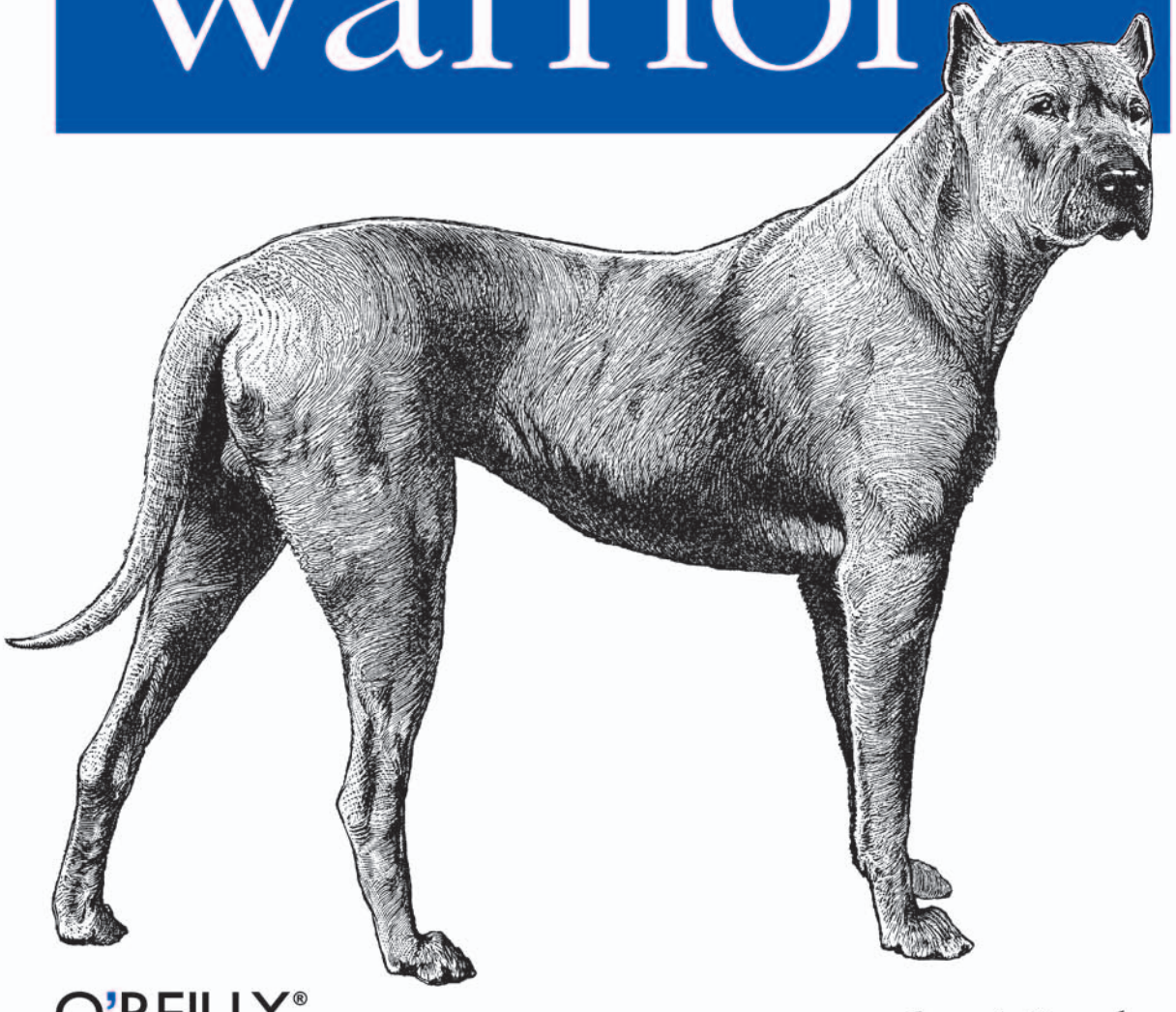
Everything You Need to Know That Wasn't on the CCNA Exam

Network Warrior



O'REILLY®

Gary A. Donahue

Network Warrior

Other resources from O'Reilly

Related titles

- BGP
- Cisco IOS Cookbook™
- DNS & BIND Cookbook™
- Essential SNMP
- Ethernet: The Definitive Guide
- Internet Core Protocols: The Definitive Guide
- IPv6 Essentials
- IPv6 Network Administration
- TCP/IP Network Administration

oreilly.com

oreilly.com is more than a complete catalog of O'Reilly's books. You'll also find links to news, events, articles, weblogs, sample chapters, and code examples.

oreillynet.com is the essential portal for developers interested in open and emerging technologies, including new platforms, programming languages, and operating systems.

Conferences

O'Reilly brings diverse innovators together to nurture the ideas that spark revolutionary industries. We specialize in documenting the latest tools and systems, translating the innovator's knowledge into useful skills for those in the trenches.

Visit *conferences.oreilly.com* for our upcoming events.

Safari Bookshelf (*safari.oreilly.com*) is the premier online reference library for programmers and IT professionals. Conduct searches across more than 1,000 books. Subscribers can zero in on answers to time-critical questions in a matter of seconds. Read the books on your Bookshelf from cover to cover or simply flip to the page you need. Try it today for free.

Network Warrior

Gary A. Donahue

O'REILLY®
Beijing • Cambridge • Farnham • Köln • Paris • Sebastopol • Taipei • Tokyo

Network Warrior
by Gary A. Donahue

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

Editor: Mike Loukides
Production Editor: Sumita Mukherji
Copyeditor: Rachel Head
Proofreader: Sumita Mukherji
Indexer: Ellen Troutman
Cover Designer: Karen Montgomery
Interior Designer: David Futato
Illustrators: Robert Romano and Jessamyn Read

Printing History:

June 2007: First Edition.

This book uses RepKover™, a durable and flexible lay-flat binding.

ISBN-10: 0-596-10151-1
ISBN-13: 978-0-596-10151-0
[C]

For my girls:
Lauren, Meghan, and Colleen,
and Cozy and Daisy.

—Gary A. Donahue

Table of Contents

Part IV. Telecom

Part V. Security and Firewalls

Part VIII. Designing Networks

Preface

The examples used in this book are taken from my own experiences, as well as from the experiences of those with or for whom I have had the pleasure of working. Of course, for obvious legal and honorable reasons, the exact details and any information that might reveal the identities of the other parties involved have been changed.

Cisco equipment is used for the examples within this book, and, with very few exceptions, the examples are TCP/IP-based. You may argue that a book of this type should include examples using different protocols and equipment from a variety of vendors, and, to a degree, that argument is valid. However, a book that aims to cover the breadth of technologies contained herein, while also attempting to show examples of these technologies from the point of view of different vendors, would be quite an impractical size.

The fact is that Cisco Systems (much to the chagrin of its competitors, I'm sure) is the premier player in the networking arena. Likewise, TCP/IP is the protocol of the Internet, and the protocol used by most networked devices. Is it the best protocol for the job? Perhaps not, but it is the protocol in use today, so it's what I've used in all my examples. Not long ago, the Cisco CCIE exam still included Token Ring Source Route Bridging, AppleTalk, and IPX. Those days are gone, however, indicating that even Cisco understands that TCP/IP is where everyone is heading.

WAN technology can include everything from dial-up modems (which, thankfully, are becoming quite rare in metropolitan areas) to ISDN, T1, DS3, SONET, and so on. We will cover many of these topics, but we will not delve too deeply into them, for they are the subject of entire books unto themselves—some of which may already sit next to this one on your O'Reilly bookshelf.

Again, all the examples used in this book are drawn from real experiences, most of which I faced myself during my career as a networking engineer, consultant, manager, and director. I have run my own company, and have had the pleasure of working with some of the best people in the industry, and the solutions presented in these chapters are those my teams and I discovered or learned about in the process of resolving the issues we encountered.

Who Should Read This Book

This book is intended for use by anyone with first-level certification knowledge of data networking. Anyone with a CCNA or equivalent (or greater) knowledge should benefit from this book. My goal in writing *Network Warrior* is to explain complex ideas in an easy-to-understand manner. While the book contains introductions to many topics, you can also consider it as a reference for executing common tasks related to those topics. I am a teacher at heart, and this book allows me to teach more people than I'd ever thought possible. I hope you will find the discussions I have included both informative and enjoyable.

I have noticed over the years that people in the computer, networking, and telecom industries are often misinformed about the basics of these disciplines. I believe that in many cases, this is the result of poor teaching, or the use of reference material that does not convey complex concepts well. With this book, I hope to show people how easy some of these concepts are. Of course, as I like to say, "It's easy when you know how," so I have tried very hard to help anyone who picks up my book understand the ideas contained herein.

If you are reading this, my guess is that you would like to know more about networking. So would I! Learning should be a never-ending adventure, and I am honored that you have let me be a part of your journey. I have been studying and learning about computers, networking, and telecom for the last 24 years, and my journey will never end.

This book attempts to teach you what you need to know in the real world. When should you choose a layer-3 switch over a layer-2 switch? How do you tell if your network is performing as it should? How do you fix a broadcast storm? How do you know you're having one? How do you know you have a spanning-tree loop, and how do you fix it? What is a T1, or a DS3 for that matter? How do they work? In this book, you'll find the answers to all of these questions, and many, many more. *Network Warrior* includes configuration examples from real-world events and designs, and is littered with anecdotes from my time in the field—I hope you enjoy them.

Conventions Used in This Book

The following typographical conventions are used in this book:

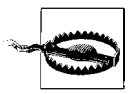
Italic
: Used for new terms where they are defined, for emphasis, and for URLs

`Constant width`
: Used for commands, output from devices as it is seen on the screen, and samples of Request for Comments (RFC) documents reproduced in the text

`Constant width italic`
: Used to indicate arguments within commands for which you should supply values

`Constant width bold`
: Used for commands to be entered by the user and to highlight sections of output from a device that have been referenced in the text or are significant in some way

Indicates a tip, suggestion, or general note



Indicates a warning or caution

Using Code Examples

This book is here to help you get your job done. In general, you may use the code in this book in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*Network Warrior* by Gary A. Donahue. Copyright 2007 O'Reilly Media, Inc., 978-0-596-10151-0."

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at *permissions@oreilly.com*.

We'd Like to Hear from You

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international or local)
707-829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at:

http://www.oreilly.com/catalog/9780596101510

To comment or ask technical questions about this book, send email to:

bookquestions@oreilly.com

For more information about our books, conferences, Resource Centers, and the O'Reilly Network, see our web site at:

http://www.oreilly.com

Safari® Enabled

When you see a Safari® Enabled icon on the cover of your favorite technology book, that means the book is available online through the O'Reilly Network Safari Bookshelf.

Safari offers a solution that's better than e-books. It's a virtual library that lets you easily search thousands of top tech books, cut and paste code samples, download chapters, and find quick answers when you need the most accurate, current information. Try it for free at *http://safari.oreilly.com*.

Acknowledgments

Writing a book is hard work—far harder than I ever imagined. Though I spent countless hours alone in front of a keyboard, I could not have accomplished the task without the help of many others.

I would like to thank my lovely wife, Lauren, for being patient, loving, and supportive. Lauren, being my in-house proofreader, was also the first line of defense against grammatical snafus. Many of the chapters no doubt bored her to tears, but I know she enjoyed at least a few. Thank you for helping me achieve this goal in my life.

I would like to thank Meghan and Colleen for trying to understand that when I was writing, I couldn't play. I hope I've helped instill in you a sense of perseverance by completing this book. If not, you can be sure that I'll use it as an example for the rest of your lives. I love you both "bigger than the universe" bunches.

I would like to thank my mother—because she's my mom, and because she never gave up on me, always believed in me, and always helped me even when she shouldn't have (Hi, Mom!).

I would like to thank my father for being tough on me when he needed to be, for teaching me how to think logically, and for making me appreciate the beauty in the details. I have fond memories of the two of us sitting in front of my Radio Shack Model III computer while we entered basic programs from a magazine. I am where I am today largely because of your influence, direction, and teachings. You made me the man I am today. Thank you, Papa. I miss you.

I would like to thank my Cozy, my faithful Newfoundland dog who was tragically put to sleep in my arms so she would no longer have to suffer the pains of cancer. Her body failed while I was writing this book, and if not for her, I probably would not be published today. Her death caused me great grief, which I assuaged by writing. I miss you my Cozy—may you run pain free at the rainbow bridge until we meet again.

I would like to thank Matt Maslowski for letting me use the equipment in his lab that was lacking in mine, and for helping me with Cisco questions when I wasn't sure of myself. I can't think of anyone I would trust more to help me with networking topics. Thanks, buddy.

I would like to thank Adam Levin for answering my many Solaris questions, even the really nutty ones. Sorry the book isn't any shorter.

I would like to thank Jeff Cartwright for giving me my first exciting job at an ISP and for teaching me damn-near everything I know about telecom. I still remember being taught about one's density while Jeff drove us down Interstate 80, scribbling waveforms on a pad on his knee while I tried not to be visibly frightened. Thanks also for proofreading some of my telecom chapters. There is no one I would trust more to do so.

I would like to thank Mike Stevens for help with readability and for some of the more colorful memories that have been included in this book. His help with PIX firewalls was instrumental to the completion of those chapters.

I would like to thank Peter Martin for helping me with some subjects in the lab for which I had no previous experience. And I'd like to extend an extra thank you for your aid as one of the tech reviewers for *Network Warrior*—your comments were always spot-on, and your efforts made this a better book.

I would like to thank another tech reviewer, Yves Eynard: you caught some mistakes that floored me, and I appreciate the time you spent reviewing. This is a better book for your efforts.

I would like to thank Paul John for letting me use the lab while he was using it for his CCIE studies.

I would like to thank Henri Tohme and Lou Marchese for understanding my need to finish this book, and for accommodating me within the limits placed upon them.

I would like to thank Sal Conde and Ed Hom for access to 6509E switches and modules.

I would like to thank Christopher Leong for doing some last-minute technical reviews on a couple of the telecom chapters.

I would like to thank Mike Loukides, my editor, for not cutting me any slack, for not giving up on me, and for giving me my chance in the first place. You have helped me become a better writer, and I cannot thank you enough.

I would like to thank Rachel Head, the copyeditor who made this a much more readable book.

I would like to thank Robert Romano, senior technical illustrator at O'Reilly, for working to keep the illustrations in this book as close to my original drawings as possible.

I would like to thank all the wonderful people at O'Reilly. Writing this book was an awesome experience, due in large part to the people I worked with at O'Reilly.

I would like to thank my good friend, John Tocado, who once told me, "If you want to write, then write!" This book is proof that you can change someone's life with a single sentence. You'll argue that I changed my own life, and that's fine, but you'd be wrong. When I was overwhelmed with the amount of remaining work to be done, I seriously considered giving up. Your words are the reason I did not. Thank you.

I cannot begin to thank everyone else who has given me encouragement. Living and working with a writer must, at times, be maddening. Under the burden of deadlines, I've no doubt been cranky, annoying, and frustrating, for which I apologize.

My purpose for the last year has been the completion of this book. All other responsibilities, with the exception of health and family, took a back seat to my goal. Realizing this book's publication is a dream come true for me. You may have dreams yourself, for which I can offer only this one bit of advice: work toward your goals, and you will realize them. It really is that simple.

PART I

Hubs, Switches, and Switching

This section begins with a brief introduction to networks. It then moves on to describe the benefits and drawbacks of hubs and switches in Ethernet networks. Finally, many of the protocols commonly used in a switched environment are covered.

This section is composed of the following chapters:

Chapter 1, *What Is a Network?*
Chapter 2, *Hubs and Switches*
Chapter 3, *Auto-Negotiation*
Chapter 4, *VLANs*
Chapter 5, *Trunking*
Chapter 6, *VLAN Trunking Protocol*
Chapter 7, *EtherChannel*
Chapter 8, *Spanning Tree*

CHAPTER 1
What Is a Network?

Before we get started, I would like to define some terms and set some ground rules. For the purposes of this book (and your professional life, I hope), a *computer network* can be defined as "two or more computers connected by some means through which they are capable of sharing information." Don't bother looking for that in an RFC because I just made it up, but it suits our needs just fine.

There are many types of networks: Local Area Networks (LANs), Wide Area Networks (WANs), Metropolitan Area Networks (MANs), Campus Area Networks (CANs), Ethernet networks, Token Ring networks, Fiber Distributed Data Interface (FDDI) networks, Asynchronous Transfer Mode (ATM) networks, frame-relay networks, T1 networks, DS3 networks, bridged networks, routed networks, and point-to-point networks, to name a few. If you're old enough to remember the program Laplink, which allowed you to copy files from one computer to another over a special parallel port cable, you can consider that connection a network as well. It wasn't very scalable (only two computers), or very fast, but it was a means of sending data from one computer to another via a connection.

Connection is an important concept. It's what distinguishes a *sneaker net*, in which information is physically transferred from one computer to another via removable media, from a real network. When you slap a floppy disk into a computer, there is no indication that the files came from another computer—there is no connection. A connection involves some sort of addressing, or identification of the nodes on the network (even if it's just master/slave or primary/secondary).

The machines on a network are often connected physically via cables. However, wireless networks, which are devoid of physical connections, are connected through the use of radios. Each node on a wireless network has an address. Frames received on the wireless network have a specific source and destination, as with any network.

Networks are often distinguished by their reach. LANs, WANs, MANs, and CANs are all examples of network types defined by their areas of coverage. LANs are, as their name implies, local to something—usually a single building or floor. WANs

cover broader areas, and are usually used to connect LANs. WANs can span the globe, and there's nothing that says they couldn't go farther. MANs are common in areas where technology like Metropolitan Area Ethernet is possible; they typically connect LANs within a given geographical region such as a city or town. A CAN is similar to a MAN, but is limited to a *campus* (a campus is usually defined as a group of buildings under the control of one entity, such as a college or a single company).

An argument could be made that the terms MAN and CAN can be interchanged, and in some cases, this is true. (Conversely, there are plenty of people out there who would argue that a CAN exists only in certain specific circumstances, and that calling a CAN by any other name is madness.) The difference is usually that in a campus environment, there will probably be conduits to allow direct physical connections between buildings, while running fiber between buildings in a city is generally not possible. Usually, in a city, telecom providers are involved in delivering some sort of technology that allows connectivity through their networks.

MANs and CANs may, in fact, be WANs. The differences are often semantic. If two buildings are in a campus, but are connected via frame relay, are they part of a WAN, or part of a CAN? What if the frame relay is supplied as part of the campus infrastructure, and not through a telecom provider? Does that make a difference? If the campus is in a metropolitan area, can it be called a MAN?

Usually, a network's designers start calling it by a certain description that sticks for the life of the network. If a team of consultants builds a WAN, and refers to it in the documentation as a MAN, the company will probably call it a MAN for the duration of its existence.

Add into all of this the idea that LANs may be connected with a CAN, and CANs may be connected with a WAN, and you can see how confusing it can be, especially to the uninitiated.

The point here is that a lot of terms are thrown around in this industry, and not everyone uses them properly. Additionally, as in this case, the definitions may be nebulous; this, of course, leads to confusion.

You must be careful about the terminology you use. If the CIO calls the network a WAN, but the engineers call the network a CAN, you must either educate whomever is wrong, or opt to communicate with each party using their own language. This issue is more common than you might think. In the case of MAN versus WAN versus CAN, beware of absolutes. In other areas of networking, the terms are more specific.

For our purposes, we will define these network types as follows:

Local Area Network (LAN)
: A LAN is a network that is confined to a limited space, such as a building or floor. It uses short-range technologies such as Ethernet, Token Ring, and the like. A LAN is usually under the control of the company or entity that requires its use.

Wide Area Network (WAN)
: A WAN is a network that is used to connect LANs by way of a third-party provider. An example would be a frame-relay cloud (provided by a telecom provider) connecting corporate offices in New York, Boston, Los Angeles, and San Antonio.

Campus Area Network (CAN)
: A CAN is a network that connects LANs and/or buildings in a discrete area owned or controlled by a single entity. Because that single entity controls the environment, there may be underground conduits between the buildings that allow them to be connected by fiber. Examples include college campuses and industrial parks.

Metropolitan Area Network (MAN)
: A MAN is a network that connects LANs and/or buildings in an area that is often larger than a campus. For example, a MAN might be used to connect a company's various offices within a metropolitan area via the services of a telecom provider. Again, be careful of absolutes. Many companies in Manhattan have buildings or data centers across the river in New Jersey. These New Jersey sites are considered to be in the New York metropolitan area, so they are part of the MAN, even though they are in a different state.

Terminology and language are like any protocol: be careful how you use the terms that you throw around in your daily life, but don't be pedantic to the point of annoying other people by telling them when and how they're wrong. Instead, listen to those around you, and help educate them. A willingness to share knowledge is what separates the average IT person from the good one.

CHAPTER 2

Hubs and Switches

Hubs

In the beginning of Ethernet, 10Base-5 used a very thick cable that was hard to work with (it was nicknamed *thicknet*). 10Base-2, which later replaced 10Base-5, used a much smaller cable, similar to that used for cable TV. Because the cable was much thinner than that used by 10Base-5, 10Base-2 was nicknamed *thin-net*. These cable technologies required large metal couplers called N connectors (10Base-5) and BNC connectors (10Base-2). These networks also required special terminators to be installed at the end of cable runs. When these couplers or terminators were removed, the entire network would stop working. These cables formed the physical backbones for Ethernet networks.

With the introduction of Ethernet running over unshielded twisted pair (UTP) cables terminated with RJ45 connectors, *hubs* became the new backbones in most installations. Many companies attached hubs to their existing thin-net networks to allow greater flexibility as well. Hubs were made to support UTP and BNC 10Base-2 installations, but UTP was so much easier to work with that it became the de facto standard.

A hub is simply a means of connecting Ethernet cables together so that their signals can be repeated to every other connected cable on the hub. Hubs may also be called *repeaters* for this reason, but it is important to understand that while a hub is a repeater, a repeater is not necessarily a hub.

A repeater repeats a signal. Repeaters are usually used to extend a connection to a remote host, or to connect a group of users who exceed the distance limitation of 10Base-T. In other words, if the usable distance of a 10Base-T cable is exceeded, a repeater can be placed inline to increase the usable distance.

I was surprised to learn that there is no specific distance limitation included in the 10Base-T standard. While 10Base-5 and 10Base-2 do include distance limitations (500 meters and 200 meters, respectively), the 10Base-T spec instead describes certain characteristics that a cable should meet. To be safe, I usually try to keep my 10Base-T cables within 100 meters.

Segments are divided by repeaters or hubs. Figure 2-1 shows a repeater extending the distance between a server and a personal computer.

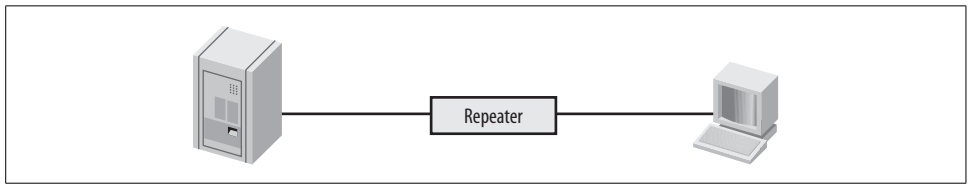


Figure 2-1. Repeater extending a single 10Base-T link

A hub is like a repeater, except that while a repeater may have only two connectors, a hub can have many more; that is, it repeats a signal over many cables as opposed to just one. Figure 2-2 shows a hub connecting several computers to a network.

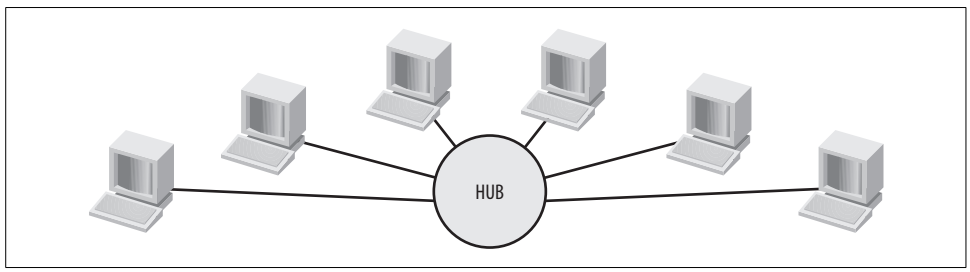


Figure 2-2. Hub connecting multiple hosts to a network

When designing Ethernet networks, repeaters and hubs get treated the same way. The 5-4-3 rule of Ethernet design states that between any two nodes on an Ethernet network, there can be only five segments, connected via four repeaters, and only three of the segments can be populated. This rule, which seems odd in the context of today's networks, was the source of much pain for those who didn't understand it.

As hubs became less expensive, extra hubs were often used as repeaters in more complex networks. Figure 2-3 shows an example of how two remote groups of users could be connected using hubs on each end and a repeater in the middle.

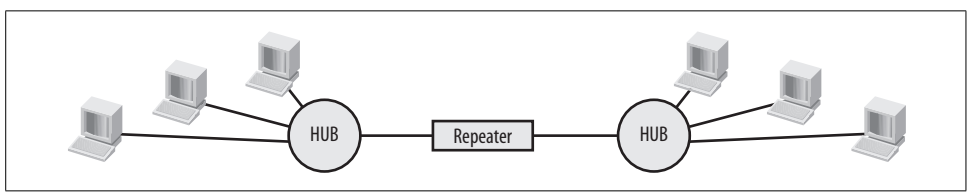


Figure 2-3. Repeater joining hubs

Hubs are very simple devices. Any signal received on any port is repeated out every other port. Hubs are purely physical and electrical devices, and do not have a presence on the network (except possibly for management purposes). They do not alter frames or make decisions based on them in any way.

Figure 2-4 illustrates how hubs operate. As you might imagine, this model can become problematic in larger networks. The traffic can become so intensive that the network becomes saturated—if someone prints a large file, everyone on the network will suffer while the file is transferred to the printer over the network.

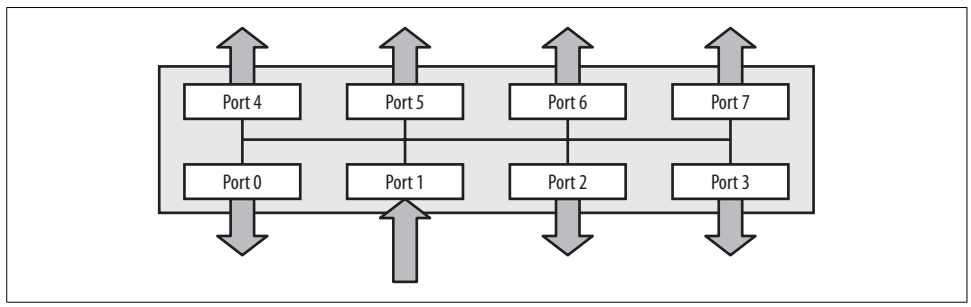


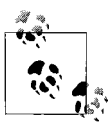
Figure 2-4. Hubs repeat inbound signals to all ports, regardless of type or destination

If another device is already using the wire, the sending device will wait a bit, and then try to transmit again. When two stations transmit at the same time, a *collision* occurs. Each station records the collision, backs off again, and then retransmits. On very busy networks, a lot of collisions will occur.

With a hub, more stations are capable of using the network at any given time. Should all of the stations be active, the network will appear to be slow because of the excessive collisions.

Collisions are limited to network *segments*. An Ethernet network segment is a section of network where devices can communicate using layer-2 MAC addresses. To communicate outside of an Ethernet segment, an additional device, such as a router, is required. Collisions are also limited to *collision domains*. A collision domain is an area of an Ethernet network where collisions can occur. If one station can prevent another from sending because it has the network in use, these stations are in the same collision domain.

A *broadcast domain* is the area of an Ethernet network where a broadcast will be propagated. Broadcasts stay within a layer-3 network (unless forwarded), which is usually bordered by a layer-3 device such as a router. Broadcasts are sent through switches (layer-2 devices), but stop at routers.



Many people mistakenly think that broadcasts are contained within switches or virtual LANs (VLANs). I think this is due to the fact that they are so contained in a properly designed network. If you connect two switches with a crossover cable—one configured with VLAN 10 on all ports, and the other configured with VLAN 20 on all ports—hosts plugged into each switch will be able to communicate if they are on the same IP network. Broadcasts and IP networks are not limited to VLANs, though it is very tempting to think so.

Figure 2-5 shows a network of hubs connected via a central hub. When a frame enters the hub on the bottom left on port 1, the frame is repeated out every other port on that hub, which includes a connection to the central hub. The central hub in turn repeats the frame out every port, propagating it to the remaining hubs in the network. This design replicates the backbone idea, in that every device on the network will receive every frame sent on the network.

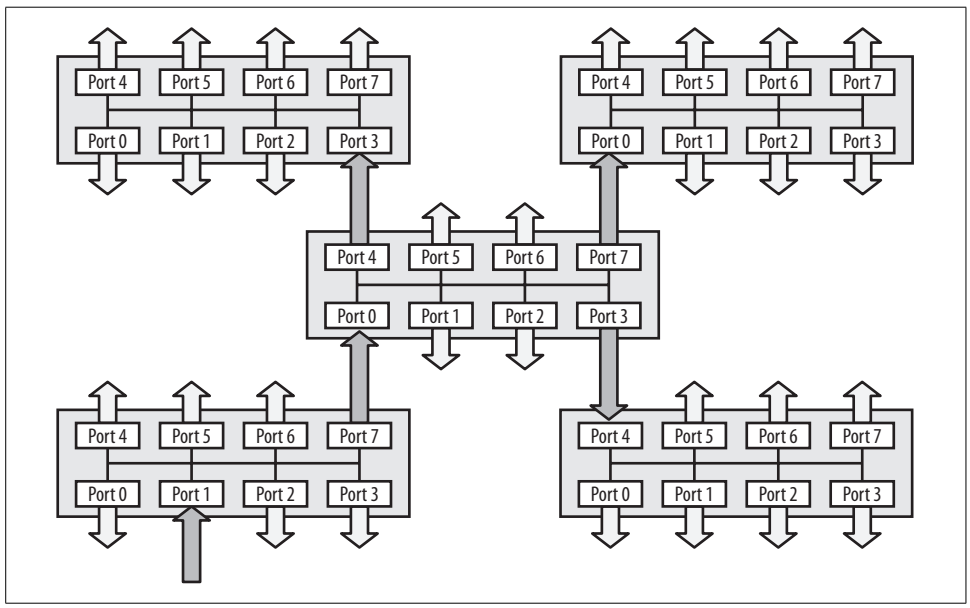


Figure 2-5. Hub-based network

In large networks of this type, new problems can arise. *Late collisions* occur when two stations successfully test for a clear network, and then transmit, only to then encounter a collision. This condition can occur when the network is so large that the propagation of a transmitted frame from one end of the network to the other takes longer than the test used to detect whether the network is clear.

One of the other major problems when using hubs is the possibility of *broadcast storms*. Figure 2-6 shows two hubs connected with two connections. A frame enters the network on Switch 1, and is replicated on every port, which includes the two connections to Switch 2, which now repeats the frame out all of its ports, including the two ports connecting the two switches. Once Switch 1 receives the frame, it again repeats it out every interface, effectively causing an endless loop.

Anyone who's ever lived through a broadcast storm on a live network knows how much fun it can be—especially if you consider your boss screaming at you to be fun. Symptoms include every device essentially being unable to send any frames on the network due to constant network traffic, all status lights on the hubs staying on

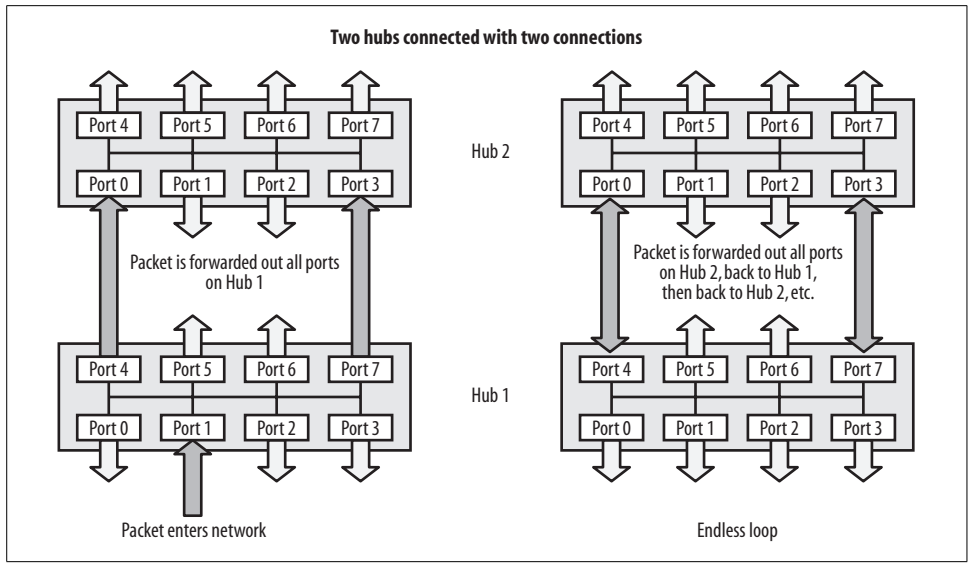


Figure 2-6. Broadcast storm

constantly instead of blinking normally, and (perhaps most importantly) senior executives threatening you with bodily harm.

The only way to resolve a broadcast storm is to break the loop. Shutting down and restarting the network devices will just start the cycle again. Because hubs are not generally manageable, it can be quite a challenge to find a layer-2 loop in a crisis.

Hubs have a lot of drawbacks, and modern networks rarely employ them. Hubs have long since been replaced by *switches*, which offer greater speed, automatic loop detection, and a host of additional features.

Switches

The next step in the evolution of Ethernet after the hub was the switch. Switches differ from hubs in that switches play an active role in how frames are forwarded. Remember that a hub simply repeats every signal it receives via any of its ports out every other port. A switch, in contrast, keeps track of what devices are on what ports, and forwards frames only to the devices for which they are intended.

What we refer to as a *packet* in TCP/IP is called a *frame* when speaking about hubs, bridges, and switches. Technically, they are different things, since a TCP packet is encapsulated with layer-2 information to form a frame. However, the terms "frames" and "packets" are often thrown around interchangeably (I'm guilty of this myself). To be perfectly correct, always refer to frames when speaking of hubs and switches.

When other companies began developing switches, Cisco had all of its energies concentrated in routers, so it did not have a solution that could compete. Hence, Cisco did the smartest thing it could do at the time—it acquired the best of the new switching companies, like Kalpana, and added their devices to the Cisco lineup. As a result, Cisco switches did not have the same operating system that their routers did. While Cisco routers used the Internetwork Operating System (IOS), the Cisco switches sometimes used menus, or an operating system called *CatOS*. (Cisco calls its switch line by the name *Catalyst*; thus, the Catalyst Operating System was CatOS.)

A quick word about terminology is in order. The words "switching" and "switch" have multiple meanings, even in the networking world. There are Ethernet switches, frame-relay switches, layer-3 switches, multilayer switches, and so on. Here are some terms that are in common use:

Switch
: The general term used for anything that can switch, regardless of discipline or what is being switched. In the networking world, a switch is generally an Ethernet switch. In the telecom world, a switch can be many things.

Ethernet switch
: Any device that forwards frames based on their layer-2 MAC addresses using Ethernet. While a hub repeats all frames to all ports, an Ethernet switch forwards frames only to the ports for which they are destined. An Ethernet switch creates a collision domain on each port, while a hub generally expands a collision domain through all ports.

Layer-3 switch
: This is a switch with routing capabilities. Generally, VLANs can be configured as virtual interfaces on a layer-3 switch. True layer-3 switches are rare today; most switches are now multilayer switches.

Multilayer switch
: Same as a layer-3 switch, but may also allow for control based on higher layers in packets. Multilayer switches allow for control based on TCP, UDP, and even details contained within the data payload of a packet.

Switching
: In Ethernet, switching is the act of forwarding frames based on their destination MAC addresses. In telecom, switching is the act of making a connection between two parties. In routing, switching is the process of forwarding packets from one interface to another within a router.

Switches differ from hubs in one very fundamental way: a signal that comes into one port is *not* replicated out every other port on a switch as it is in a hub. While modern switches offer a variety of more advanced features, this is the one that makes a switch a switch.

Figure 2-7 shows a switch with paths between ports four and six, and ports one and seven. The beauty is that frames can be transmitted along these two paths simultaneously, which greatly increases the perceived speed of the network. A dedicated path is created from the source port to the destination port for the duration of each frame's transmission. The other ports on the switch are not involved at all.

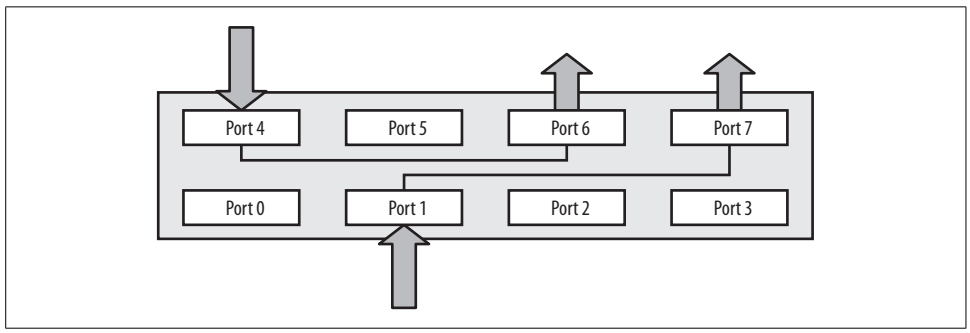


Figure 2-7. A switch forwards frames only to the ports that need to receive them

So, how does the switch determine where to send the frames being transmitted from different stations on the network? Every Ethernet frame contains the source and destination MAC address for the frame. The switch opens the frame (only as far as it needs to), determines the source MAC address, and adds that MAC address to a table if it is not already present. This table, called the *content-addressable memory table* (or CAM table) in CatOS, and the *MAC address table* in IOS, contains a map of what MAC addresses have been discovered on what ports. The switch then determines the frame's destination MAC address, and checks the table for a match. If a match is found, a path is created from the source port to the appropriate destination port. If there is no match, the frame is sent to all ports.

When a station using IP needs to send a packet to another IP address on the same network, it must first determine the MAC address for the destination IP address. To accomplish this, IP sends out an Address Resolution Protocol (ARP) request packet. This packet is a broadcast, so it is sent out all switch ports. The ARP packet, when encapsulated into a frame, now contains the requesting station's MAC address, so the switch knows what port to assign for the source. When the destination station replies that it owns the requested IP address, the switch knows which port the destination MAC address is located on (the reply frame will contain the replying station's MAC address).

Running the `show mac-address-table` command on an IOS-based switch displays the table of MAC addresses and the ports on which they can be found. Multiple MAC addresses on single port usually indicate that the port in question is a connection to another switch or networking device:

```
Switch1-IOS> sho mac-address-table
Legend: * - primary entry
```

```
        age - seconds since last seen
        n/a - not available

  vlan   mac address     type    learn     age              ports
------+----------------+--------+-----+----------+-------------------------
*   24  0013.bace.e5f8   dynamic  Yes        165   Gi3/4
*   24  0013.baed.4881   dynamic  Yes         25   Gi3/4
*   24  0013.baee.8f29   dynamic  Yes         75   Gi3/4
*    4  0013.baeb.ff3b   dynamic  Yes          0   Gi2/41
*   24  0013.baee.8e89   dynamic  Yes        108   Gi3/4
*   18  0013.baeb.01e0   dynamic  Yes          0   Gi4/29
*   24  0013.2019.3477   dynamic  Yes        118   Gi3/4
*   18  0013.bab3.a49f   dynamic  Yes         18   Gi2/39
*   18  0013.baea.7ea0   dynamic  Yes          0   Gi7/8
*   18  0013.bada.61ca   dynamic  Yes          0   Gi4/19
*   18  0013.bada.61a2   dynamic  Yes          0   Gi4/19
*    4  0013.baeb.3993   dynamic  Yes          0   Gi3/33
```

From the preceding output, you can see that should the device with the MAC address `0013.baeb.01e0` wish to talk to the device with the MAC address `0013.baea.7ea0`, the switch will set up a connection between ports Gi4/29 and Gi7/8.

You may notice that I specify the command `show` in my descriptions, and then use the shortened version `sho` while entering commands. Cisco devices allow you to abbreviate commands, so long as the abbreviation cannot be confused with another command.

This information is also useful if you need to figure out where a device is connected to a switch. First, get the MAC address of the device you're looking for. Here's an example from Solaris:

```
[root@unix /]$ ifconfig -a
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
dmfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 172.16.1.9 netmask ffff0000 broadcast 172.16.255.255
        ether 0:13:ba:da:d1:ca
```

Then, take the MAC address (shown on the last line) and include it in the IOS command `show mac-address-table | include` *`mac-address`*:

```
Switch1-IOS> sho mac-address-table | include 0013.bada.d1ca
*   18  0013.bada.61ca   dynamic  Yes          0   Gi3/22
```

Take notice of the format when using MAC addresses, as different systems display MAC addresses differently. You'll need to convert the address to the appropriate format for IOS or CatOS. IOS displays each group of two-byte pairs separated by a period. Solaris and most other operating systems display each octet separated by a colon or hyphen (CatOS uses a hyphen as the delimiter when displaying MAC addresses in hexidecimal). Some systems may also display MAC addresses in decimal, while others use hexadecimal.

The output from the preceding command shows that port Gi3/22 is where our server is connected.

On a switch running CatOS, this is accomplished a little differently because the `show cam` command contains an option to show a specific MAC address:

```
Switch1-CatOS: (enable) sho cam 00-00-13-ba-da-d1-ca
* = Static Entry. + = Permanent Entry. # = System Entry. R = Router Entry.
X = Port Security Entry $ = Dot1x Security Entry

VLAN  Dest MAC/Route Des    [CoS]  Destination Ports or VCs / [Protocol Type]
----  ------------------    -----  -------------------------------------------
20    00-13-ba-da-d1-ca             3/48 [ALL]
Total Matching CAM Entries Displayed  =1
```

Switch Types

Cisco switches can be divided into two types: fixed-configuration and modular switches. Fixed-configuration switches are smaller—usually 1 rack unit (RU) in size. These switches typically contain nothing but Ethernet ports, and are designed for situations where larger switches are unnecessary.

Examples of fixed-configuration switches include the Cisco 2950, 3550, and 3750 switches. The 3750 is capable of being *stacked*. Stacking is a way of connecting multiple switches together to form a single logical switch. This can be useful when more than the maximum number of ports available on a single fixed-configuration switch (48) are needed. The limitation of stacking is that the backplane of the stack is limited to 32 gigabits per second (Gbps). For comparison, some of the larger modular switches can support 720 Gbps on their backplanes. These large modular switches are usually more expensive then a stack of fixed-configuration switches, however.

The benefits of fixed-configuration switches include:

Price
: Fixed-configuration switches are generally much less expensive than their modular cousins.

Size
: Fixed-configuration switches are usually only 1 RU in size. They can be used in closets, and in small spaces where chassis-based switches do not fit. Two switches stacked together are still smaller than the smallest chassis switch.

Weight
: Fixed-configuration switches are lighter than even the smallest chassis switches. A minimum of two people are required to install most chassis-based switches.

Power
: Fixed-configuration switches are all capable of operating on normal household power, and hence can be used almost anywhere. The larger chassis-based switches require special power supplies and AC power receptacles when fully loaded with modules. Many switches are also available with DC power options.

On the other hand, Cisco's larger, modular chassis-based switches have the following advantages over their smaller counterparts:

Expandability
: Larger chassis-based switches can support hundreds of Ethernet ports, and the chassis-based architecture allows the processing modules (supervisors) to be upgraded easily. Supervisors are available for the 6500 chassis that provide 720 Gbps of backplane speed. While you can stack up to seven 3750s for an equal number of ports, remember that the backplane speed of a stack is limited to 32 Gbps.

Flexibility
: The Cisco 6500 chassis will accept modules that provide services outside the range of a normal switch. Such modules include:

  - Firewall Services Modules (FWSMs)
  - Intrusion Detection System Modules (IDSMs)
  - Content Switching Modules (CSMs)
  - Network Analysis Modules (NAMs)
  - WAN modules (FlexWAN)

Redundancy
: Some fixed-configuration switches support a power distribution unit, which can provide some power redundancy at additional cost. However, Cisco's chassis-based switches all support multiple power supplies (older 4000 chassis switches actually required three power supplies for redundancy and even more to support Voice over IP). Most chassis-based switches support dual supervisors as well.

Speed
: The Cisco 6500 chassis employing Supervisor-720 (Sup-720) processors supports up to 720 Gbps of throughput on the backplane. The fastest backplane in a fixed-configuration switch—the Cisco 4948—supports only 48 Gbps. (The 4948 switch is designed to be placed at the top of a rack in order to support the devices in the rack. Due to the specialized nature of this switch, it cannot be stacked, and is therefore limited to 48 ports.)

