IBM documentation, it is defined as a processor.

host processor A processor that controls all or part of a user application network. Normally, the data communication access method resides on this host.

hpterm According to Hewlett-Packard documentation, a type of terminal window, sometimes called a *terminal emulator program,* that emulates HP2622 terminals, complete with softkeys. In the HP-UX environment, the hpterm window is the default window for the X environment.

icon A small, graphic representation of an object on the root window. Icons are found in Apple hosts as well as the X-Windows system.

icon family In the Apple computer family of products, the set of icons that represent an object, such as an application or document, on the desktop.

idle state A state in which the Macintosh portable computer slows from its normal 16-MHz clock speed to a 1-MHz clock speed. The power manager puts the Macintosh portable in the idle state when the system has been inactive for 15 s.

image Procedures and data bound together by the linker to form an executable program. This executable program is executed by the process. There are three types of images: executable, sharable, and system.

image name The name of the file in which an image is stored.

image mode The default screen mode using multiple image planes for a single screen. The number of image planes determines the variety of colors that are available to the screen.

image planes The primary display planes on a device that supports two sets of planes. The other set of display planes is known as *overlay planes.* These two sets of planes are treated as two separate screens in stacked mode and one screen in combined mode.

image privileges The privileges assigned to an image when it is installed.

inactive This term has a variety of meanings depending on the context or environment. It is, however, generally agreed to mean something that is not operational or pertaining to a node or device not connected or not available for connection to another node or device. In IBM's AIX operating system it pertains to a window that does not have an input focus. In SNA, particularly VTAM, the state of a resource, a major or minor node not activated or for which the VARY INACT command has been issued.

incoming-call packet A call supervision packet transmitted by data circuit-terminating equipment (DCE) to inform called data terminal equipment (DTE) that another DTE has requested a call.

independent logical unit (ILU) In SNA, a type of LU that does not require VTAM for session establishment after the initial download of parameters.

index A structure that permits retrieval of records in an indexed file by key value.

indexed file organization A Digital Equipment–type file organization in which a file contains records and a primary key index used to process the records sequentially or randomly by index.

indexed sequential file According to Digital Equipment documentation, a record file in which each record has one or more data keys embedded in it. Records in the file are individually accessible by specifying a key associated with the record.

index file According to Digital Equipment documentation, the file on Files-11 volume that contains the access information for all files on the volume and enables the operating system to identify and access the volume.

index file bitmap According to Digital Equipment documentation, a table in the index file of a Files-11 volume that indicates which file headers are in use.

index path According to NetWare documentation, a logical ordering of records in a Btrieve file based on the values of an index. An index path for each index in a file exists. A file may have up to 24 separate index paths.

indirect activation According to IBM documentation, in VTAM, the activation of a lower-level resource of the resource hierarchy as a result of SCOPE or ISTATUS specifications related to an activation command naming a higher-level resource.

indirect deactivation According to IBM documentation, in VTAM, the deactivation of a lower-level resource of the resource hierarchy as a result of a deactivation command naming a higher-level resource.

information (I) format A format used for information transfer.

information (I) frame A frame in I format used for numbered information transfer.

Information Management System/Virtual Storage (IMS/VS) This is a software subsystem offering by IBM. It is a database/data-communication (DB/DC) system that can manage complex databases and networks.

information management A feature of the information system that provides interactive systems management applications for problem, change, and configuration management.

inhibited According to IBM, a logical unit (LU) that has indicated to its system services control point (SSCP) that it is temporarily not ready to establish LU-LU sessions. An INITIATE request for a session with an inhibited LU will be rejected by the SSCP. The LU can separately indicate whether this applies to its ability to act as a primary logical unit (PLU) or a secondary logical unit (SLU).

initial program load (IPL) An IBM term referring to the initialization procedure that causes an operating system to commence operation. The process by which a configuration image is loaded into storage at the beginning of a workday or after a system malfunction. The process of loading system programs and preparing a system to run jobs.

INITIATE In SNA, a network services request sent from a logical unit (LU) to a system services control point (SSCP) requesting that an LU-LU session be established.

INITIATING LU (ILU) In SNA, the LU that first requests that a session be set up. The ILU may be one of the LUs that will participate in the session, or it may be a third-party LU. If it is one of the session participants, the ILU is also called the *origin LU* (OLU).

initiator In OSI, a file service user which requests an FTAM establishment.

inoperative The condition of a resource that has been active, but is no longer. The resource may have failed, received an INOP request, or been suspended while a reactivate command is being processed.

input/output channel In a data-processing system, a functional unit that handles transfer of data between internal and peripheral equipment. In a computing system, a functional unit, controlled by a processor, that handles transfer of data between processor storage and local peripheral devices. In IBM terminology, this term refers to a specific type of path, either parallel or serial.

installation exit The means (or way) by which an IBM software product may be modified by a customer's system programmers to change or extend the functions of the IBM software product. Such modifications consist of exit routines written to replace one or more existing modules of an IBM software product, or to add one or more modules or subroutines to an IBM software product, for the purpose of modifying or extending the functions of the IBM software product.

installation exit routine A routine written by a user to take control at an installation exit of an IBM software product.

installationwide exit Synonym for installation exit.

integrated communication adapter (ICA) A communication adapter that is an integral part of the host processor.

Integrated Services Digital Network (ISDN) A set of standards being developed within ANSI, ISO, and CCITT for the delivery of various services over digital networks.

integrity control According to Novell documentation, the method used to ensure the completeness of files. Specifically, Btrieve uses preimaging and NetWare's transaction tracking system to guarantee integrity.

intensive-mode recording (IMR) An NCP function that forces recording of temporary errors for a specified resource.

interactive problem control system (IPCS) According to IBM documentation, a VM component that permits online problem management, interactive problem diagnosis, online debugging for disk-resident CP abend dumps, problem tracking, and problem reporting.

Interactive System Productivity Facility (ISPF) An IBM-licensed program that serves as a full-screen editor and dialog manager. Used for writing application programs, it provides a means of generating standard screen panels and interactive dialogs between the application programmer and terminal user.

interapplication communication (IAC) In Apple terminology, a collection of features, provided by the edition manager, Apple event manager, event manager, and PPC toolbox, that help applications work together.

interchange node A new type of node supported by VTAM beginning in version 4 release 1, that acts as both an APPN network node and a subarea type 5 node to transform APPN protocols to subarea protocols and vice versa.

interconnected networks According to IBM, SNA networks connected by gateway NCPs.

interface A shared boundary between two functional units, defined by functional characteristics, signal characteristics, or other characteristics, as appropriate. The concept includes the specification of the connection of two devices having different functions. Hardware, software, or both, that links systems, programs, or devices.

intermediate node A node that is at the end of more than one branch.

intermediate routing node (IRN) A node containing intermediate routing functions.

intermediate session routing (ISR) A term used in APPN that is a type of routing function within an APPN network node that provides session-level flow control and outage reporting for all sessions that pass through the node but whose endpoints are elsewhere.

intermediate SSCP In SNA, this is an SSCP along a session initiation path that owns neither of the LUs involved in a cross-network LU-LU session.

International Organization for Standardization (ISO) An organization of national standards bodies from various countries established to promote development of standards to facilitate international exchange of goods and services and develop cooperation in intellectual, scientific, technological, and economic activity.

Internet According to different documents describing the Internet, a collection of networks, routers, gateways, and other networking devices that use the TCP/IP protocol suite and function as a single, cooperative virtual network. The Internet provides universal connectivity and three levels of network services: unreliable, connectionless packet delivery; reliable, full-duplex stream delivery; and application-level services such as electronic mail (email) that build on the first two. The Internet reaches many universities, government research labs, and military installations and over a dozen countries.

Internet address According to Apple documentation, an AppleTalk address that includes the socket number, node ID, and network number. According to TCP/IP documentation, the 32-bit address assigned to the host. It is a software address that is locally managed on local internets, but on the "big I" Internet is dictated to the user (entity desiring access to the Internet) on the central Internet.

Internet Architecture Board (IAB) A group related to TCP/IP protocol. Specifically, it is a group of people who set policy and review standards for TCP/IP and the Internet. The IAB was reorganized in 1989; technically oriented individuals moved to research and engineering subgroups. See **Internet Engineering Task Force** and **Internet Research Task Force.**

Internet Control Message Protocol (ICMP) Specific to the TCP/IP protocol suite. It is an integral part of the Internet Protocol. It handles error and control messages. Routers and hosts use ICMP to send reports of problems about datagrams back to the original source that sent the datagram. ICMP also includes an echo request/reply used to test whether a destination is reachable and responding.

Internet Engineering Task Force (IETF) A group of people concerned with short- and medium-term problems with TCP/IP and the connected Internet. IETF is divided into six areas which are further divided into working groups.

Internet Packet eXchange (IPX) A Novell protocol that operates at layer 3 of the OSI model. It is used in the NetWare protocols and is similar to IP in TCP/IP.

Internet Protocol (IP) A protocol used to route data from their source to its destination; the TCP/IP standard protocol that defines the IP datagram as the unit of information passed across an Internet and provides the basis for connectionless, best-effort packet delivery service. IP includes the ICMP control and error message protocol as an integral part. The entire protocol suite is often referred to as TCP/IP because TCP and IP are the two fundamental protocols.

Internet Research Steering Group (IRSG) A committee consisting of the IRTF research group chairpersons plus the IRTF chairperson, who direct and coordinate research related to TCP/IP and the connected Internet.

Internet Research Task Force (IRTF) A group of people working on research problems related to TCP/IP and the connected Internet.

interpersonal message According to OSI-related documents, this is defined as a message type used for human-to-human communication in MHS.

interpersonal messaging system According to multiple explanations in the OSI community, an MHS system supporting the communication of interpersonal messages.

InterPoll According to Apple documentation, this is software from Apple that helps administrators monitor the network and diagnose the source of problems that arise.

interuser communication vehicle (IUCV) According to IBM documentation, a VM facility for passing data between virtual machines and VM components.

interpret table In IBM's VTAM, an installation-defined correlation list that translates an argument into a string of eight characters. This table can translate logon data into the name of an application program for which the logon is intended.

IP address The 32-bit dotted decimal address assigned to hosts that want to participate in a local TCP/IP internet or the central Internet. IP addresses are software addresses. Actually, an IP address consists of a network portion and a host portion. The partition makes routing efficient.

IP datagram A term used with TCP/IP networks; a basic unit of information passed across a TCP/IP internet. An IP datagram is to an internet what a hardware packet is to physical network. It contains a source address and a destination address along with data.

IS-IS routing Routing between ISs within a routing domain.

ISODE A term used in the ISO development environment. A set of public-domain software subroutines that provide an interface between the GOSIP-specified session layer (ISO) and the DoD-specified transport layer (TCP/IP). Allows the development of applications that will execute over both OSI and TCP/IP protocol stacks as a migration path from TCP/IP networks to GOSIP networks.

ISTATUS According to IBM documentation, in VTAM and NCP, a definition specification method for indication the initial status of resources.

job A means by which an accounting unit is assigned to a process and its subprocesses, if any, and all subprocesses that they create. Jobs are classified as batch and interactive. For example, the job controller creates an interactive job to handle a user's requests when the user logs onto the system, and it creates a batch job when the symbiont manager passes a command input file to it.

job controller The system process that establishes a job's process context, starts a process running the LOGIN image for the job, maintains the accounting record for the job, manages symbionts, and terminates a process and its subprocesses.