Chassis-based switches do have some disadvantages. They can be very heavy, take up a lot of room, and require a lot of power. If you need the power and flexibility offered by a chassis-based switch, however, the disadvantages are usually just considered part of the cost of doing business.

Cisco's two primary chassis-based switches are the 4500 series and the 6500 series. There is an 8500 series as well, but these switches are rarely seen in corporate environments.

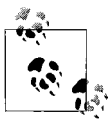
Planning a Chassis-Based Switch Installation

Installing chassis-based switches requires more planning than installing smaller switches. There are many elements to consider when configuring a chassis switch. You must choose the modules (sometimes called *blades*) you will use, and then determine what size power supplies you need. You must decide whether your chassis will use AC or DC power, and what amperage the power supplies will require. Chassis-based switches are large and heavy, so adequate rack space must also be provided. Here are some of the things you need to think about when planning a chassis-based switch installation.

Rack space

Chassis switches can be quite large. The 6513 switch occupies 19 RU of space. The NEBS version of the 6509 takes up 21 RU. A seven-foot telecom rack is 40 RU, so these larger switches use up a significant portion of the available space.

The larger chassis switches are very heavy, and should be installed near the bottom of the rack whenever possible. Smaller chassis switches (such as the 4506, which takes up only 10 RU) can be mounted higher in the rack.



Always use a minimum of two people when lifting heavy switches. Often, a third person can be used to guide the chassis into the rack. The chassis should be moved only after all the modules and power supplies have been removed.

Power

Each module will draw a certain amount of power (measured in watts). When you've determined what modules will be present in your switch, you must add up the power requirements for all the modules. The result will determine what size power supplies you should order. To provide redundancy, each of the power supplies in the pair should be able to provide all the power necessary to run the entire switch, including all modules. If your modules require 3,200 watts in total, you'll need two 4,000-watt power supplies for redundant power. You can use two 3,000-watt power supplies, but they will both be needed to power all the modules. Should one module fail, some modules will be shut down to conserve power.

Depending on where you install your switch, you may need power supplies capable of using either AC or DC power. In the case of DC power supplies, make sure you specify *A and B feeds*. For example, if you need 40 amps of DC power, you'd request *40 amps DC—A and B feeds*. This means that you'll get two 40-amp power circuits for failover purposes. Check the Cisco documentation regarding grounding information. Most collocation facilities supply positive ground DC power.

For AC power supplies, you'll need to specify the voltage, amperage, and socket needed for each feed. Each power supply typically requires a single feed, but some will take two or more. You'll need to know the electrical terminology regarding plugs and receptacles. All of this will be included in the documentation for the power supply, which is available on Cisco's web site. For example, the power cord for a power supply may come with a NEMA L6-20P plug. This will require NEMA L6-20R receptacles. The P and R on the ends of the part numbers describe whether the part is a *plug* or a *receptacle*. (The NEMA L6-20 is a twist-lock 250-volt AC 16-amp connector.)

The power cables will connect to the power supplies via a large rectangular connector. This plug will connect to a receptacle on the power supply, which will be surrounded by a clamp. Always tighten this clamp to avoid the cable popping out of the receptacle when stressed.

Cooling

On many chassis switches, cooling is done from side to side: the air is drawn in on one side, pulled across the modules, then blown out the other side. Usually, rack-mounting the switches allows plenty of airflow. Be careful if you will be placing these switches in cabinets, though. Cables are often run on the sides of the switches, and if there are a lot of them, they can impede the airflow.

The NEBS-compliant 6509 switch moves air vertically, and the modules sit vertically in the chassis. With this switch, the air vents can plainly be seen on the front of the chassis. Take care to keep them clear.

I once worked on a project where we needed to stage six 6506 switches. We pulled them out of their crates, and put them side by side on a series of pallets. We didn't stop to think that the heated exhaust of each switch was blowing directly into the input of the next switch. By the time the air got from the intake of the first switch to the exhaust of the last switch, it was so hot that the last switch shut itself down. Always make sure you leave ample space between chassis switches when installing them.

Installing and removing modules

Modules for chassis-based switches are inserted into small channels on both sides of the slot. Be very careful when inserting modules, as it is very easy to miss the channels and get the modules stuck. Many modules—especially service modules like FWSMs, IDSMs, and CSMs—are densely packed with components. I've seen $40,000 modules ruined by engineers who forced them into slots without properly aligning them. Remember to use a static strap, too.

Any time you're working with a chassis or modules, you should use a static strap. They're easy to use, and come with just about every piece of hardware these days.

Routing cables

When routing cables to modules, remember that you may need to remove the modules in the future. Routing 48 Ethernet cables to each of 7 modules can be a daunting task. Remember to leave enough slack in the cables so that each module's cables can be moved out of the way to slide the module out. When one of your modules fails, you'll need to pull aside all the cables attached to that module, replace the module, then place all the cables back into their correct ports. The more planning you do ahead of time, the easier this task will be.

CHAPTER 3

Auto-Negotiation

When I get called to a client's site to diagnose a network slowdown or a "slow" device, the first things I look at are the error statistics and the auto-negotiation settings on the switches and the devices connected to them. If I had to list the most common problems I've seen during my years in the field, auto-negotiation issues would be in the top five, if not number one.

Why is auto-negotiation such a widespread problem? The truth is, too many people don't really understand what it does and how it works, so they make assumptions that lead to trouble.

What Is Auto-Negotiation?

Auto-negotiation is the feature that allows a port on a switch, router, server, or other device to communicate with the device on the other end of the link to determine the optimal duplex mode and speed for the connection. The driver then dynamically configures the interface to the values determined for the link. Let's examine these parameters:

Speed
: Speed is the rate of the interface, usually listed in megabits per second (Mbps). Common Ethernet speeds include 10 Mbps, 100 Mbps, and 1,000 Mbps. 1,000 Mbps Ethernet is also referred to as *Gigabit Ethernet*.

Duplex
: Duplex refers to how data flows on the interface. On a half-duplex interface, data can only be transmitted or received at any given time. A conversation on a two-way radio is usually half-duplex—each person must push a button to talk, and, while talking, that person cannot listen. A full-duplex interface, on the other hand, can send and receive data simultaneously. A conversation on a telephone is full duplex.

How Auto-Negotiation Works

First, let's cover what auto-negotiation does *not* do: when auto-negotiation is enabled on a port, it does not automatically determine the configuration of the port on the other side of the Ethernet cable and then match it. This is a common misconception that often leads to problems.

Auto-negotiation is a protocol, and as with any protocol, it only works if it's running on both sides of the link. In other words, if one side of a link is running auto-negotiation, and the other side of the link is not, auto-negotiation *cannot* determine the speed and duplex configuration of the other side. If auto-negotiation *is* running on the other side of the link, the two devices decide *together* on the best speed and duplex mode. Each interface advertises the speeds and duplex modes at which it can operate, and the best match is selected (higher speeds and full duplex are preferred).

The confusion exists primarily because auto-negotiation always seems to work. This is because of a feature called *parallel detection*, which kicks in when the auto-negotiation process fails to find auto-negotiation running on the other end of the link. Parallel detection works by sending the signal being received to the local 10Base-T, 100Base-TX, and 100Base-T4 drivers. If any one of these drivers detects the signal, the interface is set to that speed.

Parallel detection determines only the link speed, not the supported duplex modes. This is an important consideration because the common modes of Ethernet have differing levels of duplex support:

10Base-T
: 10Base-T was originally designed without full-duplex support. Some implementations of 10Base-T support full duplex, but most do not.

100Base-T
: 100Base-T has long supported full duplex, which has been the preferred method for connecting 100-Mbps links for as long as the technology has existed. However, the default behavior of 100Base-T is usually half duplex, and it must be set to full duplex, if so desired.

Because of the lack of widespread full-duplex support on 10Base-T, and the typical default behavior of 100Base-T, when auto-negotiation falls through to the parallel detection phase (which only detects speed), the safest thing for the driver to do is to choose half-duplex mode for the link.

When Auto-Negotiation Fails

When auto-negotiation fails on 10/100 links, the most likely cause is that one side of the link has been set to 100/full, and the other side has been set to auto-negotiation. This results in one side being 100/full, and the other side being 100/half.

Figure 3-1 shows a half-duplex link. In a half-duplex environment, the receiving (RX) line is monitored. If a frame is present on the RX link, no frames are sent until the RX line is clear. If a frame is received on the RX line while a frame is being sent on the transmitting (TX) line, a collision occurs. Collisions cause the collision error counter to be incremented—and the sending frame to be retransmitted—after a random back-off delay.

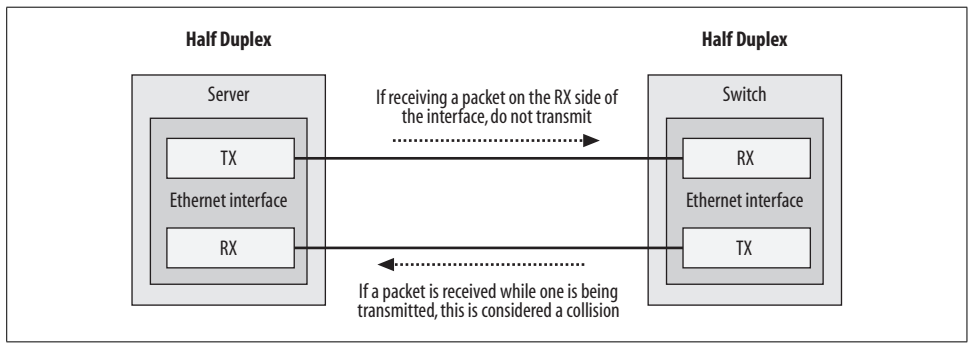


Figure 3-1. Half duplex

Figure 3-2 shows a full-duplex link. In full-duplex operation, the RX line is not monitored, and the TX line is always considered available. Collisions do not occur in full-duplex mode because the RX and TX lines are completely independent.

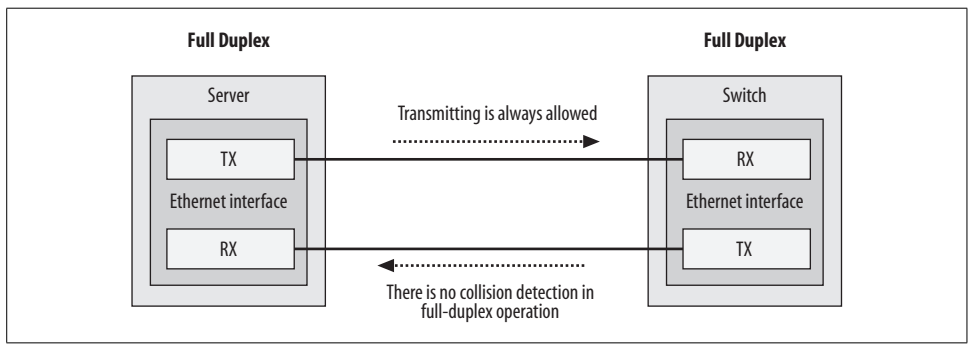


Figure 3-2. Full duplex

When one side of the link is full-duplex, and the other side is half-duplex, a large number of collisions will occur on the half-duplex side. Because the full-duplex side sends frames without checking the RX line, if it's a busy device, chances are it will be sending frames constantly. The other end of the link, being half-duplex, will listen to the RX line, and will not transmit unless the RX line is available. It will have a hard time getting a chance to transmit, and will record a high number of collisions, resulting in the device appearing to be slow on the network. The issue may not be

obvious because a half-duplex interface normally shows collisions. The problem should present itself as excessive collisions.

Figure 3-3 shows a link where auto-negotiation has failed.

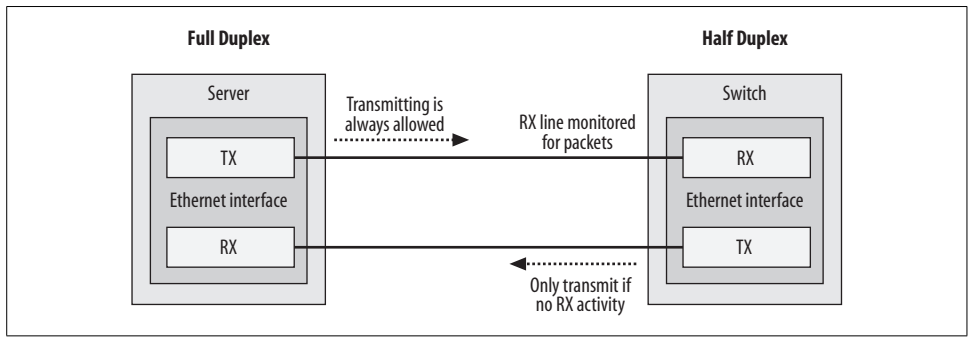


Figure 3-3. Common auto-negotiation failure scenario

In the real world, if you see that an interface that is set to auto-negotiation has negotiated to 100/half, chances are the other side is set to 100/full. 100-Mbps interfaces that do not support full duplex are rare, so properly configured auto-negotiation ports should almost never end up configured for half duplex.

Auto-Negotiation Best Practices

Using auto-negotiation to your advantage is as easy as remembering one simple rule:

> Make sure that both sides of the link are configured the same way.

If one side of the link is set to auto-negotiation, make sure the other side is also set to auto-negotiation. If one side is set to 100/full, make sure the other side is also set to 100/full.

Be careful about using 10/full, as full duplex is not supported on all 10Base-T Ethernet devices.

Gigabit Ethernet uses a substantially more robust auto-negotiation mechanism than the one described in this chapter. Gigabit Ethernet should thus always be set to auto-negotiation, unless there is a compelling reason not to do so (such as an interface that will not properly negotiate). Even then, this should be considered a temporary workaround until the misbehaving part can be replaced.

Configuring Auto-Negotiation

For Cisco switches, auto-negotiation is enabled by default. You can configure the speed and duplex mode manually with the `speed` and `duplex` interface commands in IOS.

You cannot set the duplex mode without first setting the speed. The switch will complain if you attempt to do so:

```
2950(config-if)# duplex half
Duplex can not be set until speed is set to non-auto value
```

To set the speed of the interface, use the `speed` command. If the interface has previously been configured, you can return it to auto-negotiation with the `auto` keyword:

```
2950(config-if)# speed ?
  10    Force 10 Mbps operation
  100   Force 100 Mbps operation
  auto  Enable AUTO speed configuration
```

Once you've set the speed, you can set the duplex mode to `auto`, `full`, or `half`:

```
2950(config-if)# duplex ?
  auto  Enable AUTO duplex configuration
  full  Force full duplex operation
  half  Force half-duplex operation
```

CHAPTER 4

VLANs

Virtual LANs, or VLANs, are virtual separations within a switch that provide distinct logical LANs that each behave as if they were configured on a separate physical switch. Before the introduction of VLANs, one switch could serve only one LAN. VLANs enabled a single switch to serve multiple LANs. Assuming no vulnerabilities exist in the switch's operating system, there is no way for a frame that originates on one VLAN to make its way to another.

Connecting VLANs

Figure 4-1 shows a switch with multiple VLANs. The VLANs have been numbered 10, 20, 30, and 40. In general, VLANs can be named or numbered; Cisco's implementation uses numbers to identify VLANs by default. The default VLAN is numbered 1. If you plug a number of devices into a switch without assigning its ports to specific VLANs, all the devices will be in VLAN 1.

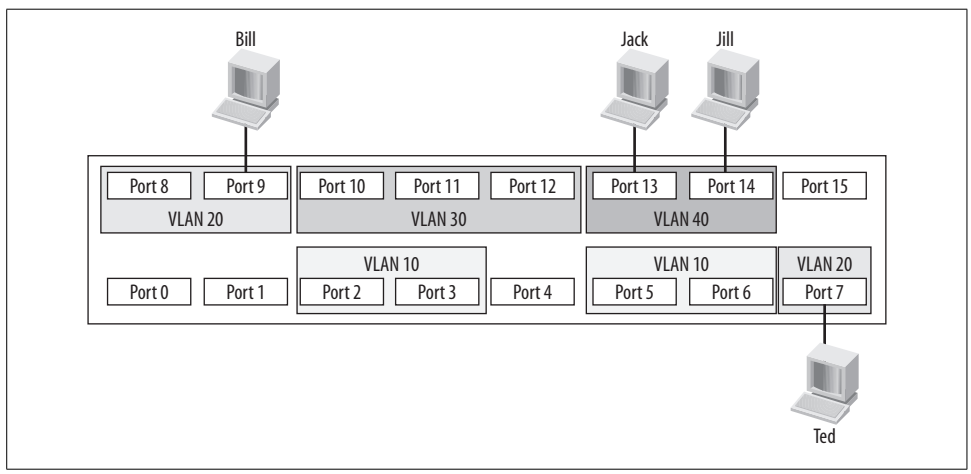


Figure 4-1. VLANs on a switch

Frames cannot leave the VLANs from which they originate. This means that in the example configuration, Jack can communicate with Jill, and Bill can communicate with Ted, but Bill and Ted cannot communicate with Jack or Jill in any way.

For a packet on a layer-2 switch to cross from one VLAN to another, an outside router must be attached to each of the VLANs to be routed. Figure 4-2 shows an external router connecting VLAN 20 with VLAN 40. Assuming a proper configuration on the router, Bill will now be able to communicate with Jill, but neither workstation will show any indication that they reside on the same physical switch.

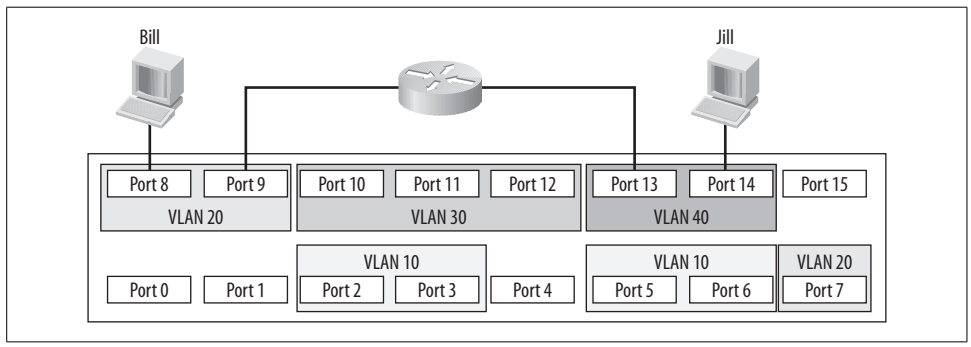


Figure 4-2. External routing between VLANs

When expanding a network using VLANs, the same limitations apply. If you connect another switch to a port that is configured for VLAN 20, the new switch will be able to forward frames only to or from VLAN 20. If you wanted to connect two switches, each containing four VLANs, you would need four links between the switches: one for each VLAN. A solution to this problem is to deploy *trunks* between switches. Trunks are links that carry frames for more than one VLAN.

Figure 4-3 shows two switches connected with a trunk. Jack is connected to VLAN 20 on Switch B, and Diane is connected to VLAN 20 on Switch A. Because there is a trunk connecting these two switches together, assuming the trunk is allowed to carry traffic for all configured VLANs, Jack will be able to communicate with Diane. Notice that the ports to which the trunk is connected are not assigned VLANs. These ports are *trunk ports*, and as such, do not belong to a single VLAN.

Trunks also allow another possibility with switches. Figure 4-2 showed how two VLANs can be connected with a router, as if the VLANs were separate physical networks. Imagine if you wanted to route between *all* of the VLANs on the switch. How would you go about such a design? Traditionally, the answer would be to provide a single connection from the router to each of the networks to be routed. On this switch, each of the networks is a VLAN, so you'd need a physical connection between the router and each VLAN.

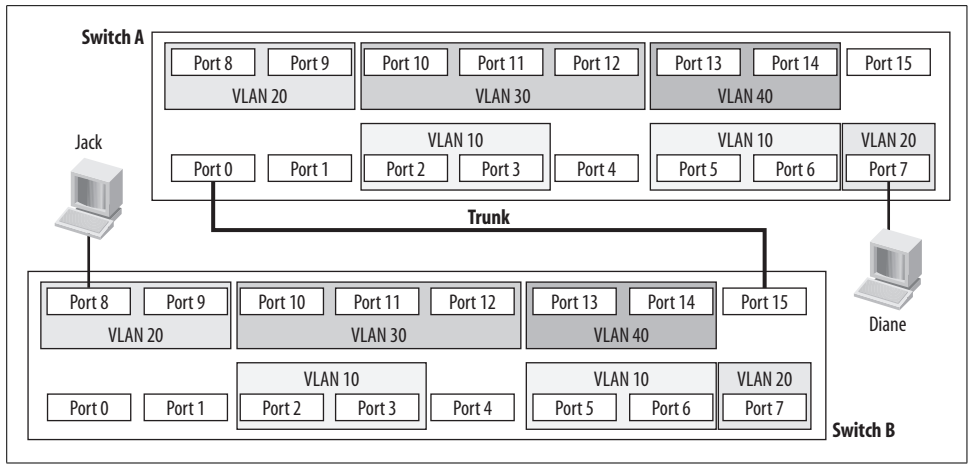


Figure 4-3. Two switches connected with a trunk

As you can see in Figure 4-4, with this setup, four interfaces are being used both on the switch and on the router. Smaller routers rarely have four Ethernet interfaces, though, and Ethernet interfaces on routers can be costly. Additionally, switches are bought with a certain port density in mind. In this configuration, a quarter of the entire switch has been used up just for routing between VLANs.

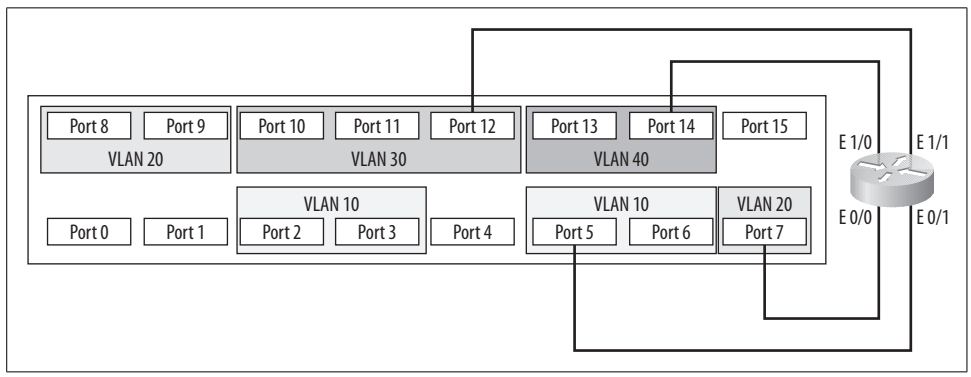


Figure 4-4. Routing between multiple VLANs

Another way to route between VLANs is commonly known as the *router on a stick* configuration. Instead of running a link from each VLAN to a router interface, you can run a single trunk from the switch to the router. All the VLANs will then pass over a single link, as shown in Figure 4-5.

Deploying a router on a stick saves a lot of interfaces on both the switch and the router. The downside is that the trunk is only one link, and the total bandwidth available on that link is only 10 Mbps. In contrast, when each VLAN has its own

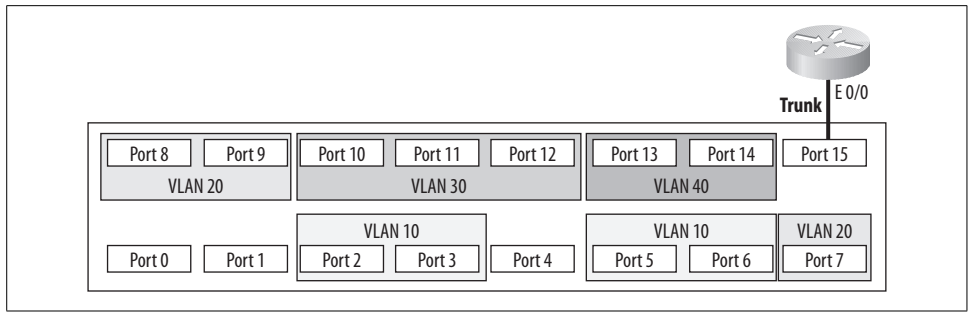


Figure 4-5. Router on a stick

link, each VLAN has 10 Mbps to itself. Also, don't forget that the router is passing traffic between VLANs, so chances are each frame will be seen twice on the same link—once to get to the router, and once to get back to the destination VLAN.

Using a switch with a router is not very common anymore because most vendors offer switches with layer-3 functionality built-in. Figure 4-6 shows conceptually how the same design would be accomplished with a layer-3 switch. Because the switch contains the router, no external links are required. With a layer-3 switch, every port can be dedicated to devices or trunks to other switches.

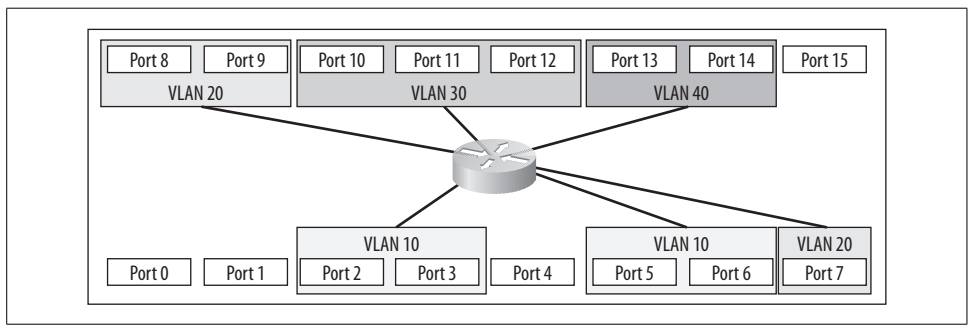


Figure 4-6. Layer-3 switch

Configuring VLANs

VLANs are typically configured via the CatOS or IOS command-line interpreter (CLI), like any other feature. However, some IOS models, such as the 2950 and 3550 switches, have a configurable *VLAN database* with its own configuration mode and commands. This can be a challenge for the uninitiated, especially because the configuration for this database is completely separate from the configuration for the rest of the switch. Even a `write erase` followed by a `reload` will not clear the VLAN database on these switches. Configuring through the VLAN database is a throwback to

older models that offered no other way to manage VLANs. All newer switches (including those with a VLAN database) offer the option of configuring the VLANs through the normal IOS CLI. Switches like the 6500, when running in native IOS mode, only support IOS commands for switch configuration.

Cisco recommends that the VLAN Trunking Protocol (VTP) be configured as a first step when configuring VLANs. This idea has merit, as trunks will not negotiate without a VTP domain. However, setting a VTP domain is not required to make VLANs function on a single switch. Configuring VTP is covered later (see Chapter 5 and Chapter 6).

CatOS

For CatOS, creating a VLAN is accomplished with the `set vlan` command:

```
Switch1-CatOS# (enable) set vlan 10 name Lab-VLAN
VTP advertisements transmitting temporarily stopped,
and will resume after the command finishes.
Vlan 10 configuration successful
```

There are a lot of options when creating a VLAN, but for the bare minimum, this is all that's needed. To show the status of the VLANs, execute the `show vlan` command:

```
Switch1-CatOS# (enable) sho vlan
VLAN Name                             Status    IfIndex Mod/Ports, Vlans
---- -------------------------------- --------- ------- ------------------------
1    default                          active    7       1/1-2
                                                        2/1-2
                                                        3/5-48
                                                        6/1-48
10   Lab-VLAN                         active    112
20   VLAN0020                         active    210     3/1-4
1002 fddi-default                     active    8
1003 token-ring-default               active    11
1004 fddinet-default                  active    9
1005 trnet-default                    active    10
1006 Online Diagnostic Vlan1          active    0       internal
1007 Online Diagnostic Vlan2          active    0       internal
1008 Online Diagnostic Vlan3          active    0       internal
1009 Voice Internal Vlan              active    0       internal
1010 Dtp Vlan                         active    0       internal
1011 Private Vlan Reserved Vlan       suspend   0       internal
1016 Online SP-RP Ping Vlan           active    0       internal
```

Notice that VLAN 10 has the name you assigned; VLAN 20's name, which you did not assign, defaulted to VLAN0020. The output shows which ports are assigned to VLAN 20, and that most of the ports still reside in VLAN 1. (Because VLAN 1 is the default VLAN, all ports reside there by default.)

There are no ports in VLAN 10 yet, so add some, again using the `set vlan` command:

```
Switch1-CatOS# (enable) set vlan 10 6/1,6/3-4
VLAN 10 modified.
VLAN 1 modified.
VLAN  Mod/Ports
---- ----------------------
10  6/1,6/3-4
```

You've now added ports 6/1, 6/3, and 6/4 to VLAN 10. A `show vlan` will reflect these changes:

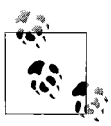
```
Switch1-CatOS# (enable) sho vlan
VLAN Name                             Status    IfIndex Mod/Ports, Vlans
---- -------------------------------- --------- ------- -------------------------
1    default                          active    7       1/1-2
                                                        2/1-2
                                                        3/5-48
                                                        6/2,6/5-48
10   Lab-VLAN                         active    112     6/1,6/3-4
20   VLAN0020                         active    210     3/1-4
1002 fddi-default                     active    8
1003 token-ring-default               active    11
1004 fddinet-default                  active    9
1005 trnet-default                    active    10
1006 Online Diagnostic Vlan1          active    0       internal
1007 Online Diagnostic Vlan2          active    0       internal
1008 Online Diagnostic Vlan3          active    0       internal
1009 Voice Internal Vlan              active    0       internal
1010 Dtp Vlan                         active    0       internal
1011 Private Vlan Reserved Vlan       suspend   0       internal
1016 Online SP-RP Ping Vlan           active    0       internal
```

The output indicates that VLAN 1 was modified as well. This is because the ports had to be removed from VLAN 1 to be added to VLAN 10.

IOS Using VLAN Database

This method is included for the sake of completeness. Older switches that require this method of configuration are no doubt still deployed. Newer switches that support the VLAN database, such as the 3550, actually display this message when you enter VLAN database configuration mode:

```
3550-IOS# vlan database
% Warning: It is recommended to configure VLAN from config mode,
  as VLAN database mode is being deprecated. Please consult user
  documentation for configuring VTP/VLAN in config mode.
```



If you have an IOS switch with active VLANs, but no reference is made to them in the running configuration, it's possible that they were configured in the VLAN database. Another possibility is that they were learned via VTP (we will cover this in Chapter 6).

To configure VLANs in the VLAN database, you must enter VLAN database configuration mode with the command `vlan database`. Requesting help (`?`) lists the commands available in this mode:

```
2950-IOS# vlan database
2950-IOS(vlan)# ?
VLAN database editing buffer manipulation commands:
  abort  Exit mode without applying the changes
  apply  Apply current changes and bump revision number
  exit   Apply changes, bump revision number, and exit mode
  no     Negate a command or set its defaults
  reset  Abandon current changes and reread current database
  show   Show database information
  vlan   Add, delete, or modify values associated with a single VLAN
  vtp    Perform VTP administrative functions.
```

To create a VLAN, give the `vlan` command followed by the VLAN number and name:

```
2950-IOS(vlan)# vlan 10 name Lab-VLAN
VLAN 10 added:
    Name: Lab-VLAN
```

You can show the VLANs configured from within VLAN database mode with the command `show`. You have the option of displaying the current database (`show current`), the differences between the current and proposed database (`show changes`), or the proposed database as it will look after you apply the changes using the `apply` command or exit VLAN database configuration mode. The default behavior of the `show` command is `show proposed`:

```
2950-IOS(vlan)# show
  VLAN ISL Id: 1
    Name: default
    Media Type: Ethernet
    VLAN 802.10 Id: 100001
    State: Operational
    MTU: 1500
    Backup CRF Mode: Disabled
    Remote SPAN VLAN: No

  VLAN ISL Id: 10
    Name: Lab-VLAN
    Media Type: Ethernet
    VLAN 802.10 Id: 100010
    State: Operational
    MTU: 1500
    Backup CRF Mode: Disabled
    Remote SPAN VLAN: No
```

Nothing else is required to create a simple VLAN. The database will be saved upon exit:

```
2950-IOS(vlan)# exit
APPLY completed.
Exiting....
```

Now, when you execute the `show vlan` command in IOS, you'll see the VLAN you've created:

```
2950-IOS# sho vlan

VLAN Name                             Status    Ports
---- -------------------------------- --------- -------------------------------
1    default                          active    Fa0/1, Fa0/2, Fa0/3, Fa0/4
                                                Fa0/5, Fa0/6, Fa0/7, Fa0/8
                                                Fa0/9, Fa0/10, Fa0/11, Fa0/12
                                                Fa0/13, Fa0/14, Fa0/15, Fa0/16
                                                Fa0/17, Fa0/18, Fa0/19, Fa0/20
                                                Fa0/21, Fa0/22, Fa0/23, Fa0/24
                                                Gi0/1, Gi0/2
10   Lab-VLAN                         active
1002 fddi-default                     active
1003 token-ring-default               active
1004 fddinet-default                  active
1005 trnet-default                    active
```

Adding ports to the VLAN is accomplished in IOS interface configuration mode, and is covered in the next section.

IOS Using Global Commands

Adding VLANs in IOS is relatively straightforward when all of the defaults are acceptable, which is usually the case. First, enter configuration mode. From there, issue the `vlan` command with the identifier for the VLAN you're adding or changing. Next, specify a name for the VLAN with the `name` subcommand (as with CatOS, a default name of VLANxxxx is used if you do not supply one):

```
2950-IOS# conf t
Enter configuration commands, one per line. End with CNTL/Z.
2950-IOS(config)# vlan 10
2950-IOS(config-vlan)# name Lab-VLAN
```

Exit configuration mode, then issue the `show vlan` command to see the VLANs present:

```
2950-IOS# sho vlan

VLAN Name                             Status    Ports
---- -------------------------------- --------- -------------------------------
1    default                          active    Fa0/1, Fa0/2, Fa0/3, Fa0/4
                                                Fa0/5, Fa0/6, Fa0/7, Fa0/8
                                                Fa0/9, Fa0/10, Fa0/11, Fa0/12
                                                Fa0/13, Fa0/14, Fa0/15, Fa0/16
                                                Fa0/17, Fa0/18, Fa0/19, Fa0/20
                                                Fa0/21, Fa0/22, Fa0/23, Fa0/24
                                                Gi0/1, Gi0/2
10   Lab-VLAN                         active
1002 fddi-default                     active
1003 token-ring-default               active
1004 fddinet-default                  active
1005 trnet-default                    active
```

Assigning ports to VLANs in IOS is done in interface configuration mode. Each interface must be configured individually with the `switchport access` command (this is in contrast to the CatOS switches, which allow you to add all the ports at once with the `set vlan` command):

```
2950-IOS(config)# int f0/1
2950-IOS(config-if)# switchport access vlan 10
2950-IOS(config-if)# int f0/2
2950-IOS(config-if)# switchport access vlan 10
```

Newer versions of IOS allow commands to be applied to multiple interfaces with the `interface range` command. Using this command, you can accomplish the same result as before while saving some precious keystrokes:

```
2950-IOS (config)# interface range f0/1 - 2
2950-IOS (config-if-range)# switchport access vlan 10
```

Now, when you execute the `show vlan` command, you'll see that the ports have been assigned to the proper VLAN:

```
2950-IOS# sho vlan

VLAN Name                             Status    Ports
---- -------------------------------- --------- -------------------------------
1    default                          active    Fa0/3, Fa0/4, Fa0/5, Fa0/6
                                                Fa0/7, Fa0/8, Fa0/9, Fa0/10
                                                Fa0/11, Fa0/12, Fa0/13, Fa0/14
                                                Fa0/15, Fa0/16, Fa0/17, Fa0/18
                                                Fa0/19, Fa0/20, Fa0/21, Fa0/22
                                                Fa0/23, Fa0/24, Gi0/1, Gi0/2
10   Lab-VLAN                         active    Fa0/1, Fa0/2
1002 fddi-default                     active
1003 token-ring-default               active
1004 fddinet-default                  active
1005 trnet-default                    active
```

CHAPTER 5
Trunking

A *trunk*, using Cisco's terminology, is an interface or link that can carry frames for multiple VLANs at once. As we saw in the previous chapter, a trunk can be used to connect two switches so that devices in VLANs on one switch can communicate with devices in the same VLANs on another switch. Unless there is only one VLAN to be connected, switches are connected at layer two using trunks. Figure 5-1 shows two switches connected with a trunk.

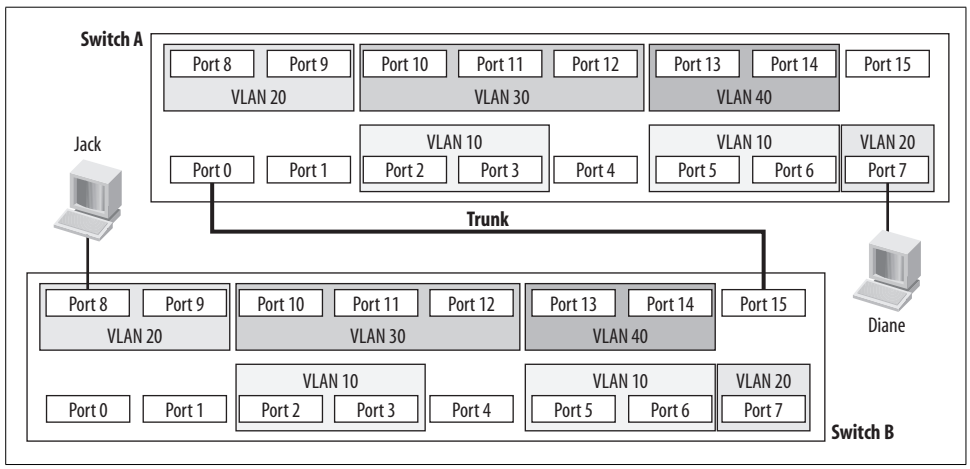


Figure 5-1. A trunk connecting two switches

Trunking is generally related to switches, but a router can connect to a trunk as well. The router on a stick scenario described in Chapter 4 requires a router to communicate with a trunk port on a switch.

How Trunks Work

Figure 5-2 shows a visual representation of a trunk. VLANs 10, 20, 30, and 40 exist on both sides of the trunk. Any traffic from VLAN 10 on Switch-1 that is destined for VLAN 10 on Switch-2 must traverse the trunk. (Of course, the reverse is true as well.)

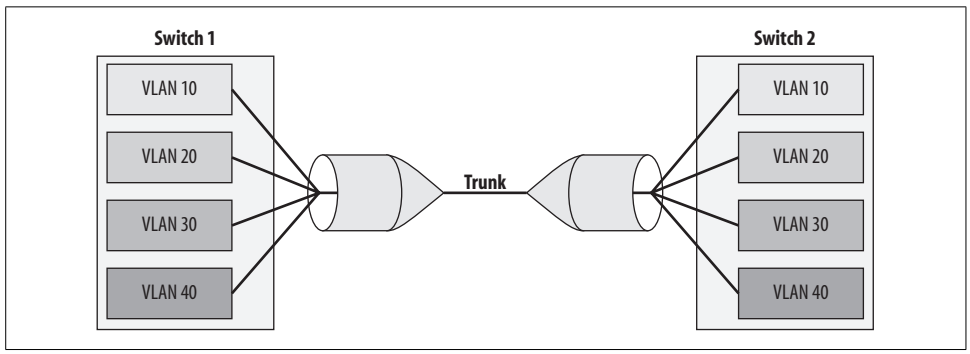


Figure 5-2. Visual representation of a trunk

For the remote switch to know how to forward the frame, the frame must contain a reference to the VLAN to which it belongs. IP packets have no concept of VLANs, though, and nor does TCP, UDP, ICMP, or any other protocol above layer two. Remember that a VLAN is a layer-two concept, so if there were to be any mention of a VLAN, it would happen at the data-link layer. Ethernet was invented before VLANs, so there is no mention of VLANs in any Ethernet protocols, either.

To accomplish the marking or *tagging* of frames to be sent over a trunk, both sides must agree to a protocol. Currently, the protocols for trunking supported on Cisco switches are Cisco's Inter-Switch Link (ISL) and the IEEE standard 802.1Q. Not all Cisco switches support both protocols. For example, the Cisco 2950 and 4000 switches only support 802.1Q. To determine whether a switch can use a specific trunking protocol, use the IOS command `show interface capabilities`, or the CatOS command `show port capabilities`:

```
Switch1-CatOS# sho port capabilities
Model                     WS-X6K-SUP2-2GE
Port                      1/1
Type                      1000BaseSX
Auto MDIX                 no
AuxiliaryVlan             no
Broadcast suppression     percentage(0-100)
Channel                   yes
COPS port group           1/1-2
CoS rewrite               yes
Dot1q-all-tagged          yes
Dot1x                     yes
```

```
Duplex                      full
Fast start                  yes
Flow control                receive-(off,on,desired),send-(off,on,desired)
Inline power                no
Jumbo frames                yes
Link debounce timer         yes
Link debounce timer delay   yes
Membership                  static,dynamic
Port ASIC group             1/1-2
QOS scheduling              rx-(1p1q4t),tx-(1p2q2t)
Security                    yes
SPAN                        source,destination
Speed                       1000
Sync restart delay          yes
ToS rewrite                 DSCP
Trunk encap type            802.1Q,ISL
Trunk mode                  on,off,desirable,auto,nonegotiate
UDLD                        yes
```

ISL differs from 802.1Q in a couple of ways. First, ISL is a Cisco proprietary protocol, whereas 802.1Q is an IEEE standard. Second, ISL encapsulates Ethernet frames within an ISL frame, while 802.1Q alters existing frames to include VLAN tags. Furthermore, ISL is only capable of supporting 1,000 VLANs, while 802.1Q is capable of supporting 4,096.

On switches that support both ISL and 802.1Q, either may be used. The protocol is specific to each trunk. While both sides of the trunk must agree on a protocol, you may configure ISL and 802.1Q trunks on the same switch and in the same network.

ISL

To add VLAN information to a frame to be sent over a trunk, ISL encapsulates the entire frame within a new frame. An additional header is prepended to the frame, and a small suffix is added to the end. Information regarding the VLAN number and some other information is present in the header, while a checksum of the frame is included in the footer. A high-level overview of an ISL frame is shown in Figure 5-3.

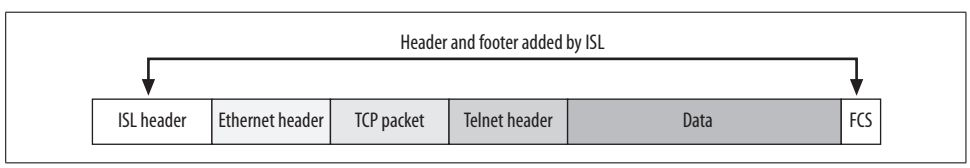


Figure 5-3. ISL encapsulated frame

The frame check sequence (FCS) footer is in addition to the FCS field already present in Ethernet frames. The ISL FCS frame computes a checksum based on the frame including the ISL header; the Ethernet FCS checksum does not include this header.

Adding more information to an Ethernet frame can be problematic. If an Ethernet frame has been created at the maximum size of 1,518 bytes, ISL will add an additional 30 bytes, for a total frame size of 1,548 bytes. These frames may be counted as "giant" frame errors, though Cisco equipment has no problem accepting them.

802.1Q

802.1Q takes a different approach to VLAN tagging. Instead of adding additional headers to a frame, 802.1Q inserts data into existing headers. An additional 4-byte tag field is inserted between the Source Address and Type/Length fields. Because 802.1Q has altered the frame, the FCS of the frame is altered to reflect the change.

Because only 4 bytes are added, the maximum size for an 802.1Q frame is 1522 bytes. This may result in "baby giant" frame errors, though the frames will still be supported on Cisco devices.

Which Protocol to Use

Why are there two protocols at all? There is a history at work here. Cisco developed and implemented ISL before 802.1Q existed. Older switches from Cisco only support ISL. Oddly enough, other switches, like the Catalyst 4000, only support 802.1Q. In some cases, the blade within a switch chassis may be the deciding factor. As an example, the 10-Gb blade available for the Catalyst 6509 only supports 802.1Q, while the switch itself supports 802.1Q and ISL; in this case, 802.1Q must be used.

In many installations, either protocol can be used, and the choice is not important. When trunking between Cisco switches, there is no real benefit of using one protocol over the other, except for the fact that 802.1Q can support 4,096 VLANs, whereas ISL can only support 1,000. Some purists may argue that ISL is better because it doesn't alter the original frame, and some others may argue that 802.1Q is better because the frames are smaller, and there is no encapsulation. What usually ends up happening is that whoever installs the trunk uses whatever protocol she is used to.

Cisco has recommendations on how to set up trunks between Cisco switches. This document (Cisco document ID 24067) is titled "System Requirements to Implement Trunking."

When connecting Cisco switches to non-Cisco devices, the choice is 802.1Q. Remember, there are no restrictions regarding protocol choice on a switch that supports both. If you need to connect a 3Com switch to your Cisco network, you can do so with an 802.1Q trunk even if your Cisco network uses ISL trunks elsewhere. The

trunking protocol is local to each individual trunk. If you connect to a 3Com switch using 802.1Q, the VLANs on that switch will still be accessible on switches connected using ISL elsewhere.

Trunk Negotiation

Some Cisco switches support the Dynamic Trunking Protocol (DTP). This protocol attempts to determine what trunking protocols are supported on each side and to establish a trunk, if possible.

Trunk negotiation includes the VLAN Trunking Protocol (VTP) domain name in the process. For DTP to successfully negotiate, both switches must have the same VTP domain name. See Chapter 6 for details on configuring VTP domains.

An interface running DTP sends frames every 30 seconds in an attempt to negotiate a trunk. If a port has been manually set to either "trunk" or "prevent trunking," DTP is unnecessary, and can be disabled. The IOS command `switchport nonegotiate` disables DTP:

```
SW1(config-if)# switchport mode trunk
SW1(config-if)# switchport nonegotiate
```

Figure 5-4 shows the possible `switchport` modes. Remember, not all switches support all modes. A port can be set to the mode `access`, which means it will never be a trunk; `dynamic`, which means the port may become a trunk; or `trunk`, which means the port will be a trunk regardless of any other settings.

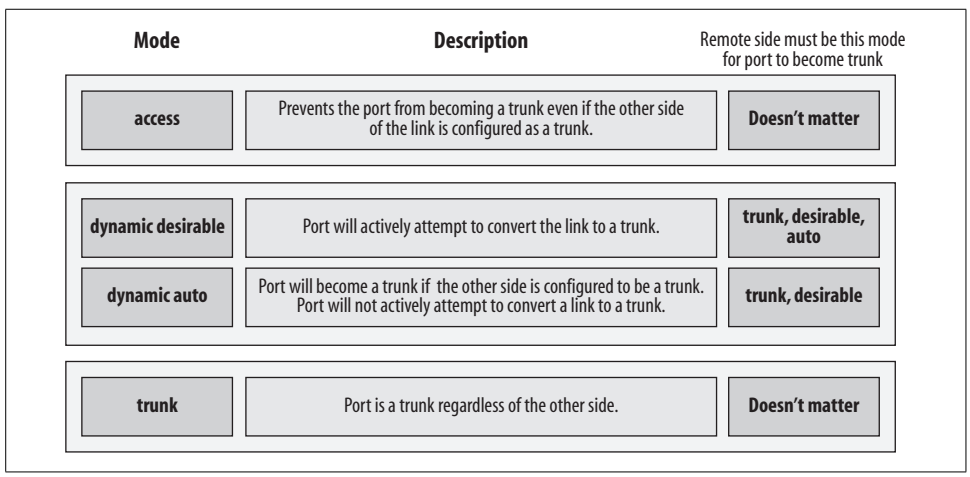


Figure 5-4. Possible switch port modes related to trunking

The two `dynamic` modes, `desirable` and `auto`, refer to the method in which DTP will operate on the port. `desirable` indicates that the port will initiate negotiations and try to make the link a trunk. `auto` indicates that the port will listen for DTP but will not actively attempt to become a port.

The default mode for most Cisco switches is `dynamic auto`. A port in this condition will automatically become a trunk should the remote switch port connecting to it be hardcoded as a trunk or set to `dynamic desirable`.

Configuring Trunks

Configuring a trunk involves determining what port will be a trunk, what protocol the trunk will run, and whether and how the port will negotiate. Optionally, you may also wish to limit what VLANs are allowed on the trunk link.

IOS

The Cisco 3550 is an excellent example of an IOS switch. This section will walk you through configuring one of the Gigabit ports to be an 802.1Q trunk using a 3550 switch.

You might think that the first thing to do would be to specify that the port is a trunk, but as you're about to see, that's not the case:

```
3550-IOS(config-if)# switchport mode trunk
Command rejected: An interface whose trunk encapsulation is "Auto" can not be
configured to "trunk" mode.
```

On an IOS switch capable of both ISL and 802.1Q, you must specify a *trunk encapsulation* before you can configure a port as a trunk. (`trunk encapsulation` is an unfortunate choice for the command because, as you now know, 802.1Q does not encapsulate frames like ISL does. Still, you must follow Cisco's syntax.) Once you've chosen a trunking protocol, you are free to declare the port a trunk:

```
interface GigabitEthernet0/1
 switchport trunk encapsulation dot1q
 switchport mode trunk
```

Should you wish to subsequently remove trunking from the interface, the command to do so is `switchport mode access`.

By default, all VLANs on a switch are included in a trunk. But you may have 40 VLANs, and only need to trunk 3 of them. Because broadcasts from all allowed VLANs will be sent on every trunk port, excluding unneeded VLANs can save a lot

of bandwidth on your trunk links. You can specify which VLANs are able to traverse a trunk with the `switchport trunk allowed` command. These are the options for this command:

```
3550-IOS(config-if)# switchport trunk allowed vlan ?
  WORD    VLAN IDs of the allowed VLANs when this port is in trunking mode
  add     add VLANs to the current list
  all     all VLANs
  except  all VLANs except the following
  none    no VLANs
  remove  remove VLANs from the current list
```

To allow only one VLAN (VLAN 100, in this case) on a trunk, use a command like this:

```
3550-IOS(config-if)# switchport trunk allowed vlan 100
```

As you can see from the output of the `show interface trunk` command, only VLAN 100 is now allowed. IOS has removed the others:

```
3550-IOS# sho int trunk

Port        Mode         Encapsulation  Status        Native vlan
Fa0/15      on           802.1q         trunking      1

Port      Vlans allowed on trunk
Fa0/15      100

Port        Vlans allowed and active in management domain
Fa0/15      none

Port        Vlans in spanning tree forwarding state and not pruned
Fa0/15      none
```

If you wanted to allow all VLANs except VLAN 100, you could do it with the following command:

```
3550-IOS(config-if)# switchport trunk allowed vlan except 100
```

This command will override the previous command specifying VLAN 100 as the only allowed VLAN, so now all VLANs *except* VLAN 100 will be allowed. (Executing the `switchport trunk allowed vlan 100` command again would again reverse the state of the VLANs allowed on the trunk.) `show interface trunk` shows the status:

```
3550-IOS# sho int trunk

Port        Mode         Encapsulation  Status        Native vlan
Fa0/15      on           802.1q         trunking      1

Port      Vlans allowed on trunk
Fa0/15      1-99,101-4094
```

```
Port        Vlans allowed and active in management domain
Fa0/15      1,3-4,10

Port        Vlans in spanning tree forwarding state and not pruned
Fa0/15      1,3-4,10
```

VLANs 1–99 and 101–4096 are now allowed on the trunk. Let's say you want to remove VLANs 200 and 300 as well. Using the `remove` keyword, you can do just that:

```
3550-IOS(config-if)# switchport trunk allowed vlan remove 200
3550-IOS(config-if)# switchport trunk allowed vlan remove 300
```

`show interface trunk` now shows that all VLANs—except 100, 200, and 300—are allowed on the trunk:

```
3550-IOS# sho int trunk

Port        Mode         Encapsulation  Status        Native vlan
Fa0/15      on           802.1q         trunking      1

Port      Vlans allowed on trunk
Fa0/15      1-99,101-199,201-299,301-4094

Port        Vlans allowed and active in management domain
Fa0/15      1,3-4,10

Port        Vlans in spanning tree forwarding state and not pruned
Fa0/15      1,3-4,10
```

CatOS

Configuring a trunk on a CatOS switch is done via the `set trunk` command. Options for the `set trunk` command are as follows:

```
Switch1-CatOS# (enable) set trunk 3/1 ?
  none                      No vlans
  <mode>                    Trunk mode (on,off,desirable,auto,nonegotiate)
  <type>                    Trunk type (isl,dot1q,dot10,lane,negotiate)
  <vlan>                    VLAN number
```

The mode `on` indicates that the port has been hardcoded to be a trunk, and the mode `off` indicates that the port will never be a trunk. The modes `desirable` and `auto` are both dynamic, and refer to the method in which DTP will operate on the port. `desirable` indicates that the port will initiate negotiations, and try to make the link a trunk. `auto` indicates that the port will listen for DTP, but will not actively attempt to become a port. You can use the mode `nonegotiate` to turn off DTP in the event that either `on` or `off` has been chosen as the mode on the opposing port.

The trunk types `isl` and `dotq1` specify ISL and 802.1Q as the protocols, respectively; `negotiate` indicates that DTP should be used to determine the protocol. The trunk types `dot10` and `lane` are for technologies such as ATM, and will not be covered here.

One of the nice features of CatOS is that it allows you to stack multiple arguments in a single command. This command sets the port to mode `desirable`, and the protocol to 802.1Q:

```
Switch1-CatOS# (enable) set trunk 3/5 desirable dot1q
Port(s)  3/1 trunk mode set to desirable.
Port(s)  3/1 trunk type set to dot1q.
Switch1-CatOS# (enable)
2006 May 23 11:29:31 %ETHC-5-PORTFROMSTP:Port 3/5 left bridge port 3/5
2006 May 23 11:29:34 %DTP-5-TRUNKPORTON:Port 3/5 has become dot1q trunk
```

The other side of the link was not configured, but the trunk became active because the default state of the ports on the other side is `auto`.

The command to view trunk status on CatOS is `show port trunk`:

```
Switch1-CatOS# sho port trunk
* - indicates vtp domain mismatch
# - indicates dot1q-all-tagged enabled on the port
$ - indicates non-default dot1q-ethertype value
Port      Mode         Encapsulation  Status        Native vlan
--------  -----------  -------------  ------------  -----------
 3/5      desirable    dot1q          trunking      1
15/1      nonegotiate  isl            trunking      1
16/1      nonegotiate  isl            trunking      1

Port      Vlans allowed on trunk
--------  --------------------------------------------------------------------
 3/5      1-4094
15/1      1-4094
16/1      1-4094

Port      Vlans allowed and active in management domain
--------  --------------------------------------------------------------------
 3/5      1,10,20
15/1
16/1

Port      Vlans in spanning tree forwarding state and not pruned
--------  --------------------------------------------------------------------
 3/5      1,10,20
15/1
16/1
```

The trunks 15/1 and 16/1 shown in this output are internal trunks. On a 6500 switch running CatOS, trunks exist from the supervisors to the multilayer switch feature cards (MSFCs). The MSFCs are known as slot 15 and 16 when two supervisors are installed.

To specify which VLANs can traverse a trunk, use the same `set trunk` command, and append the VLANs you wish to allow. CatOS works a little differently from IOS in that it will not remove all of the active VLANs in favor of ones you specify:

```
Switch-2# (enable) set trunk 3/5 100
Vlan(s) 100 already allowed on the trunk
Please use the 'clear trunk' command to remove vlans from allowed list.
```

Remember that all VLANs are allowed by default. Preventing a single VLAN from using a trunk is as simple as using the `clear trunk` command:

```
Switch-2# (enable) clear trunk 3/5 100
Removing Vlan(s) 100 from allowed list.
Port  3/5 allowed vlans modified to 1-99,101-4094.
```

You don't have to do a `show trunk` command to see what VLANs are allowed, because the `clear trunk` tells you the new status of the port.

To limit a CatOS switch so that only one VLAN is allowed, disallow all the remaining VLANs. Just as you removed one VLAN with the `clear trunk` command, you can remove all of the VLANs *except* the one you want to allow:

```
Switch-2# (enable) clear trunk 3/5 1-99,101-4094
Removing Vlan(s) 1-99,101-4094 from allowed list.
Port  3/5 allowed vlans modified to 100.
```

Finally, a `show trunk` will show you the status of the trunks. As you can see, only VLAN 100 is now allowed on trunk 3/5:

```
Switch-2# (enable) sho trunk
* - indicates vtp domain mismatch
# - indicates dot1q-all-tagged enabled on the port
$ - indicates non-default dot1q-ethertype value
Port      Mode         Encapsulation  Status        Native vlan
--------  -----------  -------------  ------------  -----------
 3/5      auto         dot1q          trunking      1
15/1      nonegotiate  isl            trunking      1
16/1      nonegotiate  isl            trunking      1

Port      Vlans allowed on trunk
--------  --------------------------------------------------------------------
 3/5      100
15/1      1-4094
16/1      1-4094

Port      Vlans allowed and active in management domain
--------  --------------------------------------------------------------------
 3/5
15/1
16/1

Port      Vlans in spanning tree forwarding state and not pruned
--------  --------------------------------------------------------------------
 3/5
15/1
16/1
```

CHAPTER 6

VLAN Trunking Protocol

In complex networks, managing VLANs can be time-consuming and error-prone. The VLAN Trunking Protocol (VTP) is a means whereby VLAN names and numbers can be managed at central devices, with the resulting configuration distributed automatically to other devices. Take for example the network shown in Figure 6-1. This typical three-tier network is composed completely of layer-2 switches. There are 12 switches in all: 2 in the core, 4 in the distribution layer, and 6 in the access layer. (A real network employing this design might have hundreds of switches.)

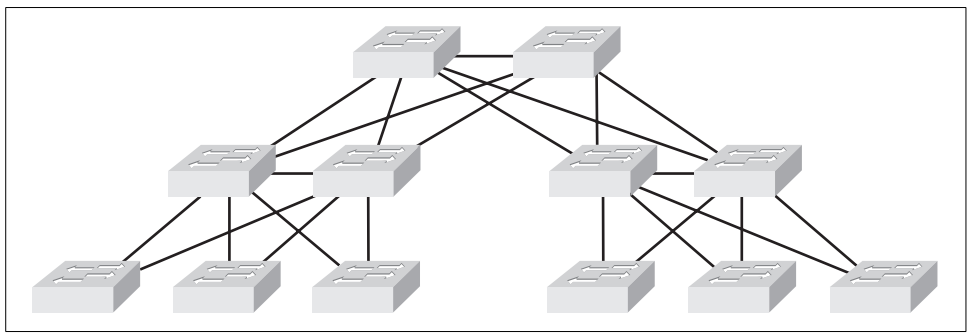


Figure 6-1. Three-tier switched network

Let's assume that the network has 10 VLANs throughout the entire design. That's not so bad, right? Here's what a 10-VLAN configuration might look like on a 2950:

```
vlan 10
 name IT
!
vlan 20
 name Personnel
!
vlan 30
 name Accounting
!
```

```
vlan 40
 name Warehouse1
!
vlan 50
 name Warehouse2
!
vlan 60
 name Shipping
!
vlan 70
 name MainOffice
!
vlan 80
 name Receiving
!
vlan 90
 name Lab
!
vlan 100
 name Production
```

Now, consider that every switch in the design needs to have information about every VLAN. To accomplish this, you'll need to enter these commands, exactly the same each time, into every switch. Sure, you can copy the whole thing into a text file, and paste it into each of the switches, but the process still won't be fun. Look at the VLAN names. There are two warehouses, a lab, a main office—this is a big place! You'll have to haul a laptop and a console cable out to each switch, and the whole process could take quite a while.

Now, add into the equation the possibility that you'll need to add or delete a VLAN at some point, or change the name of one of them. You'll have make the rounds all over again each time there's a change to make.

I can hear you thinking, "But I can just telnet to each switch to make the changes!" Yes, you can, but when you change the VLAN you're connected through without thinking, you'll be back out there working on the consoles—and this time you'll have the foreman threatening you with whatever tool he happened to find along the way because the network's been down since you mucked it up. (Don't worry, things like that almost never happen…more than once.)

While the telnet approach is an option, you need to be very careful about typos. Human error has to be the primary cause of outages worldwide. Fortunately, there's a better way: VTP.

VTP allows VLAN configurations to be managed on a single switch. Those changes are then propagated to every switch in the VTP domain. A *VTP domain* is a group of connected switches with the same VTP domain string configured. Interconnected switches with differently configured VTP domains will not share VLAN information. A switch can only be in one VTP domain; the VTP domain is null by default.

Switches with mismatched VTP domains will not negotiate trunk protocols. If you wish to establish a trunk between switches with mismatched VTP domains, you must have their trunk ports set to `mode trunk`. See Chapter 5 for more information.

The main idea of VTP is that changes are made on *VTP servers*. These changes are then propagated to *VTP clients*, and any other VTP servers in the domain. Switches can be configured manually as VTP servers, VTP clients, or the third possibility, *VTP transparent*. A VTP transparent switch receives and forwards VTP updates, but does not update its configuration to reflect the changes they contain. Some switches default to VTP server, while others default to VTP transparent. VLANs cannot be locally configured on a switch in client mode.

Figure 6-2 shows a simple network with four switches. SW1 and SW2 are both VTP servers. SW3 is set to VTP transparent, and SW4 is a VTP client. Any changes to the VLAN information on SW1 will be propagated to SW2 and SW4. The changes will be passed through SW3, but will not be acted upon by that switch. Because the switch does not act on VTP updates, its VLANs must be configured manually if users on that switch are to interact with the rest of the network.

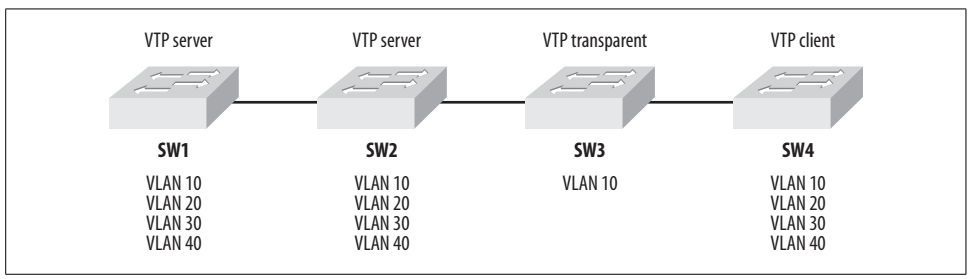


Figure 6-2. VTP modes in action

Looking at Figure 6-2, it is important to understand that both VTP servers can accept and disseminate VLAN information. This leads to an interesting problem. If someone makes a change on SW1, and someone else simultaneously makes a change on SW2, which one wins?

Every time a change is made on a VTP server, the configuration is considered *revised*, and the configuration revision number is incremented by one. When changes are made, the server sends out VTP updates (called *summary advertisements*) containing the revision numbers. The summary advertisements are followed by *subset advertisements*, which contain specific VLAN information.

When a switch receives a VTP update, the first thing it does is compare the VTP domain name in the update to its own. If the domains are different, the update is ignored. If they are the same, the switch compares the update's configuration revision

number to its own. If the revision number of the update is lower than or equal to the switch's own revision number, the update is ignored. If the update has a higher revision number, the switch sends an *advertisement request*. The response to this request is another summary advertisement, followed by subset advertisements. Once it has received the subset advertisements, the switch has all the information necessary to implement the required changes in the VLAN configuration.

When a switch's VTP domain is null, if it receives a VTP advertisement over a trunk link, it will inherit the VTP domain and VLAN configuration from the switch on the other end of the trunk. This will happen only over manually configured trunks, as DTP negotiations cannot take place unless a VTP domain is configured.

Switches also send advertisement requests when they are reset, and when their VTP domains are changed.

To answer the question posed earlier, assuming that both SW1 and SW2 started with the same configuration revision number, whichever switch submits the change first will "win," and have its change propagated throughout the domain, as it will be the first switch to advertise a higher configuration revision number. The changes made on the other switch will be lost, having effectively been overwritten. There will be no indication that these changes were lost or even made.

VTP Pruning

On large or congested networks, VTP can create a problem when excess traffic is sent across trunks needlessly. Take, for example, the network shown in Figure 6-3. The switches in the gray box all have ports assigned to VLAN 100, while the rest of the switches do not. With VTP active, all of the switches will have VLAN 100 configured, and as such will receive broadcasts initiated on that VLAN. However, those without ports assigned to VLAN 100 have no use for the broadcasts.

On a busy VLAN, broadcasts can amount to a significant percentage of traffic. In this case, all that traffic is being needlessly sent over the entire network, and is taking up valuable bandwidth on the inter-switch trunks.

VTP pruning prevents traffic originating from a particular VLAN from being sent to switches on which that VLAN is not *active* (i.e., switches that do not have ports connected and configured for that VLAN). With VTP pruning enabled, the VLAN 100 broadcasts will be restricted to switches on which VLAN 100 is actively in use, as shown in Figure 6-4.

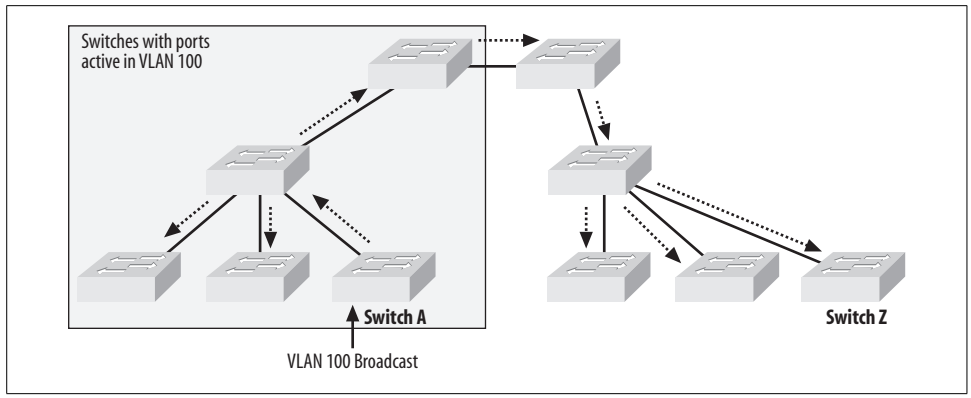


Figure 6-3. Broadcast sent to all switches in VTP domain

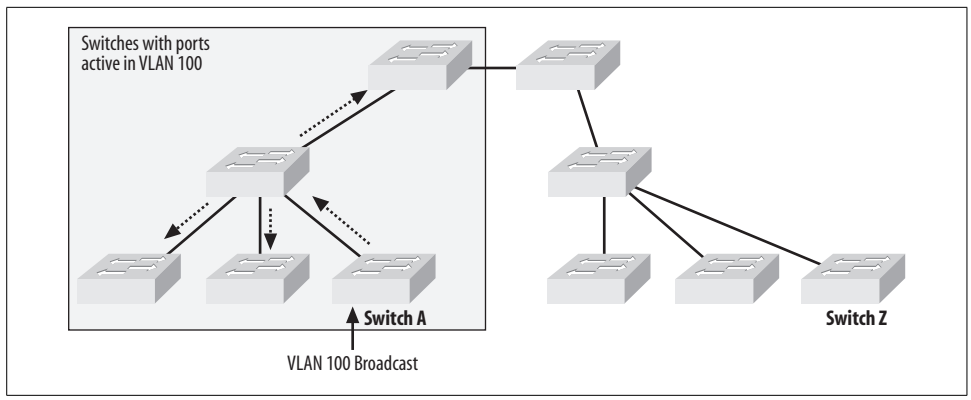


Figure 6-4. VTP pruning limits traffic to switches with active ports in VLANs

Cisco documentation states that pruning is not designed to work with switches in VTP transparent mode.

Dangers of VTP

VTP offers a lot of advantages, but it can have some pretty serious drawbacks, too, if you're not careful.

Imagine a network in an active office. The office is in Manhattan, and spans 12 floors in a skyscraper. There are a pair of 6509s in the core, a pair of 4507Rs on each floor in the distribution layer, and 3550 access-layer switches throughout the environment.

The total number of switches is close to 100. VTP is deployed with the core 6509s being the only VTP servers. The rest of the switches are all configured as VTP clients. All in all, this is a pretty well-built scenario very similar to the one shown in Figure 6-1 (but on a grander scale).

Now, say that somewhere along the way, someone needed some switches for the lab, and managed to get a couple of 3550s of his own. These 3550s were installed in the lab but were not connected into the network. For months, the 3550s in the lab stayed as a standalone pair, trunked only to each other. The VLAN configuration was changed often, as is usually the case in a lab environment. More importantly, the lab was created as a mirror of the production network, including the same VTP domain.

Then, months after the 3550s were installed, someone else decided that the lab needed to connect to the main network. He successfully created a trunk to one of the distribution-layer 4507R switches. Within a minute, the entire network was down. Remember the angry foreman with the threatening pipe wrench? He's got nothing on a financial institution's CTO on a rampage!

What went wrong? Remember that many switches are VTP servers by default. Remember also that when a switch participating in VTP receives an update that has a higher revision number than its own configuration's revision number, the switch will implement the new scheme. In our scenario, the lab's 3550s had been functioning as a standalone network with the same VTP domain as the regular network. Multiple changes were made to their VLAN configurations, resulting in a high configuration revision number. When these switches, which were VTP servers, were connected to the more stable production network, they automatically sent out updates. Each of the switches on the main network, including the core 6509s, received an update with a higher revision number than its current configuration. Consequently, they all requested the VLAN configuration from the rogue 3550s, and implemented that design.

What's especially scary in this scenario is that the people administering the lab network may not even have been involved in the addition of the rogue 3550s to the production network. If they were involved, they might have recognized the new VLAN scheme as coming from the lab. If not, troubleshooting this problem could take some time.

The lesson here is that VTP can be dangerous if it is not managed well. In some cases, such as in smaller networks that are very stable, VTP should not be used. A good example of a network that should not use VTP is an e-commerce web site. Changes to the VLAN design should occur rarely, if ever, so there is little benefit to deploying VTP.

In larger, more dynamic environments where VTP is of use, proper procedures must be followed to ensure that unintended problems do not occur. In the example described above, security measures such as enabling VTP passwords (discussed in

the next section) would probably have prevented the disaster. More importantly, perhaps, connecting rogue switches to a production network should not be allowed without change-control procedures being followed. A good way to prevent the connection of rogue switches is to shut down all switch ports that are not in use. This forces people to request that ports be turned up when they want to connect devices to a switch.

Configuring VTP

To use VTP, you must configure a VTP domain. You'll also need to know how to set the mode on your switches. Further configuration options include setting VTP passwords and configuring VTP pruning.

VTP Domains

The default VTP domain is null. Bear this in mind when implementing VTP because trunks negotiated using the null VTP domain will break if you assign a different domain to one side.

This behavior differs from switch to switch. For example, the Catalyst 5000 will not negotiate trunks unless a VTP domain has been set for each switch.

On some switches, such as the Cisco 6500, the null domain will be overwritten if a VTP advertisement is received over a trunk link, and the switch will inherit the VTP domain from the advertisement. (If a VTP domain has been previously configured, this will not occur.)

Note also that once you've changed a switch's VTP domain to something other than null, there is no way to change it back to null short of erasing the configuration and rebooting.

IOS

Setting or changing the VTP domain in IOS is done with the `vtp domain` command:

```
3550-IOS(config)# vtp domain GAD-Lab
Changing VTP domain name from NULL to GAD-Lab
3550-IOS(config)#
1w4d: %DTP-5-DOMAINMISMATCH: Unable to perform trunk negotiation on port Fa0/20
because of VTP domain mismatch.
```

In this case, changing the domain has resulted in a VTP domain mismatch that will prevent trunk negotiation from occurring on port Fa0/20.

CatOS

You can set or change the VTP domain on CatOS with the `set vtp domain` command:

```
Switch1-CatOS# (enable) set vtp domain GAD-Lab
VTP domain GAD-Lab modified
```

In this case, I resolved the trunk issue, so no error was reported. Had my change resulted in a VTP domain mismatch, the switch would have alerted me with a similar message to the one reported on the IOS switch.

VTP Mode

Chances are you will need to change the default VTP mode on one or more switches in the VTP domain. When this is the case, you'll need the relevant commands for IOS and CatOS.

IOS

There are three VTP modes on an IOS-based switch: `server`, `client`, and `transparent`. They are set using the `vtp mode` command:

```
3550-IOS(config)# vtp mode ?
  client       Set the device to client mode.
  server       Set the device to server mode.
  transparent  Set the device to transparent mode.
```

Setting a switch to the mode already in use results in an error message:

```
3550-IOS(config)# vtp mode server
Device mode already VTP SERVER.
```

Changing the VTP mode results in a simple message showing your change:

```
3550-IOS(config)# vtp mode transparent
Setting device to VTP TRANSPARENT mode.
```

CatOS

CatOS has an additional mode: `off`. This mode is similar to `transparent` mode, in that advertisements are ignored, but they are not forwarded as they would be using `transparent`. The modes are set using the `set vtp mode` command:

```
Switch1-CatOS# (enable) set vtp mode ?
  client                    VTP client mode
  off                       VTP off
  server                    VTP server mode
  transparent               VTP transparent mode
```

Changing the VTP mode on a CatOS switch results in a status message indicating that the VTP domain has been modified:

```
Switch1-CatOS# (enable) set vtp mode transparent
Changing VTP mode for all features
VTP domain GAD-Lab modified
```

Unlike with IOS, setting the mode to the mode already in use does not result in an error message.

VTP Password

Setting a VTP password ensures that only switches configured with the same VTP password will be affected by VTP advertisements.

IOS

In IOS, you can set a password for VTP with the `vtp password` command:

```
3550-IOS(config)# vtp password MilkBottle
Setting device VLAN database password to MilkBottle
```

There is no option to encrypt the password, but the password is not displayed in the configuration. To show the password, execute the `show vtp password` command from the enable prompt:

```
3550-IOS# sho vtp password
VTP Password: MilkBottle
```

To remove the VTP password, negate the command:

```
3550-IOS(config)# no vtp password
Clearing device VLAN database password.
```

CatOS

Setting the password for VTP in CatOS is done with the `set vtp passwd` command:

```
Switch1-CatOS# (enable) set vtp passwd MilkBottle
Generating the secret associated to the password.
VTP domain GAD-Lab modified
```

To encrypt the password so it cannot be read in the configuration, append the word `hidden` to the command:

```
Switch1-CatOS# (enable) set vtp passwd MilkBottle hidden
Generating the secret associated to the password.
The VTP password will not be shown in the configuration.
VTP domain GAD-Lab modified
```

To clear the password on a CatOS switch, set the password to the number zero:

```
Switch1-CatOS# (enable) set vtp passwd 0
Resetting the password to Default value.
VTP domain GAD-Lab modified
```

VTP Pruning

VTP pruning must be enabled or disabled throughout the entire VTP domain. Failure to configure VTP pruning properly can result in instability in the network.

By default, all VLANs up to VLAN 1001 are eligible for pruning, except VLAN 1, which can never be pruned. The extended VLANs above VLAN 1001 are not supported by VTP, and as such cannot be pruned. CatOS allows the pruning of VLANs 2–1000.

If you enable VTP pruning on a VTP server, VTP pruning will automatically be enabled for the entire domain.

IOS

VTP pruning is enabled with the `vtp pruning` command on IOS:

```
3550-IOS(config)# vtp pruning
Pruning switched on
```

Disabling VTP pruning is done by negating the command (`no vtp pruning`).

To show which VLANs are eligible for pruning on a trunk, execute the `show interface` *`interface-id`* `switchport` command:

```
3550-IOS# sho int f0/15 switchport
Name: Fa0/15
Switchport: Enabled
Administrative Mode: trunk
Operational Mode: trunk
Administrative Trunking Encapsulation: dot1q
Operational Trunking Encapsulation: dot1q
Negotiation of Trunking: On
Access Mode VLAN: 1 (default)
Trunking Native Mode VLAN: 1 (default)
Administrative Native VLAN tagging: enabled
Voice VLAN: none
Administrative private-vlan host-association: none
Administrative private-vlan mapping: none
Administrative private-vlan trunk native VLAN: none
Administrative private-vlan trunk Native VLAN tagging: enabled
Administrative private-vlan trunk encapsulation: dot1q
Administrative private-vlan trunk normal VLANs: none
Administrative private-vlan trunk private VLANs: none
Operational private-vlan: none
Trunking VLANs Enabled: 1-99,101-199,201-299,301-4094
Pruning VLANs Enabled: 2-1001
Capture Mode Disabled
Capture VLANs Allowed: ALL
Protected: false
Unknown unicast blocked: disabled
Unknown multicast blocked: disabled
Appliance trust: none
```

Configuring which VLANs are eligible for pruning is done at the interface level in IOS. The command `switchport trunk pruning vlan` is used on each trunking interface on the switch where pruning is desired:

```
3550-IOS(config-if)# switchport trunk pruning vlan ?
  WORD    VLAN IDs of the allowed VLANs when this port is in trunking mode
```

```
  add     add VLANs to the current list
  except  all VLANs except the following
  none    no VLANs
  remove  remove VLANs from the current list
```

All VLANs are pruning-eligible by default. If you configure VLAN 10 to be eligible for pruning, IOS considers this to mean that *only* VLAN 10 should be eligible:

```
3550-IOS(config-if)# switchport trunk pruning vlan 10
3550-IOS(config-if)#
```

No message is displayed telling you that you have just disabled pruning for VLANs 2–99 and 101–1001. You have to look at the interface again to see:

```
3550-IOS# sho int f0/15 swi
Name: Fa0/15
Switchport: Enabled
Administrative Mode: trunk
Operational Mode: trunk
Administrative Trunking Encapsulation: dot1q
Operational Trunking Encapsulation: dot1q
Negotiation of Trunking: On
Access Mode VLAN: 1 (default)
Trunking Native Mode VLAN: 1 (default)
Administrative Native VLAN tagging: enabled
Voice VLAN: none
Administrative private-vlan host-association: none
Administrative private-vlan mapping: none
Administrative private-vlan trunk native VLAN: none
Administrative private-vlan trunk Native VLAN tagging: enabled
Administrative private-vlan trunk encapsulation: dot1q
Administrative private-vlan trunk normal VLANs: none
Administrative private-vlan trunk private VLANs: none
Operational private-vlan: none
Trunking VLANs Enabled: 1-99,101-199,201-299,301-4094
Pruning VLANs Enabled: 100
Capture Mode Disabled
Capture VLANs Allowed: ALL

Protected: false
Unknown unicast blocked: disabled
Unknown multicast blocked: disabled
Appliance trust: none
```

You can add VLANs to the list of pruning-eligible VLANs with the `add` keyword, and remove them with the `remove` keyword:

```
3550-IOS(config-if)# switchport trunk pruning vlan add 20-30
3550-IOS(config-if)#
3550-IOS(config-if)# switchport trunk pruning vlan remove 30
3550-IOS(config-if)#
```

You can also specify that all VLANs except one or more that you list be made pruning-eligible with the `switchport trunk pruning vlan except` *`vlan-id`* command.

Remember to double-check your work with the `show interface` *`interface-id`* `switchport` command. Adding and removing VLANs can quickly get confusing, especially with IOS managing VTP pruning on an interface basis.

CatOS

CatOS gives you a nice warning about running VTP in the entire domain when you enable VTP pruning. Pruning is enabled with the `set vtp pruning enable` command:

```
Switch1-CatOS# (enable) set vtp pruning enable
This command will enable the pruning function in the entire management domain.
All devices in the management domain should be pruning-capable before enabling.
Do you want to continue (y/n) [n]? y
VTP domain GAD-Lab modified
```

Disabling pruning results in a similar prompt. To disable VTP pruning on CatOS, use the `set vtp pruning disable` command:

```
Switch1-CatOS# (enable) set vtp pruning disable
This command will disable the pruning function in the entire management domain.
Do you want to continue (y/n) [n]? y
VTP domain GAD-Lab modified
```

Once pruning has been enabled, VLANs 2–1000 are eligible for pruning by default. To remove a VLAN from the list of eligible VLANs, use the `clear vtp pruneeligible` command. Unlike IOS, CatOS manages pruning-eligible VLANs on a switch level as opposed to an interface level:

```
Switch1-CatOS# (enable) clear vtp pruneeligible 100
Vlans 1,100,1001-4094 will not be pruned on this device.
VTP domain GAD-Lab modified.
```

To add a VLAN back into the list of VLANs eligible for pruning, use the `set vtp pruneeligible` command:

```
Switch1-CatOS# (enable) set vtp pruneeligible 100
Vlans 2-1000 eligible for pruning on this device.
VTP domain GAD-Lab modified.
```

CHAPTER 7
EtherChannel

EtherChannel is the Cisco term for the technology that enables the bonding of up to eight physical Ethernet links into a single logical link. EtherChannel was originally called *Fast EtherChannel* (FEC), as it was only available on Fast Ethernet at the time.

With EtherChannel, the single logical link's speed is equal to the aggregate of the speeds of all the physical links used. For example, if you were to create an EtherChannel out of four 100-Mbps Ethernet links, the EtherChannel would have a speed of 400 Mbps.

This sounds great, and it is, but the idea is not without problems. For one thing, the bandwidth is not truly the aggregate of the physical link speeds in all situations. For example, on an EtherChannel composed of four 1-Gbps links, each conversation will still be limited to 1 Gbps by default.

The default behavior is to assign one of the physical links to each packet that traverses the EtherChannel, based on the packet's destination MAC address. This means that if one workstation talks to one server over an EtherChannel, only one of the physical links will be used. In fact, all of the traffic destined for that server will traverse a single physical link in the EtherChannel. This means that a single user will only ever get 1 Gbps from the EtherChannel at a time. (This behavior can be changed to send each packet over a different physical link, but as we'll see, there are limits to how well this works for applications like VoIP.) The benefit arises when there are multiple destinations, which can each use a different path.

EtherChannels are referenced by different names on IOS and CatOS devices. As Figure 7-1 shows, on a switch running CatOS, an EtherChannel is called a *channel*, while on a switch running IOS, an EtherChannel is called a *port channel interface*. The command to configure an EtherChannel on CatOS is `set port channel`, and the commands to view channels include `show port channel` and `show channel`. EtherChannels on IOS switches are actually virtual interfaces, and they are referenced like any other interfaces (for example, `interface port-channel 0` or `int po0`).

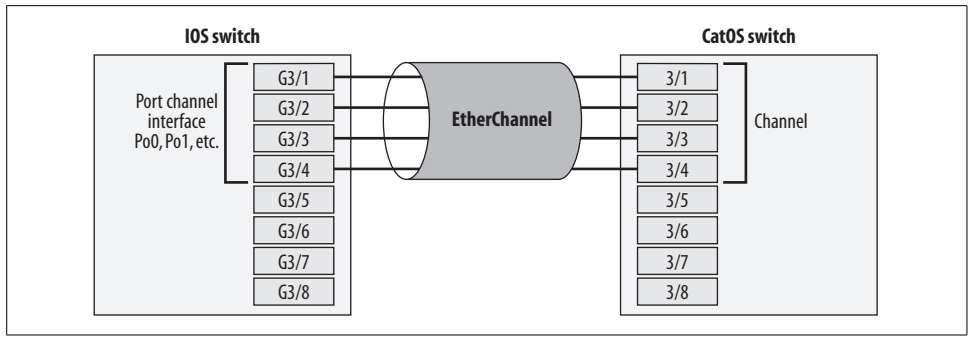


Figure 7-1. EtherChannel on IOS and CatOS

There is another terminology problem that can be a source of many headaches for network administrators. While a group of physical Ethernet links bonded together is called an *EtherChannel* in Cisco parlance, Solaris refers to the same configuration as a *trunk*. Of course, in the Cisco world the term "trunk" refers to something completely different: a link that labels frames with VLAN information so that multiple VLANs can traverse it.

Figure 7-2 shows how Cisco and Solaris label the same link differently. This can cause quite a bit of confusion, and result in some amusing conversations when both sides fail to understand the differences in terminology.

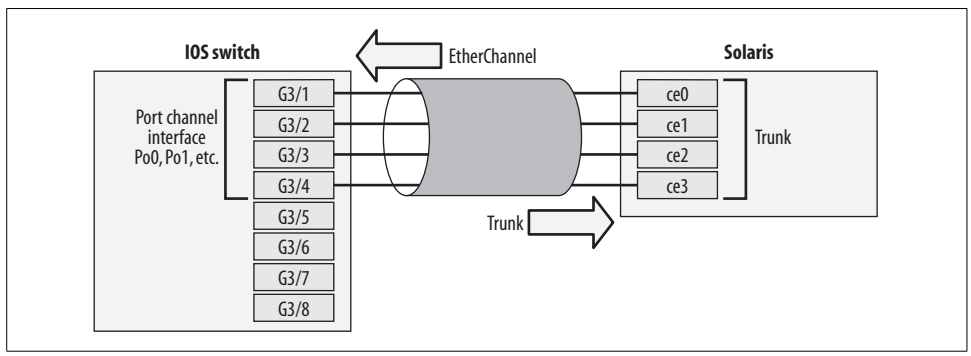


Figure 7-2. Cisco and Solaris terminology regarding EtherChannels

Load Balancing

As stated earlier, EtherChannel by default does not truly provide the aggregate speed of the included physical links. EtherChannel gives the perceived speed of the combined links by passing certain packets through certain physical links. By default, the physical link used for each packet is determined by the packet's destination MAC address. The algorithm used is Cisco-proprietary, but it is deterministic, in that

packets with the same destination MAC address will always travel over the same physical link. This ensures that packets sent to a single destination MAC address never arrive out of order.

The hashing algorithm for determining the physical link to be used may not be public, but the weighting of the links used in the algorithm is published. What is important here is the fact that a perfect balance between the physical links is not necessarily assured.

The hashing algorithm takes the destination MAC address (or another value, as you'll see later), and hashes that value to a number in the range of 0–7. The same range is used regardless of how many links are actually in the EtherChannel. Each physical link is assigned one or more of these eight values, depending on how many links are in the EtherChannel.

Figure 7-3 shows how packets are distributed according to this method. Notice that the distribution is not always even. This is important to understand because link usage statistics—especially graphs—will bear it out.

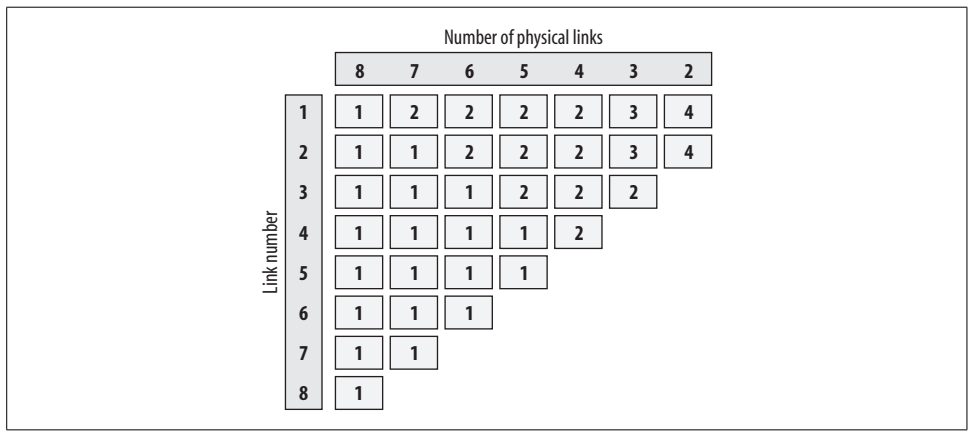


Figure 7-3. EtherChannel physical link balancing

On an EtherChannel with eight physical links, each of the links is assigned a single value. On an EtherChannel with six links, two of the links are assigned two values, and the remaining four links are each assigned one value. This means that two of the links (assuming a theoretical perfect distribution) will receive twice as much traffic as the other four. Having an EtherChannel thus does not imply that all links are used equally. Indeed, it should be obvious looking at Figure 7-3 that the only possible way to distribute traffic equally across all links in an EtherChannel (again, assuming a perfect distribution) is to design one with eight, four, or two physical links. Regardless of the information used to determine the link, the method will still hash the value to a value of 0–7, which will be used to assign a link according to this table.

The method the switch uses to determine which path to assign can be changed. The default behavior is that the destination MAC address is used. However, depending on the version of the software and hardware in use, the options may include:

- The source MAC address
- The destination MAC address
- The source and destination MAC addresses
- The source IP address
- The destination IP address
- The source and destination IP addresses
- The source port
- The destination port
- The source and destination ports

The reasons for changing the default behavior vary by circumstance. Figure 7-4 shows a relatively common layout: a group of users connected to Switch A reach a group of servers on Switch B through an EtherChannel. By default, the load-balancing method will be based on the destination MAC address in each packet. The issue here is one of usage patterns. You might think that with the MAC addresses being unique, the links will be used equally. However, the reality is that it is very common for one server to receive a good deal more traffic than others.