Job Control Language A language used in IBM's MVS operating system environment used to identify a job to an operating system and to describe the job's requirements.

job information block A data structure associated with a job that contains the quotas pooled by all processes in the job.

katakana A character set of symbols used in one of the two common Japanese phonetic alphabets, used primarily to write foreign words phonetically.

keyboard binding In an X-Windows environment, an association of a special key press with a window manager function. For example, pressing the special keys SHIFT and ESC displays the system menu of the active window.

keyboard resources According to Apple documentation, a category of files that are stored in a resource file by the resource manager and used by the Macintosh script management system, including the international utilities package.

keyboard script The script for keyboard input. It determines the character input method and the keyboard mapping, that is, what character codes are produced when a sequence of keys is pressed.

keyword (1) According to Apple documentation, a four-character code used to uniquely identify the descriptor record for either an attribute or a parameter in an Apple event. In Apple event manager functions, constants are typically used to represent the four-character codes. (2) In programming languages, a lexical unit that, in certain contexts, characterizes some language construct. In some contexts, IF characterizes an `if-statement.` A keyword normally has the form of an identifier. One of the predefined words of an artificial language. A significant and informative word in a title or document that describes the content of that document. A name or symbol that identifies a parameter.

keyword operand A term used in the IBM environment, particularly with JCL. It is an operand that consists of a keyword followed by one or more values (such as `DSNAME =` HELLO).

keyword parameter A parameter that consists of a keyword followed by one or more values.

LAP manager According to Apple documentation, a set of operating system utilities that provide a standard interface between the AppleTalk protocols and the various link access protocols, such as LocalTalk (LLAP), EtherTalk (ELAP), and TokenTalk (TLAP).

least-weight route According to IBM's documentation, in APPN, the one route calculated by topology and routing services (TRS) to have the lowest total weight after TRS compares the node characteristics and TG characteristics of each intermediate node and intermediate TG of each possible route for the class of service requested, and computes the total combined weight for nodes and TGs in each route. After a least-weight route is calculated between two given nodes, the result may be stored to prevent repetition of this calculation in future route selections.

LEN connection A link over which LEN protocols are used.

LEN node A term used in APPN. According to IBM documentation, a node that supports independent LU protocols but does not support CP-CP sessions. It may be a peripheral node attached to a boundary node in a subarea network, an end node attached to an APPN network node in an APPN network, or a peer-connected node directly attached to another LEN node or APPN end node.

level 1 router According to Digital Equipment documentation, a DECnet for Open VMS node that can send and receive packets, and route packets from one node to another node within a single area.

level 2 router According to Digital Equipment documentation, a DECnet for Open VMS node that can send and receive packets, and route packets from one node to another within its own area and between areas.

lexical function According to Digital Equipment documentation, a command language construct that the Digital Command Language command interpreter evaluates and substitutes before it parses a command string.

Librarian According to Digital Equipment, a program that allows the user to create, update, modify, list, and maintain object library, help library, text library, and assembler macrolibrary files.

limited resource According to IBM documentation, a connection facility that causes a session traversing it to be terminated if no session activity is detected for a specified period of time.

limited-resource session According to IBM documentation, a session that traverses a limited resource link. This session is terminated if no session activity is detected for a specified period of time.

line The portion of a data circuit external to data circuit-terminating equip ment (DCE) that connects the DCE to a data switching exchange (DSE), that connects a DCE to one or more other DCEs, or that connects a DSE to another DSE.

line control Synonym for Data-Link Control Protocol.

line-control discipline Synonym for Link Protocol.

line discipline Synonym for Link Protocol.

line group One or more telecommunication lines of the same type that can be activated and deactivated as a unit.

line speed The number of binary digits that can be sent over a telecommunication line in one second, expressed in bits per second (bits/s).

line switching Synonym for circuit switching.

link The combination of the link connection (the transmission medium) and two link stations, one at each end of the link connection.

Link Access Protocol (1) According to Digital Equipment documentation, a set of procedures used for link control on a packet-switching data network. X.25 defines two sets of procedures: *LAP*—the DTE/DCE interface is defined as operating in two-way simultaneous asynchronous response mode with the DTE and DCE containing a primary and secondary function; and *LAPB*—the DTE/DCE interface is defined as operating in 2-way asynchronous balanced mode. (2) According to Apple documentation, an AppleTalk protocol that controls the access of a node to the network hardware. A LAP makes it possible for many nodes to share the same communications hardware.

link-attached Pertaining to devices that are connected to a controlling unit by a data link.

link connection The physical equipment providing two-way communication between one link station and one or more other link stations; for example, a telecommunication line and DCE equipment.

link connection segment A part of the configuration that is located between two resources listed consecutively in the service point command service (SPCS) query link configuration request list.

link level A reference to the physical connection between two nodes and/or the protocols used to govern that connection.

link problem determination aid (LPDA) According to IBM documentation, a series of procedures used to test the status of and to control DCEs, the communication line, and the remote device interface. These procedures, or a subset of them, are implemented by host programs (such as the NetView program and VTAM), communication controller programs (such as NCP), and IBM LPDA DCEs.

Link Protocol Rules for sending and receiving data over a medium.

link services layer (LSL) Routes packets between LAN boards with their MLIDs and protocol stacks. The LSL maintains LAN board, protocol stack, and packet buffer information.

link station According to IBM documentation, the hardware and software components within a node representing a connection to an adjacent node over a specific link. In VTAM, a named resource within an APPN or a subarea node that represents the connection to another APPN or subarea node that is attached by an APPN or a subarea link. In the resource hierarchy in a subarea network, the link station is subordinate to the subarea link.

Little Endian A storage format or transmission of binary data in which the least significant byte comes first.

local Pertaining to a device accessed directly without use of a telecommunication line.

local access The ability to execute a program on the computer to which you are attached.

local address According to IBM documentation, in SNA, an address used in a peripheral node in place of a network address and transformed to or from a network address by the boundary function in a subarea node.

local area network (LAN) A collection of computers and other related devices connected together on the premises within a limited geographic area.

local area transport According to Digital Equipment documentation, this is a communications protocol that the operating system uses within a local area network to communicate with terminal servers.

local area VAXcluster system According to Digital Equipment documentation, a type of VAXcluster configuration in which cluster communication is carried out over the Ethernet by software that emulates certain computer interconnect (CI) port functions. A VAXcluster node can be a VAX or a microVAX processor; hierarchical storage controllers (HSCs) are not used.

local client A term used in an X-Windows environment. It refers to a program running on your local computer, the same system that is running your X server.

local directory database According to IBM documentation, a set of LUs in a network known at a particular node. The resources included are all those in the node's domain as well as any cache entries.

local management interface (LMI) A set of operational procedures and messages as well as DLCI 1023 which are defined in *Frame-Relay Specification with Extensions*, a document based on proposed T1S1 standards, which are copyrighted by Digital Equipment Corporation, Northern Telecom, Inc., and StrataCom, Inc. In this context, the term *local management interface* (LMI) is a deprecated term for link integrity verification tests (LIVTs). Current meaning: any frame-relay management interface procedures, such as DLCI 1023 or DLCI 0.

local non-SNA major node According to IBM documentation, in VTAM, a major node whose minor nodes are channel-attached non-SNA terminals.

local SNA major node According to IBM documentation, in VTAM, a major node whose minor nodes are channel-attached peripheral nodes.

local symbol According to Digital Equipment documentation, a symbol meaningful only to the module that defines it. Symbols not identified to a language processor as global symbols are considered to be local symbols. A language processor resolves local symbols. They are not known to the linker and cannot be made available to another object module. They can, however, be passed through the linker to the symbolic debugger.

LocalTalk According to Apple documentation, a type of AppleTalk network that is inexpensive and easy to set up. LocalTalk is commonly used to connect small to medium-sized workgroups.

local topology database According to IBM documentation, a database in an APPN NN or LEN node containing an entry for each transmission group (TG) having at least one end node for a given endpoint. In an APPN END node, the database has one entry for each TG connecting to the node. In a network node, the database has an entry for each TG connecting the network node to an end node. Each entry describes the current characteristics of the TG that it represents. A network node has both a local and a network topology database while an end node has only a local topology database.

locate mode According to Digital Equipment documentation, an Open VMS RMS record access technique in which a program accesses records in a Open VMS RMS I/O buffer area to reduce overhead.

location name An identifier for the network location of the computer on which a port resides. The PPC toolbox provides the location name. It contains an object string, a type string, and a zone. An application can specify an alias for its location name by modifying its type string.

log off To request that a session be terminated.

log on In SNA products, to initiate a session between an application program and a logical unit (LU).

logical channel In packet-mode operation, a sending channel and a receiving channel that together are used to send and receive data over a data link at the same time.

logical channel identifier A bit string in the header of a packet that associates the packet with a specific switched virtual circuit or permanent virtual circuit.

logical link control (LLC) According to OSI documentation, a sublayer in the data-link layer of the OSI model. The LLC provides the basis for an unacknowledged connectionless service or connection-oriented service on the local area network.

Logical link control (LLC) protocol In a local area network, the protocol that governs the exchange of transmission frames between data stations independently of how the transmission medium is shared.

logical name According to Digital Equipment documentation, a user-specified name for any portion or all of a file specification. For example, the logical name INPUT can be assigned to a terminal device from which a program reads data entered by a user. Logical name assignments are maintained in logical name tables for each process, each group, and the system. Logical names can be assigned translation attributes, such as terminal and concealed.

logical name table According to Digital Equipment documentation, a table that contains a set of logical names and their equivalent names for a particular process, a particular group, or the system.

logical record A group of related fields treated as a unit.

logical unit (LU) An addressable endpoint.

logical unit 6.2 Those protocols and that type of LU which supports advanced program-to-program communication (APPC).

login directory The default directory a user is assigned to on logon into a system.

logoff According to IBM documentation, in VTAM, an unformatted session-termination request. In general, termination of interaction with a system; actually entering a command of some sort to close the connection.

logon According to IBM documentation, in VTAM, an unformatted session-initiation request for a session between two logical units. In general, it means signing on or getting to the point where work can be done.

logon manager A VTAM application program that provides logon services for the transaction-processing facility (TPF).

logon mode According to IBM documentation, in VTAM, a subset of session parameters specified in a logon mode table for communication with a logical unit.

logon mode table According to IBM documentation, in VTAM, a set of entries for one or more logon modes. Each logon mode is identified by a logon mode name.

low-entry networking (LEN) According to IBM documentation, a capability in nodes allowing them to directly attach to one another using peer-to-peer protocols and to support multiple and parallel sessions between logical units. However, LEN does not provide all the capabilities of APPN; for example, it does not provide CP-CP session support.

Low-Entry Networking (LEN) end node According to IBM documentation, an end node that provides all SNA end-user services, can attach directly to other nodes using peer protocols, and derives network services implicitly from an adjacent network node when attached to an APPN network without a session between its local CP and another CP.

low-entry networking (LEN) node According to IBM documentation, a node that supports independent LU protocols but does not support CP-CP sessions.

LU group According to IBM documentation, in the NetView Performance Monitor (NPM), a file containing a list of related or unrelated logical units. The LU group is used to help simplify data collection and analysis.

LU-LU session A logical connection between two Logical Units in an SNA network that provides communication between two end users.

LU-mode pair According to IBM documentation, in the VTAM implementation of the LU6.2 architecture, the coupling of an LU name entry and a mode name entry. This coupling allows a pool of sessions with the same characteristics to be established.

LU type According to IBM documentation, the classification of an LU in terms of SNA protocols and options it supports for a given session.

LU type 6.2 (LU6.2) According to IBM documentation, a type of logical unit that supports general communication between programs in a distributed processing environment. LU6.2 is characterized by a peer relationship between transaction programs and efficient utilization of a session for multiple transactions.

LU6.2 session A logical connection utilizing LU6.2 protocols.

macroinstruction According to IBM documentation, an instruction in a source language that is to be replaced by a defined sequence of instructions in the same source language and that may also specify values for parameters in the replaced instructions.

main screen The screen on which a menu bar appears.

maintain system history program (MSHP) According to IBM documentation, a program used for automating and controlling various installation, tailoring, and service activities for a VSE system.

maintenance and operator subsystem (MOSS) According to IBM documentation, a subsystem of an IBM communication controller, such as the 3725 or the 3720, that contains a processor and operates independently of the rest of the controller. It loads and supervises the controller, runs problem determination procedures, and assists in maintaining both hardware and software.

major node According to IBM documentation, in VTAM, a set of resources that can be activated and deactivated as a group.

management information base A collection of managed objects. A term used with the concept of SNMP-based network management.

management information tree A tree structure of the management information base.

management services (MS) According to IBM documentation, one of the types of network services in control points (CPs) and physical units (PUs). Management services are provided to assist in the management of SNA networks, such as problem management, performance and accounting management, configuration management, and change management.

manufacturing messaging service (MMS) According to OSI documentation a messaging service between programmable devices.