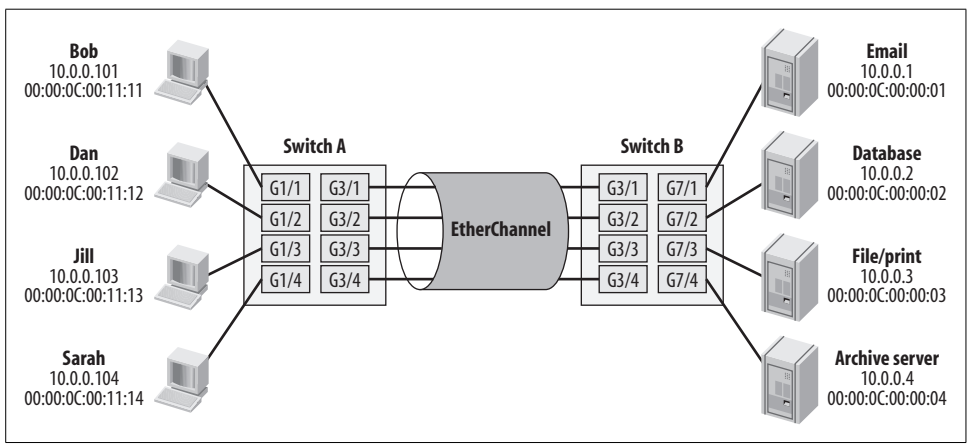


Figure 7-4. EtherChannel load-balancing factors

Let's assume that the email server in this network is receiving more than 1 Gbps of traffic, while the other servers average about 50 Mbps. Using the destination MAC address method will cause packets to be lost on the EtherChannel because every packet destined for the email server's MAC address will ride on the same physical link within the EtherChannel. Overflow does not spill over to the other links—when a physical link becomes saturated, packets are dropped.

In the case of one server receiving the lion's share of the traffic, destination MAC address load balancing does not make sense. Given this scenario, balancing with the source MAC address might make more sense.

Another important idea to remember is that the load-balancing method is only applied to packets being *transmitted* over the EtherChannel. This is not a two-way function. While changing the method to source MAC address on Switch A might be a good idea, it would be a terrible idea on Switch B, given that the email server is the most-used server. Remember, when packets are being returned from the email server, the source MAC address is that of the email server itself. So, if we use the source MAC address to determine load balancing on Switch B, we'll end up with the same problem we were trying to solve.

In this circumstance, the solution would be to have source MAC address load balancing on Switch A, and destination MAC address load balancing on Switch B. If all your servers are on one switch, and all your users are on another, as in this example, this solution will work. Unfortunately, the real world seldom provides such simple problems. A far more common scenario is that all of the devices are connected in one large switch, such as a 6509. Changing the load-balancing algorithm is done on a chassis-wide basis, so with all the devices connected to a single switch, you're out of luck.

Figure 7-5 shows an interesting problem. Here we have a single server connected to Switch A via an EtherChannel, and a single network attached storage (NAS) device that is also attached to Switch A via an EtherChannel. All of the filesystems for the server are mounted on the NAS device, and the server is heavily used—it's a database server that serves more than 5,000 users at any given time. The bandwidth required between the server and the NAS device is in excess of 2 Gbps.

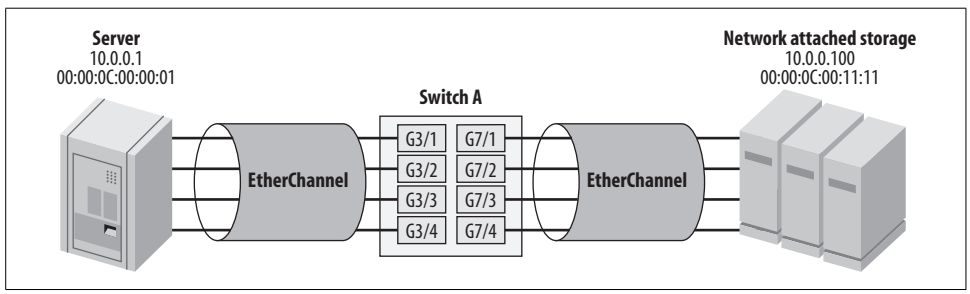


Figure 7-5. Single server to single NAS

Unfortunately, there is no easy solution to this problem. We can't use the destination MAC address or the source MAC address for load balancing because in each case there is only one address, and it will always be the same. We can't use a combination of source and destination MAC addresses, or the source and/or destination IP addresses, for the same reason. And we can't use the source or destination port numbers, because once they're negotiated, they don't change. One possibility, assuming

the drivers support it, is to change the server and/or NAS device so that each link has its own MAC address, but the packets will still be sourced from and destined for only one of those addresses.

The only solutions for this problem are manual load balancing or faster links. Splitting the link into four 1 Gbps links, each with its own IP network, and mounting different filesystems on each link will solve the problem. However, that's too complicated for my tastes. A better solution, if available, might be to use a faster physical link, such as 10 Gbps Ethernet.

Configuring and Managing EtherChannel

The device on the other end of the EtherChannel is usually the determining factor in how the EtherChannel is configured. One design rule that must always be applied is that each of the links participating in an EtherChannel must have the same configuration. The descriptions can be different, but each of the physical links must be the same type and speed, and they must all be in the same VLAN. If they are trunks, they must all be configured with the same trunk parameters.

EtherChannel Protocols

EtherChannel will negotiate with the device on the other side of the link. Two protocols are supported on Cisco devices. The first is the Link Aggregation Control Protocol (LACP), which is defined in IEEE specification 802.3ad. LACP is used when connecting to non-Cisco devices, such as servers. As an example, Solaris will negotiate with a Cisco switch via LACP. The other protocol used in negotiating EtherChannel links is the Port Aggregation Control Protocol (PAgP), which is a Cisco-proprietary protocol. Since PAgP is Cisco-proprietary, it is used only when connecting two Cisco devices via an EtherChannel. Each protocol supports two modes: a passive mode (`auto` in PAgP and `passive` in LACP), and an active mode (`desirable` in PAgP and `active` in LACP). Alternatively, you can set the mode to `on`, thus forcing the creation of the EtherChannel. The available protocols and modes are outlined in Figure 7-6.

Generally, when you are configuring EtherChannels between Cisco switches, the ports will be EtherChannels for the life of the installation. Setting all interfaces in the EtherChannel on both sides to `desirable` makes sense. When connecting a Cisco switch to a non-Cisco device such as a Solaris machine, use the `active` LACP setting. Also be aware that some devices use their own channeling methods, and require the Cisco side of the EtherChannel to be set to `on` because they don't negotiate with the other sides of the links. NetApp NAS devices fall into this category.

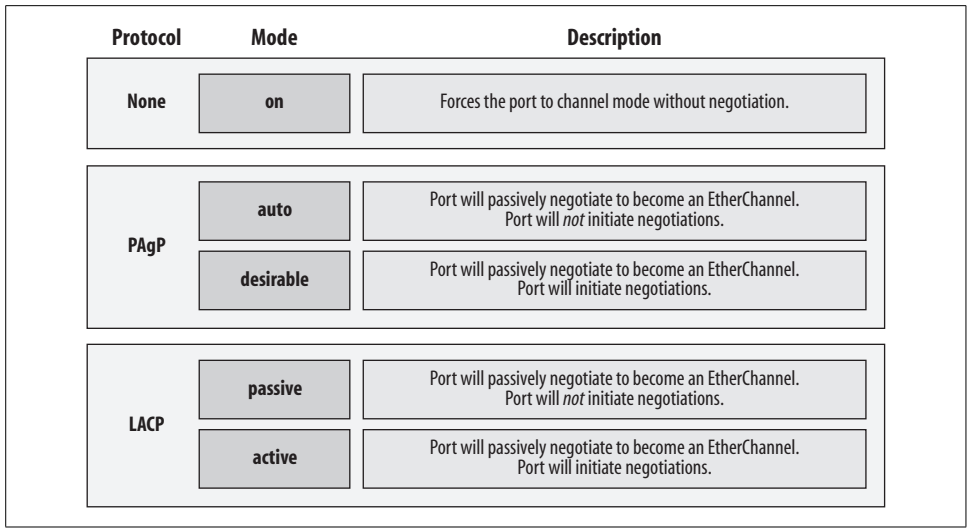


Figure 7-6. EtherChannel protocols and their modes

CatOS Example

Creating an EtherChannel in CatOS is relatively straightforward, once you know what you need. As an example, we'll create the EtherChannel shown in Figure 7-1. Because the devices on both sides are Cisco switches, we will configure the ports on both sides to be `desirable` (running PAgP and initiating negotiation):

```
set port name       3/1  Link #1 in Channel
set port name       3/2  Link #2 in Channel
set port name       3/3  Link #3 in Channel
set port name       3/4  Link #4 in Channel

set vlan 20    3/1-4

set port channel 3/1-4 mode desirable
```

Assuming the other side is set properly, this is all we need to get an EtherChannel working. The names are not necessary, of course, but everything should be labeled and documented regardless of perceived need.

Now that we've configured an EtherChannel, we need to be able to check on its status. First, let's look at the output of `show port channel`:

```
Switch-2-CatOS: (enable) sho port channel
Port  Status     Channel              Admin Ch
                 Mode                 Group Id
----- ---------- -------------------- ----- -----
 3/1 connected  desirable               74   770
 3/2 connected  desirable               74   770
 3/3 connected  desirable               74   770
 3/4 connected  desirable               74   770
```

The `show channel info` command shows a very similar output, but contains even more information. In most cases, this command is far more useful, as it shows the channel ID, admin group, interface type, duplex mode, and VLAN assigned all in one display:

```
Switch-2-CatOS: (enable) sho channel info
Chan Port  Status     Channel              Admin Speed Duplex Vlan
id                   mode                 group
---- ----- ---------- -------------------- ----- ----- ------ ----
 770  3/1  connected  desirable               74 a-1Gb a-full   20
 770  3/2  connected  desirable               74 a-1Gb a-full   20
 770  3/3  connected  desirable               74 a-1Gb a-full   20
 770  3/4  connected  desirable               74 a-1Gb a-full   20
```

The `show channel` command shows a very brief output of what ports are assigned to what channels:

```
Switch-2-CatOS: (enable) sho channel
Channel Id   Ports
-----------  ------------------------------------------------
770          3/1-4
```

`show channel traffic` is another very useful command. This command shows how the links have been used, with the traffic distribution reported as actual percentages:

```
Switch-2-CatOS: (enable) sho channel traffic
ChanId Port  Rx-Ucst Tx-Ucst Rx-Mcst Tx-Mcst Rx-Bcst Tx-Bcst
------ ----- ------- ------- ------- ------- ------- -------
   770  3/1   21.80%  18.44%  87.48%  87.70%  26.49%  21.20%
   770  3/2   34.49%  37.97%   4.02%   4.98%  19.38%  11.73%
   770  3/3   21.01%  23.47%   3.99%   3.81%  29.46%  28.60%
   770  3/4   22.66%  20.06%   4.13%   2.79%  23.69%  38.32%
```

Note that the percentages do not always add up to 100 percent. This tool is not about specifics, but rather, trends.

IOS Example

Configuring EtherChannels on an IOS-based switch is not difficult, although, as discussed earlier, if you're used to CatOS switches, the terminology may seem a bit odd. The major difference is that a port-channel virtual interface is created. This actually gives you a lot of leeway: you can configure this interface with an IP address if you wish, or just leave it as a normal switch port. Remember that each interface must be configured with identical settings, with the exception of the description. I like to configure meaningful descriptions on all my physical ports. This helps keep me track of how the interfaces are assigned, as the `show interface` command does not indicate whether an interface is a member of an EtherChannel.

Again, we'll design the EtherChannel shown in Figure 7-1 as an example, so there are Cisco switches on both sides of the links:

```
interface Port-channel1
 description 4G Etherchannel Po1
 no ip address
 switchport
 switchport access vlan 20

interface GigabitEthernet3/1
 description Link #1 in Po1
 no ip address
 switchport
 channel-group 1 mode desirable

interface GigabitEthernet3/2
 description Link #2 in Po1
 no ip address
 switchport
 channel-group 1 mode desirable

interface GigabitEthernet3/3
 description Link #3 in Po1
 no ip address
 switchport
 channel-group 1 mode desirable

interface GigabitEthernet3/4
 description Link #4 in Po1
 no ip address
 switchport
 channel-group 1 mode desirable
```

On IOS switches, the quick way to see the status of an EtherChannel is to use the `show etherchannel summary` command. CatOS users may be frustrated by the complexity of the output. First, you must figure out the codes, as outlined in the included legend; then, you can determine the status of your EtherChannel. In this example, the EtherChannel is Layer2 and is in use (`Po1(SU)`). The individual physical links are all active, as they have (`P`) next to their port numbers:

```
Switch-1-IOS# sho etherchannel summary
Flags:  D - down        P - in port-channel
        I - stand-alone s - suspended
        H - Hot-standby (LACP only)
        R - Layer3      S - Layer2
        U - in use      f - failed to allocate aggregator

        u - unsuitable for bundling
Number of channel-groups in use: 1
Number of aggregators:           1

Group  Port-channel  Protocol    Ports
------+-------------+-----------+-----------------------------------------------
1      Po1(SU)         PAgP      Gi3/1(P)   Gi3/2(P)    Gi3/3(P)     Gi3/4(P)
```

A more useful command, though missing the real status of the interfaces, is the `show etherchannel` command. This command is interesting, in that it shows the number of bits used in the hash algorithm for each physical interface, as shown previously in Figure 7-3. Also of interest in this command's output is the fact that it shows the last time at which an interface joined the EtherChannel:

```
Switch-1-IOS # sho etherchannel 1 port-channel
               Port-channels in the group:
               ----------------------

Port-channel: Po1
------------

Age of the Port-channel   = 1d:09h:22m:37s
Logical slot/port   = 14/6            Number of ports = 4
GC                  = 0x00580001      HotStandBy port = null
Port state          = Port-channel Ag-Inuse
Protocol            =   PAgP

Ports in the Port-channel:

Index   Load   Port     EC state        No of bits
------+------+------+------------------+-----------
  1     11     Gi3/1    Desirable-Sl    2
  2     22     Gi3/2    Desirable-Sl    2
  0     44     Gi3/3    Desirable-Sl    2
  3     88     Gi3/4    Desirable-Sl    2

Time since last port bundled:    1d:09h:21m:08s    Gi3/4
```

Because EtherChannels are assigned virtual interfaces on IOS, you can show the interface information as if it were a physical or virtual interface. Notice that the bandwidth is set to the aggregate speed of the links in use, but the duplex line shows the interface as `Full-duplex, 1000Mb/s`. The hardware is listed as EtherChannel, and there is a line in the output that shows the members of this EtherChannel to be Gi3/1, Gi3/2, Gi3/4, and Gi3/4:

```
Switch-1-IOS# sho int port-channel 1
Port-channel1 is up, line protocol is up (connected)
  Hardware is EtherChannel, address is 0011.720a.711d (bia 0011.720a.711d)
  Description: 4G Etherchannel Po1
  MTU 1500 bytes, BW 4000000 Kbit, DLY 10 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation ARPA, loopback not set
  Full-duplex, 1000Mb/s
  input flow-control is off, output flow-control is unsupported
  Members in this channel: Gi3/1 Gi3/2 Gi3/4 Gi3/4
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input never, output never, output hang never
  Last clearing of "show interface" counters 30w6d
  Input queue: 0/2000/1951/0 (size/max/drops/flushes); Total output drops: 139
  Queueing strategy: fifo
```

```
  Output queue: 0/40 (size/max)
  5 minute input rate 3906000 bits/sec, 628 packets/sec
  5 minute output rate 256000 bits/sec, 185 packets/sec
     377045550610 packets input, 410236657639149 bytes, 0 no buffer
     Received 66730119 broadcasts (5743298 multicast)
     0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 1951 overrun, 0 ignored
     0 watchdog, 0 multicast, 0 pause input
     0 input packets with dribble condition detected
     255121177828 packets output, 159098829342337 bytes, 0 underruns
     0 output errors, 0 collisions, 0 interface resets
     0 babbles, 0 late collision, 0 deferred
     0 lost carrier, 0 no carrier, 0 PAUSE output
     0 output buffer failures, 0 output buffers swapped out
```

Because the individual links are physical, these interfaces can be shown in the same manner as any physical interface on an IOS device, via the `show interface` command:

```
Switch-1-IOS# sho int g3/1
GigabitEthernet5/1 is up, line protocol is up (connected)
  Hardware is C6k 1000Mb 802.3, address is 0011.7f1a.791c (bia 0011.7f1a.791c)
  Description: Link #1 in Po1
  MTU 1500 bytes, BW 1000000 Kbit, DLY 10 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation ARPA, loopback not set
  Full-duplex, 1000Mb/s
  input flow-control is off, output flow-control is off
  Clock mode is auto
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input 00:00:45, output 00:00:03, output hang never
  Last clearing of "show interface" counters 30w6d
  Input queue: 0/2000/1054/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: fifo
  Output queue: 0/40 (size/max)
  5 minute input rate 924000 bits/sec, 187 packets/sec
  5 minute output rate 86000 bits/sec, 70 packets/sec
     190820216609 packets input, 207901078937384 bytes, 0 no buffer
     Received 48248427 broadcasts (1757046 multicast)
     0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 1054 overrun, 0 ignored
     0 watchdog, 0 multicast, 0 pause input
     0 input packets with dribble condition detected
     129274163672 packets output, 80449383231904 bytes, 0 underruns
     0 output errors, 0 collisions, 0 interface resets
     0 babbles, 0 late collision, 0 deferred
     0 lost carrier, 0 no carrier, 0 PAUSE output
     0 output buffer failures, 0 output buffers swapped out
```

Notice that no mention is made in the output of the fact that the interface is a member of an EtherChannel, other than in the description. This reinforces the notion that all ports should be labeled with the `description` command.

CHAPTER 8
Spanning Tree

The Spanning Tree Protocol (STP) is used to ensure that no layer-2 loops exist in a LAN. As you'll see in this chapter, layer-2 loops can cause havoc.

Spanning tree is designed to prevent loops among bridges. A *bridge* is a device that connects multiple segments within a single collision domain. Hubs and switches are both considered bridges. While the spanning tree documentation always refers to bridges generically, my examples will show switches. Switches are the devices in which you will encounter spanning tree.

When a switch receives a broadcast, it repeats the broadcast on every port (except the one on which it was received). In a looped environment, the broadcasts are repeated forever. The result is called a *broadcast storm*, and it will quickly bring a network to a halt.

Figure 8-1 illustrates what can happen when there's a loop in a network.

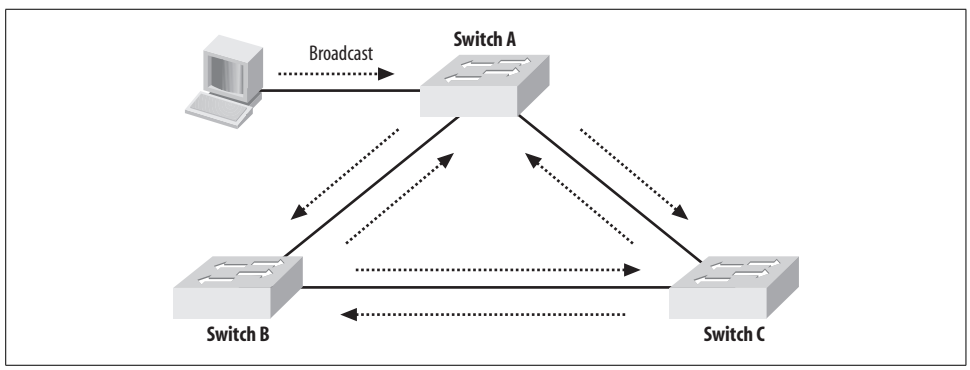


Figure 8-1. Broadcast storm

The computer on Switch A sends out a broadcast frame. Switch A then sends a copy of the broadcast to Switch B and Switch C. Switch B repeats the broadcast to Switch C, and Switch C repeats the broadcast to Switch B; Switch B and Switch C also repeat the broadcast back to Switch A. Switch A then repeats the broadcast it heard from Switch B to Switch C—and the broadcast it heard from Switch C—to Switch B. This progression will continue indefinitely until the loop is somehow broken. Spanning tree is an automated mechanism used to discover and break loops of this kind.

Spanning tree was developed by Dr. Radia Perlman of Sun Microsystems, Inc., who summed up the idea in a poem titled "Algorhyme" that's based on Joyce Kilmer's "Trees":

I think that I shall never see
A graph more lovely than a tree.
A tree whose crucial property
Is loop-free connectivity.
A tree which must be sure to span.
So packets can reach every LAN.
First the Root must be selected
By ID it is elected.
Least cost paths from Root are traced
In the tree these paths are placed.
A mesh is made by folks like me
Then bridges find a spanning tree.

Broadcast Storms

In the network shown in Figure 8-2, there's a simple loop between two switches. Switch A and Switch B are connected to each other with two links: F0/14 and F0/15 on Switch A are connected to the same ports on Switch B. I've disabled spanning tree, which is on by default, to demonstrate the power of a broadcast storm. Both ports are trunks. There are various devices on other ports on the switches, which create normal broadcasts (such as ARP and DHCP broadcasts). There is nothing unusual about this network, aside from spanning tree being disabled.

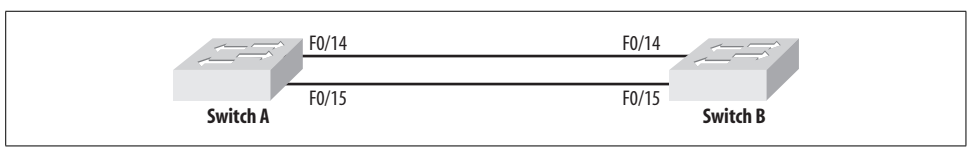


Figure 8-2. Simple layer-2 loop

Interface F0/15 has already been configured and is operating properly. The output from the `show interface f0/15` command shows the input and output rates to be very low (both are around 1,000 bits per second and 2–3 packets per second):

```
3550-IOS# sho int f0/15
FastEthernet0/15 is up, line protocol is up (connected)
  Hardware is Fast Ethernet, address is 000f.8f5c.5a0f (bia 000f.8f5c.5a0f)
  MTU 1500 bytes, BW 100000 Kbit, DLY 100 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation ARPA, loopback not set
  Keepalive set (10 sec)
  Full-duplex, 100Mb/s, media type is 10/100BaseTX
  input flow-control is off, output flow-control is unsupported
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input 00:00:10, output 00:00:00, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: fifo
  Output queue: 0/40 (size/max)
  5 minute input rate 1000 bits/sec, 2 packets/sec
  5 minute output rate 1000 bits/sec, 3 packets/sec
     5778444 packets input, 427859448 bytes, 0 no buffer
     Received 5707586 broadcasts (0 multicast)
     0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored
     0 watchdog, 5707585 multicast, 0 pause input
     0 input packets with dribble condition detected
     2597516 packets output, 213866427 bytes, 0 underruns
```

A useful tool when troubleshooting a broadcast storm is the `show processes cpu history` command. This command displays an ASCII histogram of the CPU utilization over the past 72 hours. It produces three graphs:

- CPU percent per second (last 60 seconds)
- CPU percent per minute (last 60 minutes)
- CPU percent per hour (last 72 hours)

Here is the output from the `show process cpu history` command on switch B, which shows 0–3 percent CPU utilization over the course of the last minute (the remaining graphs have been removed for brevity):

```
3550-IOS# sho proc cpu history
       11111          33333          11111                    1
100
 90
 80
 70
 60
 50
 40
 30
```

```
  20
  10
     0....5....1....1....2....2....3....3....4....4....5....5....
               0    5    0    5    0    5    0    5    0    5

                     CPU% per second (last 60 seconds)
```

The numbers on the left side of the graph are the CPU utilization percentages. The numbers on the bottom are seconds in the past (0 = the time of command execution). The numbers on the top of the graph show the integer values of CPU utilization for that time period on the graph. For example, according to the graph above, CPU utilization was normally 0 percent, but increased to 1 percent 5 seconds ago, and 3 percent 20 seconds ago. When the values exceed 10 percent, visual peaks will be seen in the graph itself.

This switch is a 3550, and has EIGRP neighbors, so it's an important device that is providing layer-3 functionality:

```
3550-IOS# sho ip eigrp neighbors
IP-EIGRP neighbors for process 55
H   Address                 Interface       Hold Uptime   SRTT   RTO  Q    Seq Type
                                            (sec)         (ms)       Cnt  Num
0   10.55.1.10              Fa0/13          14 00:25:30      1   200  0    27
2   10.55.10.3              Vl10            13 1w0d         18   200  0    25
```

Now I'll turn up interface F0/14 as a trunk:

```
3550-IOS(config)# int f0/14
3550-IOS(config-if)# switchport
3550-IOS(config-if)# switchport trunk encapsulation dot1q
3550-IOS(config-if)# switchport mode trunk
```

There are now two trunks connecting Switch A and Switch B. Remember that I've disabled spanning tree. Mere seconds after I converted F0/14 to a trunk, the input and output rates on F0/15 have shot up from 1,000 bits per second and 2–3 packets per second to 815,000 bits per second and 1,561 packets per second:

```
3550-IOS# sho int f0/15 | include minute
  5 minute input rate 815000 bits/sec, 1565 packets/sec
  5 minute output rate 812000 bits/sec, 1561 packets/sec
```

Ten seconds later, the input and output have more than doubled to 2.7 Mbps and 4,500+ packets per second:

```
3550-IOS# sho int f0/15 | include minute
  5 minute input rate 2744000 bits/sec, 4591 packets/sec
  5 minute output rate 2741000 bits/sec, 4587 packets/sec
```

Now I start to get warning messages on the console. The EIGRP neighbors are bouncing:

```
1w0d: %DUAL-5-NBRCHANGE: IP-EIGRP(0) 55: Neighbor 10.55.1.10 (FastEthernet0/13) is
down: holding time expire
```

```
1w0d: %DUAL-5-NBRCHANGE: IP-EIGRP(0) 55: Neighbor 10.55.1.10 (FastEthernet0/13) is
up: new adjacency
1w0d: %DUAL-5-NBRCHANGE: IP-EIGRP(0) 55: Neighbor 10.55.10.3 (Vlan10) is down: retry
limit exceeded
1w0d: %DUAL-5-NBRCHANGE: IP-EIGRP(0) 55: Neighbor 10.55.10.3 (Vlan10) is up: new
adjacency
```

A quick look at the CPU utilization for the past minute explains why. The histogram shows that for the last 25 seconds, CPU utilization has been at 99 percent. The trouble started 40 seconds ago, which is when I enabled the second trunk:

```
3550-IOS# sho proc cpu hist

    999999999999999999999999333339999966666
    99999999999999999999999999999999993333311111
100 **************************     *****
 90 **************************     *****
 80 **************************     *****
 70 **************************     *****
 60 **************************     **********
 50 **************************     **********
 40 ****************************************
 30 ****************************************
 20 ****************************************
 10 ****************************************
   0....5....1....1....2....2....3....3....4....4....5....5....
            0    5    0    5    0    5    0    5    0    5

              CPU% per second (last 60 seconds)
```

CPU utilization rocketed to 99 percent within seconds of the loop being created. The switch is expending most of its processing power forwarding broadcasts through the loop. As a result, all other processes start to suffer. Using telnet or SSH to administer the switch is almost impossible at this point. Even the console is beginning to show signs of sluggishness.

Within five minutes, the two links are at 50 percent utilization. They are also sending and receiving in excess of 70,000 packets per second:

```
3550-IOS# sho int f0/15
FastEthernet0/15 is up, line protocol is up (connected)
  Hardware is Fast Ethernet, address is 000f.8f5c.5a0f (bia 000f.8f5c.5a0f)
  MTU 1500 bytes, BW 100000 Kbit, DLY 100 usec,
     reliability 255/255, txload 143/255, rxload 143/255
  Encapsulation ARPA, loopback not set
  Keepalive set (10 sec)
  Full-duplex, 100Mb/s, media type is 10/100BaseTX
  input flow-control is off, output flow-control is unsupported
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input 00:00:00, output 00:00:00, output hang never
  Last clearing of "show interface" counters never
```

```
Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
Queueing strategy: fifo
Output queue: 0/40 (size/max)
5 minute input rate 56185000 bits/sec, 71160 packets/sec
5 minute output rate 56277000 bits/sec, 70608 packets/sec
   48383882 packets input, 112738864 bytes, 0 no buffer
   Received 48311185 broadcasts (0 multicast)
   0 runts, 0 giants, 0 throttles
   0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored
   0 watchdog, 36608855 multicast, 0 pause input
   0 input packets with dribble condition detected
   45107032 packets output, 4212355164 bytes, 356 underruns
   0 output errors, 0 collisions, 1 interface resets
   0 babbles, 0 late collision, 0 deferred
   0 lost carrier, 0 no carrier, 0 PAUSE output
   361 output buffer failures, 0 output buffers swapped out
```

Remember that the baseline traffic load for this interface was 1,000 bits per second and 2–3 packets per second. 99.9 percent of the traffic we're seeing now is recycled broadcasts. This is a broadcast storm. Rest assured that experiencing one on a live network—particularly one for which you are responsible—is not fun. The storm occurs quickly and affects services immediately.

Because this example is in a lab environment, I am having fun (though my wife would no doubt have some interesting comments regarding what I consider fun), but all good things must come to an end. Within seconds of disabling one of the trunks, EIGRP stabilizes, and the network problems disappear. I can now telnet to the switch with ease. The CPU returns almost immediately back to normal utilization rates of 0–1 percent:

```
3550-IOS(config)# int f0/14
3550-IOS(config-if)# no switchport
3550-IOS# sho proc cpu hist
                 666669999999999999999999999999999999999999999999
           111111111199999999999999999999999999999999999999999999
100                  **********************************************
 90                  **********************************************
 80                  **********************************************
 70                  **********************************************
 60             ***********************************************
 50             ***********************************************
 40             ***********************************************
 30             ***********************************************
 20             ***********************************************
 10             ***********************************************
   0....5....1....1....2....2....3....3....4....4....5....5....
             0    5    0    5    0    5    0    5    0    5

                  CPU% per second (last 60 seconds)
```

This example showed how devastating a broadcast storm can be. When the switches involved become unresponsive, diagnosing the storm can become very difficult. If you can't access the switch via the console, SSH, or telnet, the only way to break the loop is by disconnecting the offending links. If you're lucky, the looped port's activity lights will be flashing more than those of the other ports. In any case, you won't be having a good day.

MAC Address Table Instability

Another problem caused by a looped environment is MAC address tables (CAM tables in CatOS) being constantly updated. Take the network in Figure 8-3, for example. With all of the switches interconnected, and spanning tree disabled, Switch A will come to believe that the MAC address for the PC directly connected to it is sourced from a different switch. This happens very quickly during a broadcast storm, and, in the rare instances when you see this behavior without a broadcast storm, chances are things are about to get very bad very quickly.

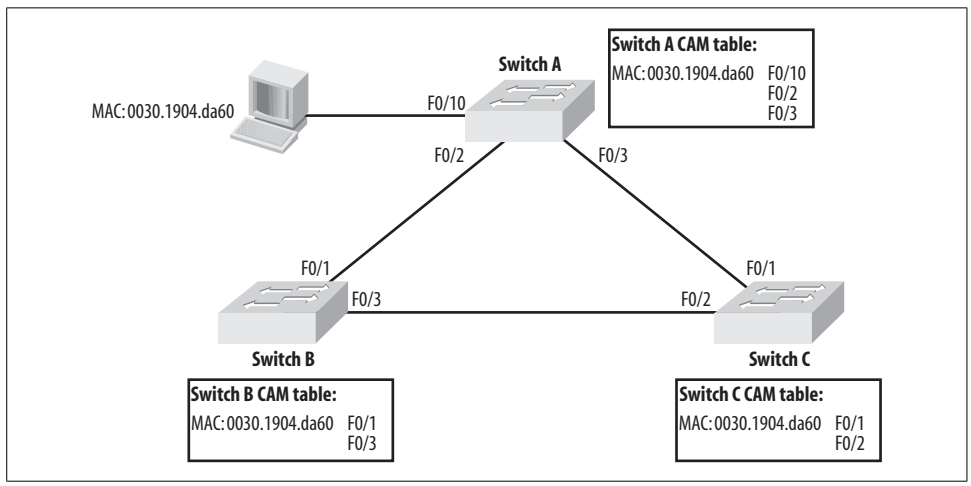


Figure 8-3. MAC address table inconsistencies

With this network in place, and spanning tree disabled, I searched for the MAC address of the PC on Switch A using the `show mac-address-table | include 0030.1904.da60` command. I repeated the command as fast as I could and got the following results:

```
3550-IOS# sho mac-address-table | include 0030.1904.da60
   1    0030.1904.da60    DYNAMIC     Fa0/10
3550-IOS# sho mac-address-table | include 0030.1904.da60
   1    0030.1904.da60    DYNAMIC     Fa0/10
3550-IOS# sho mac-address-table | include 0030.1904.da60
```

```
   1    0030.1904.da60    DYNAMIC     Fa0/2
3550-IOS# sho mac-address-table | include 0030.1904.da60
   1    0030.1904.da60    DYNAMIC     Fa0/3
3550-IOS# sho mac-address-table | include 0030.1904.da60
   1    0030.1904.da60    DYNAMIC     Fa0/2
3550-IOS# sho mac-address-table | include 0030.1904.da60
```

This switch is directly connected to the device in question, yet at different times it seems to believe that the best path to the device is via Switch B or Switch C.

Remember that a switch examines each packet that arrives on a port and assigns the packet's source MAC address to that port in its MAC address/CAM table. Because devices can and do move, the switch will assume that the last port on which the MAC address was observed is where the address now resides. As the broadcast packets originating from the PC are constantly cycling through the looped network, wherever the packet comes into the switch is where the switch will believe that MAC address belongs.

Preventing Loops with Spanning Tree

The obvious way to prevent loops is to follow the same advice a doctor might give you when you complain, "It hurts when I do this"—don't do it! Of course, in the real world, there are many variables that are out of your control. I've seen more than one network go down because someone decided to plug both network drops under his desk into the little switch she'd brought in from home. Heck, I've seen network administrators do it themselves.

Having more than one link between switches is a good idea in terms of redundancy—in fact, it's recommended. The trick is to have only one link active at a time. If you configure two links between two switches and shut one down, you'll solve the loop problem, but when the live link fails, you'll need to manually bring up the second link.

Spanning tree is a protocol designed to discover network loops and break them before they can cause any damage. Properly configured, spanning tree is an excellent tool that should always be enabled on any network. Improperly configured, however, spanning tree can cause subtle problems that can be hard to diagnose.

How Spanning Tree Works

Spanning tree elects a *root bridge* (switch) in the network. The root bridge is the bridge that all other bridges need to reach via the shortest path possible. Spanning tree calculates the cost for each path from each bridge in the network to the root bridge. The path with the lowest cost is kept intact, while all others are broken. Spanning tree breaks paths by putting ports into a *blocking* state.

Every bridge on the network that supports spanning tree sends out frames called *bridge protocol data units* (BPDUs) every two seconds. The format of the BPDU frame is shown in Figure 8-4.

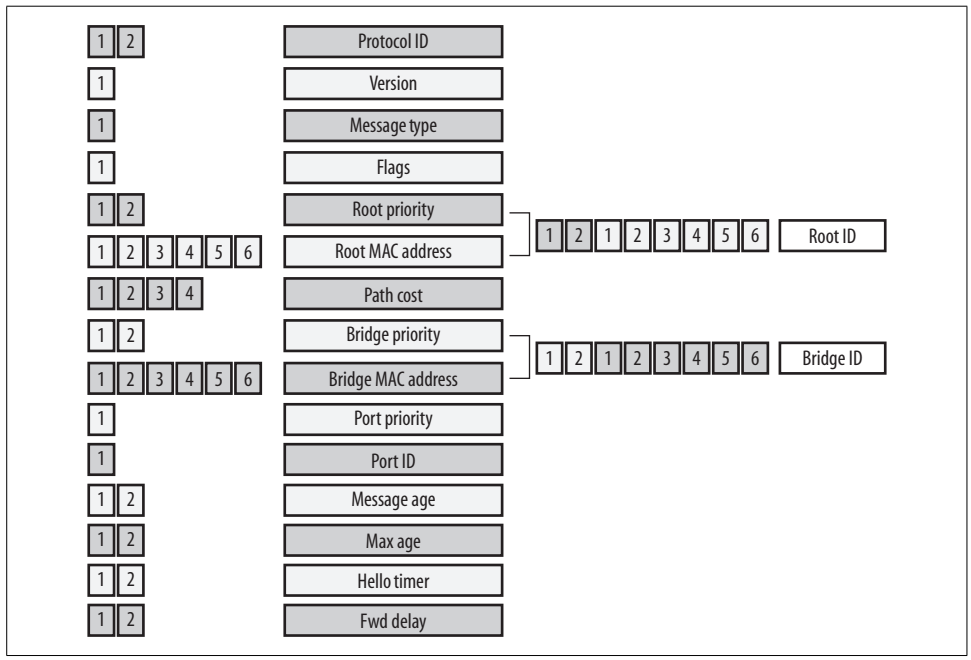


Figure 8-4. BPDU format

These frames contain the information necessary for switches in the network to perform the following functions:

Elect a root bridge

When a switch boots, it assumes that it is the root bridge, and sets the root ID to the local bridge ID in all outgoing BPDUs. If it receives a BPDU that has a lower root ID, the switch considers the switch identified by the root ID in the BPDU to be the root switch. The local switch then begins using that root ID in the BPDUs it sends.

Every bridge has a *bridge ID*. The bridge ID is a combination of the bridge priority and the bridge's MAC address. The bridge priority is a configurable two-byte field with a default value of 32,768. The lower the bridge ID value is, the more likely it is that the bridge will become the root bridge (the bridge with the lowest bridge ID becomes the root bridge).

The *root ID* is similarly composed of two fields: the root priority and the root MAC address. The root priority is also configured with a value of 32768 (0x8000) by default. Should there be a tie between root priorities, the lower root MAC address is used to break the tie.

Determine the best path to the root bridge

If BPDUs from the root bridge are received on more than one port, there is more than one path to the root bridge. The best path is considered to be via the port on which the BPDU with the lowest *path cost* was received.

Path costs are determined by adding each bridge's port priority to the initial path cost as BPDUs are forwarded from bridge to bridge.

Determine the root port on each bridge

The *root port* is the port on the switch that has the shortest path to the root bridge. The root bridge does not have root ports; it only has designated ports.

Determine the designated port on each segment

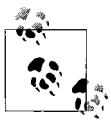
The *designated port* is the port on the segment that has the shortest path to the root bridge. On segments that are directly connected to the root bridge, the root bridge's ports are the designated ports.

Elect a designated bridge on each segment

The bridge on a given segment with the designated port is considered the *designated bridge*. The root bridge is the designated bridge for all directly connected segments. In the event that two bridges on a segment have root ports, the bridge with the lowest bridge ID becomes the designated bridge.

Block nonforwarding ports

Ports that have received BPDUs, and are neither designated nor root ports, are placed into a *blocking* state. These ports are administratively up, but are not allowed to forward traffic (though they still send and receive BPDUs).



Always configure a switch to be the root bridge. Letting the switches configure themselves is dangerous because they will choose the switch with the lowest MAC address, which will usually be a switch other than the one it should be. As a general rule, you should not let networking devices make critical decisions using default values. It will cause your network to behave in unexpected ways, and will cause you to fail higher-level certification exams, which are designed to catch you in exactly this way. Usually, the device that should be the root bridge will be obvious. The root bridge should generally be one of the core switches in your design.

Every port on a switch goes through a series of spanning tree states when it is brought online, as illustrated in the flowchart in Figure 8-5. These states transition in a pattern depending on the information learned from BPDUs received on the port.

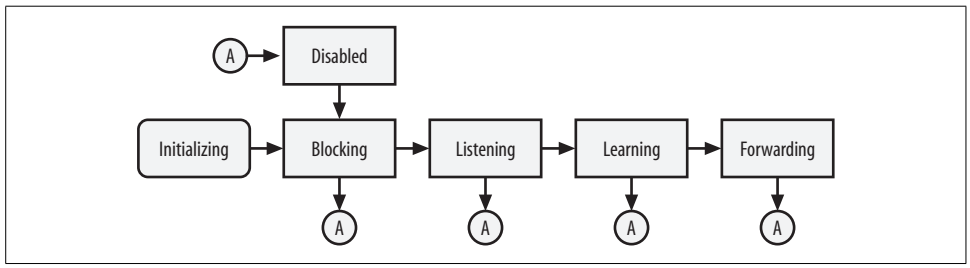


Figure 8-5. Spanning tree port states

These are the spanning tree states:

Initializing
: A port in the initializing state has either just been powered on, or just taken out of the administratively down state.

Blocking
: A port in the blocking state is essentially unused. It does not forward or receive frames, with the following exceptions:

  - The port receives and processes BPDUs.
  - The port receives and responds to messages relating to network management.

Listening
: The listening state is similar to the blocking state, except that in this state, BPDUs are sent as well as received. Frame forwarding is still not allowed, and no addresses are learned.

Learning
: A port in the learning state still does not forward frames, but it does analyze frames that come into the port and retrieve the MAC addresses from those frames for inclusion in the MAC address/CAM table. After the frames are analyzed, they are discarded.

Forwarding
: The forwarding state is what most people would consider the "normal" state. A port in this state receives and transmits BPDUs, analyzes incoming packets for MAC address information, and forwards frames from other switch ports. When a port is in the forwarding state, the device or network attached to the port is active and able to communicate.

Disabled
: A port in the disabled state does not forward frames, and does not participate in spanning tree. It receives and responds to only network-management messages.

Per-VLAN Spanning Tree

Because VLANs can be pruned from trunks (as discussed in Chapter 6), it is possible that some VLANs may form loops while others do not. For this reason, Cisco switches now default to a multiple-VLAN form of spanning tree called *Per-VLAN Spanning Tree* (PVST). PVST allows for a spanning tree instance for each VLAN when used with ISL trunks. Per-VLAN Spanning Tree Plus (PVST+) offers the same features when used with 802.1Q trunks.

By default, all VLANs will inherit the same values for all spanning tree configurations. However, each VLAN can be configured differently. For example, each VLAN may have a different spanning tree root bridge. This functionality is an advanced topic, and is not covered in this book.

Managing Spanning Tree

Spanning tree is enabled by default. To see its status, use the `show spanning-tree` command in IOS:

```
Cat-3550# sho spanning-tree

VLAN0001
  Spanning tree enabled protocol ieee
  Root ID    Priority    24577
             Address     0009.43b5.0f80
             Cost        23
             Port        20 (FastEthernet0/20)
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID  Priority    32769  (priority 32768 sys-id-ext 1)
             Address     000f.8f5c.5a00
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
             Aging Time 300

Interface        Role Sts Cost      Prio.Nbr Type
---------------- ---- --- --------- -------- --------------------------------
Fa0/13           Altn BLK 19        128.13   P2p
Fa0/14           Altn BLK 19        128.14   P2p
Fa0/15           Altn BLK 19        128.15   P2p
Fa0/20           Root FWD 19        128.20   P2p
Fa0/23           Desg FWD 19        128.23   P2p
[-text removed-]
```

The bolded text shows the priority and MAC address of the root bridge, as well as what port the switch is using to get there (this is the root port). This is very useful information when you're trying to figure out where the root bridge is on a network. By running this command on every switch in the network, you should be able to map your connections and figure out which switch is the root.

Switch-specific information is located below the root information, and below the local switch information is information specific to each port on the switch that is actively participating in spanning tree. This information will be repeated for every VLAN.

In CatOS, the equivalent command is `show spantree`. The command produces very similar information, with a slightly different layout:

```
CatOS-6509: (enable) sho spantree
VLAN 1
Spanning tree mode          RAPID-PVST+
Spanning tree type          ieee
Spanning tree enabled

Designated Root             00-00-00-00-00-00
Designated Root Priority    0
Designated Root Cost        0
Designated Root Port        1/0
Root Max Age   0  sec   Hello Time 0  sec   Forward Delay 0  sec

Bridge ID MAC ADDR          00-00-00-00-00-00
Bridge ID Priority          32768
Bridge Max Age 20 sec   Hello Time 2  sec   Forward Delay 15 sec

Port                      State         Role Cost      Prio Type
------------------------ ------------- ---- --------- ---- --------------------
 1/1                      not-connected  -           4   32
 1/2                      not-connected  -           4   32
 2/1                      not-connected  -           4   32
 2/2                      not-connected  -           4   32
[-text removed-]
```

Notice that the designated root MAC address is all zeros. This indicates that the switch considers itself to be the root bridge.