MASSBUS adapter According to Digital Equipment documentation, an interface device between the backplane interconnect and the MASSBUS device.

Mass Storage Control Protocol According to Digital Equipment documentation, the software protocol used to communicate I/O commands between a VAX processor and DSA-compliant devices on the system.

master file directory (MFD) According to Digital Equipment documentation, the file directory on a disk volume that contains the name of all user file directories on a disk, including its own.

matte A term used in window-based environments. It is the border located inside the window between the client area and the frame and used to create a three-dimensional effect for the frame and window.

maximum transfer unit (MTU) The largest amount of data that can be transferred across a given physical network. For local area networks such as the Ethernet, the MTU is determined by the network hardware. For long-haul networks that use aerial lines to interconnect packet switches, the MTU is determined by software.

media access control According to OSI nomenclature, a sublayer in the data-link layer which controls access to the physical medium of a network.

medium A physical carrier of electrons or photons. The medium may be hard, as in a type of cable; or soft, in the sense of microwave, for example.

Medium Access Control (MAC) Protocol A protocol in the lower part of the second layer in the OSI model that governs access to the transmission medium to enable the exchange of data between nodes.

Medium Access Control (MAC) sublayer The MAC sublayer supports topology-dependent functions and uses services of the physical layer to provide services to the LLC sublayer.

menu A list of selections from which to make a choice.

MERGE disk According to IBM documentation, a virtual disk in the VM operating system that contains program temporary fixes (PTFs) after the VMFMERGE EXEC is invoked.

message Generically, a reference to meaningful data passed from one end user to another. The end user may be a human or a program. Also according to OSI documentation, a structured set of data sent from a user agent to one or more recipient user agents.

message block According to Apple documentation, a byte stream that an open application uses. It is used to send data to and receive data from another open application. The PPC toolbox delivers message blocks to an application in the same sequence in which they were sent.

message-handling service (MHS) The service provided by the CCITT X.400 series of standards, consisting of a user agent to allow users to create and read electronic mail; a message transfer agent to provide addressing, sending, and receiving services; and a reliable transfer agent to provide routing and delivery services.

message store A term used in a TCP/IP environment referring to an entity acting as an intermediary between a user agent and its local message transfer agent.

message transfer agent In TCP/IP a subpart of the email component known as *Simple Mail Transfer Protocol* (SMTP). It is an object in the message transfer system. MTAs use a store-and-forward method to relay messages from originator to recipient. They interact with user agents when a message is submitted, and on delivery.

message unit According to IBM documentation, in SNA, the unit of data processed by any layer; for example, a basic information unit (BIU), a path information unit (PIU), or a request-response unit (RU).

migration data host According to IBM documentation, a VTAM node support that acts as both an APPN end node and an SNA subarea node.

MILNET Originally this network was part of the ARPAnet. In 1984 it was segmented for military installation usage.

minimize To turn a window into an icon.

minor node According to IBM documentation, in VTAM, a uniquely defined resource within a major node.

mixed interconnect VMScluster system According to Digital Equipment documentation, any VMScluster system that utilizes more than one interconnect for SCA traffic. Mixed interconnect VMScluster systems provide maximum flexibility in combining CPUs, storage, and workstations into highly available configurations.

mode name In SNA, a name used by the initiator of a session to designate the characteristics desired for that session.

modem (modulator/demodulator) A device that converts digital to analog signals and vice versa for the purpose of using computer devices in remote locations.

monitor Generally considered the same as display. It can also mean to watch a task, program execution, or the like.

monitor console routine According to Digital Equipment documentation, the command interpreter in an RSX-11 system; also an optional command interpreter in the operating system.

mounting a volume According to Digital Equipment documentation, the logical association of a volume with the physical unit on which it is loaded. Loading or placing a magnetic tape or disk pack on a drive and placing the drive on line.

mount verification According to Digital Equipment documentation, a feature that suspends I/O to and from volumes while they are changing status. Mount verification also ensures that, following a suspension in disk I/O, the volume being accessed is the same as was previously mounted.

move mode According to Digital Equipment documentation, an Open VMS RMS record I/O access technique in which a program accesses records in its own working storage area.

multicast A technique that allows copies of a single packet to be passed to a selected subset of all possible destinations. Some hardware supports multicast by allowing a network interface to belong to one or more multicast groups. Broadcast is a special form of multicast in which the subset of machines to receive a copy of a packet consists of the entire set. IP supports an Internet multicast facility.

multicast address According to Apple documentation, an Ethernet address for which the node accepts packets just as it does for its permanently assigned Ethernet hardware address. The low-order bit of the high-order byte is set to 1. Each node can have any number of multicast addresses, and any number of nodes can have the same multicast address. The purpose of a multicast address is to allow a group of Ethernet nodes to receive the same transmission simultaneously, in a fashion similar to the AppleTalk broadcast service.

multicasting A directory service agent (DSA) uses this mode to chain a request to many other DSAs.

MultiFinder According to Apple documentation, prior to version 7.0 system software, this was a multitasking operating system for Macintosh computers that enabled several applications to be open at the same time. In addition, processes (such as print spooling) could operate in the background to enable users to perform one task while the computer performed another.

multihomed host A TCP/IP host connected to two or more physical networks; thus they have more than one address. They can serve as router-type devices.

multilink transmission group According to IBM documentation, a transmission group containing two or more links.

multimode The transmission of multiple modes of light.

multipath channel (MPC) According to IBM documentation, a channel protocol that uses multiple unidirectional subchannels for VTAM-to-VTAM bidirectional communication.

multiple-domain network According to IBM documentation, a network with more than one system services control point (SSCP) in traditional subarea SNA. In APPN it is an APPN network with more than one network node.

multiple-link interface driver (MLID) According to Novell documentation, this type of driver accepts multiple protocol packets. When an MLID device driver receives a packet, the MLID does not interpret the packet; it copies identification information and passes the packet to the link support layer. MLIDs are supplied by either Novell, the network board manufacturer, or a third-party supplier.

Multiple Virtual Storage/eXtended Architecture (MVS/XA) An IBM operating system.

multipoint line A telecommunication line or circuit that connects two or more stations.

mutex A term used in Digital Equipment network environments. According to Digital Equipment documentation, a semaphore is used to control exclusive access to a region of code that can share a data structure or other resource. The mutex semaphore ensures that only one process at a time has write-access to the region of code.

name-binding protocol According to Apple documentation, this is the AppleTalk transport-level protocol that translates a character string into a network address and maintains a table that contains the Internet address and name of each entity in the node that is visible to other entities on the Internet (i.e., each entity that has registered a name with NBP).

name resolution The process of locating an entry by sequentially matching each relative distinguished name in a purported name to a vertex of the directory information tree.

name block According to Digital Equipment documentation, an Open VMS RMS user data structure that contains supplementary information used in parsing file specifications.

name translation According to IBM documentation, an SNA network interconnection. It includes the conversion of logical unit names, logon mode table names, and class-of-service names used in one network to equivalent names for use in another network.

naming context A term used in OSI networks that refers to a substructure of the directory information tree. It starts at a vertex and extends downward to a leaf and/or nonleaf structure.

naming context tree According to popular OSI documentation, a tree structure in which each node represents a naming context.

National Science Foundation (NSF) A government agency that has enabled scientists to connect to networks making up the Internet. The *National Science Foundation NETwork* (NSFNET) reference to a network that spans the United States.

NCP major node According to IBM documentation, this refers to VTAM, where a set of minor nodes represent resources, such as lines and peripheral nodes, controlled by IBM's network control program.

NCP/EP Definition Facility (NDF) According to IBM documentation, a program that is part of system support programs (SSPs). It is used to generate a partitioned emulation program (PEP) load module or a load module for a network control program (NCP) or for an emulation program (EP).

negative response (NR) According to IBM documentation, a term used in SNA referring to a response indicating that a request did not arrive successfully or was not processed successfully by the receiver.

negotiable BIND According to IBM documentation, a term used in SNA that refers to the capability allowing two half-sessions to negotiate the parameters of a session when the session is being activated.

NetBIOS NetBIOS (Net Basic Input/Output System) is the standard interface to networks that is used by IBM PCs and compatibles. With a TCP/IP network, NetBIOS refers to a set of guidelines that describes how to map NetBIOS operations into equivalent TCP/IP operations.

NetView Performance Monitor (NPM) According to IBM, a program that collects, monitors, analyzes, and displays data relevant to the performance of VTAM.

NetView According to IBM, a program used to monitor and manage a network and diagnose network problems.

NetWare-loadable module According to Novell documentation, a program that is part of a file server memory with NetWare. An NLM can be loaded or unloaded while the file server is running, can become part of the operating system, and can access NetWare directly.

network A collection of computers and related devices connected together in such a way that collectively they can be more productive than standalone equipment.

network address In general, each participating entity on a network has an address so that it can be identified when exchanging data. According to IBM documentation, in a subarea network, an address consists of subarea and element fields that identify a link, link station, PU, LU, or SSCP.

network address translation According to IBM documentation, in an SNA network interconnection, the conversion of the network address assigned to an LU in one network into an address in an another network.

network architecture The logical and physical structure of a computer network.

Network Communications Control Facility (NCCF) A part of IBM's NetView. It is the command that starts the NetView command facility. It is a command line in NetView in which various commands can be offered.

network connect block According to Digital Equipment documentation, a user-generated data structure used in a nontransparent task to identify a remote task and optionally send user data in calls to request, accept, or reject a logical link connection. For the VAX Packetnet System Interface, a block that contains the information necessary to set up an X.25 virtual circuit or to accept or reject a request to set up an X.25 virtual circuit.

network control The meaning in SNA, according to IBM documentation, is a request or response unit (RU) category used for request and responses exchanged between PUs. The purpose of this is to activate and deactivate explicit and virtual routes. The term is used for sending load modules to adjust peripheral nodes.

network control program (NCP) (1) According to Digital Equipment documentation, an interactive utility program that allows control and monitoring of a network. (2) According to IBM documentation, a program that controls the operation of a communication controller.

Network File System (NFS) According to Sun Microsystems, Inc., this is a protocol developed by Sun that uses IP to allow a set of cooperating computers to access each other's file systems as if they were local. NFS hides differences between local and remote files by placing them in the same name space. Originally designed for UNIX systems, it is now implemented on many other systems, including personal computers like the IBM PC and Apple computers.

Network Logical Data Manager (NLDM) According to IBM documentation, a subset of NetView. NLDM is a command that starts the NetView sessions' monitor. NLDM also identifies various panels and functions as part of the session monitor.

network layer According to OSI documentation, this is layer 3, which is responsible for data transfer across the network. It functions independent of the network media and the topology.

network management vector transport (NMVT) According to IBM documentation, a protocol used for management services in an SNA network.

network name According to IBM documentation, in SNA, the symbolic identifier by which end users refer to a network-accessible unit, a link, or a link station within a given network. Another definition by IBM is used in reference to APPN networks; network names are also used for routing purposes. In a multi-domain network, this is the name of the APPL statement defining a VTAM application program. This network name must be unique across domains.

network node server A term defined by IBM documentation used in reference to an APPN network node that provides network services for its local LUs and client end nodes.

network number According to Apple documentation, it is a 16-bit number that provides a unique identifier for a network in an AppleTalk internet.

network operator A person who performs a variety of functions to a network, some of which are control functions.

Network Problem Determination Application (NPDA) According to IBM documentation, this is a part of NetView. It is also a command that starts the NetView hardware monitor. NPDA identifies various panels and functions as part of the hardware monitor.

Network Protocol Data Unit (NPDU) In OSI terminology, this term refers to a packet; a logical block of control symbols and data transmitted by the network-layer protocol.

network-qualified name A name that uniquely identifies a specific resource within a specific network. It consists of a network identifier and a resource name, each of which is a 1- to 8-byte symbol string.

network range According to Apple documentation, a unique range of contiguous network numbers used to identify each Ethernet and Token-Ring network on an AppleTalk internet.

network routing facility (NRF) According to IBM documentation, an IBM program that resides in an NCP. NRF provides a path for routing messages between terminals and routes messages over this path without going through the host processor.

network services According to IBM documentation, those services within network-accessible units that control network operation.

network services header According to IBM documentation, in traditional SNA this is a 3-byte field in a function management data (FMD) request or response unit (RU) that flows in an SSCP-LU, SSCP-PU, or SSCP-SSCP session. This is used primarily to identify the network services category of the request unit (RU).

Network Services Protocol According to Digital Equipment documentation, a formal set of conventions used in a DECnet for Open VMS network to perform network management and to exchange messages over logical links.

network terminal option (NTO) According to IBM documentation, this is a program, used in conjunction with NCP, that allows some non-SNA devices to participate in sessions with SNA application programs in the host processor.

network topology database In an APPN network, according to IBM documentation, the representation of the current connectivity between the network nodes within an APPN network. It includes entries for all network nodes and the transmission groups interconnecting them and entries for all virtual routing nodes to which network nodes are attached.

Network Voice Protocol (NVP) A TCP/IP protocol for handling voice information.

no response According to IBM documentation, in SNA, a protocol requested in the for-of-response-requested field of the request header. It directs the receiver of the request not to return any response, regardless of whether the request is received and processed successfully.

node (1) Generally, a term used to refer to a computer or related device. In IBM's SNA, certain node types reflect certain functions that they can perform. (2) According to Digital Equipment documentation, an individual computer system in a network that can communicate with other computer systems in the network. A VAXBI interface—such as a central processor, controller, or memory subsystem—that occupies one of 16 logical locations on a VAXBI bus. A VAX processor or HSC that is recognized by system communications services software.

node initialization block (NIB) According to IBM documentation, in VTAM this is a control block associated with a particular node or session that contains information used by the application program. The information identifies the node or session and indicates how communication requests on a session are to be handled by VTAM.
node name According to IBM documentation, in VTAM, the symbolic name assigned to a specific major or minor node during network definition.

node number A unique number used to identify each node on a network.

node type According to IBM documentation, a node designation according to the protocols the node supports and the network-accessible units that it can contain. Five types are defined: 1, 2.0, 2.1, 4, and 5. Within a subarea network, type 1, type 2.0, and type 2.1 nodes are peripheral nodes, while type 4 and type 5 nodes are subarea nodes.

nonclient A program that is written to run on a terminal and must be fooled by a terminal emulation window into running in the window environment.

noncommand image According to Digital Equipment documentation, a program not associated with a DCL command. To invoke a noncommand image, use the filename containing the program as the parameter to the RUN command.

nonprivileged According to Digital Equipment documentation, an account with no privilege other than `TMPMBX` and `NETMBX` and a user identification code greater than the system parameter `MAXSYSGROUP.` In DECnet for Open VMS, this term means no privileges in addition to `NETMBS,` which is the minimal requirement for any network activity.

normal flow According to IBM documentation referencing SNA, a data flow designated in the transmission header (TH) that is used primarily to carry end-user data. It refers to the rate at which requests flow. On normal flow, regulation can be achieved by session-level pacing. Normal and expedited flows move in both the primary-to-secondary and secondary-to-primary directions.

normalize A term used in windowing environments meaning to change an icon back into its original appearance. The opposite of *iconify*.

notification An indication that something in the network requires the operator's attention.

NOTIFY According to IBM documentation, this is a network services request sent by an SSCP to an LU. It is used to inform the LU of the status of a procedure requested by the LU.

NULL key According to Novell documentation, this is a key field that allows the value of the field to be a user-defined null character. For this type of key, Btrieve does not index a record if the record's key value matches the null value.

object (1) According to Apple documentation, the first field in the name of an AppleTalk entity. The object is assigned by the entity itself and can be anything the user or application assigns. (2) According to Digital Equipment documentation, a passive repository of information to which the system controls access. Access to an object implies access to the information it contains. Examples of protected objects are files, volumes, global sections, and devices. A DECnet for an Open VMS process that receives a logical link request. It performs a specific network function or is a user-defined image for a special

node name According to IBM documentation, in VTAM, the symbolic name assigned to a specific major or minor node during network definition.

node number A unique number used to identify each node on a network.

node type According to IBM documentation, a node designation according to the protocols the node supports and the network-accessible units that it can contain. Five types are defined: 1, 2.0, 2.1, 4, and 5. Within a subarea network, type 1, type 2.0, and type 2.1 nodes are peripheral nodes, while type 4 and type 5 nodes are subarea nodes.

nonclient A program that is written to run on a terminal and must be fooled by a terminal emulation window into running in the window environment.

noncommand image According to Digital Equipment documentation, a program not associated with a DCL command. To invoke a noncommand image, use the filename containing the program as the parameter to the RUN command.

nonprivileged According to Digital Equipment documentation, an account with no privilege other than `TMPMBX` and `NETMBX` and a user identification code greater than the system parameter `MAXSYSGROUP.` In DECnet for Open VMS, this term means no privileges in addition to `NETMBS,` which is the minimal requirement for any network activity.

normal flow According to IBM documentation referencing SNA, a data flow designated in the transmission header (TH) that is used primarily to carry end-user data. It refers to the rate at which requests flow. On normal flow, regulation can be achieved by session-level pacing. Normal and expedited flows move in both the primary-to-secondary and secondary-to-primary directions.

normalize A term used in windowing environments meaning to change an icon back into its original appearance. The opposite of *iconify.*

notification An indication that something in the network requires the operator’s attention.

NOTIFY According to IBM documentation, this is a network services request sent by an SSCP to an LU. It is used to inform the LU of the status of a procedure requested by the LU.

NULL key According to Novell documentation, this is a key field that allows the value of the field to be a user-defined null character. For this type of key, Btrieve does not index a record if the record’s key value matches the null value.

object (1) According to Apple documentation, the first field in the name of an AppleTalk entity. The object is assigned by the entity itself and can be anything the user or application assigns. (2) According to Digital Equipment documentation, a passive repository of information to which the system controls access. Access to an object implies access to the information it contains. Examples of protected objects are files, volumes, global sections, and devices. A DECnet for an Open VMS process that receives a logical link request. It performs a specific network function or is a user-defined image for a special purpose application. A VAX Packetnet System Interface management component that contains records to specify account information for incoming calls and to specify a command procedure that is initiated when the incoming call arrives.

object class According to Digital Equipment documentation, on VAX systems, a set of protected objects with common characteristics. For example, all files belong to the FILE class whereas all devices belong to the DEVICE class.

object entry In Open Systems networking, this refers to a directory entry which is the primary collection of information in the directory information base about an object in the real world, not an alias entry.

object ID-based name Names that are based on the OBJECT IDENTIFIER type.

object identifier tree In OSI terminology, this term means a tree where edges are labeled with integers.

object identifier type A term used in OSI and other environments that implement ASN.1. It is an ASN.1 type whose values are the pathnames of the nodes of the object identifier tree.

offline Refers to a resource not being available.

online Refers to a resource being available.

open application event According to Apple documentation, an Apple event that asks an application to perform the tasks—such as displaying untitled windows—associated with opening itself; one of the four required Apple events.

open data-link interface According to Novell documentation, a set of specifications defining relationships between one or more protocol stacks, the LSL, and one or more MLIDs. These specifications allow multiple communication protocols such as IPX/SPX, TCP/IP, and AppleTalk to share the same driver and adapter.

open shortest path first (OSPF) A routing protocol based on least-cost routing.

operand In SNA, an entity on which an operation is performed; that which is operated on. An operand is usually identified by an address part of an instruction.

optional parameter According to Apple documentation, a supplemental parameter in an Apple event used to specify data that the server application should use in addition to the data specified in the direct parameter.

oscillation The periodic movement between two values.

other-domain resource According to IBM documentation, a representation for an LU that is owned by another domain and is referenced by a symbolic name, which can be qualified by a network identifier.

owner According to Digital Equipment documentation, a user with the same user identification code as the protected object. An owner always has control access to the object and can therefore modify the object's security profile. When the system processes an access request from an owner, it considers the access rights in the owner field of a protection code.

pacing A term used in IBM's SNA referring to a technique by which a receiving component controls the rate of transmission of a sending component to prevent overrun or congestion.

pacing response According to IBM documentation, in SNA, an indicator that signifies the readiness of a receiving component to accept another pacing group. The indicator is carried in a response header for session-level pacing and in a transmission header for virtual route pacing.

pacing window According to IBM documentation, this refers to the path information units (PIUs) that can be transmitted on a virtual route before a virtual route pacing response is received indicating that the virtual route receiver is ready to receive more PIUs on the route.

packet A term used generically in many instances. It is a small unit of control information and data processed by the network protocol.

packet assembly/disassembly facility A term used in packet-switching technology referring to a device at a packet-switching network permitting access from an asynchronous terminal. Terminals connect to a PAD, and a PAD puts the terminal's input data into packets, then takes the terminal's output data out of packets.

Packet InterNet Groper (PING) A program found in TCP/IP-based networks. It is the name of a program used with TCP/IP networks to test reachability of destinations by sending them an ICMP echo request and then waiting for a reply.

page A term used when virtual storage is being discussed. It refers to a fixed-length block that has a virtual address and is transferred as a unit between real storage and secondary storage.

pagelet According to Digital Equipment documentation, a 512-byte unit of memory in an AXP environment. On AXP systems, certain DCL and utility commands, system services, and system routines accept as input or provide as output memory requirements and quotas in terms of pagelets.

page table base register According to Digital Equipment documentation, on AXP systems, the processor register or its equivalent, in a hardware privileged context block that contains the page frame number of the process's first level page table.

panel According to IBM documentation, an arrangement of information that is presented in a window.

parallel links According to IBM documentation, in SNA, two or more links between adjacent subarea nodes.

parallel sessions According to IBM documentation, two or more concur rently active sessions between the same two network-accessible units (NAUs) using different pairs of network addresses or local-form session identifiers. Each session can have independent session parameters.

parallel transmission groups According to IBM documentation, multiple transmission groups (TGs) connecting two adjacent nodes.

parameter A generic term used to refer to a given constant value for a specified application and that may denote the application.

parent window In windowing environments, a window that causes another window to appear. Specifically, it refers to windows that "own" other windows.

partitioned data set (PDS) According to IBM documentation, this is a type storage, divided into partitions, called *members*. Each member contains records that are the actual data which are stored.

path In a network, any route between two or more nodes.

path control (PC) According to IBM documentation, the function that routes message units between NAUs in the network and provides the paths between them. It is depicted in traditional SNA layers. PC converts the basic information units (BIUs) from transmission control (possibly segmenting them) into path information units (PIUs) and exchanges basic transmission units containing one or more PIUs with data-link control.

path information unit (PIU) According to IBM documentation, a message unit containing only a transmission header (TH), or a TH followed by a BIU or a BIU segment.

pending active session According to IBM documentation, in VTAM, the state of an LU-LU session recorded by the SSCP when it finds both LUs available and has sent a CINIT request to the primary logical unit (PLU) of the requested session.

performance assist According to Digital Equipment documentation, the Open VMS Volume Shadowing uses controller performance assists to improve full-copy and merge operation performance. There are two distinct types of performance assists: the full-copy assist and the minimerge assist.

peripheral host node According to IBM documentation, a type of node defined in SNA terminology. It does not provide SSCP functions and is not aware of the network configuration. The peripheral host node does not provide subarea node services. It has boundary function provided by its adjacent subarea.

peripheral logical unit (PLU) A logical unit in a peripheral node found in SNA networks. It should not be confused with a primary logical unit, also abbreviated PLU.

peripheral node According to IBM documentation, a node that uses local addresses for routing and is not affected by changes in network addresses. A peripheral node requires boundary-function assistance from an adjacent subarea node.

peripheral path control According to IBM documentation, the function in a peripheral node that routes message units between units with local addresses and provides the paths between them.

permanent virtual circuit A term used in many different types of network environments. Generally, it is a permanent logical association between two DTEs, which is analogous to a leased line. Packets are routed directly by the network from one DTE to the other.

personal computer (PC) A term becoming more vague with the passing of time. It basically refers to an individual's home or office computer. By this definition, that means that it has its own processor, memory, storage, and display.

phase The place of a wave in an oscillation cycle.

physical connection A link that makes transmission of data possible. Generally agreed as a tangible link; it may support electron, photon, or other data type representation transfer.

physical layer A term used in OSI circles. It refers to the lowest layer defined by the OSI model. However, layer 0 would be the lowest of layers in such a model. This layer (layer 0) represents the medium, either hard or soft.

physical unit (PU) According to IBM documentation, a component (either software or firmware) that manages and monitors specified resources associated with a node. The PU type, indicated by number, is typically either 5, 4, 2.0, or 2.1, and dictates what supporting services are available.

physical unit (PU) services According to IBM documentation, that component within a PU that provides configuration services and maintenance services for SSCP-PU sessions.

pixel The smallest dot that can be drawn on the screen.

point A unit of measurement for type.