To get a summary of spanning tree, use the IOS command `show spanning-tree summary`. This command is useful to see the status of features like UplinkFast and BackboneFast (discussed in the following section):

```
Cat-3550# sho spanning-tree summary
Switch is in pvst mode
Root bridge for: VLAN0002, VLAN0200
Extended system ID           is enabled
Portfast Default             is disabled
PortFast BPDU Guard Default  is disabled
Portfast BPDU Filter Default is disabled
Loopguard Default            is disabled
EtherChannel misconfig guard is enabled
UplinkFast                   is disabled
BackboneFast                 is disabled
```

```
Configured Pathcost method used is short

Name                 Blocking Listening Learning Forwarding STP Active
-------------------- -------- --------- -------- ---------- ----------
VLAN0001                    3         0        0          2          5
VLAN0002                    0         0        0          5          5
VLAN0003                    2         0        0          2          4
VLAN0004                    2         0        0          2          4
VLAN0010                    2         0        0          2          4
VLAN0100                    2         0        0          2          4
VLAN0200                    0         0        0          4          4
-------------------- -------- --------- -------- ---------- ----------
7 vlans                    11         0        0         19         30
```

In CatOS, the summary command is `show spantree summary`:

```
CatOS-6509: (enable) sho spantree summ
Spanning tree mode: RAPID-PVST+
Runtime MAC address reduction: disabled
Configured MAC address reduction: disabled
Root switch for vlans: 20.
Global loopguard is disabled on the switch.
Global portfast is disabled on the switch.
BPDU skewing detection disabled for the bridge.
BPDU skewed for vlans:  none.
Portfast bpdu-guard disabled for bridge.
Portfast bpdu-filter disabled for bridge.
Uplinkfast disabled for bridge.
Backbonefast disabled for bridge.

Summary of connected spanning tree ports by vlan

VLAN  Blocking Listening Learning Forwarding STP Active
----- -------- --------- -------- ---------- ----------
  20         0         0        0          1          1

      Blocking Listening Learning Forwarding STP Active
----- -------- --------- -------- ---------- ----------
Total        0         0        0          1          1
```

An excellent command in IOS is `show spanning-tree root`, which shows you the information regarding the root bridge for every VLAN:

```
Cat-3550# sho spanning-tree root

                                        Root    Hello Max Fwd
Vlan                 Root ID            Cost    Time  Age Dly  Root Port
---------------- ------------------- --------- ----- --- ---  ------------
VLAN0001         24577 0009.43b5.0f80        23    2   20  15  Fa0/20
VLAN0002         32770 000f.8f5c.5a00         0    2   20  15
VLAN0003         32771 000d.edc2.0000        19    2   20  15  Fa0/13
```

```
VLAN0004         32772 000d.edc2.0000        19    2   20  15  Fa0/13
VLAN0010         32778 000d.edc2.0000        19    2   20  15  Fa0/13
VLAN0100         32868 000d.edc2.0000        19    2   20  15  Fa0/13
VLAN0200         32968 000f.8f5c.5a00         0    2   20  15
```

There is no equivalent command in CatOS.

Additional Spanning Tree Features

Spanning tree was originally designed for bridges with few ports. With the advent of Ethernet switches, some enhancements were made to spanning tree. These commonly seen enhancements helped make spanning tree more palatable by decreasing the time a host needs to wait for a port, and decreasing the convergence time in a layer-2 network.

PortFast

PortFast is a feature on Cisco switches that allows a port to bypass all of the other spanning tree states (see Figure 8-5) and proceed directly to the forwarding state. PortFast should be enabled only on ports that will not have switches connected. Spanning tree takes about 30 seconds to put a normal port into the forwarding state, which can cause systems using DHCP to time out and not get an IP address (on a Windows machine, a default IP address may be used). Enabling the PortFast feature on a port alleviates this problem, but you should be very careful when using this feature. If a switch were to be connected to a port configured with PortFast active, a loop could occur that would not be detected.

To enable PortFast on an IOS switch, use the `spanning-tree portfast` interface command. The switch will deliver a nice warning about the dangers of PortFast when you enable the feature:

```
Cat-3550(config-if)# spanning-tree portfast
%Warning: portfast should only be enabled on ports connected to a single
 host. Connecting hubs, concentrators, switches, bridges, etc... to this
 interface  when portfast is enabled, can cause temporary bridging loops.
 Use with CAUTION

%Portfast has been configured on FastEthernet0/20 but will only
 have effect when the interface is in a non-trunking mode.
```

To disable PortFast on an interface in IOS, simply negate the command. There is no fanfare when disabling PortFast:

```
Cat-3550(config-if)# no spanning-tree portfast
Cat-3550(config-if)#
```

On a CatOS switch, the command to enable PortFast is `set spantree portfast <mod/port> enable`. Executing this command will also result in a nice message about the dangers of PortFast:

```
CatOS-6509: (enable) set spantree portfast 3/10 enable

Warning: Connecting Layer 2 devices to a fast start port can cause
temporary spanning tree loops. Use with caution.

Spantree port  3/10 fast start enabled.
```

To disable PortFast, use the same command with `disable` instead of `enable`:

```
CatOS-6509: (enable) set spantree portfast 3/10 disable
Spantree port  3/10 fast start disabled.
```

BPDU Guard

Ports configured for PortFast should never receive BPDUs as long as they are connected to devices other than switches/bridges. If a PortFast-enabled port is connected to a switch, a bridging loop will occur. To prevent this, Cisco developed a feature called *BPDU Guard*. BPDU Guard automatically disables a port configured for PortFast in the event that it receives a BPDU. The port is not put into blocking mode, but is put into the ErrDisable state. Should this happen, the interface must be reset. BPDU Guard is enabled with the `spanning-tree bpduguard enable` command in IOS:

```
Cat-3550(config-if)# spanning-tree bpduguard enable
```

To disable this feature, change the `enable` keyword to `disable`.

In CatOS, use the `set spantree bpdu-guard <mod/port> enable` (or `disable`) command:

```
CatOS-6509: (enable) set spantree bpdu-guard 3/10 enable
Spantree port  3/10 bpdu guard enabled.
```

UplinkFast

UplinkFast is a feature designed for access-layer switches. These switches typically have links to other switches to connect to the distribution layer. Normally, when the link on the designated port fails, a port with an alternate path to the root bridge is cycled through the spanning tree listening and learning states until it returns to the forwarding state. Only then can the port pass traffic. This process can take 45 seconds or more.

UplinkFast allows a blocked uplink port to bypass the listening and learning states when the designated port fails. This allows the network to recover in five seconds or

less. This feature affects all VLANs on the switch. It also sets the bridge priority to 49,152 to all but ensure that the switch will not become the root bridge.

Figure 8-6 shows where UplinkFast would be applied in a simple network. Switches A and B would be either core or distribution switches. Switch C would be an access-layer switch. The only links to other switches on Switch C are to the distribution or core.

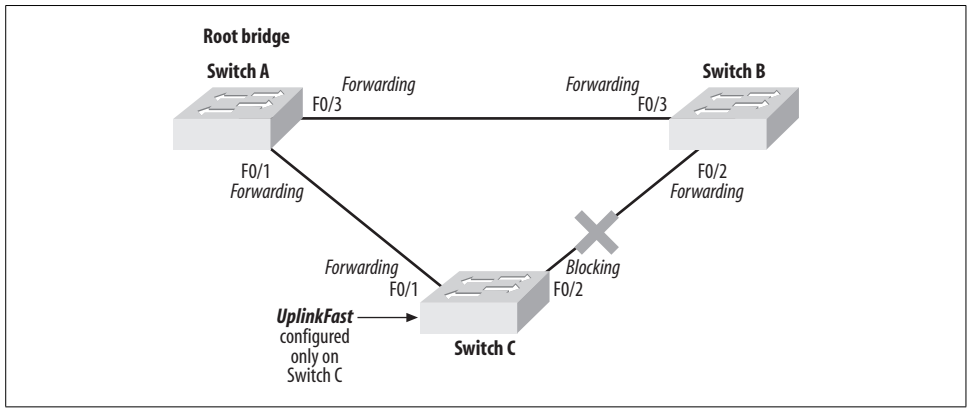


Figure 8-6. UplinkFast example

UplinkFast should be configured only on access-layer switches. It should never be enabled on distribution-layer or core switches because it prevents the switch from becoming the root bridge, which is usually counterindicated in core switches.

To configure UplinkFast on a CatOS switch, use the `set spantree uplinkfast enable` command:

```
CatOS-6509: (enable) set spantree uplinkfast enable
VLANs 1-4094 bridge priority set to 49152.
The port cost and portvlancost of all ports set to above 3000.
Station update rate set to 15 packets/100ms.
uplinkfast all-protocols field set to off.
uplinkfast enabled for bridge.
```

When disabling UplinkFast, be careful, and remember that a lot of other values were changed when you enabled the feature:

```
CatOS-6509: (enable) set spantree uplinkfast disable
uplinkfast disabled for bridge.
Use clear spantree uplinkfast to return stp parameters to default.
```

That last line of output is important. If you're moving a switch from an access-layer role to a role where you want it to become the root bridge, you'll need to change the priorities back to their defaults.

Here's what happens when I follow the switch's advice:

```
CatOS-6509: (enable) clear spantree uplinkfast
This command will cause all portcosts, portvlancosts, and the
bridge priority on all vlans to be set to default.
Do you want to continue (y/n) [n]? y
VLANs 1-4094 bridge priority set to 32768.
The port cost of all bridge ports set to default value.
The portvlancost of all bridge ports set to default value.
uplinkfast all-protocols field set to off.
uplinkfast disabled for bridge.
```

The values are not necessarily set to what they were before I enabled UplinkFast—they are returned to their defaults.

When configuring UplinkFast on IOS switches, there are no scary messages like there are in CatOS:

```
Cat-3550(config)# spanning-tree uplinkfast
Cat-3550(config)#
```

That's it. Simple! But don't let the simplicity fool you—enabling UplinkFast changes priorities in IOS, too. Unlike in CatOS, however, disabling the feature in IOS (via `no spanning-tree uplinkfast`) automatically resets the priorities to their defaults. Again, this might not be what you want or expect, so be careful.

BackboneFast

When a switch receives a BPDU advertising a root bridge that's less desirable than the root bridge it already knows about, the switch discards the BPDU. This is true for as long as the switch knows about the better root bridge. If the switch stops receiving BPDUs for the better root bridge, it will continue to believe that that bridge is the best bridge until the max_age timeout is exceeded. max_age defaults to 20 seconds.

Figure 8-7 shows a network with three switches. All of these switches are core or distribution switches, though they could also be access-layer switches. Switch A is the root bridge. Through normal convergence, the F0/2 port on Switch C is blocking, while all the others are forwarding.

Say an outage occurs that brings down the F0/3 link between Switch A and Switch B. This link is not directly connected to Switch C. The result is an *indirect link failure* on Switch C. When Switch B recognizes the link failure, it knows that it has no path to the root, and starts advertising itself as the root. Until this point, Switch B had been advertising BPDUs showing the more desirable Switch A as the root. Switch C still has that information in memory, and refuses to believe that the less desirable Switch B is the root until the max_age timeout expires.

After 20 seconds, Switch C will accept the BPDU advertisements from Switch B, and start sending its own BPDUs to Switch B. When Switch B receives the BPDUs from Switch C, it will understand that there is a path to Switch A (the more desirable root

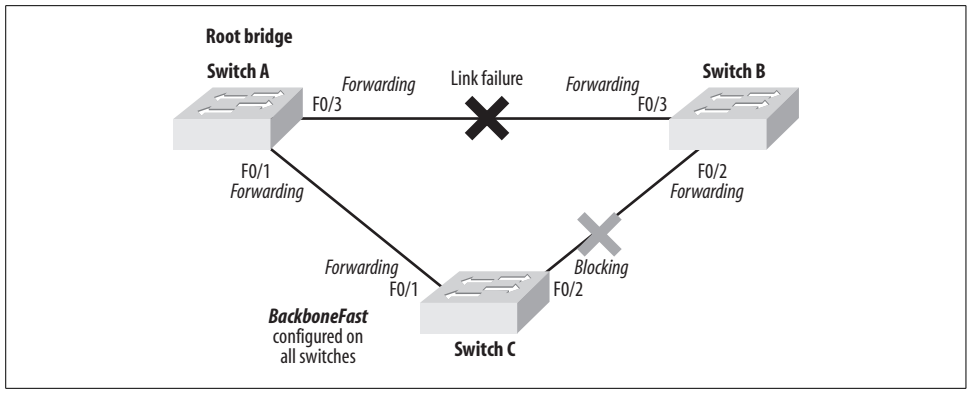


Figure 8-7. BackboneFast example

bridge) through Switch C, and accept Switch A as the root bridge again. This process takes upwards of 50 seconds with the default spanning tree timers.

BackboneFast adds functionality that detects indirect link failures. It actively discovers paths to the root by sending out *root link query* PDUs after a link failure. When it discovers a path, it sets the max_age timer to 0 so that the port can cycle through the normal listening, learning, and forwarding states without waiting an additional 20 seconds.

If BackboneFast is used, it must be enabled on every switch in the network.

Enabling BackboneFast on an IOS switch is as simple as using the `spanning-tree backbonefast` global command:

```
Cat-3550(config)# spanning-tree backbonefast
Cat-3550(config)#
```

Negating the command disables the feature.

To enable BackboneFast on a CatOS switch, use the `set spantree backbonefast enable` command:

```
CatOS-6509: (enable) set spantree backbonefast enable
Backbonefast enabled for all VLANs
```

To disable this feature, change the `enable` keyword to `disable`.

Common Spanning Tree Problems

Spanning tree can be a bit of a challenge when it misbehaves. More to the point, spanning tree problems can be hard to diagnose if the network is not properly designed. Here are a couple of common problems and how to avoid them.

Duplex Mismatch

A bridge still receives and processes BPDUs on ports even when they are in a blocked state. This allows the bridge to know that a path to the root bridge is still available should the primary path fail.

If a port in the blocking state stops receiving BPDUs, the bridge no longer considers the port to be a path to the root bridge. In this case, the port should no longer be blocked, so the bridge puts the port into the forwarding state. Would this ever happen in the real world? It's happened to me more than once.

A common spanning tree problem is shown in Figure 8-8. Here, two switches are connected with two links: F0/0 on Switch A is connected to F0/0 on Switch B, and F0/1 on Switch A is connected to F0/1 on Switch B. Switch A is the root bridge. All ports are in the forwarding state, except for F0/1 on Switch B, which is blocking. The network is stable because spanning tree has broken the potential loop. The arrows show BPDUs being sent.

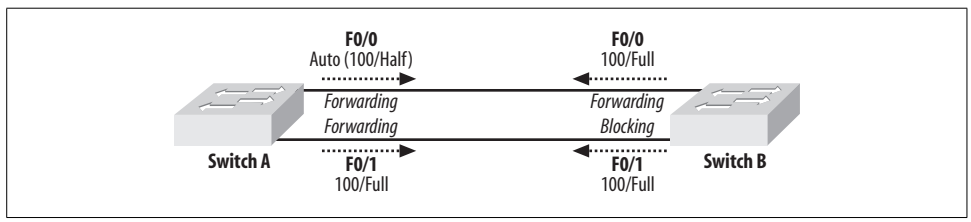


Figure 8-8. Spanning tree half-duplex problem

Port F0/0 on Switch A is the only port that is set to auto-negotiation. Auto-negotiation has determined that the port should be set to 100 Mbps and half-duplex mode. The other ports are all hardcoded to 100/full. Spanning tree is sending BPDUs out of all ports and is receiving them on all ports—even the one that is blocking.

Always make sure that both sides of an Ethernet link are configured the same way regarding speed and duplex. See Chapter 3 for details.

When a port is in half-duplex mode, it listens for collisions before transmitting. A port in full-duplex mode does not. When a half-duplex port is connected with a full-duplex port, the full-duplex port will send continuously, causing the half-duplex port to encounter many collisions. After a collision, the port will perform the back-off algorithm, and wait to resend the packet that collided. In our example, the half-duplex port is the active link with data being sent across it. When the data rate gets high, the collision problem gets worse, resulting in frames—including BPDUs—being dropped.

Switch B will listen for BPDUs over the two links shown in the diagram. If no BPDUs are seen over the F0/0 link for a set amount of time, Switch B will no longer consider the F0/0 link to be a valid path to the root bridge. Because this was the primary path to the root bridge, and the root bridge can no longer be seen, Switch B will change F0/1 from blocking to forwarding to reestablish a path to the root bridge. At this point, there are no blocking ports on the two links connecting the switches, and a bridging loop exists.

Unidirectional Links

When a link is able to transmit in one direction but not another, the link is said to be *unidirectional*. While this can happen with copper Ethernet, the problem is most often seen when using fiber.

A common issue when installing fiber plants is the cross-connection of individual fibers. Should a fiber pair be split, one fiber strand can end up on a different port or switch from the other strand in the pair.

Figure 8-9 shows four switches. Switch A is supposed to be connected to Switch B by two pairs of fiber—one between the G0/1 ports on each switch, and another between the G0/2 ports on each switch. Somewhere in the cabling plant, the fiber pair for the G0/2 link has been split. Though the pair terminates correctly on Switch B, Switch A has only one strand from the pair. The other strand has been routed to Switch C and connected to port G0/3.

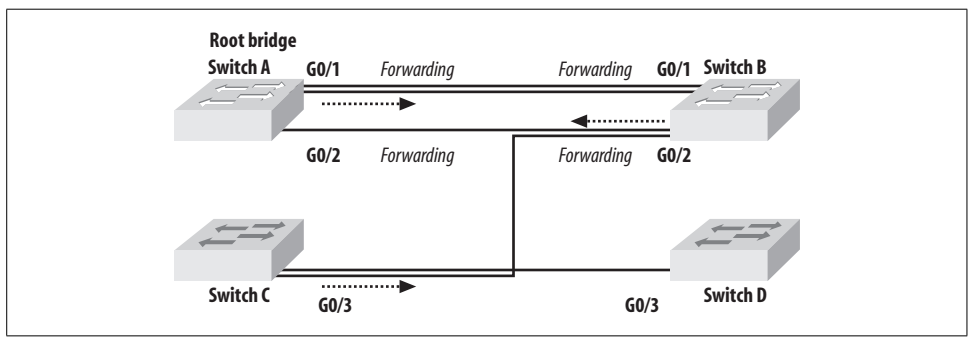
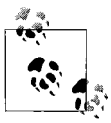


Figure 8-9. Unidirectional link problem

Fiber interfaces test for link integrity by determining whether there is a signal on the RX side of the pair. Switch B's port G0/2 has link integrity because its RX is active from Switch C, Switch A's port G0/2 has link integrity because its RX is active from Switch B, and Switch C's port G0/3 has link integrity because its RX is active from Switch D.

Spanning tree is sending BPDUs out each interface on Switch A and Switch B because the links are active. Switch A is the root bridge. Switch B is only receiving BPDUs from the root bridge on one port: G0/1. Because Switch B is not receiving BPDUs from the root bridge on port G0/2, spanning tree does not block the port. Broadcasts received on Switch B on G0/1 will be retransmitted out G0/2. A loop is born.

This problem can be difficult to uncover. A bridging loop causes mayhem in the network because the CPU utilization on network devices can quickly reach 100 percent, causing outages. The first thing inexperienced engineers try is rebooting one or more devices in the network. In the case of a unidirectional link, rebooting will not resolve the issue. When that fails, the loop is usually detected, and the engineer shuts down one of the links. But when he shuts down the link, the proof of the unidirectional link is often lost.



Physical layer first! Always suspect that something physical is wrong when diagnosing connectivity problems. It can save you hours of headaches, especially if all the other clues don't seem to add up to anything substantial. Also, don't assume that it works today just because it worked yesterday. It doesn't take much for someone to crush a fiber strand when closing a cabinet door.

With the latest versions of IOS and CatOS, unidirectional link problems are handled by a protocol called Unidirectional Link Detection (UDLD). UDLD is on by default and should be left on. If you see UDLD errors, look for issues similar to what I've just described.

Designing to Prevent Spanning Tree Problems

Proper design can help minimize spanning tree problems. One of the simplest ways to help keep trouble to a minimum is to document and know your network. If you have to figure out how your network operates when there's a problem, the problem may last longer than your job.

Use Routing Instead of Switching for Redundancy

The saying used to be, "Switch when you can and route when you have to." But in today's world of fast layer-3 switching, this mantra no longer holds. With layer-3 switches, you can route at switching speeds.

For many people, layer-3 redundancy is easier to understand than layer-2 redundancy. As long as the business needs are met, and the end result is the same, using routing to solve your redundancy concerns is perfectly acceptable.

If you decide to use routing instead of switching, don't turn off spanning tree. Spanning tree will still protect against loops you might have missed. If you're using switches—even layer-3 switches—spanning tree can be a lifesaver if someone plugs in a switch where it doesn't belong.

Always Configure the Root Bridge

Don't let spanning tree elect the root bridge dynamically. Decide which switch in your network should be the root, and configure it with a bridge priority of 1. If you let the switches decide, not only may they choose one that doesn't make sense, but switches added later may assume the role of root bridge. This will cause the entire network to reconverge and links to change states as the network discovers paths to the new root bridge.

I once saw a large financial network in a state of confusion because the root bridge was a switch under someone's desk. The root bridge had not been configured manually, and when this rogue switch was added to the network, all the switches reconverged to it because it had the lowest MAC address of all the switches. While this may not sound like a big deal on the surface, the problem manifested itself because the main trunk between the core switches, which was the farthest link from the root bridge, became blocked.

PART II
Routers and Routing

This section introduces routing and explains how the routing table works. It then moves on to more advanced topics associated with routers or routing.

This section is composed of the following chapters:

Chapter 9, *Routing and Routers*
Chapter 10, *Routing Protocols*
Chapter 11, *Redistribution*
Chapter 12, *Tunnels*
Chapter 13, *Resilient Ethernet*
Chapter 14, *Route Maps*
Chapter 15, *Switching Algorithms in Cisco Routers*

CHAPTER 9
Routing and Routers

Routing is a term with multiple meanings in different disciplines. In general, it refers to determining a path for something. In telecom, a call may be routed based on the number being dialed or some other identifier. In either case, a path is determined for the call.

Mail is also routed—I'm not talking about email here (though email is routed too)—but rather, snail mail. When you write an address and a zip code on a letter, you are providing the means for the post office to route your letter. You provide a destination and, usually, a source address, and the post office determines the best path for the letter. If there is a problem with the delivery of your letter, the return address is used to route it back to you. The exact path the letter takes to get from its source to its destination doesn't really matter; all you care about is that it (hopefully) makes it in a timely fashion, and in one piece.

In the IP world, packets or frames are forwarded on a local network by switches, hubs, or bridges. If the address of the destination is not on the local network, the packet must be forwarded to a *gateway*. The gateway is responsible for determining how to get the packet to where it needs to be. RFC 791, titled INTERNET PROTOCOL, defines a gateway thusly:

```
2.4.  Gateways

  Gateways implement internet protocol to forward datagrams between
  networks. Gateways also implement the Gateway to Gateway Protocol
  (GGP) [7] to coordinate routing and other internet control
  information.

  In a gateway the higher level protocols need not be implemented and
  the GGP functions are added to the IP module.
```

When a station on a network sends a packet to a gateway, the station doesn't care *how* the packet gets to its destination—just that it does (at least, in the case of TCP). Much like a letter in the postal system, each packet contains its source and destination addresses so that routing can be accomplished with ease.

In the realm of semantics and IP, a gateway is a device that forwards packets to a destination other than the local network. For all practical purposes, a gateway is a router. Router is the term I will generally use in this book, although you will also see the phrase *default gateway*.

In the olden days of data communication, a "gateway" was a device that translated one protocol to another. For example, if you had a device that converted a serial link into a parallel link, that device would be called a gateway. Similarly, a device that converted Ethernet to Token Ring might be called a gateway. Nowadays, such devices are called *media converters*. (This wouldn't be the first time I've been accused of harkening back to the good old days. Pull up a rocker here on the porch, and have a mint julep while I spin you a yarn.)

Routers usually communicate with each other by means of one or more *routing protocols*. These protocols let the routers learn information about networks other than the ones directly connected to them.

Network devices used to be limited to bridges and routers. Bridges, hubs, and switches operated only on layer two of the OSI stack, and routers only on layer three. Now these devices are often merged into single devices, and routers and switches often operate on all seven layers of the OSI stack.

In today's world, where every device seems to be capable of anything, when should you pick a router rather than a switch? Routers tend to be WAN-centric, while switches tend to be LAN-centric. If you're connecting T1s, you probably want a router. If you're connecting Ethernet, you probably want a switch.

Routing Tables

Routing is a fundamental process common to almost every network in use today. Still, many engineers don't understand how routing works. While the Cisco certification process should help you understand how to configure routing, in this section, I'll show you what you need to know about routing in the real world. I'll focus on the foundations, because that's what most engineers seem to be lacking—we spend a lot of time studying the latest technologies, and sometimes forget the core principles on which everything else is based.

In a Cisco router, the routing table is called the *route information base* (RIB). When you execute the command `show ip route`, the output you receive is a formatted view of the information in the RIB.

Each routing protocol has its own table of information. For example, EIGRP has the topology table, and OSPF has the OSPF database. Each protocol makes decisions on what routes will be held in its database. Routing protocols use their own metrics to determine which route is the best route, and the metrics vary widely. The metric

value is determined by the routing protocol from which the route was learned. Thus, the same link may have very different metrics depending on the protocol used. For example, the same path may be described with a metric of 2 in RIP, 200 in OSPF, and 156160 in EIGRP.

Routing protocols and metrics are covered in more detail in Chapter 10.

If the same route is learned from two sources within a single routing protocol, the one with the best metric will win. Should the same route be learned from two routing protocols within a single router, the protocol with the lowest administrative distance will win. The *administrative distance* is the value assigned to each routing protocol to allow the router to prioritize routes learned from multiple sources. The administrative distances for the various routing protocols are shown in in Table 9-1.

Table 9-1. Routing protocols and their administrative distances

Route type	Administrative distance
Connected interface	0
Static route	1
Enhanced Interior Gateway Routing Protocol (EIGRP) summary route	5
External Border Gateway Protocol (BGP)	20
Internal EIGRP	90
Interior Gateway Routing Protocol (IGRP)	100
Open Shortest Path First (OSPF)	110
Intermediate System–Intermediate System (IS-IS)	115
Routing Information Protocol (RIP)	120
Exterior Gateway Protocol (EGP)	140
On Demand Routing (ODR)	160
External EIGRP	170
Internal BGP	200
Unknown	255

Spanning tree, discussed in Chapter 8, isn't really a routing protocol because the protocol doesn't care about the data being passed; spanning tree is only concerned with loops and with preventing them from a physical and layer-2 perspective. In other words, spanning tree is concerned more with determining that all possible paths within its domain are loop-free than with determining the paths along which data should be sent.

When a packet arrives at a router, the router determines whether the packet needs to be forwarded to another network. If it does, the RIB is checked to see whether it contains a route to the destination network. If there is a match, the packet is adjusted and forwarded on to where it belongs. (See Chapter 15 for more information on this process.) If no match is found in the RIB, the packet is forwarded to the default gateway, if one exists. If no default gateway exists, the packet is dropped.

Originally, the destination network was described by a network address and a subnet mask. Today, destination networks are often described by a network address and a prefix length. The network address is an IP address that references a network. The prefix length is the number of bits set to 1 in the subnet mask. Networks are described in the format *network-address/prefix-length*. For example, the network 10.0.0.0 with a subnet mask of 255.0.0.0 would be described as 10.0.0.0/8. When shown in this format, the route is called simply a *prefix*. The network 10.0.0.0/24 is said to be a longer prefix than the network 10.0.0.0/8. The more bits that are used to identify the network portion of the address, the longer the prefix is said to be.

The RIB may include multiple routes to the same network. For example, in Figure 9-1, R2 learns the network 10.0.0.0 from two sources: R1 advertises the route 10.0.0.0/8 and R3 advertises the route 10.0.0.0/24. Because the prefix lengths are different, these are considered to be different routes. As a result, they will both end up in the routing table.

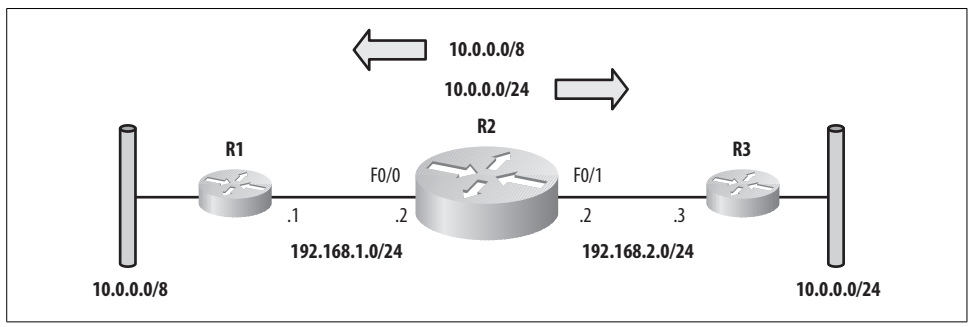


Figure 9-1. Same network with different prefix lengths

Here are the routes as seen in R2:

```
      10.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
D        10.0.0.0/8 [90/30720] via 192.168.1.1, 00:12:01, FastEthernet0/0
D        10.0.0.0/24 [90/30720] via 192.168.2.3, 00:12:01, FastEthernet0/1
```

When a packet is received in R2, the destination IP address is matched against the routing table. If R2 receives a packet destined for 10.0.0.1, which route will it choose? There are two routes in the table that seem to match: 10.0.0.0/8 and 10.0.0.0/24. The route with the longest prefix length (also called the *most specific* route) is the more

desirable route. Thus, when a packet destined for 10.0.0.1 arrives on R2, it will be forwarded to R3. The important thing to realize about this example is that there may be legitimate addresses within the 10.0.0.0/24 range behind R1 that R2 will never be able to access.

Technically, 10.0.0.0/8 is a network, and 10.0.0.0/24 is a subnet. Read on for further clarification.

Route Types

The routing table can contain six types of routes:

Host route
: A host route is a route to a host. In other words, the route is not to a network. Host routes have a subnet mask of 255.255.255.255, and a prefix length of /32.

Subnet
: A subnet is a portion of a major network. The subnet mask is used to determine the size of the subnet. 10.10.10.0/24 (255.255.255.0) is a subnet.

Summary (group of subnets)
: A summary route is a single route that references a group of subnets. 10.10.0.0/16 (255.255.0.0) would be a summary, provided that subnets with longer masks (such as 10.10.10.0/24) existed.

Major network
: A major network is any classful network, along with its native mask. 10.0.0.0/8 (255.0.0.0) is a major network.

Supernet (group of major networks)
: A supernet is single route that references a group of major networks. 10.0.0.0/7 is a supernet that references 10.0.0.0/8 and 11.0.0.0/8.

Default route
: A default route is shown as 0.0.0.0/0 (0.0.0.0). This route is also called the *route of last resort*. This is the route that is used when no other route matches the destination IP address in a packet.

The IP Routing Table

To show the IP routing table, use the `show ip route` command:

```
R2# sho ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
```

```
        E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
        i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
        ia - IS-IS inter area, * - candidate default, U - per-user static route
        o - ODR, P - periodic downloaded static route

Gateway of last resort is 11.0.0.1 to network 0.0.0.0

     172.16.0.0/16 is variably subnetted, 6 subnets, 2 masks
D       172.16.200.0/23 is a summary, 00:56:18, Null0
C       172.16.200.0/24 is directly connected, Loopback2
C       172.16.201.0/24 is directly connected, Serial0/0
C       172.16.202.0/24 is directly connected, Loopback3
C       172.16.100.0/23 is directly connected, Loopback4
D       172.16.101.0/24 [90/2172416] via 11.0.0.1, 00:53:07, FastEthernet0/1
C    10.0.0.0/8 is directly connected, FastEthernet0/0
C    11.0.0.0/8 is directly connected, FastEthernet0/1
     192.168.1.0/32 is subnetted, 1 subnets
D       192.168.1.11 [90/156160] via 11.0.0.1, 00:00:03, FastEthernet0/1
S*   0.0.0.0/0 [1/0] via 11.0.0.1
D    10.0.0.0/7 is a summary, 00:54:40, Null0
```

The first block of information is shown every time the command is executed. In the interest of brevity, I will remove it from most of the examples in this book. This block is a key that explains the codes listed down the left side of the routing table.

The next line lists the default gateway, if one is present:

```
Gateway of last resort is 11.0.0.1 to network 0.0.0.0
```

If there are two or more default gateways, they will all be listed. This is common when the default gateway is learned from a routing protocol that allows equal-cost load sharing. If two links provide access to the advertised default, and they both have the same metric, they will both be listed as default routes. In this case, packets will be equally balanced between the two links using per-packet load balancing.

If no default gateway has been configured or learned, you'll instead see this message:

```
Gateway of last resort is not set
```

The next block of text contains the rest of the routing table:

```
     172.16.0.0/16 is variably subnetted, 6 subnets, 2 masks
D       172.16.200.0/23 is a summary, 00:56:18, Null0
C       172.16.200.0/24 is directly connected, Loopback2
C       172.16.201.0/24 is directly connected, Serial0/0
C       172.16.202.0/24 is directly connected, Loopback3
C       172.16.100.0/23 is directly connected, Loopback4
D       172.16.101.0/24 [90/2172416] via 11.0.0.1, 00:53:07, FastEthernet0/1
C    10.0.0.0/8 is directly connected, FastEthernet0/0
C    11.0.0.0/8 is directly connected, FastEthernet0/1
     192.168.1.0/32 is subnetted, 1 subnets
D       192.168.1.11 [90/156160] via 11.0.0.1, 00:00:03, FastEthernet0/1
S*   0.0.0.0/0 [1/0] via 11.0.0.1
D    10.0.0.0/7 is a summary, 00:54:40, Null0
```

Let's examine a single entry from the routing table, so you can see what's important:

```
D       172.16.101.0/24 [90/2172416] via 11.0.0.1, 00:53:07, FastEthernet0/1
```

First is the route code. In this case it's `D`, which indicates that the route was learned via EIGRP. (You can look this up in the block of codes at the top of the `show ip route` output.)

Next is the route itself. In this example, the route is to the subnet 172.16.101.0/24. After that are two numbers in brackets: the first number is the administrative distance (see Table 9-1), and the second number is the metric for the route. The metric is determined by the routing protocol from which the route was learned (in this case, EIGRP).

The next piece of information is the next hop the router needs to send packets to in order to reach this subnet. In this case, `via 11.0.0.1` indicates that packets destined for the subnet 172.16.101.0/24 should be forwarded to the IP address 11.0.0.1. Finally, you have the age of the route (`00:53:07`), followed by the interface out which the router will forward the packet (`FastEthernet0/1`).

I've built the sample router so that the routing table will have one of each type of route. Again, those route types are *host*, *subnet*, *summary*, *major network*, *supernet*, and *default*. The following sections explain the types in more detail. I'll show the routing table entries for each type in bold.

Host Route

A host route is simply a route with a subnet mask of all ones (255.255.255.255), or a prefix length of /32. In the sample routing table, the route to 192.168.1.11 is a host route:

```
     172.16.0.0/16 is variably subnetted, 6 subnets, 2 masks
D       172.16.200.0/23 is a summary, 00:56:18, Null0
C       172.16.200.0/24 is directly connected, Loopback2
C       172.16.201.0/24 is directly connected, Serial0/0
C       172.16.202.0/24 is directly connected, Loopback3
C       172.16.100.0/23 is directly connected, Loopback4
D       172.16.101.0/24 [90/2172416] via 11.0.0.1, 00:53:07, FastEthernet0/1
C    10.0.0.0/8 is directly connected, FastEthernet0/0
C    11.0.0.0/8 is directly connected, FastEthernet0/1
     192.168.1.0/32 is subnetted, 1 subnets
D       192.168.1.11 [90/156160] via 11.0.0.1, 00:00:03, FastEthernet0/1
S*   0.0.0.0/0 [1/0] via 11.0.0.1
D    10.0.0.0/7 is a summary, 00:54:40, Null0
```

Notice that the route is shown to be a part of a larger network (in this case, 192.168.1.0). We know this because the host route is shown indented under the major network. The router will attempt to show you what classful (major) network contains the route. If the router only knows about a single subnet mask, it will assume that the network has been divided equally with that mask. In this case, the router has

assumed that the major network 192.168.1.0/24 has been equally subnetted, with each subnet having a /32 mask. Hence, the natively /24 network 192.168.1.0 is shown as 192.168.1.0/32.

Subnet

Subnets are shown indented under their source major networks. In our example, the major network 172.16.0.0/16 has been subnetted; in fact, it has been subnetted under the rules of Variable Length Subnet Masks (VLSM), which allow each subnet to have a different subnet mask (within certain limits—see Chapter 34 for more detail). The one route in the middle that is not in bold is a summary route, which I'll cover next.

```
     172.16.0.0/16 is variably subnetted, 6 subnets, 2 masks
D       172.16.200.0/23 is a summary, 00:56:18, Null0
C       172.16.200.0/24 is directly connected, Loopback2
C       172.16.201.0/24 is directly connected, Serial0/0
C       172.16.202.0/24 is directly connected, Loopback3
C       172.16.100.0/23 is directly connected, Loopback4
D       172.16.101.0/24 [90/2172416] via 11.0.0.1, 00:53:07, FastEthernet0/1
C    10.0.0.0/8 is directly connected, FastEthernet0/0
C    11.0.0.0/8 is directly connected, FastEthernet0/1
     192.168.1.0/32 is subnetted, 1 subnets
D       192.168.1.11 [90/156160] via 11.0.0.1, 00:00:03, FastEthernet0/1
S*   0.0.0.0/0 [1/0] via 11.0.0.1
D    10.0.0.0/7 is a summary, 00:54:40, Null0
```

Summary (Group of Subnets)

The term *summary* is used in the routing table to represent any group of routes. Technically, according to the Cisco documentation, a summary is a group of subnets, while a supernet is a group of major networks. Both are called summaries in the routing table. Thus, while the example routing table shows two summary entries, only the first is technically a summary route:

```
     172.16.0.0/16 is variably subnetted, 6 subnets, 2 masks
D       172.16.200.0/23 is a summary, 00:56:18, Null0
C       172.16.200.0/24 is directly connected, Loopback2
C       172.16.201.0/24 is directly connected, Serial0/0
C       172.16.202.0/24 is directly connected, Loopback3
C       172.16.100.0/23 is directly connected, Loopback4
D       172.16.101.0/24 [90/2172416] via 11.0.0.1, 00:53:07, FastEthernet0/1
C    10.0.0.0/8 is directly connected, FastEthernet0/0
C    11.0.0.0/8 is directly connected, FastEthernet0/1
     192.168.1.0/32 is subnetted, 1 subnets
D       192.168.1.11 [90/156160] via 11.0.0.1, 00:00:03, FastEthernet0/1
S*   0.0.0.0/0 [1/0] via 11.0.0.1
D    10.0.0.0/7 is a summary, 00:54:40, Null0
```

The last entry in the routing table, which is also reported as a summary, is a group of major networks, and is technically a supernet.

The differentiation between supernets and summary routes is primarily an academic one. In the real world, both are routinely called summary routes or aggregate routes. (Different routing protocols use different terms for groups of routes, be they subnets or major networks—BGP uses the term "aggregate," while OSPF uses the term "summary.")

The destination for both summary routes is `Null0`. `Null0` as a destination indicates that packets sent to this network will be dropped. The summary routes point to `Null0` because they were created within EIGRP on this router.

The `Null0` route is there for the routing protocol's use. The more specific routes must also be included in the routing table because the local router must use them when forwarding packets. The specific routes will not be advertised in the routing protocol—only the summary will be advertised. We can see this if we look at an attached router:

```
     172.16.0.0/16 is variably subnetted, 4 subnets, 2 masks
D       172.16.200.0/23 [90/156160] via 11.0.0.2, 04:30:21, FastEthernet0/1
D       172.16.202.0/24 [90/156160] via 11.0.0.2, 04:30:21, FastEthernet0/1
D       172.16.100.0/23 [90/156160] via 11.0.0.2, 04:30:21, FastEthernet0/1
C       172.16.101.0/24 is directly connected, Serial0/0
```

On the connected router, the summary route for 172.16.200.0/23 is present, but the more specific routes 172.16.200.0/24 and 172.16.201.0/24 are not.

Major Network

A major network is a network that is in its native form. For example, the 10.0.0.0/8 network has a native subnet mask of 255.0.0.0. The network 10.0.0.0/8 is therefore a major network. Referencing 10.0.0.0 with a prefix mask longer than /8 changes the route to a subnet, while referencing it with a mask shorter than /8 changes the route to a supernet.

Two major networks are shown in the routing table:

```
     172.16.0.0/16 is variably subnetted, 6 subnets, 2 masks
D       172.16.200.0/23 is a summary, 00:56:18, Null0
C       172.16.200.0/24 is directly connected, Loopback2
C       172.16.201.0/24 is directly connected, Serial0/0
C       172.16.202.0/24 is directly connected, Loopback3
C       172.16.100.0/23 is directly connected, Loopback4
D       172.16.101.0/24 [90/2172416] via 11.0.0.1, 00:53:07, FastEthernet0/1
C    10.0.0.0/8 is directly connected, FastEthernet0/0
C    11.0.0.0/8 is directly connected, FastEthernet0/1
     192.168.1.0/32 is subnetted, 1 subnets
```

```
D       192.168.1.11 [90/156160] via 11.0.0.1, 00:00:03, FastEthernet0/1
S*   0.0.0.0/0 [1/0] via 11.0.0.1
D    10.0.0.0/7 is a summary, 00:54:40, Null0
```

172.16.0.0/16 is also shown, but only as a reference to group all of the subnets underneath it. The entry for 172.16.0.0/16 is not a route.

Supernet (Group of Major Networks)

A supernet is a group of major networks. In this example, there is a route to 10.0.0.0/7, which is a group of the major networks 10.0.0.0/8 and 11.0.0.0/8:

```
     172.16.0.0/16 is variably subnetted, 6 subnets, 2 masks
D       172.16.200.0/23 is a summary, 00:56:18, Null0
C       172.16.200.0/24 is directly connected, Loopback2
C       172.16.201.0/24 is directly connected, Serial0/0
C       172.16.202.0/24 is directly connected, Loopback3
C       172.16.100.0/23 is directly connected, Loopback4
D       172.16.101.0/24 [90/2172416] via 11.0.0.1, 00:53:07, FastEthernet0/1
C    10.0.0.0/8 is directly connected, FastEthernet0/0
C    11.0.0.0/8 is directly connected, FastEthernet0/1
     192.168.1.0/32 is subnetted, 1 subnets
D       192.168.1.11 [90/156160] via 11.0.0.1, 00:00:03, FastEthernet0/1
S*   0.0.0.0/0 [1/0] via 11.0.0.1
D    10.0.0.0/7 is a summary, 00:54:40, Null0
```

Notice the route is again destined to `Null0`. Sure enough, on a connected router we will only see the summary, and not the more specific routes:

```
D    10.0.0.0/7 [90/30720] via 11.0.0.2, 04:30:22, FastEthernet0/1
```

Default Route

The default route, or "route of last resort" is shown in a special place above the routing table, so it can easily be seen:

```
Gateway of last resort is 11.0.0.1 to network 0.0.0.0

     172.16.0.0/16 is variably subnetted, 6 subnets, 2 masks
D       172.16.200.0/23 is a summary, 00:56:18, Null0
C       172.16.200.0/24 is directly connected, Loopback2
C       172.16.201.0/24 is directly connected, Serial0/0
C       172.16.202.0/24 is directly connected, Loopback3
C       172.16.100.0/23 is directly connected, Loopback4
D       172.16.101.0/24 [90/2172416] via 11.0.0.1, 00:53:07, FastEthernet0/1
C    10.0.0.0/8 is directly connected, FastEthernet0/0
C    11.0.0.0/8 is directly connected, FastEthernet0/1
     192.168.1.0/32 is subnetted, 1 subnets
D       192.168.1.11 [90/156160] via 11.0.0.1, 00:00:03, FastEthernet0/1
S*   0.0.0.0/0 [1/0] via 11.0.0.1
D    10.0.0.0/7 is a summary, 00:54:40, Null0
```

In this case, the default route is a static route, as shown by the S in the first column, but it could be learned from a routing protocol as well. The asterisk next to the S indicates this route is a candidate for the default route. There can be more than one candidate, in which case there will be multiple entries with asterisks. There can even be multiple default routes, but only one will be listed in the first line.

This output shows a router with two active default gateways, though only one is listed in the first line:

```
Gateway of last resort is 10.0.0.1 to network 0.0.0.0

     20.0.0.0/24 is subnetted, 1 subnets
S       20.0.0.0 [1/0] via 10.0.0.1
     10.0.0.0/24 is subnetted, 3 subnets
C       10.0.0.0 is directly connected, FastEthernet0/0
C    192.168.1.0/24 is directly connected, FastEthernet0/1
S*   0.0.0.0/0 [1/0] via 10.0.0.1
                [1/0] via 10.0.0.2
```

When in doubt, look at the 0.0.0.0/0 entry in the routing table, as it will always have the most accurate information.

CHAPTER 10
Routing Protocols

A routing protocol is a means whereby devices interchange information about the state of the network. The information collected from other devices is used to make decisions about the best path for packets to flow to each destination network.

Routing protocols are applications that reside at layer seven in the OSI model. There are many routing protocols in existence, though only a few are in common use today. Older protocols are rarely used, though some networks may contain legacy devices that support only those protocols. Some firewalls and servers may support a limited scope of routing protocols—most commonly RIP and OSPF—but for the sake of simplicity, I will refer to all devices that participate in a routing protocol as *routers*.