Point-to-Point Protocol (PPP) A protocol used over asynchronous and synchronous connections for router-to-router or a host-to-network communications.

polling The process whereby data stations are invited, one at a time, to transmit on a multipoint or point-to-point connection.

port A term used in TCP/IP-based networks. In TCP/IP there are two transport protocols: TCP and UDP. Applications that reside on top of these protocols have a port number assigned to them for addressing purposes. Generally, it is an addressable point.

port name That which contains a name string, a type string, and a script code. A term used in Apple documentation.

power manager According to Apple documentation, firmware that provides an interface to the power management hardware in the Macintosh portable computer.

presentation layer According to the OSI model for networks, this is layer 6. Data representation occurs here. Syntax of data such as ASCII or EBCDIC is determined at this layer.

presentation protocol data unit (PPDU) In OSI terminology, a term referring to logical blocks of control symbols and data transmitted at the presentation-layer protocol.

primary application program According to IBM documentation, in VTAM, this is an application program acting as the primary end of an LU-LU session.

primary end of a session According to IBM documentation, the end of a session that uses primary protocols. The primary end establishes the session. For an LU-LU session, the primary end of the session is the primary logical unit.

PrimaryInit record According to Apple documentation, a data structure in the declaration ROM of a NuBus card that contains initialization code. The slot manager executes the code in the PrimaryInit record when it first locates a declaration ROM during system startup.

primary key According to Digital Equipment documentation, the mandatory key within the data records of an indexed file; used by Open VMS RMS to determine the placement of records within the file and to build the primary index.

primary logical unit (PLU) According to IBM documentation, the logical unit that sends the BIND to activate a session with its partner LU.

PrintMonitor According to Apple documentation, a background print spooler that is included with the Macintosh MultiFinder.

print server In networking, this term is used in a general sense to convey that a computer controls spooling and other printer operations.

private partition According to IBM documentation, in VSE, this is an allocated amount of memory for the execution of a specific program or application program. Storage in a private partition is not addressable by programs running in other virtual address spaces.

privilege According to Digital Equipment documentation, this term means protecting the use of certain system functions that can affect system resources and integrity. System managers grant privileges according to users' needs and deny them to users as a means of restricting their access to the system.

process Depending on the context, this can mean an open application or an open-desk accessory.

processor That component which interprets and executes instructions.

processor status According to Digital Equipment documentation, on VAX systems, a privileged processor register, known as the *processor status longword,* consisting of a word of privileged processor status and the processor status word itself. The privileged processor status information includes the current interrupt priority level, the previous access mode, the current access mode, the interrupt stack bit, the trace trap pending bit, and the compatibility mode bit.

processor status word According to Digital Equipment, on VAX systems, the low-order word of the processor status longword. Processor status information includes the condition codes (carry, overflow, 0, negative), the arithmetic trap enable bits (integer overflow, decimal overflow, floating underflow), and the trace enable bit.

product-set identification (PSID) According to IBM documentation, a technique for identifying the hardware and software products that implement a network component.

PROFILE EXEC According to IBM documentation, in VM, a special EXEC procedure with a filename of PROFILE. The procedure is normally executed immediately after CMS is loaded into a virtual machine. It contains CP and CMS commands that are to be issued at the start of every terminal session.

program operator A term used in SNA. According to IBM documentation, a VTAM application program that is authorized to issue VTAM operator commands and receive VTAM operator awareness messages.

program temporary fix (PTF) According to IBM documentation, a temporary solution or bypass of a problem diagnosed by IBM in a current unaltered release of the program.

protocol An agreed-on way of doing something.

protocol data unit A general term referring to that which is exchanged between peer-layer entities.

PU-PU flow According to IBM documentation, in SNA this is the exchange between physical units (PUs) of network control requests and responses.

proxy ARP In TCP/IP networks, this is a technique where one machine answers ARP requests intended for another by supplying its own physical address.

pulse dispersion The spreading of pulses as they traverse an optical fiber.

queued session According to IBM documentation, in VTAM this pertains to a requested LU-LU session that cannot be started because one of the LUs is not available. If the session-initiation request specifies queuing, the system services control points (SSCPs) record the request and later continue with the session-establishment procedure when both LUs become available.

quit application event According to Apple documentation, this is an Apple event that requests that an application perform the tasks—such as releasing memory and asking the user to save documents—associated with quitting; one of the four required Apple events. The FINDER sends this event to an application immediately after sending it a PRINT DOCUMENTS event or if the user chooses restart or shutdown from the FINDERs special menu.

read-only memory (ROM) Memory in which stored data cannot be modified except under special conditions.

real resource In VTAM, a resource identified by its real name and its real network identifier.

receive pacing According to IBM documentation, the pacing of message units being received by a component.

record access block (RAB) According to Digital Equipment documentation, an Open VMS RMS user control block allocated at either assembly or run time to communicate with RMS. The control block describes the records in a particular file and associates with a file access block to form a record access stream. An RAB defines the characteristics needed to perform record-related operations, such as update, delete, or get.

record management services (RMS) According to Digital Equipment documentation, a set of operating system procedures that is called by programs to process files and records within files. RMS allows programs to issue GET and PUT requests at the record level as well as read and write blocks. RMS is an integral part of the system software; its procedures run in executive mode.

region code According to Apple documentation, a number used to indicate a particular localized version of Macintosh system software.

relative path The path through a volume's (disk's) hierarchy from one file or directory to another.

release A distribution of a new product or new function and APAR fixes for an existing product. Normally, programming support for the prior release is discontinued after some specified period of time following availability of a new release.

remote client A term used in an X-Windows environment. It is an X program running on a remote system, but the output of the program can be viewed locally.

remote operations service element A term used in open networking environments. It is an application service element that provides the basis for remote requests.

Request for Comments (also Change) (RFC) Proposed and accepted TCP/IP standards.

request header (RH) According to IBM documentation, the control information that precedes a request unit (RU).

request parameter list (RPL) According to IBM documentation, this term refers to a VTAM control block that contains the parameters necessary for processing a request for data transfer, for establishing or terminating a session, or for some other operation.

request unit (RU) According to IBM documentation, a message unit that contains control information, end-user data, or both.

request/response header (RH) According to IBM documentation, control information associated with a particular RU. The RH precedes an RU and specifies the type of RU (request or response unit).

request/response unit (RU) A generic term for a request unit or a response unit.

required apple event According to Apple documentation, it is one of four core Apple events that the Finder sends to applications. These events are called *open documents*, *open applications*, *print documents*, and *quit applications.* They are a subset of the core Apple events.

reset Generally, this is a change in the original state of operation.

resource Generally, a main storage, secondary storage, input/output devices, the processing unit, files, and control or processing programs, or anything else that can be used by a user directly or indirectly.

Resource Access Control Facility (RACF) An IBM security program package. According to IBM documentation, an IBM program that provides for access control by identifying and verifying the users of the system, authorizing access to protected resources, logging the detected unauthorized attempts to enter the system, and logging the detected accesses to protected resources.

resource definition table (RDT) According to IBM documentation, a VTAM table that describes characteristics of each node available to VTAM and associates each node with a network address. This is a main VTAM network configuration table.

resource hierarchy According to IBM documentation, a VTAM relationship among network resources in which some resources are subordinate to others as a result of their position in the network structure and architecture; for example, the logical units (LUs) of a peripheral physical unit (PU) are subordinate to the PU, which, in turn, is subordinate to the link attaching it to its subarea node.

resource registration According to IBM documentation, the process of identifying names of resources, such as LUs, to a network node server or a central directory server.

resource takeover According to IBM documentation, a VTAM action initiated by a network operator to transfer control of resources from one domain to another without breaking the connections or disrupting existing LU-LU sessions on the connection.

resource types According to IBM documentation, with reference to NetView, a concept describing the organization of panels. Resource types are defined as central processing unit, channel, control unit, and I/O device for one category; and communication controller, adapter, link, cluster controller, and terminal for another category. Resource types are combined with data types and display types to describe display organization.

response A reply to some occurrence, or the lack thereof.

response header (RH) According to IBM documentation, a header, optionally followed by a response unit, that indicates whether the response is positive or negative and may contain a pacing response.

response time According to IBM documentation, a term used with the product NetView. It refers to the elapsed time between the end of an inquiry or demand on a computer system and the beginning of the response, for example, the length of time between an indication of the end of an inquiry and the display of the first character of the response at a user terminal.

response unit (RU) According to IBM documentation, a message unit that acknowledges a request unit. It may contain prefix information received in a request unit.

restoring A term used in window-based environments to mean changing a minimized or maximized window back to its regular size.

Restructured eXtended eXecutor (REXX) According to IBM documentation, a general-purpose, procedural language for end-user personal programming, designed for ease by both casual general users and computer professionals. It is also useful for application macros.

result handler A routine that the data access manager calls to convert a data item to a character string.

return code A code used to identify the action or lack thereof of a program execution.

Reverse Address Resolution Protocol (RARP) A TCP/IP protocol for mapping Ethernet addresses to IP addresses. It is used by diskless workstations who do not know their IP addresses. In essence it asks, “Who am I?” Normally, a response occurs and is cached in the host.

RLOGIN Remote LOGIN. A logon service provided by Berkeley 4BSD UNIX systems that allows users of one machine to connect to other UNIX systems.

RMS-11 According to Digital Equipment documentation, a set of routines that are linked with compatibility mode programs and provide similar functional capabilities to Open VMS RMS. The file organizations and record formats used by RMS-11 are very similar to those of RMS.

root menu That menu which could be called the *main menu,* the menu from which other menus originate.

root window In the X-Windowing environment, this is what is presented on the screen once the X graphical user interface is visible to the user. It is the window on which other windows are based.

round-trip time (RTT) A measure of delay between two hosts.

route selection services (RSS) According to IBM documentation, this is a subcomponent of the topology and routing services component that determines the preferred route between a specified pair of nodes for a given class of service.

routing The moving of data through paths in a network.

routing information base A collection of output from route calculations.

Routing Table Maintenance Protocol (RTMP) According to Apple documentation, a protocol used by routers on an AppleTalk internet to determine how to forward a data packet to the network number to which it is addressed.

RTMP stub According to Apple documentation, the portion of the Routing Table Maintenance Protocol contained in an AppleTalk node other than a router. DDP uses the RTMP stub to determine the network number of the network cable to which the node is connected and to determine the network number and node ID on one router on that network cable.

RU chain According to IBM documentation, an SNA set of related RUs that are consecutively transmitted on a particular normal or expedited data flow.