Routing protocols allow networks to be dynamic and resistant to failure. If all routes in a network were static, the only form of dynamic routing we would be able to employ would be the *floating static route*. A floating static route is a route that becomes active only if another static route is removed from the routing table. Here's an example:

```
ip route 0.0.0.0 0.0.0.0 192.168.1.1 1
ip route 0.0.0.0 0.0.0.0 10.0.0.1 2
```

The primary default route points to 192.168.1.1, and has a metric of 1. The second default route points to 192.168.1.2, and has a metric of 2.

Routes with the best metrics are inserted into the routing table, so in this case, the first route will win. Should the network 192.168.1.0 become unavailable, all routes pointing to it will be removed from the routing table. At this time, the default route to 10.0.0.1 will be inserted into the routing table, since it now has the best metric for the 0.0.0.0/0 network.

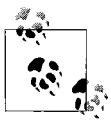
The floating static route allows routes to change if a directly connected interface goes down, but it cannot protect routes from failing if a remote device or link fails. *Dynamic* routing protocols usually allow all routers participating in the protocol to learn about any failures on the network. This is achieved through regular communication between routers.

Communication Between Routers

Routers need to communicate with one another to learn the state of the network. One of the original routing protocols, the Routing Information Protocol (RIP), sent out updates about the network using broadcasts. This was fine for smaller networks, but as networks grew, these broadcasts became troublesome. Every host on a network listens to broadcasts, and with RIP, the broadcasts could be quite large.

Most modern routing protocols communicate on broadcast networks using *multicast packets*. Multicast packets are packets with specific IP and corresponding MAC addresses that reference predetermined groups of devices.

Because routing is usually a dynamic process, existing routers must be able to discover new routers to add their information into the tables that describe the network. For example, all EIGRP routers within the same domain must be able to communicate with each other. Defining specific neighbors is not necessary with this protocol because they are discovered dynamically.



Most interior gateway protocols discover neighbors dynamically. BGP does not discover neighbors. Instead, BGP must be configured to communicate with each neighbor manually.

The Internet Assigned Numbers Authority (IANA) shows all multicast addresses in use at *http://www.iana.org/assignments/multicast-addresses*. Some of the more common multicast addresses include:

```
224.0.0.0  Base Address (Reserved)                  [RFC1112,JBP]
224.0.0.1  All Systems on this Subnet               [RFC1112,JBP]
224.0.0.2  All Routers on this Subnet                       [JBP]
224.0.0.4  DVMRP    Routers                         [RFC1075,JBP]
224.0.0.5  OSPFIGP  OSPFIGP All Routers            [RFC2328,JXM1]
224.0.0.6  OSPFIGP  OSPFIGP Designated Routers     [RFC2328,JXM1]
224.0.0.9  RIP2 Routers                           [RFC1723,GSM11]
224.0.0.10 IGRP Routers                              [Farinacci]
224.0.0.12 DHCP Server / Relay Agent                   [RFC1884]
224.0.0.18 VRRP                                       [RFC3768]
224.0.0.102 HSRP                                       [Wilson]
```

The list shows that all IGRP routers, including Enhanced IGRP routers, will listen to packets sent to the address 224.0.0.10.

Not all routing protocols use multicasts to communicate. Because BGP does not discover neighbors, it has no need for multicasts, and instead uses unicast packets. Many other routing protocols can also be configured to statically assign neighbors. This usually results in unicast messages being sent to specific routers instead of multicasts.

There may be more than one type of routing protocol on a single network. In the Ethernet network shown in Figure 10-1, for example, there are five routers, three of which are running OSPF, and two of which are running EIGRP. There is no reason for the EIGRP routers to receive OSPF updates, or vice versa. Using multicasts ensures that only the routers that are running the same routing protocols communicate with and discover each other.

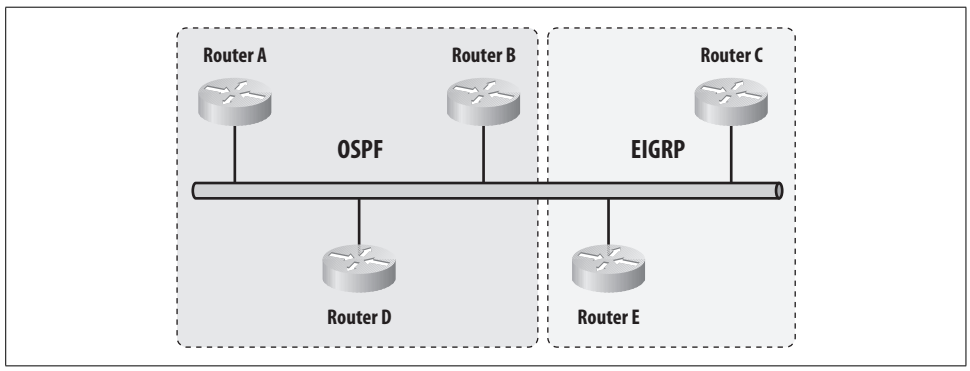


Figure 10-1. Multiple routing protocols on a single Ethernet network

A network may also contain multiple instances of the same routing protocol. These separate areas of control are called *autonomous systems* in EIGRP, and *processes* in OSPF (although the term autonomous system is often used incorrectly). Each instance is referenced with a number—either an autonomous system number (ASN), or a process ID (PID).

Figure 10-2 shows a network with two OSPF processes active. Because the multicast packets sent by an OSPF router will be destined for All OSPF Routers, all OSPF routers will listen to the updates. The updates contain the PIDs, so the individual routers can determine whether to retain or discard them. (RIP does not support the idea of separate processes, so any router running RIP will receive and process updates from all other RIP routers on the network.)

When there are two processes on the same network, the routes learned in each are not shared between the processes by default. For routes to be shared, one of the routers must participate in both processes, and be configured to share the routes between them.

The act of passing routes from one process or routing protocol to another process or routing protocol is called *redistribution*. An example of multiple OSPF routing processes being redistributed is shown in Figure 10-3.

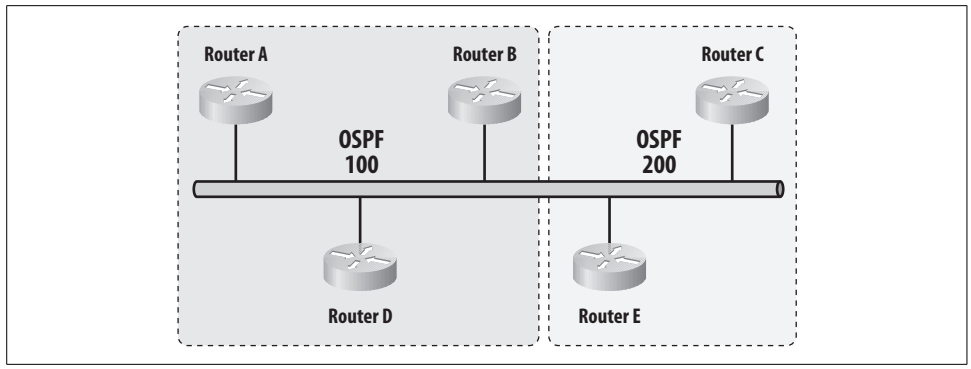


Figure 10-2. Two OSPF processes on a single network

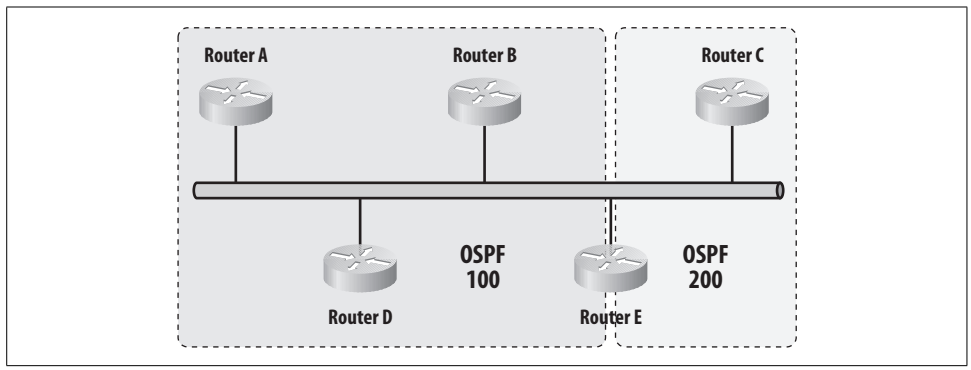


Figure 10-3. Routing protocol redistribution

In Figure 10-2, we had two OSPF processes, but there was no way for the processes to learn each other's routes. In Figure 10-3, Router E is configured to be a member of OSPF process 100 and OSPF process 200. Router E thus redistributes routes learned on each process into the other process.

When a route is learned within a routing process, the route is said to be *internal*. When a route is learned outside the routing process, and redistributed into the process, the route is said to be *external*. Internal routes are usually considered to be more reliable than external routes, by a means called *administrative distance*. Exceptions include BGP, which prefers external routes over internal ones, and OSPF, which does not assign different administrative distances to internal versus external routes.

Metrics and Protocol Types

The job of a routing protocol is to determine the best path to a destination network. The best route is chosen based on a protocol-specific set of rules. RIP uses the number of hops (routers) between networks, whereas OSPF calculates the cost of a route based on the bandwidth of all the links in the network. EIGRP uses links' reported bandwidths and delays to determine the best path by default, and it can be configured to use a few more factors as well. Each of these protocols determines a value for each route. This value is usually called a *metric*. Routes with lower metrics are more desirable.

Perhaps the simplest form of metric to understand is the one used by RIP: hop count. In RIP, the hop count is simply the number of routers between the router determining the path and the network to be reached.

Let's consider an example. In Figure 10-4, there are two networks, labeled 10.0.0.0, and 20.0.0.0. Router A considers 20.0.0.0 to be available via two paths: one through Router B, and one through Router E. The path from Router A through Router B traverses Routers B, C, and D, resulting in a hop count of 3 for this path. The path from Router A through Router E traverses routers E, F, G, and D, resulting in a hop count of 4 for this path.

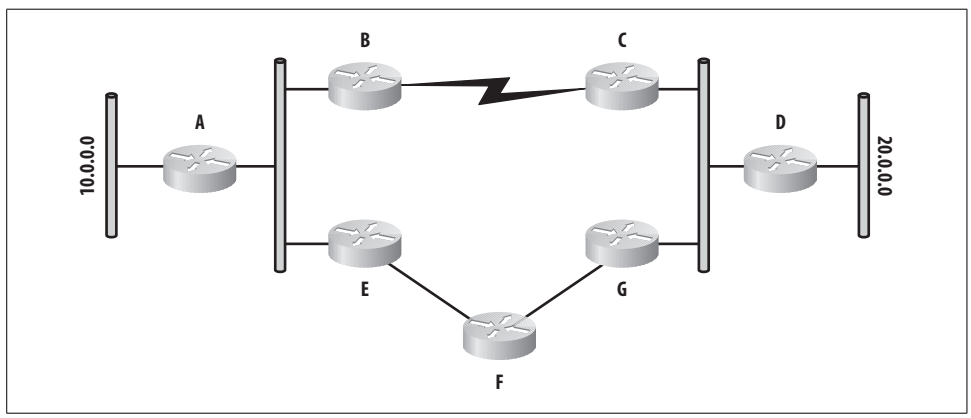


Figure 10-4. Example of metrics in routing protocols

Lower metrics always win, so Router A will consider the path through Router B to be the better path. This router will be added to the routing table with a metric of 3.

Using hop count as a metric has a limitation that can cause suboptimal paths to be chosen. Looking at Figure 10-5, you can see that the link between Routers B and C is a T1 running at 1.54 Mbps, while the links between Routers E, F, and G are all direct fiber links running at 1 Gbps. That means that the path through Routers E, F, and G

will be substantially faster than the link between Routers B and C, even though that link has fewer hops. However, RIP doesn't know about the bandwidth of the links in use, and takes into account only the number of hops.

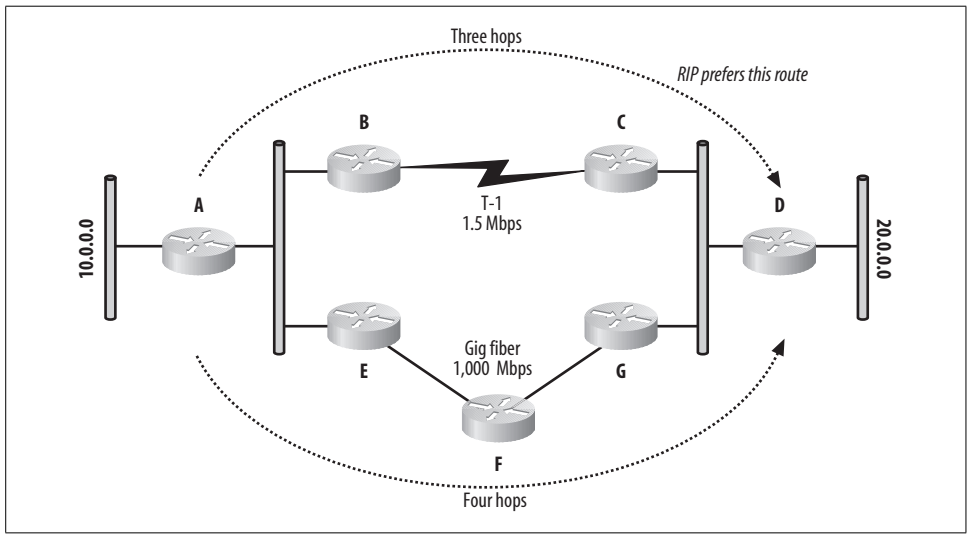


Figure 10-5. RIP uses hops to determine the best routes

A protocol such as RIP is called a *distance-vector* routing protocol, as it relies on the distance to the destination network to determine the best path. Distance-vector protocols suffer from another problem called *counting to infinity*. Protocols such as RIP place an upper limit on the number of hops allowed to reach a destination. Hop counts in excess of this number are considered to be unreachable. In RIP, the maximum hop count is 15, with a hop count of 16 being unreachable. As you might imagine, this does not scale well in modern environments, where there may easily be more than 16 routers in a given path. A more modern version of RIP called RIP Version 2 (RIPv2 or RIP2) raises the limit to 255 hops, with 256 being unreachable. However, since RIPv2 still doesn't understand the states and capabilities of the links that join the hops together, most networks employ newer, more robust routing protocols instead.

Routing protocols such as OSPF are called *link-state* routing protocols. These protocols include information about the links between the source router and destination network, as opposed to simply counting the number of routers between them.

OSPF adds up the *cost* of each link. The cost of a link is determined as 100,000,000 divided by the bandwidth of the link in bits per second (bps). The costs of some common links are therefore:

100 Mbps (100,000,000 / 100,000,000 bps) = 1
10 Mbps (100,000,000 / 10,000,000 bps) = 10
1.5 Mbps (100,000,000 / 1,540,000 bps) = 64 (results are rounded)

Figure 10-6 shows the same network used in the RIP example. This time, OSPF is determining the best path to the destination network using bandwidth-based metrics. The metric for the T1 link is 64, and the metric for the gigabit link path is 4. Because the metric for the link through Routers E, F, and G is lower than that for the link through Routers B and C, this path is the path inserted into the routing table.

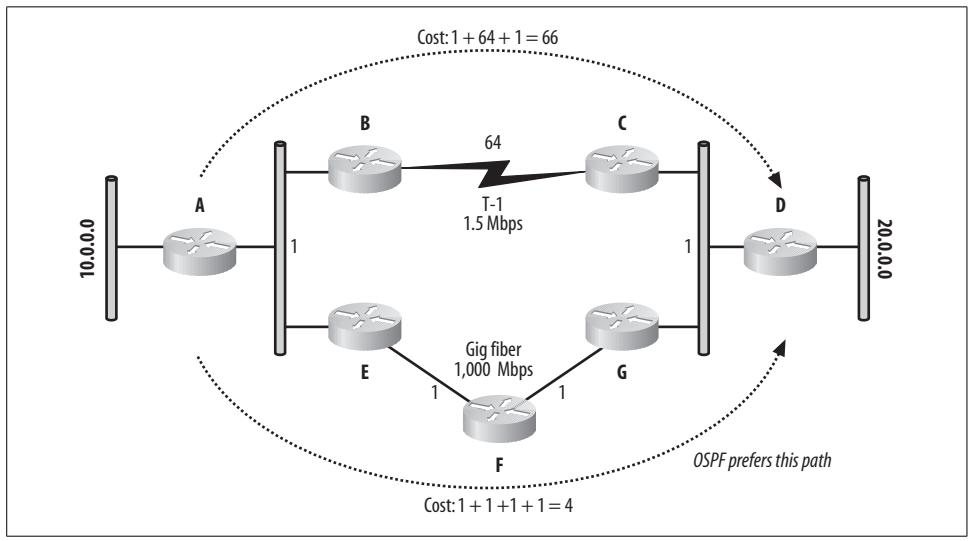


Figure 10-6. OSPF uses bandwidth to determine the best routes

EIGRP uses a more complicated formula for determining costs. It can include bandwidth, delay, reliability, effective bandwidth, and Maximum Transmission Unit (MTU) in its calculation of a metric. EIGRP is considered to be a *hybrid* protocol.

Administrative Distance

Networks often have more than one routing protocol active. In such situations, there is a high probability that the same networks will be advertised by multiple routing protocols. Figure 10-7 shows a network in which two routing protocols are running: the top half of the network is running RIP, and the bottom half is running OSPF. Router A will receive routes for the network 20.0.0.0 from RIP and OSPF. RIP's route has a better metric, but as we've seen, OSPF has a better means of determining the proper path. So, how is the best route determined?

Routers choose routes based on a predetermined set of rules. One of the factors in deciding which route to place in the routing table is *administrative distance* (AD). Administrative distance is a value assigned to every routing protocol. In the event of two protocols reporting the same route, the routing protocol with the lowest administrative distance will win.

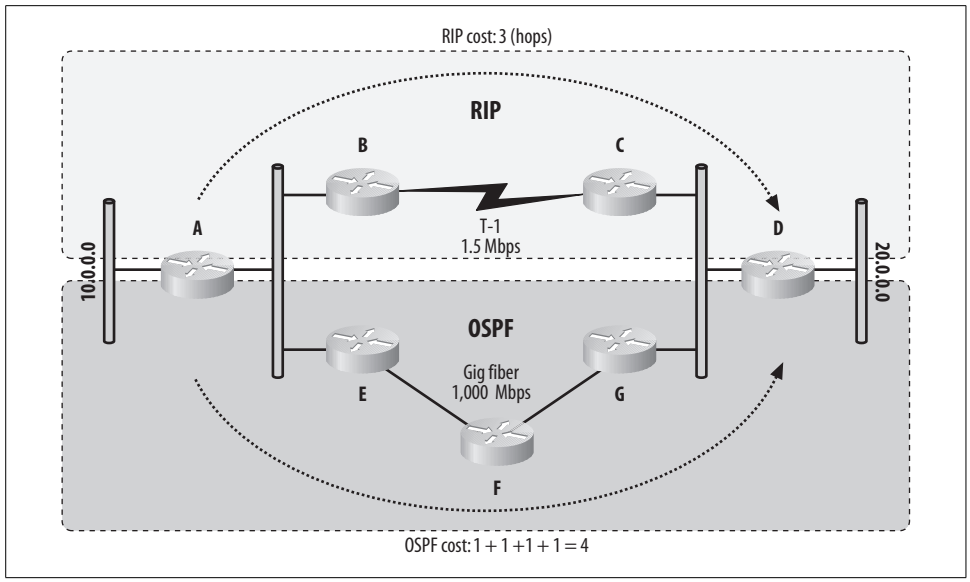


Figure 10-7. Competing routing protocols

The administrative distances of the various routing protocols are shown in Table 10-1.

Table 10-1. Administrative distances of routing protocols

Route type	Administrative distance
Connected interface	0
Static route	1
EIGRP summary route	5
External BGP	20
Internal EIGRP	90
IGRP	100
OSPF	110
IS-IS	115
RIP	120
EGP	140
ODR	160
External EIGRP	170
Internal BGP	200
Unknown	255

A static route to a connected interface has an administrative distance of 0, and is the only route that will override a normal static route. A route sourced with an administrative distance of 255 is not trusted, and will not be inserted into the routing table.

Looking at Table 10-1, you can see that RIP has an AD of 120, while OSPF has an AD of 110. This means that even though the RIP route has a better metric, in Figure 10-7, the route inserted into the routing table will be the one provided by OSPF.

Specific Routing Protocols

Entire books can and have been written on each of the routing protocols discussed in this chapter. My goal is not to teach you everything you need to know about the protocols, but rather, to introduce them and show you what you need to know to get them operational. I'll also include some of the commands commonly used to troubleshoot these protocols.

Routing protocols are divided into types based on their purpose and how they operate. The major division between routing protocols is that of *internal gateway protocols* versus *external gateway protocols*.

An internal gateway protocol, or IGP, is designed to maintain routes within an *autonomous system*. An autonomous system is any group of devices controlled by a single entity. An example might be a company or a school, but the organization does not need to be that broad—an autonomous system could be a floor in a building or a department in a company. Examples of IGPs include RIP, EIGRP, and OSPF.

An external gateway protocol, or EGP, is designed to link autonomous systems together. The Internet is the prime example of a large-scale EGP implementation. The autonomous systems—groups of devices controlled by individual service providers, schools, companies, etc.—are each self-contained. They are controlled internally by IGPs, and are interconnected using an EGP (in the case of the Internet, BGP).

Figure 10-8 shows how different autonomous systems might be connected. Within each circle is an autonomous system. The IGP running in each autonomous system is irrelevant to the external gateway protocol. The EGP knows only that a certain network is owned by a certain autonomous system. Let's say that 1.0.0.0/8 is within ASN 1, 2.0.0.0/8 is within ASN 2, 3.0.0.0/8 is within ASN 3, and so on. For a device in ASN 1 to get to the network 10.0.0.0/8, the path might be through autonomous systems 1, 2, 3, 9, and 10. It might also be through autonomous systems 1, 2, 7, 8, 9, and 10, or even 1, 2, 7, 8, 3, 9, and 10. As with a distance-vector IGP counting hops, the fewer the number of autonomous systems traversed, the more appealing the route.

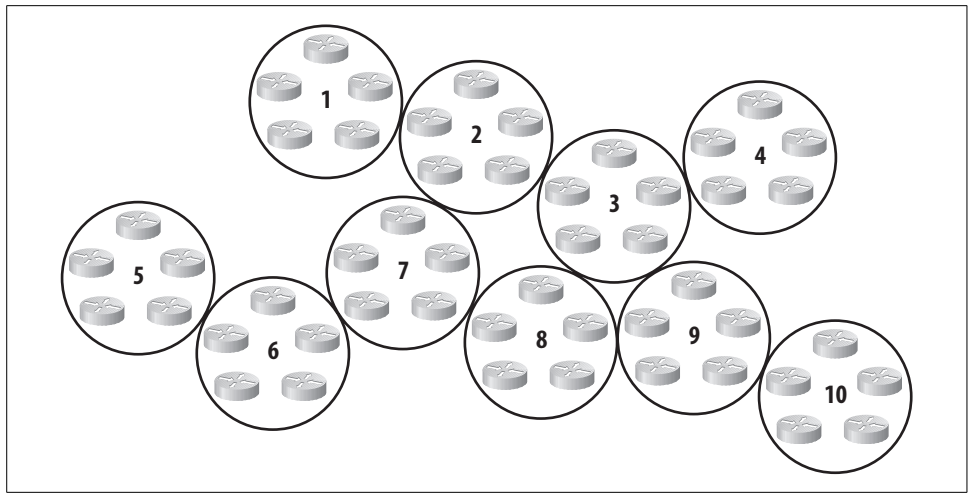


Figure 10-8. Interconnected autonomous systems

The important thing to remember with external gateway protocols is that they really don't care how many routers there are, or what the speeds of the links are. The only thing an external gateway protocol cares about is traversing the least possible number of autonomous systems in order to arrive at a destination.

Before we go any further, let's define some key routing terms:

Classful routing protocol
: A classful routing protocol is one that has no provision to support subnets. The natural state of the network is always advertised. For example, the network 10.0.0.0 will always be advertised with a subnet mask of 255.0.0.0 (/8), regardless of what subnet mask is actually in use. RIPv1 and IGRP are classful routing protocols.

Classless routing protocol
: A classless routing protocol is one that includes subnet masks in its advertisements. All modern protocols are classless. EIGRP and OSPF are classless routing protocols.

Poison reverse
: If a router needs to tell another router that a network is no longer viable, one of the methods employed is *route poisoning*. Consider RIPv1 as an example. Recall that a metric of 16 is considered unreachable. A router can send an update regarding a network with a metric of 16, thereby *poisoning* the entry in the routing table of the receiving router. When a router receives a poison update, it returns the same update to the sending router. This reflected route poisoning is called *poison reverse*. Distance-vector routing protocols (including the hybrid protocol EIGRP) use route poisoning, while link-state protocols such as OSPF do not.

Split horizon
: Split horizon is a technique used by many routing protocols to prevent routing loops. When split horizon is enabled, routes that the routing protocol learns are not advertised out the same interfaces from which they were learned. This rule can be problematic in virtual circuit topologies, such as frame relay or ATM. If a route is learned on one permanent virtual circuit (PVC) in a frame-relay interface, chances are the other PVC needs the update, but will never receive it because both PVCs exist on the same physical interface. Frame-relay subinterfaces are often the preferred method of dealing with split horizon issues.

Convergence
: A network is said to be *converged* when all of the routers in the network have received and processed all updates. Essentially, this condition exists when a network is stable. Any time a link's status changes, the routing protocols must propagate that change, whether through timed updates, or triggered updates. With timed updates, if updates are sent, but no changes need to be made, the network has converged.

There are many routing protocols in existence, but luckily, only a few are in widespread use. Each has its own idiosyncrasies. In the following sections, I'll cover the basic ideas behind the more common protocols, and show how to configure them for the most commonly seen scenarios. There is no right or wrong way to configure routing protocols, though some ways are certainly better than others. When designing any network, remember that simplicity is a worthy goal that will save you countless hours of troubleshooting misery.

RIP

The Routing Information Protocol (RIP) is the simplest of the routing protocols in use today. While I like to tell my clients that *simple is good*, I don't consider RIP to be simple goodness.

RIP broadcasts all the routes it knows about every 30 seconds, regardless of the statuses of any other routers in the network. Because it uses broadcasts, every host on the network listens to the updates, even though few can process them. On larger networks, the updates can be quite large, and consume a lot of bandwidth on expensive WAN links.

Another issue with RIP is the fact that it does not use triggered updates. A *triggered update* is one that is sent when the network changes; *nontriggered* (*timed*) *updates* are sent on a regular schedule. This, coupled with the fact that updates are only sent every 30 seconds, causes RIP networks to converge very slowly. Slow convergence

is not acceptable in most modern networks, which require failover and convergence in seconds. A RIP network with only five routers may take two minutes or more to converge.

RIP is a classful protocol, which means that subnet masks are not advertised. This is also an unacceptable limitation in most networks. Figure 10-9 illustrates one of the most common mistakes made when using classful protocols such as RIP. Router A is advertising its directly connected network 10.10.10.0/24, but because RIP is classful it advertises the network without the subnet mask. Without a subnet mask, the receiving router must assume that the entire network is included in the advertisement. Consequently, upon receiving the advertisement for 10.0.0.0 from Router A, Router B inserts the entire 10.0.0.0/8 network into its routing table. Router C has a different 10 network attached: 10.20.20.0/24. Again, RIP advertises the 10.0.0.0 network from Router C without a subnet mask. Router B has now received another advertisement for 10.0.0.0/8. The network is the same, the protocol is the same, and the hop count is the same. When a newer update is received for a route that has already been inserted into the routing table, the newer update is considered to be more reliable, and is itself inserted into the routing table, overwriting the previous entry. This means that each time Router B receives an update from Router A or Router C, it will change its routing table to show that network 10.0.0.0 is behind the router from which it received the update.

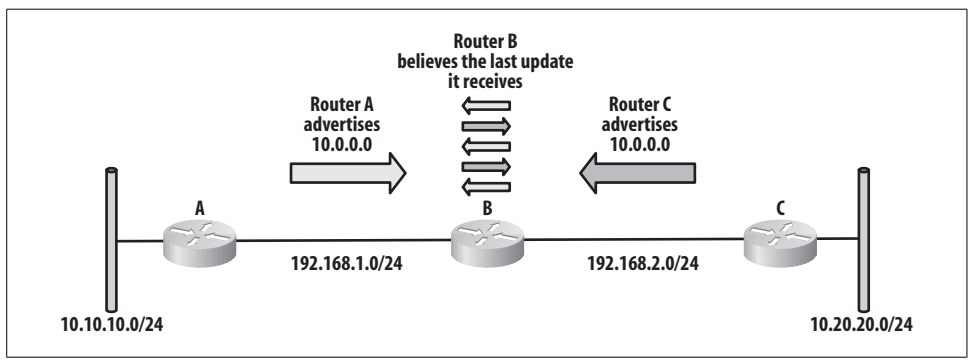


Figure 10-9. RIP classful design problem

You might be tempted to say that the networks behind Routers A and C in Figure 10-9 are different, but from RIP's point of view, you would be wrong. Technically, the *networks* behind Routers A and C are the same. They are both part of the 10.0.0.0/8 network. The routers are connected to different *subnets* within the 10.0.0.0/8 network, which is why RIP has a problem with the design.

The only other type of network that RIP understands is a host network. RIP can advertise a route for a /32 or 255.255.255.255 network. Because RIP does not include subnet masks in its updates, a route is determined to be a host route when the address of the network is other than a normal network address.

Routing protocols are configured in IOS using the `router` command. The protocol name is included in the command, which puts the router into router configuration mode:

```
Router-A (config)# router rip
Router-A (config-router)#
```

On modern routers that support RIPv2, if you wish to use RIPv1, you must specify it explicitly in the router configuration because RIPv2 is used by default:

```
router rip
 version 1
```

By default, no interfaces are included in the routing protocol. This means that no interfaces will have routing updates sent on them, and any routing updates received on the interfaces will be ignored.

To enable interfaces in a routing protocol, you specify the networks that are configured on the interfaces you wish to include. This task is accomplished with the `network` command in router configuration mode:

```
Router-A (config)# router rip
Router-A (config-router)# network 10.10.10.0
```

With a classful protocol like RIP, you must be careful because, as an example, including the network 10.0.0.0 will include every interface configured with a 10.x.x.x IP address, regardless of subnet mask. RIP does not allow the inclusion of a subnet or inverse mask in the `network` statements. You can enter a network other than 10.0.0.0, but IOS will convert the entry into the full classful network.

The preceding entry results in the following being shown in the configuration:

```
router rip
 network 10.0.0.0
```

The configuration that would include all interfaces for Router A as shown in Figure 10-10 would be as follows:

```
router rip
 version 1
 network 10.0.0.0
 network 192.168.1.0
 network 192.168.2.0
```

One entry covers both the 10.10.10.0 and 10.20.20.0 networks, but the 192.168 networks each require their own `network` statements. This is because 192.x.x.x networks are class C networks, while 10.x.x.x networks are class A networks.

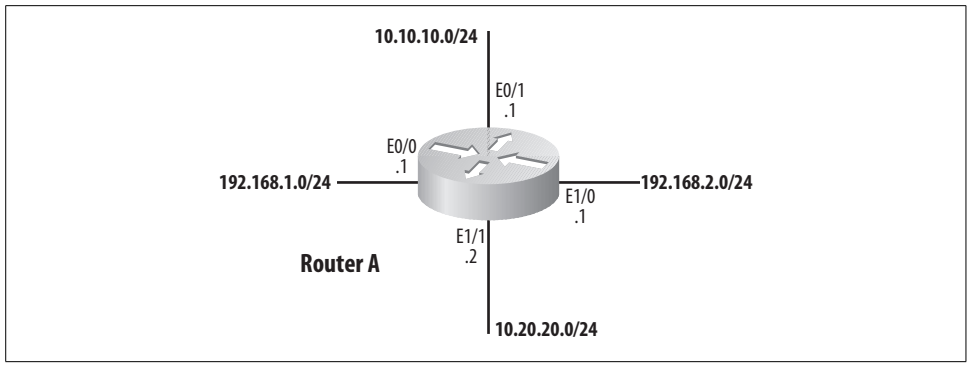


Figure 10-10. Routing protocol network interfaces

You won't always want to include every interface that the `network` statement encompasses. In the preceding example, we might want to allow RIP on E0/0, but not on E1/1. This can be accomplished with the use of the `passive-interface` command, which removes an interface from the broader range specified by the `network` command:

```
router rip
 version 1
 passive-interface Ethernet1/1
 network 10.0.0.0
```

The `passive-interface` command causes RIP to ignore updates received on the specified interface. The command also prevents RIP from sending updates on the interface.

Routes learned via RIP are identified in the routing table with an `R` in the first column. This example shows the network 172.16.0.0/16 learned via RIP. The actual network in use is 172.16.100.0/24, but because RIP is classful, the router assumes that the entire 172.16.0.0/16 network is there as well:

```
R3# sho ip route
[text removed]

Gateway of last resort is 192.168.1.2 to network 0.0.0.0

R    172.16.0.0/16 [120/1] via 10.10.10.4, 00:00:21, Ethernet0/0
     10.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
C       10.10.10.0/24 is directly connected, Ethernet0/0
C       10.100.100.100/32 is directly connected, Loopback0
C    192.168.1.0/24 is directly connected, Ethernet1/0
S*   0.0.0.0/0 [254/0] via 192.168.1.2
```

Here we see an example of a host route being received by RIP:

```
R4# sho ip route
[text removed]

Gateway of last resort is not set
```

```
     172.16.0.0/32 is subnetted, 1 subnets
C       172.16.1.1 is directly connected, Loopback0
     10.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
C       10.10.10.0/24 is directly connected, Ethernet0/0
R       10.100.100.100/32 [120/1] via 10.10.10.3, 00:00:21, Ethernet0/0
C    192.168.1.0/24 is directly connected, Ethernet0/1
```

RIPv2

RIP was updated in the mid-1990s to reflect the widespread use of Classless Internet Domain Routing (CIDR) and Variable Length Subnet Masks (VLSM). The new protocol, RIP Version 2, operates similarly to RIP Version 1, in that it still uses hops as its only metric. However, it does have some significant advantages over RIPv1, including:

- Classless routing is supported by including subnet masks in network advertisements.
- The maximum hop count is 255 instead of 15. 256 is now the unreachable metric, as opposed to 16 with RIPv1.
- Updates in RIPv2 are sent using the multicast address 224.0.0.9, instead of as broadcasts.
- Neighbors can be configured with RIPv2. When a neighbor is configured, updates are sent to that neighbor using unicasts, which can further reduce network traffic.
- RIPv2 supports authentication between routers.

Even though RIPv2 supports subnets, it still only accepts classful addresses in the `network` command, so be careful when determining what networks and interfaces you've included. Use the `passive-interface` command to limit the scope of the `network` command, if necessary.

RIPv2 is classless, and advertises routes including subnet masks, but it summarizes routes by default. This means that if you have a 10.10.10.0/24 network connected to your router, it will still advertise 10.0.0.0/8, just like RIPv1. The first thing you should do when configuring RIPv2 is turn off *auto-summarization* with the router command `no auto-summary`:

```
R3(config)# router rip
R3(config-router)# no auto-summary
```

The routing table in a Cisco router makes no distinction between RIPv1 and RIPv2. Both protocols are represented by a single `R` in the routing table.

EIGRP

The Enhanced Internal Gateway Routing Protocol (EIGRP) is a classless enhancement to the Internal Gateway Routing Protocol (IGRP), which supported only classful networks. EIGRP, like IGRP, is a Cisco-proprietary routing protocol, which means that only Cisco routers can use this protocol. If you throw a Juniper or Nortel router into your network, it will not be able to communicate with your Cisco routers using EIGRP.

EIGRP is a very popular routing protocol because it's easy to configure and manage. With minimal configuration and design, you can get an EIGRP network up and running that will serve your company for years to come.

The ease of configuring EIGRP is also the main reason I see so many misbehaving EIGRP networks in the field. A network engineer builds a small network for his company. As time goes on the network gets larger and larger, and the routing environment gets more and more complicated. EIGRP manages the routing on the network quite nicely, until one day things start to go wrong. The engineer who built the network can't figure out what's wrong, and consultants are called in who completely redesign the network.

This is not to say that EIGRP is not a good routing protocol; I believe it is a very strong protocol. My point is that it's almost too easy to configure. You can throw two EIGRP routers on an Ethernet LAN with minimal configuration, and they will communicate and share routes. You can do the same with 10 or 20 or 100 routers, and they will all communicate and share routes. You can add 100 serial links with remote sites using EIGRP, and they will all communicate and share routes. The routing table will be a mess and the routers may be converging constantly, but the packets will flow. Eventually, however, the default settings may fail to work properly because the default configurations are not designed to scale in massive networks.

When EIGRP is configured properly on a network with a well-designed IP address scheme, it can be an excellent protocol for even a large network. When configured with multiple processes, it can scale very well.

EIGRP is a hybrid routing protocol that combines features from distance-vector protocols with features usually seen in link-state protocols. EIGRP uses triggered updates, so updates are sent only when changes occur. Bandwidth and delay are used as the default metrics, and although you can add other attributes to the equation, it is rarely a good idea to do so. EIGRP converges very quickly even in large networks. A network that might take minutes to converge with RIP will converge in seconds with EIGRP.

To configure EIGRP, enter into router configuration mode with the `router eigrp` *`autonomous-system-number`* command. The autonomous system number is a number that identifies the instance of EIGRP. A router can have multiple instances of EIGRP

running on it, each with its own database containing routes. The router will choose the best route based on criteria such as metrics, administrative distance, and so on. This behavior is different from RIP's, in that RIP runs globally on the router.

Figure 10-11 shows a router with two instances of EIGRP active. Each instance is referenced by an ASN. Routes learned in one process are not shared with the other process by default. Each process is essentially its own routing protocol. In order for a route learned in one process to be known to the other, the router must be configured for redistribution. EIGRP will redistribute IGRP routes automatically within the same ASN. (Redistribution is covered in detail in Chapter 11.)

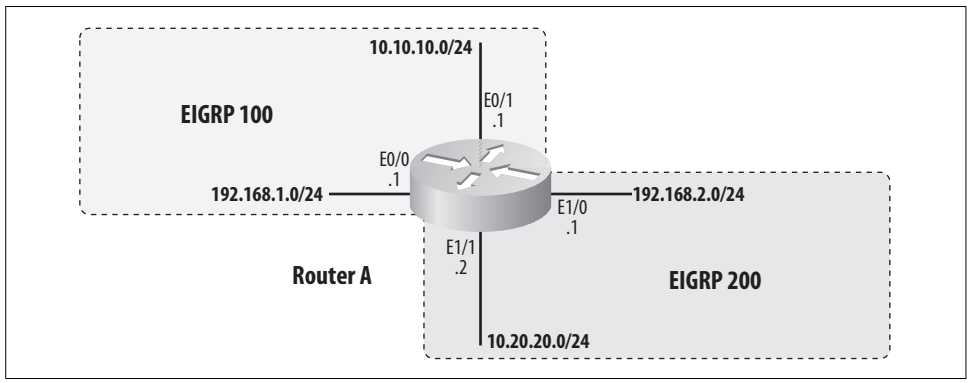


Figure 10-11. Multiple EIGRP instances

As with all IGPs, you list the interfaces you wish to include using the `network` command. EIGRP, like RIP, will automatically convert a classless network into the classful equivalent. The difference is that with EIGRP, you can add an inverse subnet mask to make the entry more specific. The following commands add all interfaces with addresses in the 10.0.0.0 network to the EIGRP 100 process:

```
Router-A(config)# router eigrp 100
Router-A(config-router)# network 10.0.0.0
```

But in the example in Figure 10-11, we'd like to add only the interface with the network 10.10.10.0/24. The subnet mask for a /24 network is 255.255.255.0, and the inverse subnet mask is 0.0.0.255 (inverse subnet masks are also called wildcard masks, and are discussed in Chapter 23). So, to add only this interface, we'd use the following `network` command:

```
Router-A(config-router)# network 10.10.10.0 0.0.0.255
```

After executing this command, the running configuration will still contain the less-specific 10.0.0.0 network statement:

```
router eigrp 100
 network 10.10.10.0 0.0.0.255
 network 10.0.0.0
```

Both commands will take effect. Be careful of this, as it can cause no end of frustration. In this example, it will cause the interface E1/1 to be included in EIGRP 100, which is not what we want. We need to remove the less-specific `network` command by negating it:

```
router eigrp 100
 no network 10.0.0.0
```

A very good practice to follow is to enable only the specific interface you wish to add in any routing process that supports it. This can be done by specifying the IP address on the interface with an all-zeros mask. In our example, the command would be `network 10.10.10.1 0.0.0.0`. This prevents surprises should network masks change, or interfaces be renumbered. Thus, my preferred configuration for EIGRP on the router shown in Figure 10-11 would be:

```
router eigrp 100
 network 10.10.10.1 0.0.0.0
 network 192.168.1.1 0.0.0.0
!
router eigrp 200
 network 10.20.20.1 0.0.0.0
 network 192.168.2.1 0.0.0.0
```

EIGRP summarizes routes the same way RIP does, but because EIGRP is a classless protocol, we can disable this behavior with the `no auto-summary` command:

```
Router-A(config-router)# no auto-summary
```

There are very few instances where you'd want to leave auto-summary on, so you should get into the habit of disabling it.

EIGRP operates by sending out hello packets using the multicast IP address 224.0.0.10 on configured interfaces. When a router running EIGRP receives these hello packets, it checks to see if the hello contains a process number matching an EIGRP process running locally. If it does, a handshake is performed. If the handshake is successful, the routers become *neighbors*.

Unlike RIP, which broadcasts routes to anyone who'll listen, EIGRP routers exchange routes only with neighbors. Once a *neighbor adjacency* has been formed, update packets are sent to the neighbor directly using unicast packets.

A useful command for EIGRP installations is the `eigrp log-neighbor-changes` command. This command displays a message to the console/monitor/log (depending on your logging configuration) every time an EIGRP neighbor adjacency changes state:

```
1d11h: %DUAL-5-NBRCHANGE: IP-EIGRP 100: Neighbor 10.10.10.4 (Ethernet0/0) is up:
new adjacency
```

On large networks, this can be annoying during a problem, but it can easily be disabled if needed.

To see the status of EIGRP neighbors on a router, use the `show ip eigrp neighbors` command:

```
R3# sho ip eigrp neighbors
IP-EIGRP neighbors for process 100
H   Address                 Interface   Hold Uptime   SRTT   RTO  Q  Seq Type
                                        (sec)         (ms)       Cnt Num
1   10.10.10.5              Et0/0         14 00:00:19    4   200  0  1
0   10.10.10.4              Et0/0         13 00:02:35    8   200  0  3
```

This command's output should be one of the first things you look at if you're having problems, because without a neighbor adjacency, EIGRP routers will not exchange routes.

Routes learned via internal EIGRP have an administrative distance of 90, and are marked with a single `D` in the first column of the routing table. Routes learned via external EIGRP have an administrative distance of 170, and are marked with the letters `D EX` at the beginning of the route:

```
R3# sho ip route
[text removed]

Gateway of last resort is 192.168.1.2 to network 0.0.0.0

     5.0.0.0/32 is subnetted, 1 subnets
D EX    5.5.5.5 [170/409600] via 10.10.10.5, 00:00:03, Ethernet0/0
     10.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
C       10.10.10.0/24 is directly connected, Ethernet0/0
C       10.100.100.100/32 is directly connected, Loopback0
C    192.168.1.0/24 is directly connected, Ethernet1/0
D    192.168.3.0/24 [90/2195456] via 10.10.10.5, 00:08:42, Ethernet0/0
S*   0.0.0.0/0 [254/0] via 192.168.1.2
```

EIGRP stores its information in three databases: the route database, the topology database, and the neighbor database. Viewing the topology database can be a tremendous help when troubleshooting routing problems. Not only can you see what EIGRP has put into the routing table, but you can also see what EIGRP considers to be alternate possibilities for routes:

```
R3# sho ip eigrp topology
IP-EIGRP Topology Table for AS(100)/ID(10.100.100.100)

Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
       r - reply Status, s - sia Status

P 5.5.5.5/32, 1 successors, FD is 409600
         via 10.10.10.5 (409600/128256), Ethernet0/0
P 10.10.10.0/24, 1 successors, FD is 281600
         via Connected, Ethernet0/0
P 192.168.3.0/24, 1 successors, FD is 2195456
         via 10.10.10.5 (2195456/2169856), Ethernet0/0
```

OSPF

In a nutshell, the premise of the Open Shortest Path First (OSPF) routing protocol is that the shortest or fastest path that is available is the one that will be used.

OSPF is the routing protocol of choice when:

- There are routers from vendors other than Cisco in the network.
- The network requires segmentation into areas or zones.
- There is a desire to avoid proprietary protocols.

OSPF is a link-state routing protocol. The metric it uses is bandwidth. The bandwidth of each link is calculated using the formula 100,000,000 divided by the bandwidth of the link in bps. Thus, a 100 Mbps link has a metric or "cost" of 1, a 10 Mbps link has a cost of 10, and a 1.5 Mbps link has a cost of 64. A 1 Gbps (or faster) link also has a cost of 1 because the cost cannot be lower than 1. The costs for each link in the path are added together to form a metric for the route.

In networks that include links faster than 100 Mbps, the formula for link cost can be changed using the `auto-cost reference-bandwidth` command. The default reference bandwidth is 100. In other words, by default, a 100 Mbps link has a cost of 1. To make a 1000 Mbps link have a cost of 1, change the reference bandwidth to 1,000:

```
R3(config)# router ospf 100
R3(config-router)# auto-cost reference-bandwidth 1000
```

If you change the reference bandwidth, you must change it on every router communicating in the OSPF process. Failure to do so will cause unstable networks, and unpredictable routing behavior.

OSPF classifies routers according to their function in the network. These are the types of OSPF routers:

Internal router
: An internal router is one that resides completely within a single area within a single OSPF autonomous system.

Area border router (ABR)
: An ABR is one that resides in more than one area within a single OSPF autonomous system.

Autonomous system border router (ASBR)
: An ASBR is one that connects to multiple OSPF autonomous systems, or to an OSPF autonomous system and another routing protocol's autonomous system.

Backbone routers
: Backbone routers are OSPF routers that reside in area zero. Area zero is considered the backbone in an OSPF network.

Designated router
: The DR is the router on a broadcast network that is elected to do the brunt of the OSPF processing. The DR will update all the other routers in the area with routes.

Backup designated router (BDR)
: The BDR is the router with the most eligibility to become the DR should the DR fail.

Unlike other routing protocols, OSPF does not send routes, but rather *link state advertisements* (LSAs). Each OSPF router determines which routes to use based on an internal database complied from these LSAs. There are six LSA types:

Router LSAs (type 1)
: Router LSAs are sent by every OSPF router into each connected area. These advertisements describe the router's links within the area.

Network LSAs (type 2)
: Network LSAs are sent by DRs, and describe the routers connected to the network from which the LSA was received.

Summary LSAs for ABRs (type 3)
: Summary LSAs for ABRs are sent by ABRs. These advertisements describe inter-area routes for networks. They are also used to advertise summary routes.

Summary LSAs for ASBRs (type 4)
: Summary LSAs for ASBRs are sent by ASBRs and ABRs. These advertisements describe links to ASBRs.

Autonomous System External (ASE) LSAs (type 5)
: ASE LSAs are sent by ASBRs and ABRs. These advertisements describe networks external to the autonomous system. They are sent everywhere, except to stub areas.

Not So Stubby Area (NSSA) LSAs (type 7)
: NSSA LSAs are sent by ABRs. These advertisements describe links within the NSSA.

OSPF separates networks into areas. The core area, which all other areas must connect with, is area zero. One of the perceived benefits of OSPF is that it forces you to design your network in such a way that there is a core with satellite areas. You can certainly build an OSPF network with only an area zero, but such a design usually doesn't scale well.

There are two main types of areas: *backbone* and *nonbackbone* areas. Area zero is the backbone area; all other areas are nonbackbone areas. Nonbackbone areas are further divided into the following types:

Normal area
: An OSPF area that is not area zero, and is not configured as one of the following types. No special configuration is required.

Stub area
: An OSPF area that does not allow ASE LSAs. When an area is configured as a stub, no `O E1` or `O E2` routes will be seen in the area.

Totally stubby area (TSA)
: An OSPF area that does not allow type-3, -4, or -5 LSAs, except for the default summary route. TSAs see only a default route, and routes local to the areas themselves.

Not so stubby area (NSSA)
: No type-5 LSAs are allowed in an NSSA. Type-7 LSAs that convert to type 5 at the ABR are allowed.

NSSA totally stub area
: NSSA totally stub areas are a combination of totally stubby and not so stubby areas. This area type does not allow type-3, -4, or -5 LSAs, except for the default summary route; it does allow type-7 LSAs that convert to type 5 at the ABR.

On Ethernet and other broadcast networks, OSPF elects a router to become the designated router, and another to be the backup designated router. Calculating OSPF routes can be CPU-intensive, especially in a dynamic network. Having one router that does the brunt of the work makes the network more stable, and allows it to converge faster. The DR calculates the best paths, then propagates that information to its neighbors within the network that are in the same area and OSPF process.

OSPF dynamically elects the DR through a relatively complicated process. The first step involves the router interface's OSPF *priority*. The default priority is 1, which is the lowest value an interface can have, and still be elected the DR. A value of 0 indicates that the router is ineligible to become the DR on the network. Setting the priority higher increases the chances that the router will be elected the DR. The OSPF interface priority is configured using the interface command `ip ospf priority`. The valid range is 0–255.

Ideally, you should plan which router is to become the DR, and set its priority accordingly. Usually, there is an obvious choice, such as a hub or core router, or perhaps just the most powerful router on the network. The designated router will be doing more work than the other routers, so it should be the one with the most horsepower. If your design includes a hub router, that router will need to be the DR because it will be the center of the topology.

If the OSPF interface priority is not set, resulting in a tie, the router will use the OSPF router ID to break the tie. Every router has an OSPF router ID. This ID can be configured manually with the `router-id` command. If the router ID is not configured manually, the router will assign it to be the IP address of the lowest-numbered loopback address, if one is configured. If a loopback address is not configured, the router ID will be the highest IP address configured on the router. The only ways to change the router ID are to remove and reinstall the OSPF configuration, or to reboot the router. Be careful, and think ahead when planning your network IP scheme.

When first deploying OSPF, engineers commonly make the mistake of neglecting the priority and router ID when configuring the routers. Left to its own devices, OSPF will usually pick routers that you would not choose as the DR and BDR.

A common network design using OSPF is to have a WAN in the core as area zero. Figure 10-12 shows such a network. Notice that all of the areas are designated with the same OSPF process number. Each of the areas borders on area zero, and there are no paths between areas other than via area zero. This is a proper OSPF network design.

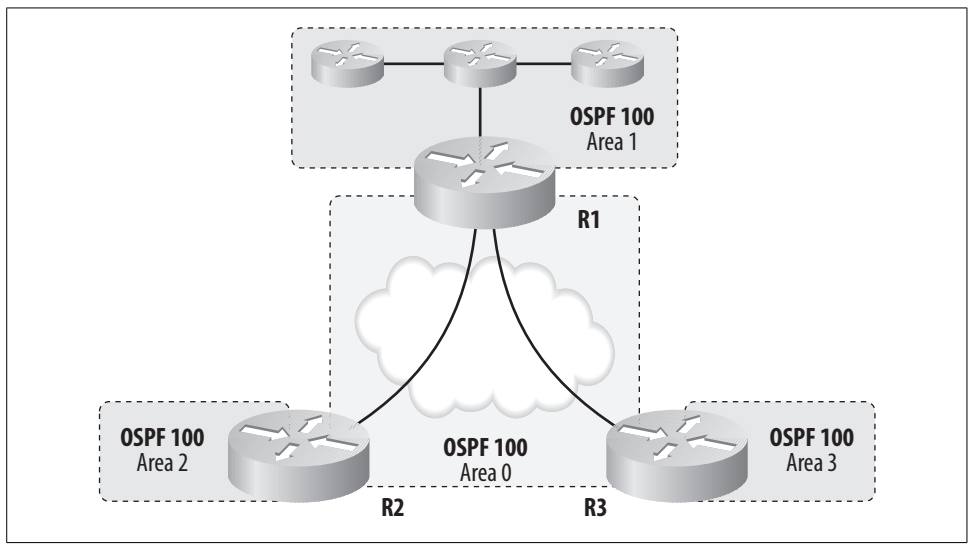


Figure 10-12. Simple OSPF network

Area zero does not have to be fully meshed when using technologies such as frame relay. This is in part because OSPF recognizes the fact that there are different types of networks. OSPF knows that networks supporting broadcasts act differently from networks that are point-to-point and thus have only two active IP addresses. OSPF supports the following network types:

Point-to-point
: A point-to-point network is one with only two nodes on it. A common example is a serial link between routers, such as a point-to-point T1. No DR is chosen in a point-to-point network because there are only two routers on the network. This is the default OSPF network type on serial interfaces with PPP or HDLC encapsulation.

Point-to-multipoint

A point-to-multipoint network is a network where one hub router connects to all the other routers, but the other routers only connect to the hub router. Specifically, the remote routers are assumed to be connected with virtual circuits, though only one IP network is used. No neighbors are configured, and no DR is chosen. Area 0 in Figure 10-12 could be configured as a point-to-multipoint OSPF network.

Broadcast

A broadcast network is an Ethernet, Token Ring, or FDDI network. Any number of hosts may reside on a broadcast network, and any host may communicate directly with any other host. A DR must be chosen, and neighbors must be discovered or configured on a broadcast network. A broadcast network uses multicasts to send hello packets to discover OSPF routers. This is the default OSPF network type for Ethernet and Token Ring networks.

Nonbroadcast multiaccess (NBMA)

In a nonbroadcast multiaccess network, all nodes may be able to communicate with one another, but they do not share a single medium. Examples include frame-relay, X.25, and Switched Multimegabit Data Service (SMDS) networks. Because NBMA networks do not use multicasts to discover neighbors, you must manually configure them. Area 0 in Figure 10-12 could be configured as an NBMA network. This is the default OSPF network type on serial interfaces with frame-relay encapsulation.

OSPF enables interfaces using the `network` router command. It is a classless protocol, so you must use inverse subnet masks to limit the interfaces included. Unlike with EIGRP, you must include the inverse mask. If you do not, OSPF will not assume a classful network, but will instead report an error:

```
R3(config-router)# network 10.10.10.0
% Incomplete command.
```

In addition to the inverse mask, you must also specify the area in which the network resides:

```
R3(config-router)# network 10.10.10.0 0.0.0.255 area 0
```

My preference is to specifically configure interfaces so there are no surprises. This is done with an inverse mask of 0.0.0.0:

```
R3(config-router)# network 10.10.10.1 0.0.0.0 area 0
```

OSPF routes are marked by the letter `O` in the first column of the routing table:

```
R3# sho ip route
[Text Removed]

Gateway of last resort is 192.168.1.2 to network 0.0.0.0
```

```
     192.192.192.0/30 is subnetted, 1 subnets
C       192.192.192.4 is directly connected, Serial0/0
     172.16.0.0/32 is subnetted, 1 subnets
O IA    172.16.1.1 [110/11] via 10.10.10.4, 00:00:09, Ethernet0/0
     10.0.0.0/24 is subnetted, 1 subnets
C       10.10.10.0 is directly connected, Ethernet0/0
C    192.168.1.0/24 is directly connected, Ethernet1/0
S*   0.0.0.0/0 [254/0] via 192.168.1.2
```

Various OSPF route types are described in the routing table. They are: `O` (OSPF), `O IA` (OSPF inter-area), `O N1` (OSPF NSSA external type 1), and `O N2` (OSPF NSSA external type 2).

OSPF stores its routes in a database, much like EIGRP. The command to show the database is `show ip ospf database`:

```
R3# sho ip ospf database

            OSPF Router with ID (192.192.192.5) (Process ID 100)

                Router Link States (Area 0)

Link ID         ADV Router      Age         Seq#       Checksum Link count
192.192.192.5   192.192.192.5   1769        0x8000002A 0x00C190 1

                Summary Net Link States (Area 0)

Link ID         ADV Router      Age         Seq#       Checksum
192.192.192.4   192.192.192.5   1769        0x8000002A 0x003415

                Router Link States (Area 1)

Link ID         ADV Router      Age         Seq#       Checksum Link count
192.192.192.5   192.192.192.5   1769        0x8000002A 0x00B046 1

                Summary Net Link States (Area 1)

Link ID         ADV Router      Age         Seq#       Checksum
10.10.10.0      192.192.192.5   1769        0x8000002A 0x0002A2

            OSPF Router with ID (192.168.1.116) (Process ID 1)
```

If all of this seems needlessly complicated to you, you're not alone. The complexity of OSPF is one of the reasons that many people choose EIGRP instead. If you're working in a multivendor environment, however, EIGRP is not an option.

BGP

The Border Gateway Protocol (BGP) is a very different protocol from the others described here. The most obvious difference is that BGP is an external gateway protocol, while all the previously discussed protocols were internal gateway protocols.

BGP can be hard to understand for those who have only ever dealt with internal protocols like EIGRP and OSPF because the very nature of the protocol is different. As BGP is not often seen in the corporate environment, I'll only cover it briefly here.

BGP does not deal with hops or links, but rather with autonomous systems. A network in BGP is referred to as a *prefix*. A prefix is advertised from an autonomous system. BGP then propagates that information through the connected autonomous systems until all the autonomous systems know about the prefix.

Routes in BGP are considered most desirable when they traverse the least possible number of autonomous systems. When a prefix is advertised, the autonomous system number is prefixed onto the autonomous system *path*. This path is the equivalent of a route in an internal gateway protocol. When an autonomous system learns of a prefix, it learns of the path associated with it. When the autonomous system advertises that prefix to another autonomous system, it prepends its own ASN to the path. As the prefix is advertised to more and more autonomous systems, the path gets longer and longer. The shorter the path, the more desirable it is.

Figure 10-13 shows a simple example of BGP routing in action. The network 10.0.0.0/8 resides in AS 105, which advertises this prefix to AS 3 and AS 2. The path for 10.0.0.0/8 within AS 3 and AS 2 is now `10.0.0.0/8 AS105`. AS 2 in turn advertises the prefix to AS 1, prepending its own ASN to the path. AS 1 now knows the path to 10.0.0.0/8 as `10.0.0.0/8 AS2, AS105`. Meanwhile, AS 3 advertises the prefix to AS 100, which then knows the path to 10.0.0.0/8 as `10.0.0.0/8 AS3, AS105`.

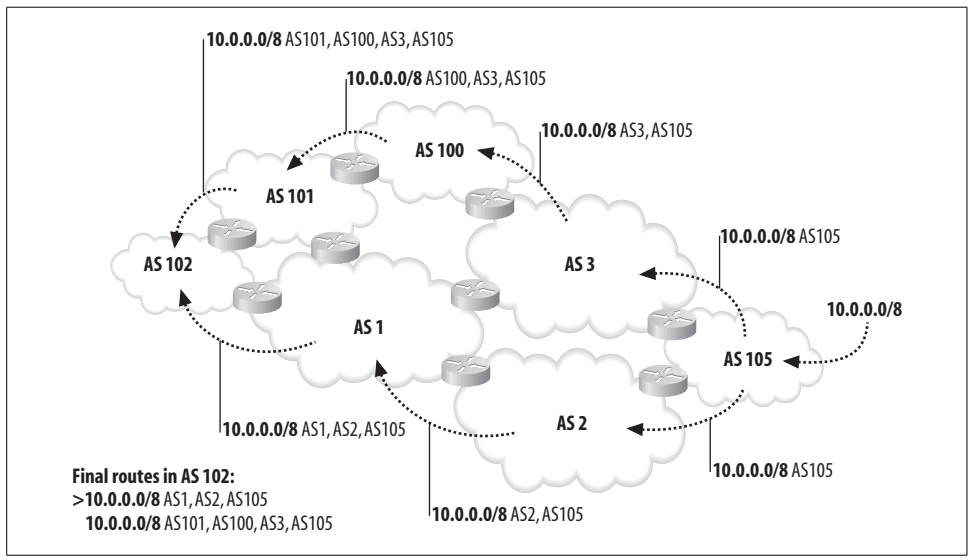


Figure 10-13. Routing in BGP

On the other side of the world, AS 102 receives two paths:

```
> 10.0.0.0/8 AS1, AS2, AS105
  10.0.0.0/8 AS101, AS100, AS3, AS105
```

The > on the first line indicates that BGP considers this the preferred path. The path is preferred because it is the shortest path among the known choices.

What makes BGP so confusing to newcomers is the many attributes that can be configured. A variety of weights can be attributed to paths, with names like local preference, weight, communities, and multiexit discriminator. To make matters worse, many of these attributes are very similar in function.

The protocol also functions differently from other protocols. For example, the `network` statement, which is used to enable interfaces in other protocols, is used to list the specific networks that can be advertised in BGP.

BGP does not discover neighbors; they must be configured manually. There can only be one autonomous system on any given router, though it may communicate with neighbors in other autonomous systems.

BGP is the routing protocol of the Internet. Many of the major service providers allow anonymous telnet into *route servers* that act just like Cisco routers. Do an Internet search for the term "looking-glass routers," and you should find plenty of links. These route servers are an excellent way to learn more about BGP, as they are a part of the largest network in the world, and have active routes to just about every public network on Earth. Unless you're working at a tier-1 service provider, where else could you get to poke around with a BGP router that has 20 neighbors, 191,898 prefixes, and 3,666,117 paths? I have a pretty cool lab, but I can't compare with that! Here is the output from an actual route server:

```
route-server> sho ip bgp summary
BGP router identifier 10.1.2.5, local AS number 65000
BGP table version is 208750, main routing table version 208750
191680 network entries using 19359680 bytes of memory
3641563 path entries using 174795024 bytes of memory
46514 BGP path attribute entries using 2605064 bytes of memory
42009 BGP AS-PATH entries using 1095100 bytes of memory
4 BGP community entries using 96 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 197854964 total bytes of memory
Dampening enabled. 2687 history paths, 420 dampened paths
191529 received paths for inbound soft reconfiguration
BGP activity 191898/218 prefixes, 3666117/24554 paths, scan interval 60 secs

Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
10.0.0.2        4  7018       0       0        0    0    0 never    Idle (Admin)
12.0.1.63       4  7018   45038     188   208637    0    0 03:04:16        0
12.123.1.236    4  7018   39405     189   208637    0    0 03:05:02   191504
12.123.5.240    4  7018   39735     189   208637    0    0 03:05:04   191504
12.123.9.241    4  7018   39343     189   208637    0    0 03:05:03   191528
12.123.13.241   4  7018   39617     188   208637    0    0 03:04:20   191529
12.123.17.244   4  7018   39747     188   208637    0    0 03:04:58   191505
12.123.21.243   4  7018   39441     188   208637    0    0 03:04:28   191528
12.123.25.245   4  7018   39789     189   208637    0    0 03:05:07   191504
```

```
12.123.29.249   4  7018   39602     188    208637    0    0 03:04:16   191505
12.123.33.249   4  7018   39541     188    208637    0    0 03:04:16   191528
12.123.37.250   4  7018   39699     188    208637    0    0 03:04:26   191529
12.123.41.250   4  7018   39463     188    208637    0    0 03:04:19   191529
12.123.45.252   4  7018   39386     188    208637    0    0 03:04:20   191505
12.123.133.124  4  7018   39720     188    208637    0    0 03:04:20   191528
12.123.134.124  4  7018   39729     188    208637    0    0 03:04:22   191529
12.123.137.124  4  7018   39480     188    208637    0    0 03:04:15   191528
12.123.139.124  4  7018   39807     188    208637    0    0 03:04:24   191528
12.123.142.124  4  7018   39748     188    208637    0    0 03:04:22   191505
12.123.145.124  4  7018   39655     188    208637    0    0 03:04:23   191529
```

These route servers can get pretty busy and very slow. If you find yourself waiting too long for a response to a query, either wait a bit and try again, or try another route server.

Choose your favorite public IP network (doesn't everyone have one?) and see how the paths look from the looking-glass router. If you don't have a favorite, choose one that you can easily figure out, like one in use by *www.cisco.com* or *www.oreilly.com*:

```
[bossman@myserver bossman]$ nslookup www.oreilly.com
Server:  localhost
Address:  127.0.0.1

Name:    www.oreilly.com
Addresses:  208.201.239.36, 208.201.239.37
```

Once you have the address, you can do a lookup for the network:

```
route-server> sho ip bgp 208.201.239.0
BGP routing table entry for 208.201.224.0/19, version 157337
Paths: (19 available, best #15, table Default-IP-Routing-Table)
  Not advertised to any peer
  7018 701 7065, (received & used)
    12.123.137.124 from 12.123.137.124 (12.123.137.124)
      Origin IGP, localpref 100, valid, external, atomic-aggregate
      Community: 7018:5000
  7018 701 7065, (received & used)
    12.123.33.249 from 12.123.33.249 (12.123.33.249)
      Origin IGP, localpref 100, valid, external, atomic-aggregate
      Community: 7018:5000
  7018 701 7065, (received & used)
    12.123.29.249 from 12.123.29.249 (12.123.29.249)
      Origin IGP, localpref 100, valid, external, atomic-aggregate
      Community: 7018:5000
  7018 701 7065, (received & used)
    12.123.41.250 from 12.123.41.250 (12.123.41.250)
      Origin IGP, localpref 100, valid, external, atomic-aggregate
      Community: 7018:5000
  7018 701 7065, (received & used)
    12.123.1.236 from 12.123.1.236 (12.123.1.236)
      Origin IGP, localpref 100, valid, external, atomic-aggregate, best
      Community: 7018:5000
```

CHAPTER 11
Redistribution

Redistribution is the process of injecting routes into a routing protocol from outside the realm of the protocol. For example, if you had a router that was running EIGRP and OSPF, and you needed the routes learned by EIGRP to be advertised in OSPF, you would redistribute the EIGRP routes into OSPF. Another common example is the redistribution of static or connected routes. Because static routes are entered manually, and not learned, they must be redistributed into a routing protocol if you wish them to be advertised.

As Figure 11-1 shows, routes learned through EIGRP are not automatically advertised out of the OSPF interfaces. To accomplish this translation of sorts, you must configure redistribution within the protocol where you wish the routes to appear.

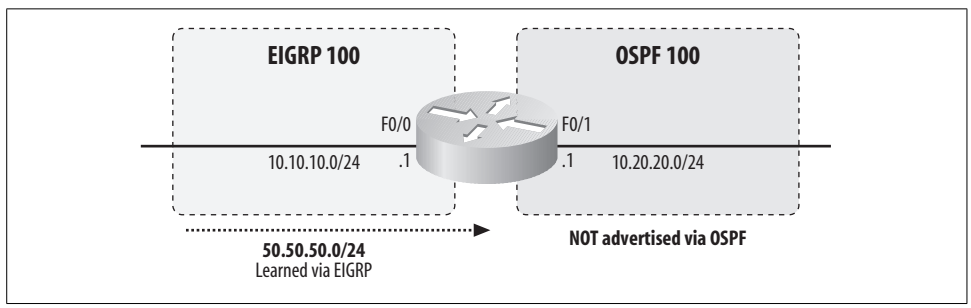


Figure 11-1. Most routing protocols do not redistribute by default

One of the main reasons that protocols do not redistribute routes automatically is that different protocols have vastly different metrics. OSPF, for example, calculates the best route based on the bandwidth of the links. EIGRP, on the other hand, uses bandwidth and delay (by default) to form a very different metric. While the router could assume you wanted to redistribute, and assign a standard metric to the learned routes, a better approach is to allow you to decide whether and how routes should be redistributed.

Two steps must be taken when redistributing routes. First, a metric must be configured. This allows the routing protocol to assign a metric that it understands to the incoming routes. Second, the `redistribute` command must be added. (The exact commands used for these purposes vary widely between protocols, and they'll be discussed individually in the sections that follow.)

One reason to redistribute routes might be the inclusion of a firewall that must participate in dynamic routing, but cannot use the protocol in use on the network. For example, many firewalls support RIP, but not EIGRP. To dynamically route between an EIGRP router and a RIP-only firewall, you must redistribute between RIP and EIGRP on the router.

The best rule to remember when redistributing is to keep it simple. It's easy to get confused when routes are being sent back and forth between routing protocols. Keeping the design as simple as possible will help keep the network manageable. You can create some pretty interesting problems when redistribution isn't working properly. The simpler the design is, the easier it is to troubleshoot.

Redistribution is about converting one protocol's routes into a form that another protocol can understand. This is done by assigning new metrics to the routes as they pass into the new protocol. Because the routes must adhere to the metrics of the new protocol, the key to understanding redistribution is understanding metrics.

When a protocol redistributes routes from any source, they become *external* routes in the new protocol. Routes can be redistributed from a limited number of sources:

Static routes
: Routes that have been entered manually into the configuration of the router doing the redistribution can be redistributed into routing protocols. Injecting static routes on one router into a dynamic routing protocol can be a useful way of propagating those routes throughout the network.

Connected routes
: Routes that are in the routing table as a result of a directly connected interface on the router doing the redistribution can also be redistributed into routing protocols. When redistributing a connected route, the network in question will be inserted into the routing protocol, but the interfaces configured within that network will not advertise or listen for route advertisements. This can be used as an alternative to the `network` command when such behavior is desired.

Other routing protocols
: Routes can be learned dynamically from other routing protocols that are active on the router doing the redistribution. Routes from any routing protocol can be redistributed into any other routing protocol. An example of redistributing between routing protocols would be OSPF redistributing into EIGRP.

The same routing protocol from a different autonomous system or process
: Protocols that support autonomous systems, such as EIGRP, OSPF, and BGP, can redistribute between these systems. An example of a single protocol redistributing between autonomous systems would be EIGRP 100 redistributing into EIGRP 200.

Regardless of what protocol you redistribute into, you can still only do it from one of the sources just listed. When redistributing routes, the command `redistribute`—followed by the route source—is used within the protocol receiving the route.

Redistribution is configured on the protocol for which the routes are destined, not the one from which they are sourced. No configuration needs to be done on the protocol providing the routes.

Redistributing into RIP

We'll start with RIP, because it has the simplest metric, and therefore the simplest configuration.

A common problem when configuring routing protocols is the inclusion of static routes. Because the routes are static, they are, by definition, not dynamic. But if they're statically defined, why include them in a dynamic routing protocol at all?

Figure 11-2 shows a simple network where redistribution of a static route is required. R1 has a directly connected interface on the 50.50.50.0/24 network, but is not running a routing protocol. R2 has a static route pointing to R1 for the 50.50.50.0/24 network. R2 and R3 are both communicating using RIPv2. In this case, R3 cannot get to the 50.50.50.0/24 network because R2 has not advertised it.

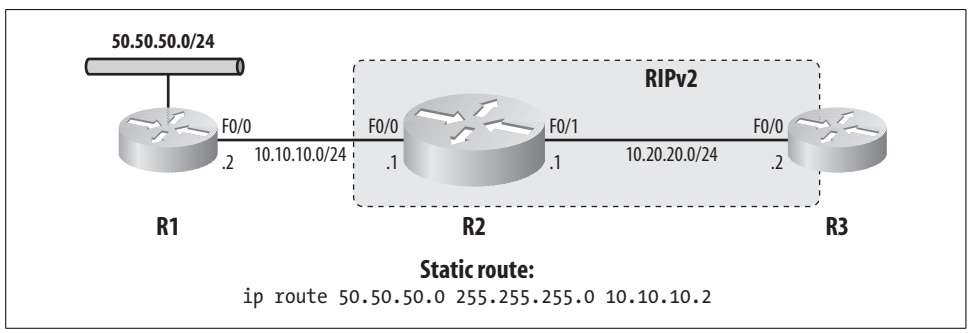


Figure 11-2. Redistributing a static route into RIPv2

For RIP to advertise the static route to R3, the route must be redistributed into RIP. Here is the full RIP configuration for R2:

```
router rip
 version 2
```

```
 redistribute static metric 1
 network 10.0.0.0
 no auto-summary
```

Notice the `metric` keyword on the `redistribute` command. This defines what the RIP metric will be for all static routes injected into RIP. Another way to accomplish this is with the `default-metric` command:

```
router rip
 version 2
 redistribute static
 network 10.0.0.0
 default-metric 3
 no auto-summary
```

Here, the default metric is set to 3. If you set a default metric as shown here, you don't need to include a metric when you use the `redistribute static` command. The router will automatically assign the default metric you've specified to all redistributed static routes.

You can see what a protocol's default metric is with the `show ip protocols` command:

```
R2# sho ip protocols
Routing Protocol is "rip"
  Sending updates every 30 seconds, next due in 2 seconds
  Invalid after 180 seconds, hold down 180, flushed after 240
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Default redistribution metric is 3
  Redistributing: static, rip
  Default version control: send version 2, receive version 2
  Automatic network summarization is not in effect
  Maximum path: 4
  Routing for Networks:
    10.0.0.0
  Routing Information Sources:
    Gateway         Distance      Last Update
  Distance: (default is 120)
```

Be careful with default metrics because they apply to all routes redistributed into the routing protocol, regardless of the source. If you now redistribute EIGRP into RIP, the metric assigned in RIP will be 3 because that is the configured default. You can override the default metric by specifying a metric on each `redistribute` command. Here, I have specified a default metric of 5, but I've also configured EIGRP routes to have a metric of 1 when redistributed into RIP:

```
router rip
 version 2
 redistribute static
 redistribute eigrp 100 metric 1
 network 10.0.0.0
 default-metric 5
 no auto-summary
```

Here is the routing table on R3 after the final configuration on R2:

```
R3# sho ip route
[text removed]

Gateway of last resort is not set

     192.192.192.0/30 is subnetted, 1 subnets
C       192.192.192.4 is directly connected, Serial0/0
     50.0.0.0/24 is subnetted, 1 subnets
R       50.50.50.0 [120/5] via 10.20.20.1, 00:00:07, Ethernet1/0
     10.0.0.0/24 is subnetted, 2 subnets
R       10.10.10.0 [120/1] via 10.20.20.1, 00:00:10, Ethernet1/0
C       10.20.20.0 is directly connected, Ethernet1/0
```

The route 50.50.50.0/24 is in the routing table, and has a metric of 5. The route below it is a result of the `network 10.0.0.0` statement on R2. This route is not a product of redistribution, and so has a normal RIP metric.

Another common issue is the need to advertise networks that are connected to a router, but are not included in the routing process. Figure 11-3 shows a network with three routers, all of which are participating in RIPv2. R1 has a network that is not included in the RIP process. The configuration for R1's RIP process is as follows:

```
router rip
 version 2
 network 10.0.0.0
 no auto-summary
```

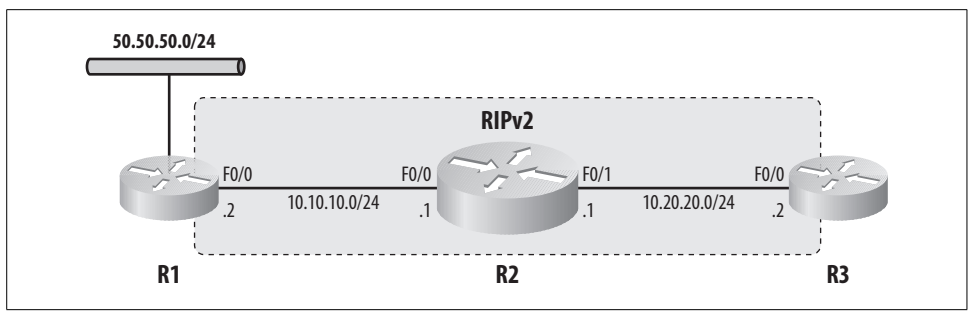


Figure 11-3. Redistributing connected routes into RIP

There are no routers on the 50.50.50.0/24 network, so enabling RIP on that interface would add useless broadcasts on that network. Still, R3 needs to be able to get to the network. To add 50.50.50.0/24 to the advertisements sent out by R1, we must redistribute connected networks into RIP using the `redistribute connected` command on R1:

```
router rip
 version 2
 redisribute connected metric 1
 network 10.0.0.0
 no auto-summary
```

While sending useless broadcasts may seem trivial, remember that RIP sends broadcasts that include the entire routing table. Only 25 destinations can be included in a single RIP update packet. On a network with 200 routes, each update will be composed of eight large broadcast packets, each of which will need to be processed by every device on the network. That's potentially 12k of data every 30 seconds.

If that's not enough proof for you, consider this: RIP updates are classified as *control* packets (IP precedence 6 or DSCP 48). That means that they have a higher precedence than voice RTP packets, which are classified as *express forwarding* packets (IP precedence 5 or DSCP 40). To put it simply, RIP updates can easily affect voice quality on VOIP-enabled networks.

Now R3 can see the 50.50.50.0/24 network in its routing table because it's been advertised across the network by RIP:

```
R3# sho ip route
[text removed]

Gateway of last resort is not set

     192.192.192.0/30 is subnetted, 1 subnets
C       192.192.192.4 is directly connected, Serial0/0
     50.0.0.0/24 is subnetted, 1 subnets
R       50.50.50.0 [120/2] via 10.20.20.1, 00:00:05, Ethernet1/0
     10.0.0.0/24 is subnetted, 2 subnets
R       10.10.10.0 [120/1] via 10.20.20.1, 00:00:24, Ethernet1/0
C       10.20.20.0 is directly connected, Ethernet1/0
     192.168.1.0/29 is subnetted, 1 subnets
R       192.168.1.0 [120/2] via 10.20.20.1, 00:00:24, Ethernet1/0
```

Redistributing into EIGRP

EIGRP was designed to automatically redistribute IGRP routes from the same ASN. This behavior can be disabled with the `no redistribute igrp` *`autonomous-system`* command:

```
router eigrp 100
 no redistribute igrp 100
```

Redistributing routes into EIGRP is done the same way as it is with RIP. It only looks harder because the metric in EIGRP is more complicated than that in RIP—whereas RIP only uses hop count as a metric, EIGRP uses the combined bandwidth and delay values from all the links in the path. In fact, EIGRP uses more than just these two measurements, but, by default, the other metrics are disabled. However, with redistribution, you must specify them, so let's take a look at what they should be.

As with RIP, you can use the `default-metric` command to specify the metric of redistributed routes, or you can specify the metric on each `redistribute` command line. Here are the arguments required for the `default-metric` command in EIGRP, and the allowed ranges of values:

- Bandwidth in Kbps: 1–4,294,967,295
- Delay in 10-microsecond units: 0–4,294,967,295
- Reliability metric, where 255 is 100 percent reliable: 0–255
- Effective bandwidth metric (loading), where 255 is 100 percent loaded: 1–255
- Maximum Transmission Unit (MTU) metric of the path: 1–4,294,967,295

How you configure these values will largely depend on your needs at the time. Remember that redistributed routes are external routes, so they will always have a higher administrative distance than internal routes in EIGRP. Such routes will be advertised with an administrative distance of 170.

You need to make redistributed routes appear as though they are links because that's what EIGRP understands. If you wanted to make redistributed routes appear as 100 Mbps Ethernet links, you would configure the default metric like this:

```
R3(config-router)# default-metric 100000 10 255 1 1500
```

The appropriate values to use in these commands are not always obvious. For example, the bandwidth is presented in Kbps, not bps (a 100 Mbps link is 100,000 Kbps). This reflects the way that bandwidth is shown in the `show interface` command:

```
R2# sho int f0/0 | include BW
  MTU 1500 bytes, BW 100000 Kbit, DLY 100 usec,
```

Conversely, notice how the `show interface` command shows the delay of an interface:

```
R2# sho int f0/0 | include DLY
  MTU 1500 bytes, BW 100000 Kbit, DLY 100 usec,
```

Here, the delay is shown in microseconds, but when you specify the delay in redistribution, you must use 10-microsecond units. That is, to achieve a delay of 100 microseconds, you would specify a delay value of 10.

When I configure redistribution, I always make the reliability 255, loading 1, and MTU 1500. In fact, I usually make the redistributed routes appear as 100 Mbps links, as shown previously. Keep it simple. While there may be instances where you'll want to alter these values, those instances will be rare.

The method of specifying a default metric, and overriding it with a specific metric described in "Redistributing into RIP," is also valid with EIGRP. Here, I've specified a default metric reflecting a 100 Mbps link with a delay of 100 microseconds, and I'm redistributing OSPF process 100 with a metric reflecting a 1,000 Mbps links with a delay of 50 microseconds:

```
router eigrp 100
 redistribute ospf 100 metric 1000000 5 255 1 1500
```

```
network 10.0.0.0
default-metric 100000 10 255 1 1500
no auto-summary
```

When redistributing OSPF routes into another protocol, you can limit the types of routes that are redistributed. For example, you can redistribute only OSPF internal routes, while ignoring all OSPF external routes. This is done by using the `match` keyword with the `redistribute ospf` command.

```
R2(config-router)# redistribute ospf 100 match ?
  external       Redistribute OSPF external routes
  internal       Redistribute OSPF internal routes
  nssa-external  Redistribute OSPF NSSA external routes
```

When matching external routes, you can also differentiate between OSPF type-1 and type-2 routes:

```
R2(config-router)# redistribute ospf 100 match external ?
  1              Redistribute external type 1 routes
  2              Redistribute external type 2 routes
  external       Redistribute OSPF external routes
  internal       Redistribute OSPF internal routes
  match          Redistribution of OSPF routes
  metric         Metric for redistributed routes
  nssa-external  Redistribute OSPF NSSA external routes
  route-map      Route map reference
  <cr>
```

Finally, you can combine route types with the `match` keyword. As an example, I have configured this router to redistribute OSPF routes from process 100 into EIGRP 100, but to include only internal routes and external type-2 routes:

```
R2(config-router)# redistribute ospf 100 match internal external 2
```

Because I have not specified a metric, the default metric will be used. I could have added a metric to the specific redistribution as well.

Redistributing RIP into EIGRP is done the same way as redistributing OSPF, but there is no option for matching route types because RIP does not support the many types of routes that OSPF does:

```
router eigrp 100
 redistribute rip metric 100000 100 255 1 1500
```

Redistributing into OSPF

Redistribution into OSPF is done in the same way as in the other protocols. The metric for an OSPF route is a derivative of the bandwidths of the links contained in the route. Setting the default metric in OSPF to 10 Mbps is done as follows:

```
R3(config-router)# default-metric 10
```

There are no other options. The metric can have a value of 1–16,777,214, with 1 being 100 Mbps (assuming default settings).

If you do not specify a default metric or a metric on the `redistribute` command line, OSPF will assign a metric of 20 to all redistributed routes, except those from BGP, which will be assigned a metric of 1.

While all redistributed routes are external, OSPF supports two types of external routes, which are cleverly described as *type-1* and *type-2* external routes. Type-1 routes are designated with `O E1` in the routing table, while type-2 routes are designated with `O E2`. `E1` routes include the metric as set at the point of redistribution, plus the metric of all the links within the OSPF autonomous system. `E2` routes only include the metric set at the point of redistribution. Figure 11-4 illustrates how the OSPF metrics change throughout a simple network depending on the external route type in use.

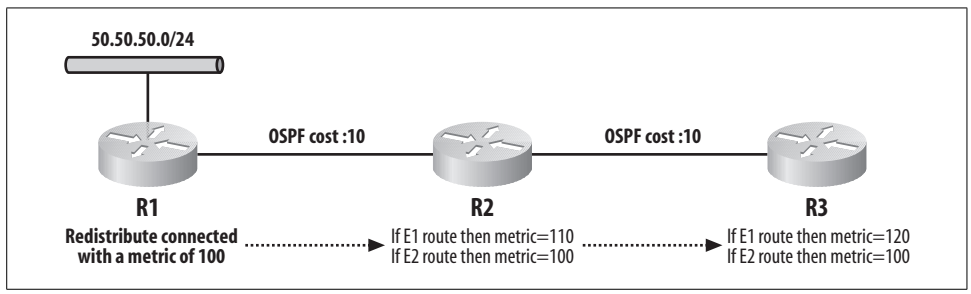


Figure 11-4. OSPF external route types

Redistribution into OSPF defaults to type-2 routes. Which type should you use? That depends on your needs at the time. Generally, in smaller networks (less than, say, 10 routers) `E2` routes may be easier to maintain, as they have the same value anywhere in the OSPF autonomous system. But to me, `E1` routes function more logically because they increment with each hop, as do most other metrics.

To set the route type, add the `metric-type` keyword to the `redistribute` command:

```
R3(config-router)# redistribute eigrp 100 metric-type ?
  1  Set OSPF External Type 1 metrics
  2  Set OSPF External Type 2 metrics
```

The metric type and the metric can be added onto one command line:

```
R3(config-router)# redistribute eigrp 100 metric-type 2 metric 100
```

In practice, every time you redistribute into OSPF you should include the keyword `subnets`:

```
R3(config-router)# redistribute eigrp 100 metric 100 subnets
```

Without the `subnets` keyword, OSPF will redistribute only routes that have not been subnetted. In the world of VLSM, where practically all networks are subnets, it is rare that you will not want subnets to be redistributed.

If you do not include the `subnets` keyword on modern versions of IOS, you will be warned about your probable mistake:

```
R3(config-router)# redistribute eigrp 100 metric 100
% Only classful networks will be redistributed
```

Mutual Redistribution

The term *mutual redistribution* is used when a router redistributes between two routing protocols in both directions instead of just one. Often, we redistribute because there is a device or entity we wish to connect with that doesn't support the routing protocol we have chosen to use. We need to share routes between protocols, but, if you will, the protocols don't speak the same language.

Figure 11-5 shows a network in which every subnet needs to be reached by every other subnet. The problem here is that the network on the left is using OSPF, and the network on the right is using EIGRP. For a host on 50.50.50.0 to be able to route to a host on the 70.70.70.0 network, EIGRP routes will need to be redistributed into OSPF. Conversely, if hosts on the 70.70.70.0 network wish to talk with the hosts on 50.50.50.0, OSPF routes will need to be redistributed into EIGRP. Because there is only one router connecting these two domains together, redistribution must occur in both directions.

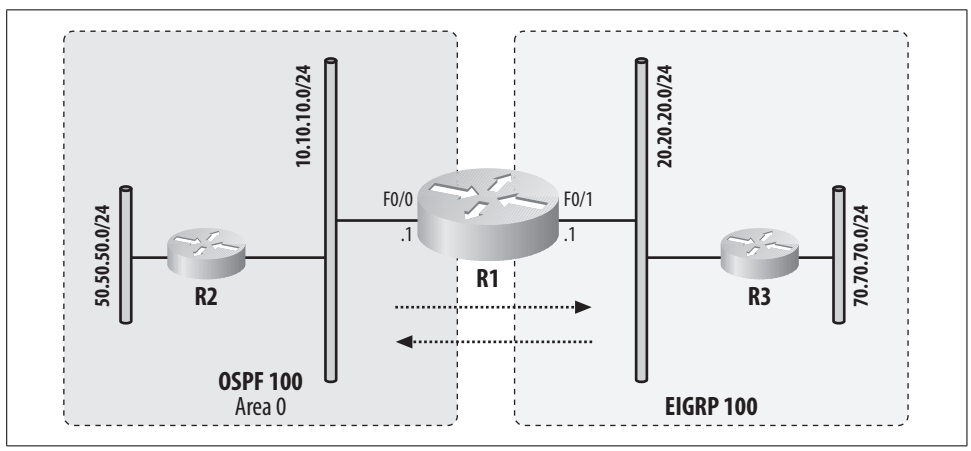


Figure 11-5. Mutual redistribution

To accomplish mutual redistribution on one router, simply configure both protocols for redistribution into the other:

```
router eigrp 100
 redistribute ospf 100
 network 20.20.20.0 0.0.0.255
 default-metric 100000 100 255 1 1500
 no auto-summary
!
router ospf 100
 redistribute eigrp 100 metric 100 subnets
 network 10.10.10.0 0.0.0.255 area 0
 default-metric 10
```

The configuration is simple. I've followed the steps outlined in the preceding sections by establishing a default metric in each protocol, then redistributed accordingly. Nothing more needs to be done when there is only one router doing mutual redistribution.

Redistribution Loops

Redistribution can get interesting when there are multiple routers doing it. Routes redistributed from one routing protocol into another can be redistributed back into the originating protocol, which can cause some pretty strange results. All of the original metrics will have been lost, so the route will inherit whatever metric was configured during redistribution.

Figure 11-6 shows a network with three routers. R3 has a network attached that is being advertised in EIGRP 100 by way of the `redistribute connected` command (50.50.50.0/24). R1 is redistributing from OSPF into EIGRP (from left to right in the drawing), and R2 is redistributing from EIGRP to OSPF (from right to left in the drawing).