RUN disk According to IBM documentation, this is a virtual disk that contains the VTAM, NetView, and VM/SNA console support (VSCS) load libraries, program temporary fixes (PTFs), and user-written modifications from the Zap disk.

same domain According to IBM documentation, this pertains to communication between entities in the same SNA domain.

satellite node According to Digital Equipment documentation, a processor that is part of a local area VMScluster system. A satellite node is booted remotely from the system disk of the boot server in this type of VMScluster system.

screen In SAA Basic Common User Access architecture, the physical surface of a display device on which information is shown to a user.

screen dump A screen capture capable of being routed to a file.

secondary end of a session According to IBM documentation, the end of a session that uses secondary protocols. For an LU-LU session, the secondary end of the session is the secondary logical unit (SLU).

SecondaryInit record According to Apple documentation, a data structure in the declaration ROM of a NuBus card that contains initialization code. The slot manager executes the code in the SecondaryInit record after RAM patches to the operating system have been loaded from disk during system startup.

secondary logical unit (SLU) According to IBM documentation, the LU that contains the secondary half-session for a particular LU-LU session. An LU may contain secondary and primary half-sessions for different active LU-LU sessions.

segment According to IBM documentation, with reference to a Token-Ring network, a section of cable between components or devices. A segment may consist of a single patch cable, several patch cables that are connected, or a combination of building cable and patch cables that are connected. In TCP/IP, this is the unit of transfer sent from TCP on one machine to TCP on another.

segmentation According to IBM documentation, a process by which path control divides basic information units into smaller units, called BIU segments, to accommodate smaller buffer sizes in adjacent nodes. Both segmentation and segment assembly are optional path control features. The support for either or both is indicated in the BIND request and response.

Selective ACKnowledgment (SACK) In TCP/IP, an acknowledgment mechanism used with sliding-window protocols. This permits a receiver to acknowledge packets received out of order within the current sliding window.

semaphore According to Digital Equipment documentation, in a DECnet for Open VMS network, a common data structure used to control the exchange of signals between concurrent processes.

send pacing According to IBM documentation, the pacing of message units (in SNA) that a component is sending.

sequential access mode According to Digital Equipment documentation, the retrieval or storage of records where a program reads or writes records one after the other in the order in which they appear, starting and ending at any arbitrary point in the file.

sequential file organization According to Digital Equipment documentation, a file organization in which records appear in the order in which they were originally written. The records can be of fixed or variable length. Sequential file organization permits sequential record access and random access by the record's file address. Sequential file organization with fixed-length records also permits random access by relative record number.

server An entity that serves the request of a client. This may be in the context of TCP/IP applications with clients and servers, or it could refer to a print or file server.

service access point (SAP) A logical addressable point.

service primitive Part of a service element. There are four types: confirm, indication, request, and response.

session A logical connection between two addressable endpoints.

session activation request According to IBM documentation, a request in SNA that activates a session between two NAUs and specifies session parameters that control various protocols during session activity; examples are BIND and ACTPU.

session awareness (SAW) data According to IBM documentation, data collected by the NetView program about a session that include the session type, the names of session partners, and information about the session activation status. The data are collected for LU-LU, SSCP-LU, SSCP-PU, and SSCP-SSCP session and for non-SNA terminals not supported by NTO. The data can be displayed in various forms, such as most-recent-sessions lists.

session connector According to IBM documentation, a session-layer component in an APPN network node or in a subarea node boundary or gateway function that connects two stages of a session. Session connectors swap addresses from one address space to another for session-level intermediate routing, segment session message units as needed, and (except for gateway function session connectors) adaptively pace the session traffic in each direction.

session control (SC) According to IBM documentation, one of the following: (1) one of the components of transmission control (session control is used to purge data flowing in a session after an unrecoverable error occurs, to resyn chronize the data flow after such an error, and to perform cryptographic verification) or (2) a request unit (RU) category used for requests and responses exchanged between the session control components of a session and for session activation and deactivation requests and responses.

session control block (SCB) According to IBM documentation, in NPM, control blocks in common storage area for session collection.

session data According to IBM documentation, data regarding a session, collected by the NetView program consisting of session awareness data, session trace data, and session response-time data.

session deactivation request According to IBM documentation, a term used in SNA that refers to a request that deactivates a session between two NAUs; examples are UNBIND and DACTPU.

session-establishment request According to IBM documentation, in VTAM where a request to an LU to establish a session. For the primary logical unit (PLU) of the requested session, the session-establishment request is the CINIT sent from the system services control point (SSCP) to the PLU. For the secondary logical unit (SLU) of the requested session, the session-establishment request is the BIND sent from the PLU to the SLU.

session ID According to IBM documentation, a number that uniquely identifies a session.

7.0-compatible According to Apple documentation, this term is used to refer to an application that runs without problems in system software version 7.0.

7.0-dependent According to Apple documentation, this term is used to refer to an application that requires the existence of features that are present only in system software version 7.0.

7.0-friendly According to Apple documentation, this term is used to refer to an application that is 7.0-compatible and takes advantage of some of the special features of system software version 7.0, but is still able to perform all its principal functions when operating in version 6.0.

session-initiation request According to IBM documentation, an Initiate or logon request from an LU to an SSCP that an LU-LU session be activated.

session layer Layer 5 in the OSI reference model. It coordinates the dialog between two communicating application processes.

session-level LU-LU verification According to IBM documentation, an LU6.2 security service that is used to verify the identity of each logical unit when a session is established.

session-level pacing According to IBM documentation, a flow-control technique that permits a receiving half-session or session connector to control the data transfer rate (the rate at which it receives request units) on the normal flow. It is used to prevent overloading a receiver with unprocessed requests when the sender can generate requests faster than the receiver can process them.

session limit According to IBM documentation, a term used to refer to the maximum number of concurrently active LU-LU sessions that a specific LU can support.

session manager (SM) Typically a third-party product that permits a user on one terminal to log onto multiple applications concurrently.

session monitor According to IBM documentation, a component of NetView that collects and correlates session-related data and provides online access to this information.

session parameters According to IBM documentation, the parameters that specify or constrain the protocols (such as bracket protocol and pacing) for a session between two NAUs.

session partner According to IBM documentation, in SNA, one of the two NAUs having an active session.

session path According to IBM documentation, the half-sessions delimiting a given session and their interconnection (including any intermediate session connectors).

session services According to IBM documentation, one type of network services in the control point (CP) and in the logical unit (LU). These services provide facilities for an LU or a network operator to request that a control point aid with initiating or terminating sessions between LUs. Assistance with session termination is needed only by SSCP-dependent LUs.

session stage According to IBM documentation, that portion of a session path consisting of two session-layer components that are logically adjacent and their interconnection. An example is the paired session-layer components in adjacent type 2.1 nodes and their interconnection over the link between them.

shadow resource According to IBM documentation, an alternate representation of a network resource that is retained as a definition for possible future use.

sharable image According to Digital Equipment documentation, an image that has all its internal references resolved, but must be linked with one or more object modules to produce an executable image. A sharable image cannot be executed. A sharable image file can be used to contain a library of routines.

shared-access transport facility (SATF) A transmission facility, such as a multipoint link connection or a Token-Ring network where multiple pairs of nodes can form concurrently active links.

shared image According to Digital Equipment documentation, an image that is installed so that multiple users in a system can share the memory pages where the image is loaded.

shared partition According to IBM documentation, in VSE this is a partition allocated for a program such as VSE/POWER that provides services for and communicates with programs in other partitions of the system's virtual address spaces. Storage in a shared partition is addressable by programs running concurrently in other partitions.

sibling networks According to Novell documentation, two or more equal networks branching off the same node in an internetwork. Workstations on these networks that use NetWare Btrieve must have access to a file server loaded with BSERVER.

sift-down effect According to IBM documentation, the copying of a value from a higher-level resource to a lower-level resource. The sift-down effect applies to many of the keywords and operands in NCP and VTAM definition statements. If an operand is coded on a macroinstruction or generation statement for a higher-level resource, it need not be coded for lower-level resources for which the same value is desired. The value "sifts down," that is, becomes the default for all lower-level resources.

silly-window syndrome In TCP/IP-based networks a scenario in which a receiver keeps indicating a small "window" and a sender continues to send small segments to it.

Simple Mail Transfer Protocol (SMTP) A TCP/IP application that provides email support. The SMTP protocol specifies how two mail systems interact and the format of control messages.

Simple Network Monitoring Protocol (SNMP) A de facto industry-standard protocol used to manage TCP/IP networks.

single-byte character set (SBCS) According to IBM documentation, a character set in which each character is represented by a one-byte code.

single-console image facility (SCIF) According to IBM documentation, a VM facility that allows multiple consoles to be controlled from a single virtual machine console.

single-domain network According to IBM documentation, a network with one SSCP.

single mode A fiber-optic cable containing only one mode.

sleep state According to Apple documentation, a low-power-consumption state of the Macintosh portable computer. In the sleep state, the Power Manager and the various device drivers shut off power or remove clocks from the computer's various subsystems, including the CPU, RAM, ROM, and I/O ports.

sliding window A scenario in which a protocol permits the transmitting station to send a stream of bytes before an acknowledgment arrives.

serial line IP (SLIP) A protocol used to run IP protocol over serial lines. An example is using telephone lines.

SNA network A collection of IBM hardware and software put together in such a way as to form a collective whole greater than the parts. The components of the network conform to the SNA format and protocol specifications defined by IBM.

SNA Network Interconnection (SNI) According to IBM, the connection of two or more independent SNA networks to allow communication between logical units in those networks. The individual SNA networks retain their independence.

socket (1) A concept from Berkeley 4BSD UNIX that allows an application program to access the TCP/IP protocols. (2) In TCP/IP networks, the Internet address of the host and the port number it uses. A TCP/IP application is identified by its socket.

solicited message According to IBM documentation, a response from VTAM to a command entered by a program operator.

source route A route determined by the source. TCP/IP implements source routing by using an option field in an IP datagram.

specific mode According to IBM documentation, in VTAM, this is in the form of a RECEIVE request that obtains input from one specific session or an ACCEPT request that completes the establishment of a session by accepting a specific queued CINIT request.

SSCP-dependent LU An LU requiring assistance from an SSCP to establish an LU-LU session.

SSCP ID According to IBM documentation, in SNA, a number that uniquely identifies an SSCP. The SSCP ID is used in session activation requests sent to physical units (PUs) and other SSCPs.

SSCP-independent LU According to IBM documentation, an LU that can activate an LU-LU session (i.e., send a BIND request) without assistance from an SSCP. It does not have an SSCP-LU session. Currently, only an LU6.2 can be an independent LU.

SSCP-LU session According to IBM documentation, in SNA, a session between the SSCP and an LU. The session enables the LU to request the SSCP to help initiate LU-LU sessions.

SSCP-PU session According to IBM documentation, in SNA, a session between an SSCP and a PU. SSCP-PU sessions allow SSCPs to send requests to and receive status information from individual nodes in order to control the network configuration.

SSCP rerouting According to IBM documentation, an SNA network interconncetion. A technique used by the gateway SSCP to send session-initiation RUs, by way of a series of SSCP-SSCP sessions, from one SSCP to another, until the owning SSCP is reached.

SSCP-SSCP session According to IBM documentation, a session between the SSCP in one domain and the SSCP in another domain. This type session is used to initiate and terminate cross-domain LU-LU sessions.

stack An area of memory in the application partition that is used to store temporary variables.

start-stop (SS) transmission Asynchronous transmission in which each signal that represents a character is preceded by a start signal and is followed by a stop signal.

station An input or output point.

statistic Significant data about a defined resource.

status Generally speaking, a condition or state of a resource. According to Digital Equipment documentation, a display type for the NCP commands SHOW and LIST. *Status* refers to dynamic information about a component that is kept in either the volatile or permanent database.

status monitor According to IBM documentation, a component of the NetView program that collects and summarizes information on the status of resources defined in a VTAM domain.

stream In IBM's SNA, a structured protocol, such as a 3270 data stream, a GDS data stream, or LU6.2 data stream. According to Digital Equipment documentation, an access window to a file associated with a record control block, supporting record operation requests. Generally, it is a full-duplex connection between a user's task and a device.

subarea According to IBM documentation, a portion of the SNA network consisting of a subarea node, attached peripheral nodes, and associated resources.

subarea address According to IBM documentation, a value in the subarea field of a network address that identifies a particular subarea.

subarea host node According to IBM documentation, a node that provides both subarea function and an application program interface (API) for running application programs. It provides SSCP functions and subarea node services, and is aware of the network configuration.

subarea link According to IBM documentation, a link that connects two subarea nodes.

subarea LU According to IBM documentation, a logical unit that resides in a subarea node.

subarea network According to IBM documentation, interconnected subareas, their directly attached peripheral nodes, and the transmission groups that connect them.

subarea node (SN) According to IBM documentation, a node that uses network addresses for routing and maintains routing tables that reflect the configuration of the network. Subarea nodes can provide gateway function to connect multiple subarea networks, intermediate routing functions, and boundary functions to support peripheral nodes. Type 4 and type 5 nodes are subarea nodes.

subarea path control According to IBM documentation, the function in a subarea node that routes message units between network-accessible units (NAUs) and provides the paths between them.

subdirectory According to Digital Equipment documentation, this is a directory file, cataloged in a higher-level directory, that lists additional files belonging to the owner of the directory.

subsystem A secondary or subordinate software system.

summary According to Digital Equipment documentation, the default display type for the NCP commands SHOW and LIST. A summary includes the most useful information for a component, selected from the status and characteristics information.

supervisor According to IBM documentation, that part of a control program that coordinates the use of resources and maintains the flow of processing unit operations.

supervisor call (SVC) According to IBM documentation, a request that serves as the interface into operating system functions, such as allocating storage. The SVC protects the operating system from inappropriate user entry. All operating system requests must be handled by SVCs.

switched connection A data-link connection that functions like a dial telephone.

switched line A line in which the connection is established by dialing.

switched major node According to IBM documentation, in VTAM, a major node whose minor nodes are physical units and logical units attached by switched SDLC links.

switched network A network that establishes connections by a dialing function.

switched network backup According to IBM documentation, an optional facility that allows a user to specify, for certain types of physical units (PUs), a switched line to be used as an alternate path if the primary line becomes unavailable or unusable.

switched virtual circuit A temporary logical association between two DTEs connected to a packet-switching data network.

symbiont According to Digital Equipment documentation, a process that transfers record-oriented data to or from a device. For example, an input symbiont transfers data from card readers to disks, and an output symbiont transfers data from disks to line printers.

symbiont manager According to Digital Equipment documentation, the function that maintains spool queues and dynamically creates symbiont processes to perform the necessary I/O operations.

symbol According to Digital Equipment documentation, an entity that, when defined, will represent a particular function or entity (e.g., a command string, directory name, or filename) in a particular context.

symbol table According to Digital Equipment documentation, that portion of an executable image that contains the definition of global symbols used by the debugger for images linked with the DEBUG qualifier. A table in which the Digital Command Language (DCL) places local symbols. DCL maintains a local symbol table for each command level.

synchronous backplane interconnect According to Digital Equipment documentation, that part of the hardware that interconnects the VAX processor, memory controllers, MASSBUS adapters, and the UNIBUS adapter.

synchronization point According to IBM documentation, an intermediate point or endpoint during processing of a transaction at which an update or modification to one or more of the transaction’s protected resources is logically complete and error free.

sync-point services (SPS) According to IBM documentation, the component of the sync-point manager that is responsible for coordinating the managers of protected resources during sync-point processing. SPS coordinates two-phase commit protocols, resync protocols, and logging.

system communication services According to Digital Equipment documentation, a protocol responsible for the formation and breaking of intersystem process connections and for flow control of message traffic over those connections. System services such as the VMScluster connection manager and the Mass Storage Control Protocol (MSCP) disk server communicate with this protocol.

system control block According to Digital Equipment documentation, on VAX systems, the data structure in system space that contains all the interrupt and exception vectors known to the system.

system definition According to IBM documentation, the process, completed before a system is put into use, by which desired functions and operations of the system are selected from various available options.

system disk According to Digital Equipment documentation, the disk that contains the operating system. In a VMScluster environment, a system disk is set up so that most of the files can be shared by several processors. In addition, each processor has its own directory on the system disk that contains its page, swap, and dump files.

system file According to Apple documentation, a file, located in the system folder, that contains the basic system software plus some system resources, such as font and sound resources.

system generation Synonym for system definition.

System GETVIS area According to IBM documentation, a storage space that is available for dynamic allocation to VSE system control programs or other application programs.

system image According to Digital Equipment documentation, the image read into memory from disk when the system is started up.

system management facility (SMF) According to IBM documentation, a feature of MVS that collects and records a variety of system and job-related information.

system menu In a windowing environment, particularly the X-Window environment, the menu that displays when you press the system menu button on the window manager window frame. Every window has a system menu that enables you to control the size, shape, and position of the window.

system modification program (SMP) According to IBM documentation, a program used to install software changes on MVS systems.

system services control point (SSCP) According to IBM documentation, a component within a subarea network for managing the configuration, coordinating network operator, and problem determination requests, and providing directory services and other session services for end users of the network. Multiple SSCPs, cooperating as peers with one another, can divide the network into domains of control, with each SSCP having a hierarchical control relationship to the PUs and LUs within its own domain.

SSCP domain According to IBM documentation, the system services control point, the PUs, the LUs, the links, the link stations, and all the resources that the SSCP has the ability to control by means of activation and deactivation requests.

System Support Program (SSP) According to IBM documentation, an IBM licensed program, made up of a collection of utilities and small programs, that supports the operation of the NCP.

Systems Network Architecture (SNA) IBM's description of the logical structure, formats, protocols, and operational sequences for their network offering called SNA.

takeover According to IBM documentation, the process by which the failing active subsystem is released from its eXtended Recovery Facility (XRF) sessions with terminal users and replaced by an alternate subsystem.

task specifier According to Digital Equipment documentation, information provided to DECnet for Open VMS software that enables it to complete a logical link connection to a remote task. This information includes the name of the remote node on which the target task runs and the name of the task itself.

telecommunications access method (TCAM) According to IBM documentation, the access method prior to VTAM.

TELNET The TCP/IP standard protocol for remote terminal service.

10-BaseT Technical name for Ethernet implemented on twisted wire.

terminal Generally agreed on as a point of entry with a display and keyboard.

terminal access facility (TAF) According to IBM documentation, in the NetView program, a facility that allows a network operator to control a number of subsystems. In a full-screen or operator control session, operators can control any combination of such subsystems simultaneously.

terminal-based program In the X-Windows environment, a program (nonclient) written to be run on a terminal (not in a window). Terminal-based programs must be fooled by terminal-emulation clients to run on the X-Windows system.

terminal emulator A term used to generally refer to a program that performs some type of simulation; typically this simulation is of a type of terminal.

TERMINATE According to IBM documentation, in SNA it is a request unit that is sent by an LU to its SSCP to cause the SSCP to start a procedure to end one or more designated LU-LU sessions.

terminal server A network device used to connect “dumb” terminals to a network medium. Consequently, these terminals have virtual terminal access to hosts and devices located on a network.

terminal type The type of terminal attached to your computer. UNIX uses the terminal type to set the TERM environment variable so that it can communicate with the terminal correctly. In the SNA environment, the terminal type is required in order to know how to configure the system so that it can function.

term0 According to Hewlett-Packard documentation, a level 0 terminal is a reference standard that defines basic terminal functions.

TG weight According to IBM documentation, a quantitative measure of how well the values of a transmission group’s characteristics satisfy the criteria specified by the class-of-service definition, as computed during route selection for a session.

thread According to Digital Equipment documentation, a single, sequential flow of control within a program. It is the active execution of a designated routine, including any nested routine invocations. A single thread has a single point of execution within it. A thread can be executed in parallel with other threads.

threshold Generally agreed on as a percentage value set for a resource.

tile In the X-Windows environment, a rectangular area used to cover a surface with a pattern or visual texture.

time-division multiplexing (TDM) A technique used to multiplex data on a channel by a timesharing of the channel.

time-domain reflectometer (TDR) A device used to troubleshoot networks. It sends signals through a network medium to check for continuity.

time to live (TTL) A technique used in best-effort delivery systems to avoid endlessly looping packets. For example, packets have a “time” associated with their lifetime.

timesharing option extensions (TSO/E) According to IBM, the base for all TSO enhancements; a program that provides enhancements to MVS/XA users.

timeout An event that occurs at the end of a predetermined period of time.

title bar A term used in the X-Windows environment; the rectangular area between the top of the window and the window frame. The title bar contains the title of the window object. For example, Xclock for clocks.

TN3270 A program that uses the TELNET protocol but produces an EBCDIC 3270 data stream. The program is normally found as a TN3270-client application that provides access into a 3270-based environment.

token The symbol of authority passed successively from one data station to another to indicate the station temporarily in control of the transmission medium.

Token Ring A network with a ring topology that passes tokens from one attaching device to another.

Token-Ring interface coupler (TIC) Interface board used to connect a device such as a 3720, 3725, or 3745 communication controller to a Token-Ring network.

Token-Ring network A ring network that allows unidirectional data transmission between data stations by a token-passing procedure.

topology and routing services (TRS) According to IBM documentation, an APPN control point component that manages the topology database, computes routes, and provides a route selection control vector (RSCV) that specifies the best route through the network for a given session according to its requested class of service.

trace A record of events captured and used to troubleshoot hardware and/or software.

transaction According to Apple documentation, a sequence of Apple events sent back and forth between a client and a server application, beginning with the client's initial request for a service.

transaction-processing facility (TPF) A software system designed to support real-time applications.

transaction program According to IBM, a program that conforms to LU6.2 protocols.

transceiver A device that connects a host's cable from the interface board to the main cable of the network.

trap An event used in SNMP-managed networks to send data to the network manager. A trap is sent from an SNMP agent.

trash folder According to Apple documentation, a directory at the root level of a volume for storing files that the user has moved to the TRASH icon. After opening the TRASH icon, the user sees the collection of all items that the user has moved to the trash icon—that is, the union of appropriate trash directories from all mounted volumes. A Macintosh set up to share files among users in a network environment maintains separate trash subdirectories for remote users within its shared, network trash directory. The FINDER for system software version 7.0 empties a trash directory only when the user of that directory chooses the EMPTY TRASH command.

translated code According to Digital Equipment documentation, the native AXP object code in a translated image. Translated code includes AXP code that reproduces the behavior of equivalent VAX code in the original image and calls to the translated image environment.

translated image According to Digital Equipment documentation, an AXP-executable or -sharable image created by translating the object code of a VAX image. The translated image, which is functionally equivalent to the VAX image from which it was translated, includes both translated code and the original image.

translated image environment (TIE) According to Digital Equipment documentation, a native AXP-sharable image that supports the execution of translated images. The TIE processes all interactions with the native AXP system and provides an environment similar to VAX for the translated image by managing VAX state; by emulating VAX features such as exception processing, AST delivery, and complex VAX instructions; and by interpreting untranslated VAX instructions.

translation According to Digital Equipment documentation, the process of converting a VAX binary image to an AXP image that runs with the assistance of the TIE on an AXP system. Translation is a static process which converts as much VAX code as possible to native ALPHA AXP instructions. The TIE interprets any untranslated VAX code at run time.

translation table A table used to replace one or more characters with alternative characters.

Transmission Control Protocol (TCP) The TCP/IP standard transport-level protocol that provides the reliable, full-duplex, stream service on which many application protocols depend. It is connection-oriented in that before transmitting data, participants must establish a connection.

transmission group (TG) According to IBM, a group of links between adjacent subarea nodes, appearing as a single logical link for routing of messages.

transmission header (TH) According to SNA, this is control information, optionally followed by a basic information unit, created and used by path control to route message units, and to control their flow within the network.

transmission priority According to IBM documentation, a rank assigned to a message unit that determines its precedence for being selected by the path control component in each node along a route for forwarding to the next node in the route.

transport layer According to the OSI model, the layer that provides a reliable end-to-end service to its users.

transport network According to IBM documentation, that part of an SNA network that includes the data-link control and path control layer.

Trivial File Transfer Protocol (TFTP) A TCP/IP standard protocol for file transfer that uses UDP as a transport mechanism. TFTP depends only on UDP so that it can be used on machines such as diskless workstations.

type According to Apple documentation, the second field in the name of an AppleTalk entity. The type is assigned by the entity itself and can be anything the user or application assigns.

type 2.1 end node According to IBM documentation, a type 2.1 node that provides full SNA end-user services, but no intermediate routing or network services to any other node; it is configured only as an endpoint in a network.

type 2.1 network According to IBM documentation, a collection of interconnected type 2.1 network nodes and type 2.1 end nodes. A type 2.1 network may consist of nodes of just one type, namely, all network nodes or all end nodes; a pair of directly attached end nodes is the simplest case of a type 2.1 network.

type 2.1 node A node that conforms to IBM's type 2.1 architecture.

type 5 node According to IBM documentation, a node that can be any one of the following: (1) advanced peer-to-peer networking (APPN) end node, (2) advanced peer-to-peer networking (APPN) network node, (3) interchange node, (4) low-entry networking (LEN) node, (5) migration data host, or (6) subarea node. It is also a node that traditionally has the SSCP.

UNBIND According to IBM, a request to deactivate a session between two logical units (LUs).

unformatted According to IBM, pertaining to commands (such as LOGON or LOGOFF) entered by an end user and sent by an LU in character form.

Unformatted System Services (USS) According to IBM documentation, in SNA products, an SSCP facility that translates a character-coded request, such as a LOGON or LOGOFF request, into a field-formatted request for processing by formatted system services and that translates field-formatted replies and responses into character-coded requests for processing by a logical unit.

unit control block (UCB) According to Digital Equipment documentation, structure in the I/O database that describes the characteristics of and current activity on a device unit. The unit control block also holds the fork block for its unit's device driver; the fork block is a critical part of a driver fork process. The UCB also provides a dynamic storage area for the driver.

universal symbol According to Digital Equipment documentation, a global symbol in a sharable image that can be used by modules linked with that sharable image. Universal symbols are typically a subset of all the global symbols in a sharable image. When creating a sharable image, the linker ensures that universal symbols remain available for reference after the symbols have been resolved.

UNIX-to-UNIX copy program (UUCP) An application program that allows one UNIX system to copy files to or from another UNIX system.

unsolicited message According to IBM documentation, a message from VTAM to a program operator that is unrelated to any command entered by the program operator.

upline dump According to Digital Equipment documentation, in DECnet for Open VMS, a function that allows an adjacent node to dump its memory to a file on a system.

User Datagram Protocol (UDP) A TCP/IP standard protocol that is in contrast to TCP. UDP is connectionless and unreliable.

user exit According to IBM documentation, a point in an IBM-supplied program at which a user exit routine may be given control.

user exit routine According to IBM documentation, a user-written routine that receives control at predefined user exit points. User exit routines can be written in assemblies or a high-level language.

user file directory According to Digital Equipment documentation, a file that briefly catalogs a set of files stored on disk or tape. The directory includes the name, type, and version number of each file in the set. It also contains a unique number that identifies that file's actual location and points to a list of its file attributes.

user privileges According to Digital Equipment documentation, those privileges granted to a user by the system manager.

VAXBI According to Digital Equipment documentation, the part of the VAX 8200, VAX 8250, VAX 8300, VAX 8350 hardware that connects I/O adapters with memory controllers and the processor. In VAX 8530, VAX 8550, VAX 8700, VAX 8800, or VAX 6200 and VAX 6300 systems, the part of the hardware that connects I/O adapters with the bus that interfaces with the processor and memory.

VAXcluster configuration According to Digital Equipment documentation, a highly integrated organization of Open VMS systems that communicate over a high-speed communications path. VAXcluster configurations have all the functions of single-node systems, plus the ability to share CPU resources, queues, and disk storage. Like a single-node system, the VAXcluster configuration provides a single security and management environment. Member nodes can share the same operating environment or serve specialized needs.

VAX Environment Software Translator (VEST) According to Digital Equipment documentation, a software migration tool that translates VAX-executable and -shareable images into translated images that run on AXP systems. VEST is part of the DECmigrate toolset.

VAX vector instruction emulation facility (VVIEF) According to Digital Equipment documentation, a standard feature of the operating system that allows vectorized applications to be written and debugged in a VAX system in which vector processors are not available. VVIEF emulates the VAX vector processing environment, including the nonprivileged VAX vector instructions and the vector system services. Use of VVIEF is restricted to user mode code.

vector According to Digital Equipment documentation, a storage location that contains the starting address of a procedure to be executed when a given interrupt or exception occurs.

vector present system According to Digital Equipment documentation, a VAX system that, in its hardware implementation, complies with the VAX vector architecture, and incorporates one or more optional vector processors.

virtual disk According to the IBM corporation, in VM, a physical disk storage device, or a logical subdivision of a physical disk storage device, that has its own address, consecutive storage space for data, and index or description of stored data so that the data can be accessed.

virtual filestore A concept in OSI that refers to the OSI abstraction of a collection of files, directories, and/or references.

virtual machine (VM) According to IBM documentation, in VM, a functional equivalent of a computing system. On the 370 feature of VM, a virtual machine operates in System/370 mode. On the ESA feature of VM, a virtual machine operates in System/370, 370-XA, ESA/370, or ESA/390 mode. Each virtual machine is controlled by an operating system. VM controls the concurrent execution of multiple virtual machines on an actual processor complex.

virtual machine group According to IBM documentation, in the group control system (GCS), two or more virtual machines associated with each other through the same named system.

Virtual Machine/Enterprise Systems Architecture (VM/ESA) According to IBM documentation, an IBM program that manages the resources of a single computer so that multiple computing systems appear to exist. Each virtual machine is the functional equivalent of a real machine.

Virtual Machine/eXtended Architecture (VM/XA) According to IBM documentation, an operating system that facilitates conversion to MVS/XA by allowing several operating systems (a production system and one or more test systems) to run simultaneously on a single 370-XA processor.

Virtual Machine/System Product (VM/SP) According to IBM, a program that manages the resources of a single computer so that multiple computing systems appear to exist.

Virtual Machine/System Product High Performance Option (VM/SP HPO) According to IBM documentation, a program that can be installed and executed in conjunction with VM/SP to extend the capabilities of VM/SP with programming enhancements, support for microcode assists, and additional functions.

virtual route (VR) According to IBM documentation, in SNA it is either a logical connection between two subarea nodes that is physically realized as a particular explicit route or a logical connection that is contained wholly within a subarea node for intranode sessions.

virtual route (VR) pacing According to IBM documentation, in SNA, a flow-control technique used by the virtual route control component of path control at each end of a virtual route to control the rate at which path information units (PIUs) flow over the virtual route.

virtual routing node According to IBM documentation, a representation of a node's connectivity to a connection network defined on a shared-access transport facility, such as a Token Ring.

virtual storage According to IBM documentation, storage space that may be regarded as addressable main storage by the user of a computer system in which virtual addresses are mapped into real addresses.

virtual storage access method (VSAM) According to IBM documentation, an access method of direct or sequential processing of fixed- and variable-length records on direct-access devices.

Virtual Storage Extended (VSE) According to IBM documentation, a program whose full name is the *Virtual Storage Extended/Advanced Function.* It is a software operating system controlling the execution of programs.

virtual telecommunication access method (VTAM) According to IBM documentation, a program that controls communication and the flow of data in an SNA network. It provides single-domain, multiple-domain, and interconnected network capability.

VM/SNA console support (VSCS) According to IBM documentation, a VTAM component for the VM environment that provides Systems Network Architecture (SNA) support. It allows SNA terminals to be virtual machine consoles.

VM/370 control program (CP) According to IBM documentation, that component of VM/370 that manages the resources of a single computer with the result that multiple computing systems appear to exist. Each virtual machine is the functional equivalent of an IBM System/370 computing system.

VSE/Advanced Functions According to IBM documentation, the basic operating system support needed for a VSE-controlled installation.

VMScluster configuration According to Digital Equipment documentation, a highly integrated organization of Open VMS AXP systems, or a combination of AXP or VAX systems, that communicate over a high-speed communication path. VMScluster configurations have all the functions of single-node systems, plus the ability to share CPU resources, queues, and disk storage. Like a single-node system, the VMScluster configuration provides a single security and management environment. Member nodes can share the same operating environment or serve specialized needs.

VTAM application program According to IBM documentation, a program that has opened an access method control block (ACB) to identify itself to VTAM and that can therefore issue VTAM macroinstructions.

VTAM Common Network Services (VCNS) According to IBM documentation, this is VTAM's support for shared physical connectivity between SNA networks and certain non-SNA networks.

VTAM definition According to IBM documentation, the process of defining the user application network to VTAM and modifying IBM-defined characteristics to suit the needs of the user.

VTAM definition library According to IBM documentation, the operating system files or data sets that contain the definition statements and start options filed during VTAM definition.

VTAM internal trace (VIT) According to IBM documentation, a trace used in VTAM to collect data on channel I/O, use of locks, and storage management services.

VTAM operator According to IBM documentation, a person or program authorized to issue VTAM operator commands.

VTAM operator command According to IBM documentation, a command used to monitor or control a VTAM domain.

waveform The representation of a disturbance as a function as it occurs in time and its relationship to space.

wavelength Defined as the distance that an electromagentic wave can travel in the amount of time it takes to oscillate through a complete cycle.

well-known port A term used with TCP/IP networks. In TCP/IP, applications and programs that reside on top of TCP and UDP, respectively, have a designated port assigned to them. This agreed-on port is known as a *well-known port.*

window A term used with environments such as X-Windows. Generally, the term is used in contrast with line or full-screen mode.

window-based program A program written for use with a windowing system; for example, this could refer to an X-Windows environment or the Microsoft Windows environment. Opposite from a window-based program is a terminal-based program.

window decoration In the X-Windows environment, the frame and window control buttons that surround windows managed by the window manager.

window manager A program in the X-Windowing system that controls size, placement, and operation of windows on the root window. The window manager includes the functional window frames that surround each window object as well as a menu for the root window.

XENIX A version of UNIX that can run on a PC.

X.21 A CCITT standard defining logical link control and media access control in X.25 networks.

X.25 A CCITT standard for packet-switched network layer services.

X.400 A CCITT and ISO combination of standards for providing electronic mail services.

X.500 A CCITT and ISO combination of standards for providing directory services.

X application An application program that conforms to X protocol standards.

X protocol A protocol that uses TCP as a transport mechanism. It supports asynchronous, event-driven distributed window environments; this can be across heterogeneous platforms.

X terminal A terminal and machine specifically designed to run an X server. In this type environment, X clients are run on remote systems.

X toolkit A collection of high-level programs based on programming from the X library.

X-Windows system A software system developed at MIT whose original design intent was to provide distributed computing support for the development of programs. It supports two-dimensional bitmapped graphics.

Zap disk According to IBM documentation, the virtual disk in the VM operating system that contains the user-written modifications to VTAM code.

zone (1) According to Digital Equipment documentation, this is a section of a fully configured VAXft fault-tolerant computing system that contains a minimum of a CPU module, memory module, I/O module, and associated devices. A VAXft system consists of two such zones with synchronized processor operations. If one zone fails, processing continues uninterrupted through automatic failover to the other zone. (2) In AppleTalk, a logical grouping of devices in an AppleTalk internet that makes it easier for users to locate network services. The network administrator defines zones during the router setup process. (3) According to Apple documentation, this is a logical grouping of a subset of the nodes on the AppleTalk internet. The zone is the third field in the name of an AppleTalk entity.

Zone Information Protocol An AppleTalk protocol that maintains a table in each router, called the *zone information table,* that lists the relationship between zone names and networks.

zone name According to Apple documentation, a name defined for each zone in an AppleTalk internet. A LocalTalk network can have only one zone name. Ethernet and Token-Ring networks can have multiple zone names, called a *zone list.*

zone of authority A term used with the domain name system to refer to the group of names authorized by a given name server.