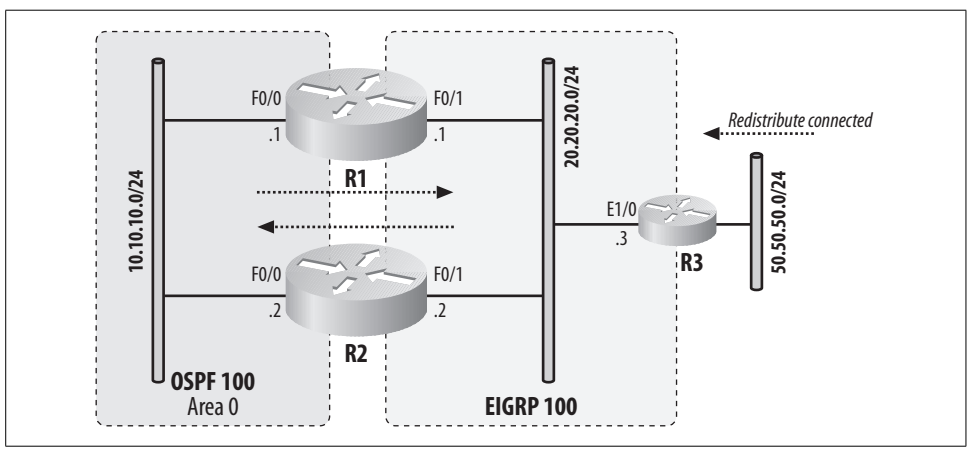


Figure 11-6. Redistribution loop

The network 50.50.50.0/24 will be advertised from R3 to R1 and R2 through EIGRP. R2 will in turn redistribute the route into OSPF 100. R2 now has an entry for 50.50.50.0/24 in the OSPF database as well as the EIGRP topology table. Because the route was originally redistributed into EIGRP, it has an administrative distance of 170 when it gets to R2. R2 advertises the router to R1 via OSPF, which has an administrative distance of 110. So, even though R1 has also learned of the route from R3, where it originated, it will prefer the route from R2 because of the more attractive administrative distance.

Here are the IP routing tables from each router. Router R1 has learned the route for 50.50.50.0/24 from router R2 via OSPF:

```
R1# sho ip route
[text removed]

Gateway of last resort is not set

     50.0.0.0/24 is subnetted, 1 subnets
O E2    50.50.50.0 [110/10] via 10.10.10.2, 00:16:28, FastEthernet0/0
     20.0.0.0/24 is subnetted, 1 subnets
C       20.20.20.0 is directly connected, FastEthernet0/1
     10.0.0.0/24 is subnetted, 1 subnets
C       10.10.10.0 is directly connected, FastEthernet0/0
```

R2 has learned the route from EIGRP as an external route from R3. The route is external because it was originally redistributed into EIGRP on R3:

```
R2# sho ip route
[text removed]

Gateway of last resort is not set

     50.0.0.0/24 is subnetted, 1 subnets
D EX    50.50.50.0 [170/156160] via 20.20.20.3, 00:17:30, FastEthernet0/1
     20.0.0.0/24 is subnetted, 1 subnets
C       20.20.20.0 is directly connected, FastEthernet0/1
     10.0.0.0/24 is subnetted, 1 subnets
C       10.10.10.0 is directly connected, FastEthernet0/0
```

R3 shows only its two connected routes and the network from the OSPF side, as it was redistributed into EIGRP on R2:

```
R3# sho ip route
[text removed]

Gateway of last resort is not set

     50.0.0.0/24 is subnetted, 1 subnets
C       50.50.50.0 is directly connected, Loopback0
     20.0.0.0/24 is subnetted, 1 subnets
C       20.20.20.0 is directly connected, Ethernet1/0
     10.0.0.0/24 is subnetted, 1 subnets
D EX    10.10.10.0 [170/537600] via 20.20.20.2, 00:00:15, Ethernet1/0
```

The key to this example lies in the fact that EIGRP has a higher administrative distance for external routes (170) than it does for internal routes (90). OSPF only has one administrative distance for all routes (110).

This type of problem can cause no end of headaches in a production environment. If you don't have experience using redistribution in complex environments, this is a very easy mistake to make. Symptoms of the problem include routes pointing to places you don't expect. Look carefully at the design, and follow the route back to its source to see where the problem starts. In networks where you're redistributing between different autonomous systems using the same routing protocol, you may see routes flip-flop back and forth between sources. This can be caused by each AS reporting the same metric, in which case the router will update its routing table each time it receives an update.

In the present example, one way to resolve the problem is to stop redistributing the connected route and include the interface into EIGRP with the `network` command. Using this approach, the route becomes an internal route with an AD of 90, which is more desirable than OSPF's AD of 110.

Limiting Redistribution

When designing complex networks with multiple redistribution points, you must somehow limit redistribution so that loops are prevented. I'm going to show you my method of choice, which involves tagging routes and filtering with route maps.

Route Tags

Many routing protocols—for example, EIGRP, OSPF, and RIPv2 (but not RIPv1)—allow you *tag* routes with values when redistributing them. The route tags are nothing more than numbers within the range of 0–4,294,967,295. (Unfortunately, the tags cannot be alphanumeric.) Route tags do not affect the protocol's actions; the tag is simply a field to which you can assign a value to use elsewhere.

To set a route tag when redistributing into OSPF, add the `tag` *`tag#`* keyword to the `redistribute` command:

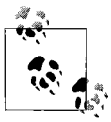
```
R2(config-router)# redistribute eigrp 100 metric 10 subnets tag 2
```

This command will redistribute routes from EIGRP 100 into OSPF. The OSPF metric will be 10, and the tag will be 2. To see the tags in OSPF routes, use the `show ip ospf database` command. Redistributed routes will be external routes. The last column will be the tags for these routes:

```
R2# sho ip ospf dat

            OSPF Router with ID (10.10.10.2) (Process ID 100)

                Router Link States (Area 0)
```

```
Link ID         ADV Router      Age         Seq#       Checksum Link count
10.10.10.2      10.10.10.2      128         0x80000002 0x00F5BA 1
20.20.20.1      20.20.20.1      129         0x80000002 0x009DD9 1

                Net Link States (Area 0)

Link ID         ADV Router      Age         Seq#       Checksum
10.10.10.1      20.20.20.1      129         0x80000001 0x00B5CA

                Type-5 AS External Link States

Link ID         ADV Router      Age         Seq#       Checksum Tag
20.20.20.0      10.10.10.2      4           0x80000001 0x00D774 2
20.20.20.0      20.20.20.1      159         0x80000001 0x002DF9 0
50.50.50.0      10.10.10.2      4           0x80000001 0x009B56 2
```

To set a route tag in EIGRP, you need to use route maps. Luckily for those of you who have a route-map phobia, the way I'll show you to use them is one of the simplest ways they can be deployed.



Route maps are cool! There, I said it. Route maps are quite powerful, and if you have a fear of them, I suggest you spend some time playing around with them. The difficulty usually lies in confusion between route maps and access lists and how they interact. It will be well worth your time to learn more about route maps. They can get you out of a technical corner (such as a redistribution loop) when no other option exists. See Chapter 14 for more information.

To apply a tag to a redistributed route in EIGRP, you must first create a route map, then call it in a `redistribute` command line. Route maps in their simplest form consist of a line including the route map name, a `permit` or `deny` statement, and a number, followed by descriptions of one or more actions to carry out. Here's a simple route map:

```
route-map TAG-EIGRP permit 10
 set tag 3
```

The first line lists the name of the route map, the keyword `permit`, and the number 10 (this is the default; the numbers are used to order multiple entries in a route map, and as there's only one entry here, it doesn't really matter what the number is). The keyword `permit` says to perform the actions specified below the opening line. The next line shows the action to be taken for this route map entry, which is "set the tag to a value of 3."

Once you've created the TAG-EIGRP route map, you can call it using the `route-map` keyword, and the route map's name in the EIGRP `redistribute` command:

```
R2(config-router)# redistribute connected route-map TAG-EIGRP
```

This command redistributes connected routes into EIGRP using the default metric, and applies the tag set in the `TAG-EIGRP` route-map.

To see whether your tag has been implemented, look in the EIGRP topology table for the specific routes you believe should be tagged. To illustrate, I've applied this route map to R3 in the network shown in Figure 11-6. Here's what R2's EIGRP topology table looks like for the route 50.50.50.0/24:

```
R2# sho ip eigrp top
IP-EIGRP Topology Table for AS(100)/ID(10.10.10.2)

Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
       r - reply Status, s - sia Status

P 10.10.10.0/24, 1 successors, FD is 28160
         via Redistributed (281600/0)
P 20.20.20.0/24, 1 successors, FD is 28160
         via Connected, FastEthernet0/1
P 50.50.50.0/24, 1 successors, FD is 156160, tag is 3
         via 20.20.20.3 (156160/128256), FastEthernet0/1
```

And here is the detail for 50.50.50.0/24 on R3. The source is *Rconnected*, which means it was learned from the `redistributed connected` command:

```
R3# sho ip eigrp top 50.50.50.0/24
IP-EIGRP (AS 100): Topology entry for 50.50.50.0/24
  State is Passive, Query origin flag is 1, 1 Successor(s), FD is 128256
  Routing Descriptor Blocks:
  0.0.0.0, from Rconnected, Send flag is 0x0
      Composite metric is (128256/0), Route is External
      Vector metric:
        Minimum bandwidth is 10000000 Kbit
        Total delay is 5000 microseconds
        Reliability is 255/255
        Load is 1/255
        Minimum MTU is 1514
        Hop count is 0
      External data:
        Originating router is 50.50.50.1 (this system)
        AS number of route is 0
        External protocol is Connected, external metric is 0
        Administrator tag is 3 (0x00000003)
```

The last line shows the *administrator tag* as 3, indicating that routes redistributed into EIGRP (specifically, redistributed connected routes) have been marked with a tag of 3.

EIGRP doesn't do anything with this information other than store it. So what does tagging do for you? Just as you can set a tag to apply when redistributing routes into a routing protocol, you can also test for a tag and permit or deny redistributions based on it. Call me a nerd, but that's pretty cool.

To check for an incoming route tag, you again must use a route map. This must be done for all routing protocols, including OSPF.

Looking back at the example from Figure 11-6, consider that we've now set a tag of 3 for the connected route 50.50.50.0/24 on R3. If we could prevent this route from being advertised into OSPF, that would solve the problem, because then R1 would never learn of the route improperly.

On R2, when we redistribute into OSPF, we need to tell the router to call a route map. We're no longer setting the tag with the `redistribute` command, as we'll set it in the route map. If I'm checking for a tag using a route map, I always set it there, too. It's easier for me to understand things when I do everything in one place. I've also seen problems where setting a tag with the `tag` keyword and then checking for it with route maps doesn't work very well. Here, I'm telling the router to redistribute EIGRP 100 routes into OSPF 100, assign them a metric of 10, and apply whatever logic is included in the route map `No-EIGRP-Tag3`:

```
router ospf 100
 redistribute eigrp 100 metric 10 subnets route-map No-EIGRP-Tag3
```

Here's how I've designed the route map:

```
route-map No-EIGRP-Tag3 deny 10
 match tag 3
!
route-map No-EIGRP-Tag3 permit 20
 set tag 2
```

This one's a little more complicated than the last route map, but it's still pretty simple. The first line is a `deny` entry. It's followed by an instruction that says, "match anything with a tag of 3." The `match` coming after the `deny` can be confusing, but the more you play with route maps, the more this will make sense. The next entry doesn't match anything; it permits everything and then sets a tag of 2 for the routes. Taken together, the route map essentially says, "match anything with a tag of 3, and deny it," then "match everything else, and set the tag to 2."

Now when we look at the OSPF database on router R2, we'll see that the route for 50.50.50.0/24 is gone. The route to 20.20.20.0/24 that was learned from R1 is still there, however, because it was not tagged with a 3:

```
R2# sho ip ospf database

            OSPF Router with ID (10.10.10.2) (Process ID 100)

                Router Link States (Area 0)

Link ID         ADV Router      Age         Seq#       Checksum Link count
10.10.10.2      10.10.10.2      769         0x80000002 0x00F5BA 1
20.20.20.1      20.20.20.1      770         0x80000002 0x009DD9 1

                Net Link States (Area 0)

Link ID         ADV Router      Age         Seq#       Checksum
```

```
10.10.10.1      20.20.20.1      771         0x80000001 0x00B5CA

                Type-5 AS External Link States

Link ID         ADV Router      Age         Seq#       Checksum Tag
20.20.20.0      10.10.10.2      224         0x80000001 0x00D774 2
20.20.20.0      20.20.20.1      800         0x80000001 0x002DF9 0
```

The route still exists in the EIGRP topology table on R2; it just wasn't redistributed into OSPF because of our cool route map.

In the routing table on R1, we'll now see that 50.50.50.0/24 is pointing to R3 the way we want it to:

```
R1# sho ip route
[text removed]

Gateway of last resort is not set

     50.0.0.0/24 is subnetted, 1 subnets
D EX    50.50.50.0 [170/156160] via 20.20.20.3, 00:00:16, FastEthernet0/1
     20.0.0.0/24 is subnetted, 1 subnets
C       20.20.20.0 is directly connected, FastEthernet0/1
     10.0.0.0/24 is subnetted, 1 subnets
C       10.10.10.0 is directly connected, FastEthernet0/0
```

A Real-World Example

Today's networks are often designed with high availability as a primary driver. When designing networks with no single points of failure, a scenario like the one shown in Figure 11-7 is a real possibility.

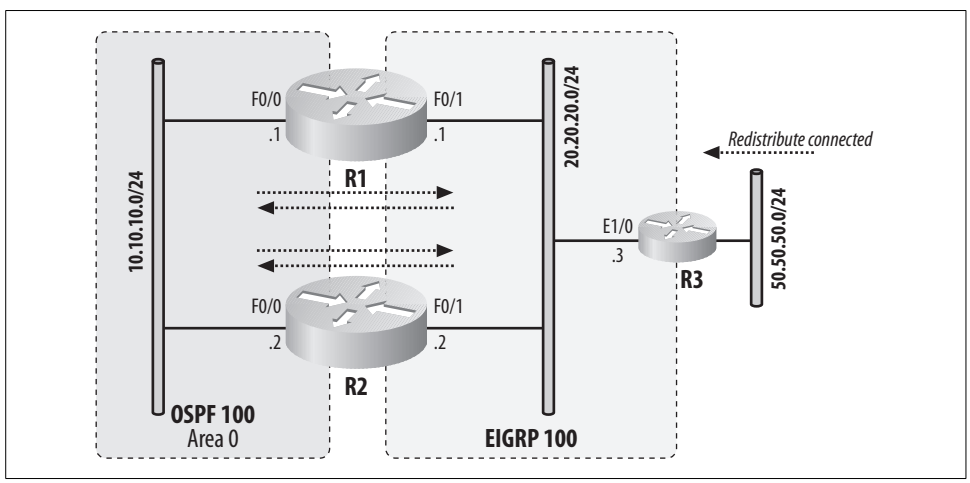


Figure 11-7. Two routers performing mutual redistribution

Here, we have two routers doing mutual redistribution (both are redistributing EIGRP into OSPF and OSPF into EIGRP). You've already seen what can happen when each one is only redistributing in one direction. The probability of router-induced mayhem is pretty strong here, but this kind of design is very common, for the reasons already discussed.

To make this scenario work, we'll again use route tags, but this time we'll add some flair. (You can never have too much flair.)

The idea behind this technique is simple: routes sent from one protocol into another will not be advertised back to the protocol from which they came.

I like to tag my routes with the number of the router I'm working on. That's not always possible, especially if you've named your router something clever like Boston-PoP-Router or Michelle. Another option is to tag your routes with the administrative distance of the routing protocols they came from—90 for EIGRP, and so on. Still another is to use autonomous system numbers. Whatever you choose, make sure it's obvious, if possible. And always document what you've done so others can understand your brilliance.

Using the administrative distance as a tag, here is the configuration for R1:

```
router eigrp 100
 redistribute ospf 100 route-map OSPF-to-EIGRP
 network 20.20.20.0 0.0.0.255
 default-metric 100000 100 255 1 1500
 no auto-summary
!
router ospf 100
 log-adjacency-changes
 redistribute eigrp 100 subnets route-map EIGRP-to-OSPF
 network 10.10.10.0 0.0.0.255 area 0
 default-metric 100
!
route-map EIGRP-to-OSPF deny 10
 match tag 110
!
route-map EIGRP-to-OSPF permit 20
 set tag 90
!
route-map OSPF-to-EIGRP deny 10
 match tag 90
!
route-map OSPF-to-EIGRP permit 20
 set tag 110
```

The route maps are the same on R2 because the same rules apply, and we need to test for any routes redistributed from other routers.

When a route is redistributed from OSPF into EIGRP, it will be assigned a tag of 110. Routes redistributed from EIGRP into OSPF will be assigned a tag of 90. When a route tagged with 90 is seen in OSPF, we'll know if was sourced from EIGRP because the tag value will have been set to 90. When this route then comes to be redistributed into EIGRP from OSPF on the other router, the route map will deny it. Thus, a route cannot be redistributed into EIGRP if it originated in EIGRP. OSPF-sourced routes are similarly blocked from redistribution back into OSPF.

While this design does prevent routing loops, it does not solve the problem of R3's 50.50.50.0/24 network being advertised through the wrong protocol. R2 is now pointing back to OSPF for this network:

```
R2# sho ip route
[text removed]

Gateway of last resort is not set

     50.0.0.0/24 is subnetted, 1 subnets
O E2    50.50.50.0 [110/100] via 10.10.10.1, 00:11:50, FastEthernet0/0
     20.0.0.0/24 is subnetted, 1 subnets
C       20.20.20.0 is directly connected, FastEthernet0/1
     10.0.0.0/24 is subnetted, 1 subnets
C       10.10.10.0 is directly connected, FastEthernet0/0
```

Every network is different, and there will always be challenges to solve. In this case, we might have been better off with the first design, where each router was only redistributing in one direction, even though that design is not very resilient.

The way to solve the problem, and provide multiple mutual redistribution points, is to combine the two scenarios. With route maps, we can match multiple tags, so in addition to denying any routes already redistributed into EIGRP on R2, we can also match on the 3 tag we set on R3:

```
route-map EIGRP-to-OSPF deny 10
 match tag 110 3
!
route-map EIGRP-to-OSPF permit 20
 set tag 90
```

The line `match tag 110 3` means, "match on tag 110 or 3." Now R2 has the right routes:

```
R2# sho ip route
[text removed]

Gateway of last resort is not set

     50.0.0.0/24 is subnetted, 1 subnets
D EX    50.50.50.0 [170/156160] via 20.20.20.3, 00:00:01, FastEthernet0/1
     20.0.0.0/24 is subnetted, 1 subnets
C       20.20.20.0 is directly connected, FastEthernet0/1
     10.0.0.0/24 is subnetted, 1 subnets
C       10.10.10.0 is directly connected, FastEthernet0/0
```

Another method

Here's another method to use, which I like for its elegance: because redistributed routes are external, only allow internal routes to be redistributed. Once a route is redistributed and becomes external, it won't be redistributed again.

When redistributing OSPF routes into another protocol, this is simple. The keyword `match` in the `redistribute` command lets you match on route type:

```
router eigrp 100
 redistribute ospf 100 match internal
 network 20.20.20.0 0.0.0.255
 default-metric 100000 100 255 1 1500
 no auto-summary
```

When redistributing other protocols, you must resort to route maps:

```
router ospf 100
 redistribute eigrp 100 route-map Only-Internal subnets
 network 10.10.10.0 0.0.0.255 area 0
 default-metric 100
!
route-map Only-Internal permit 10
 match route-type internal
```

This solution solves both our problems. As the 50.50.50.0/24 route is an external route by nature of its original redistribution into R3, it will not be redistributed into OSPF. What once required many lines of code and multiple route maps has been accomplished with one keyword and a single two-line route map. Simple is good.

Here is the final routing table from R2:

```
R2# sho ip route
[text removed]

Gateway of last resort is not set

     50.0.0.0/24 is subnetted, 1 subnets
D EX    50.50.50.0 [170/156160] via 20.20.20.3, 00:13:30, FastEthernet0/1
     20.0.0.0/24 is subnetted, 1 subnets
C       20.20.20.0 is directly connected, FastEthernet0/1
     10.0.0.0/24 is subnetted, 1 subnets
C       10.10.10.0 is directly connected, FastEthernet0/0
```

As long as you keep things simple, tagging and filtering redistributed routes is easy. The more complicated the network is, the harder it is to keep all the redistributed networks behaving properly. Additionally, a more complex network might not allow this last solution, because there might be valid external routes that need to be redistributed.

CHAPTER 12
Tunnels

A *tunnel* is a means whereby a local device can communicate with a remote device as if the remote device were local as well. There are many types of tunnels. Virtual Private Networks (VPNs) are tunnels. Generic Routing Encapsulation (GRE) creates tunnels. Secure Shell (SSH) is also a form of tunnel, though different from the other two. Let's take a closer look at these three types:

GRE

GRE tunnels are designed to allow remote networks to appear to be locally connected. GRE offers no encryption, but it does forward broadcasts and multicasts. If you want a routing protocol to establish a neighbor adjacency or exchange routes through a tunnel, you'll probably need to configure GRE. GRE tunnels are often built within VPN tunnels to take advantage of encryption. GRE is described in RFCs 1701 and 2784.

VPN

VPN tunnels are also designed to allow remote networks to appear as if they were locally connected. VPN encrypts all information before sending it across the network, but it will not usually forward multicasts and broadcasts. Consequently, GRE tunnels are often built within VPNs to allow routing protocols to function. VPNs are often used for remote access to secure networks.

There are two main types of VPNs; *point-to-point* and *remote access*. Point-to-point VPNs offer connectivity between two remote routers, creating a virtual link between them. Remote-access VPNs are single-user tunnels between a user and a router, firewall, or *VPN concentrator* (a specialized VPN-only device).

Remote-access VPNs usually require VPN client software to be installed on a personal computer. The client communicates with the VPN device to establish a personal virtual link.

SSH
: SSH is a client/server application designed to allow secure connectivity to servers. In practice, it is usually used just like telnet. The advantage of SSH over telnet is that it encrypts all data before sending it. While not originally designed to be a tunnel in the sense that VPN or GRE would be considered a tunnel, SSH can be used to access remote devices in addition to the one to which you have connected. While this does not have a direct application on Cisco routers, the concept is similar to that of VPN and GRE tunnels, and thus worth mentioning. I use SSH to access my home network instead of a VPN.

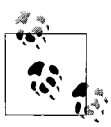
Tunnels can encrypt data so that only the other side can see it, as with SSH, or they can make a remote network appear local, as with GRE, or they can do both, as is the case with VPN.

GRE tunnels will be used for the examples in this chapter because they are the simplest to configure and the easiest to understand. GRE tunnels are solely a means of connecting remote networks as if they were local networks—they enable a remote interface on another router to appear to be directly connected to a local router, even though many other routers and networks may separate them. GRE does not encrypt data.

GRE Tunnels

To create a GRE tunnel, you must create virtual interfaces on the routers at each end, and the tunnel must begin and terminate at existing routable IP addresses. The tunnel is not a physical link—it is logical. As such, it must rely on routes already in place for its existence. The tunnel will behave like a physical link in that it will need an IP address on each side of the link. These will be the tunnel interface IPs. In addition, as the link is virtual, the tunnel will need to be told where to originate and terminate. The source and destination must be existing IP addresses on the routers at each end of the tunnel.

The best way to guarantee that the tunnel's source and destination points are available is to use the loopback interfaces on each end as targets. This way, if there are multiple paths to a router, the source and destination points of the tunnel are not dependent on any single link, but rather on a logical interface on the router itself.



Loopback interfaces are different from loopback IP addresses. A loopback interface can have any valid IP address assigned to it. A loopback interface on a router is a logical interface within the router that is always up. It can be included in routing protocols and functions like any other interface, with the exception that a loopback interface is always up/up. You can configure multiple loopback interfaces on a router.

Figure 12-1 shows an example of a simple network in which we will build a GRE tunnel. Four routers are connected. They are all running EIGRP, with all connected networks redistributed into the routing table. The purpose of this example is to show how the path from Router A to the network 10.20.20.0/24 on Router D will change with the addition of a GRE tunnel.

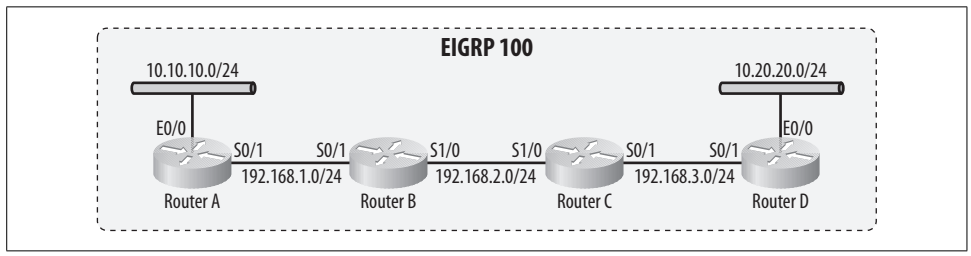


Figure 12-1. Simple network

Given the network in Figure 12-1, the routing table on Router A looks like this:

```
Router-A# sho ip route

Gateway of last resort is not set

     10.0.0.0/24 is subnetted, 2 subnets
D       10.20.20.0 [90/3196416] via 192.168.1.2, 03:39:06, Serial0/1
C       10.10.10.0 is directly connected, Ethernet0/1
C    192.168.1.0/24 is directly connected, Serial0/1
D    192.168.2.0/24 [90/2681856] via 192.168.1.2, 03:39:06, Serial0/1
D    192.168.3.0/24 [90/3193856] via 192.168.1.2, 03:39:06, Serial0/1
```

All routes except the connected routes are available through Serial0/1. Now we're going to add a tunnel between Router A and Router D. Because we prefer to link tunnels to loopback interfaces, we will add one to each router. Figure 12-2 shows the network as we will create it.

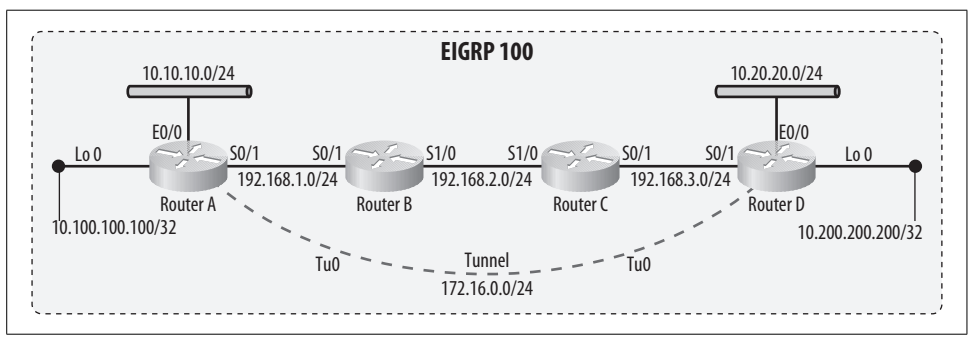


Figure 12-2. Simple network with a tunnel

First, we will add the loopback interfaces on Router A:

```
Router-A# conf t
Enter configuration commands, one per line. End with CNTL/Z.
Router-A(config)# int lo 0
Router-A(config-if)# ip address 10.100.100.100 255.255.255.255
```

Next, we will add the loopback interfaces on Router D:

```
Router-D# conf t
Enter configuration commands, one per line. End with CNTL/Z.
Router-D(config)# int lo 0
Router-D(config-if)# ip address 10.200.200.200 255.255.255.255
```

Because we are redistributing connected interfaces in EIGRP, they are now both visible in Router A's routing table:

```
Router-A# sho ip route

Gateway of last resort is not set

     10.0.0.0/8 is variably subnetted, 4 subnets, 2 masks
D       10.20.20.0/24 [90/3196416] via 192.168.1.2, 03:50:27, Serial0/1
C       10.10.10.0/24 is directly connected, Ethernet0/1
C       10.100.100.100/32 is directly connected, Loopback0
D EX    10.200.200.200/32 [170/3321856] via 192.168.1.2, 00:00:52, Serial0/1
C    192.168.1.0/24 is directly connected, Serial0/1
D    192.168.2.0/24 [90/2681856] via 192.168.1.2, 03:50:27, Serial0/1
D    192.168.3.0/24 [90/3193856] via 192.168.1.2, 03:50:27, Serial0/1
```

Now that the loopback addresses are visible in the routing table, it's time to create the tunnel. The process is simple. We'll begin by creating the virtual interfaces for each side of the tunnel. Tunnel interfaces are numbered like all interfaces in IOS, with the first being tunnel 0:

```
Router-A(config)# int tunnel ?
  <0-2147483647>  Tunnel interface number

Router-A(config)# int tunnel 0
Router-A(config-if)#
23:23:39: %LINEPROTO-5-UPDOWN: Line protocol on Interface Tunnel0, changed state to
down
```

Tunnels must have existing routable IP addresses as their beginning and end points, and these must be configured on both routers. The *source* of the tunnel is the local side, and the *destination* is the remote side (from the viewpoint of the router being configured):

```
Router-A(config-if)# ip address 172.16.0.1 255.255.255.0
Router-A(config-if)# tunnel source loopback 0
Router-A(config-if)# tunnel destination 10.200.200.200
23:25:15: %LINEPROTO-5-UPDOWN: Line protocol on Interface Tunnel0, changed state to
up
```

As soon as we add the destination IP address to the tunnel, the tunnel interface comes up on Router A:

```
Router-A# sho int tu0
Tunnel0 is up, line protocol is up
  Hardware is Tunnel
  Internet address is 172.16.0.1/24
  MTU 1514 bytes, BW 9 Kbit, DLY 500000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation TUNNEL, loopback not set
  Keepalive not set
  Tunnel source 10.100.100.100 (Loopback0), destination 10.200.200.200
  Tunnel protocol/transport GRE/IP, key disabled, sequencing disabled
  Checksumming of packets disabled,  fast tunneling enabled
  Last input never, output never, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: fifo
  Output queue :0/0 (size/max)
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     0 packets input, 0 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     0 packets output, 0 bytes, 0 underruns
     0 output errors, 0 collisions, 0 interface resets
     0 output buffer failures, 0 output buffers swapped out
```

Note that although Router A shows the interface to be up, because Router D does not yet have a tunnel interface, nothing can be sent across the link. Be careful of this, as you may get confused under pressure. The tunnel network 172.16.0.0/24 is even active in Router A's routing table (it will not be found on Router D at this time):

```
Router-A# sho ip route

Gateway of last resort is not set

     172.16.0.0/24 is subnetted, 1 subnets
C       172.16.0.0 is directly connected, Tunnel0
     10.0.0.0/8 is variably subnetted, 4 subnets, 2 masks
D       10.20.20.0/24 [90/3196416] via 192.168.1.2, 04:25:38, Serial0/1
C       10.10.10.0/24 is directly connected, Ethernet0/1
C       10.100.100.100/32 is directly connected, Loopback0
D EX    10.200.200.200/32 [170/3321856] via 192.168.1.2, 00:36:03, Serial0/1
C    192.168.1.0/24 is directly connected, Serial0/1
D    192.168.2.0/24 [90/2681856] via 192.168.1.2, 04:25:39, Serial0/1
D    192.168.3.0/24 [90/3193856] via 192.168.1.2, 04:25:39, Serial0/1
```

To terminate the tunnel on Router D, we need to add the tunnel interface there. We'll use the same commands that we did on Router A, but reverse the source and destination:

```
Router-D(config)# int tu 0
23:45:13: %LINEPROTO-5-UPDOWN: Line protocol on Interface Tunnel0, changed state to
down
```

```
Router-D(config-if)# ip address 172.16.0.2 255.255.255.0
Router-D(config-if)# tunnel source lo 0
Router-D(config-if)# tunnel destination 10.100.100.100
Router-D(config-if)#
23:47:06: %LINEPROTO-5-UPDOWN: Line protocol on Interface Tunnel0, changed state to
up
```

Now we have a live link between these routers, which appear to be directly connected. However, the Ethernet network on Router D, which was known through the serial link, is still known through the serial link, and not the tunnel. If the tunnel is theoretically directly connected to both routers, why isn't the tunnel the preferred path?

Digging into our EIGRP expertise, we can use the `show ip eigrp topology` command to see what EIGRP knows about the paths:

```
Router-A# sho ip eigrp top
5d18h: %SYS-5-CONFIG_I: Configured from console by console
IP-EIGRP Topology Table for AS(100)/ID(192.168.1.1)

Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
       r - reply Status, s - sia Status

P 10.20.20.0/24, 1 successors, FD is 3196416
         via 192.168.1.2 (3196416/2684416), Serial0/1
         via 172.16.0.2 (297246976/28160), Tunnel0
P 10.10.10.0/24, 1 successors, FD is 281600
         via Connected, Ethernet0/1
P 192.168.1.0/24, 1 successors, FD is 2169856
         via Connected, Serial0/1
```

Both paths appear in the table, but the distance on the tunnel path is huge compared with that of the serial interface. To find out why, let's take a look at the virtual tunnel interface:

```
Router-A# sho int tu 0
Tunnel0 is up, line protocol is up
  Hardware is Tunnel
  Internet address is 172.16.0.1/24
  MTU 1514 bytes, BW 9 Kbit, DLY 500000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation TUNNEL, loopback not set
  Keepalive not set
  Tunnel source 10.100.100.100 (Loopback0), destination 10.200.200.200
  Tunnel protocol/transport GRE/IP, key disabled, sequencing disabled
  Checksumming of packets disabled,  fast tunneling enabled
  Last input 00:00:00, output 00:00:00, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: fifo
  Output queue :0/0 (size/max)
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     88293 packets input, 7429380 bytes, 0 no buffer
```

```
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     80860 packets output, 6801170 bytes, 0 underruns
     0 output errors, 0 collisions, 0 interface resets
     0 output buffer failures, 0 output buffers swapped out
```

Take a look at the bandwidth and delay for the tunnel interface. The defaults are an extremely low speed, and an extremely high delay, and because EIGRP uses these metrics to determine feasible distance, the tunnel appears to be a much less desirable path than the existing serial links. This is beneficial because the tunnel is built over what could potentially be a multitude of links and routers. A tunnel is a software device, which means that the processing delay for the interface is variable (unlike with a physical interface). Tunnels should not be the most desirable paths by default.

To prove that the tunnel is running, and to show how the virtual link behaves, here is a traceroute from Router A to the closest serial interface on Router D, which is 192.168.3.2 on S0/1:

```
Router-A# trace 192.168.3.2

Type escape sequence to abort.
Tracing the route to 192.168.3.2

  1 192.168.1.2 4 msec 4 msec 0 msec
  2 192.168.2.2 4 msec 4 msec 4 msec
  3 192.168.3.2 4 msec 4 msec 4 msec
Router-A#
```

And here's a traceroute to the remote end of the tunnel on Router D (172.16.0.2), which is on the same router some three physical hops away:

```
Router-A# trace 172.16.0.2

Type escape sequence to abort.
Tracing the route to 172.16.0.2

  1 172.16.0.2 4 msec 4 msec 4 msec
Router-A#
```

The other end of the tunnel appears to the router to be the other end of a wire or link, but in reality, the tunnel is a logical construct composed of many intermediary devices that are not visible. Specifically, the tunnel hides the fact that Routers B and C are in the path.

GRE Tunnels and Routing Protocols

The introduction of a routing protocol across a GRE tunnel can cause some interesting problems. Take, for example, our network, now altered as shown in Figure 12-3. This time we have the links between the routers updating routes using RIPv2. The

other interfaces on Router A and Router D are included in RIP using the `redistribute connected` command. EIGRP is running on all interfaces on Routers A and D with the exception of the serial links, which are running RIPv2.

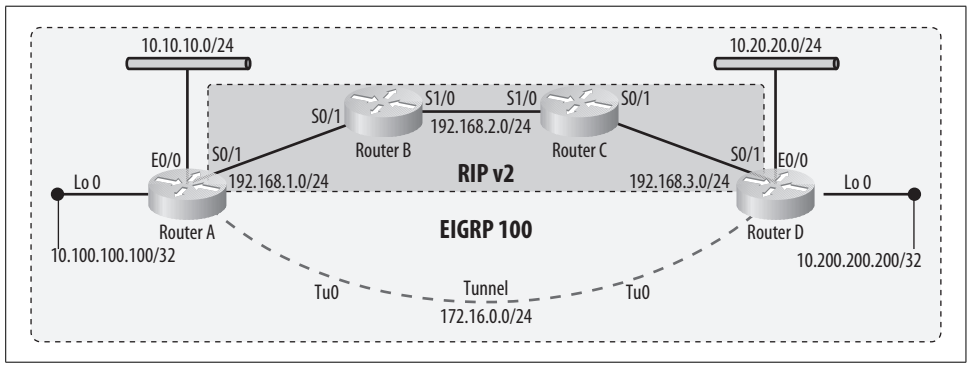


Figure 12-3. Recursive routing example

While this may look a bit odd, consider the possibility that the Routers B and C are owned and operated by a service provider. We cannot control them, and they only run RIPv2. We run EIGRP on our routers (A and D) and want to route between them using EIGRP.

Here are the pertinent configurations for Router A and Router D (remember, in this scenario, Routers B and C are beyond our control):

- Router A:

```
interface Loopback0
 ip address 10.100.100.100 255.255.255.255
!
interface Tunnel0
 ip address 172.16.0.1 255.255.255.0
 tunnel source Loopback0
 tunnel destination 10.200.200.200
!
interface Ethernet0/1
 ip address 10.10.10.1 255.255.255.0
!
interface Serial0/1
 ip address 192.168.1.1 255.255.255.0

router eigrp 100
 network 10.10.10.0 0.0.0.255
 network 10.100.100.0 0.0.0.255
 network 172.16.0.0 0.0.0.255
 no auto-summary
!
router rip
 version 2
 redistribute connected
```

```
 passive-interface Ethernet0/0
 passive-interface Loopback0
 passive-interface Tunnel0
 network 192.168.1.0
 no auto-summary
```

- Router D:

```
interface Loopback0
 ip address 10.200.200.200 255.255.255.255
!
interface Tunnel0
 ip address 172.16.0.2 255.255.255.0
 tunnel source Loopback0
 tunnel destination 10.100.100.100
!
interface FastEthernet0/0
 ip address 10.20.20.1 255.255.255.0
!
interface Serial0/1
 ip address 192.168.3.2 255.255.255.0
!
router eigrp 100
 network 10.20.20.0 0.0.0.255
 network 10.200.200.0 0.0.0.255
 network 172.16.0.0 0.0.0.255
 no auto-summary
!
router rip
 version 2
 redistribute connected
 passive-interface FastEthernet0/0
 passive-interface Loopback0
 passive-interface Tunnel0
 network 192.168.3.0
 no auto-summary
```

Everything looks fine, and in fact the tunnel comes up right away, but shortly after the tunnel comes up, we start seeing these errors on the console:

```
1d01h: %LINEPROTO-5-UPDOWN: Line protocol on Interface Tunnel0, changed state to up
1d01h: %TUN-5-RECURDOWN: Tunnel0 temporarily disabled due to recursive routing
1d01h: %LINEPROTO-5-UPDOWN: Line protocol on Interface Tunnel0, changed state to down
```

The error message `Tunnel0 temporarily disabled due to recursive routing` is a result of the destination of the tunnel being learned through the tunnel itself. With the tunnel manually shut down on Router A, the loopback interface on Router D is known through RIP, as expected:

```
Router-A# sho ip route

Gateway of last resort is not set

     10.0.0.0/8 is variably subnetted, 4 subnets, 2 masks
R       10.20.20.0/24 [120/3] via 192.168.1.2, 00:00:07, Serial0/1
C       10.10.10.0/24 is directly connected, Ethernet0/1
```

```
C       10.100.100.100/32 is directly connected, Loopback0
R       10.200.200.200/32 [120/3] via 192.168.1.2, 00:00:07, Serial0/1
C    192.168.1.0/24 is directly connected, Serial0/1
R    192.168.2.0/24 [120/1] via 192.168.1.2, 00:00:07, Serial0/1
R    192.168.3.0/24 [120/2] via 192.168.1.2, 00:00:07, Serial0/1
```

Once we bring the tunnel up and EIGRP starts working, the remote loopback becomes known through the tunnel:

```
Router-A# sho ip route

Gateway of last resort is not set

     172.16.0.0/24 is subnetted, 1 subnets
C       172.16.0.0 is directly connected, Tunnel0
     10.0.0.0/8 is variably subnetted, 4 subnets, 2 masks
D       10.20.20.0/24 [90/297246976] via 172.16.0.2, 00:00:04, Tunnel0
C       10.10.10.0/24 is directly connected, Ethernet0/1
C       10.100.100.100/32 is directly connected, Loopback0
D       10.200.200.200/32 [90/297372416] via 172.16.0.2, 00:00:04, Tunnel0
C    192.168.1.0/24 is directly connected, Serial0/1
R    192.168.2.0/24 [120/1] via 192.168.1.2, 00:00:00, Serial0/1
R    192.168.3.0/24 [120/2] via 192.168.1.2, 00:00:00, Serial0/1
```

Once this occurs, the routers on both sides immediately recognize the problem, and shut down the tunnel. The EIGRP route is lost, and the RIP route returns. The router will then bring the tunnel back up, and the cycle will continue indefinitely until something is changed.

The reason for the recursive route problem is the administrative distances of the protocols in play. RIP has an administrative distance of 120, while EIGRP has an administrative distance of 90. When the protocol with the better administrative distance learns the route, that protocol's choice is placed into the routing table. Figure 12-4 shows how the different routing protocols are both learned by Router A.

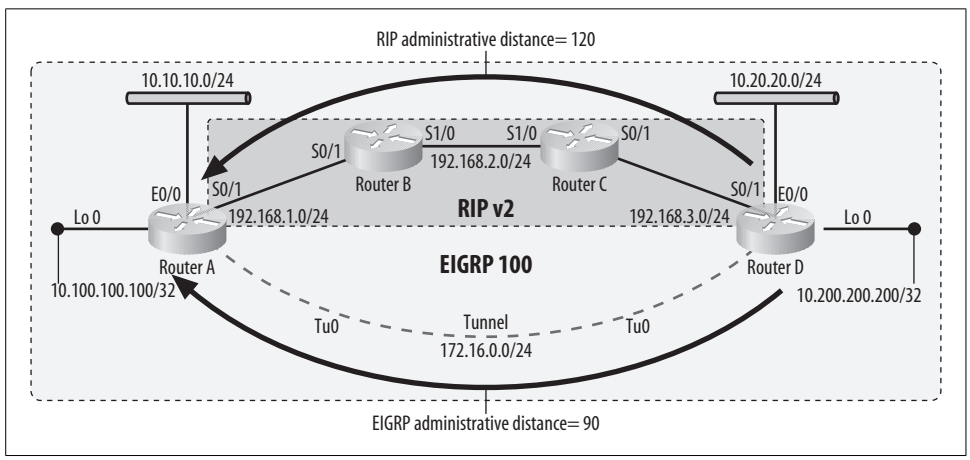


Figure 12-4. EIGRP route learned through tunnel

In the case of the tunnel running EIGRP, the issue is that the route to the remote end of the tunnel is known through the tunnel itself. The tunnel relies on the route to the remote loopback interface, but the tunnel is also providing the route to the remote loopback interface. This is not allowed, so the router shuts down the tunnel. Unfortunately, it then brings the tunnel back up, which causes the routes to constantly change, and the tunnel to become unstable.

The solutions to this problem are either to stop using tunnels (recommended in this case), or to filter the remote side of the tunnel so it is not included in the routing protocol being run through the tunnel (EIGRP). Installing a VPN would work as well, as the VPN would hide the "public" networks from the "inside" of either side, thus alleviating the problem. Looking at our configurations, the problem is that we've included the loopback networks in our EIGRP processes. Removing them solves our recursive route problems:

```
Router-A (config)# router eigrp 100
Router-A(config-router)# no network 10.100.100.0 0.0.0.255
```

Here's the new configuration for Router A:

```
router eigrp 100
 network 10.20.20.0 0.0.0.255
 network 10.200.200.0 0.0.0.255
 network 172.16.0.0 0.0.0.255
 no auto-summary
```

We'll do the same on Router D for its loopback network, and then we'll be able to see the desired result on Router A. Now, the route to the remote loopback address has been learned through RIP, and the route to the remote Ethernet has been learned through EIGRP:

```
Router-A# sho ip route

Gateway of last resort is not set

     172.16.0.0/24 is subnetted, 1 subnets
C       172.16.0.0 is directly connected, Tunnel0
     10.0.0.0/8 is variably subnetted, 4 subnets, 2 masks
D       10.20.20.0/24 [90/297246976] via 172.16.0.2, 00:03:23, Tunnel0
C       10.10.10.0/24 is directly connected, Ethernet0/1
C       10.100.100.100/32 is directly connected, Loopback0
R       10.200.200.200/32 [120/3] via 192.168.1.2, 00:00:11, Serial0/1
C    192.168.1.0/24 is directly connected, Serial0/1
R    192.168.2.0/24 [120/1] via 192.168.1.2, 00:00:12, Serial0/1
R    192.168.3.0/24 [120/2] via 192.168.1.2, 00:00:12, Serial0/1
```

GRE tunnels are not usually a good idea, because they complicate networks. If someone were troubleshooting the network we just built, the tunnel would only add complexity. Usually, the introduction of a GRE tunnel into a network without a clear need is the result of poor planning or design. Routing across a VPN is one of the few legitimate needs for a GRE tunnel. (IPSec, the protocol widely used in VPN, does not forward multicast or broadcast packets, so GRE is required.)

Running a GRE tunnel through a VPN tunnel can get you the routing protocol link you need. Why not just run the GRE tunnel? Remember that GRE does not encrypt data. Figure 12-5 shows a common layout incorporating GRE over VPN.

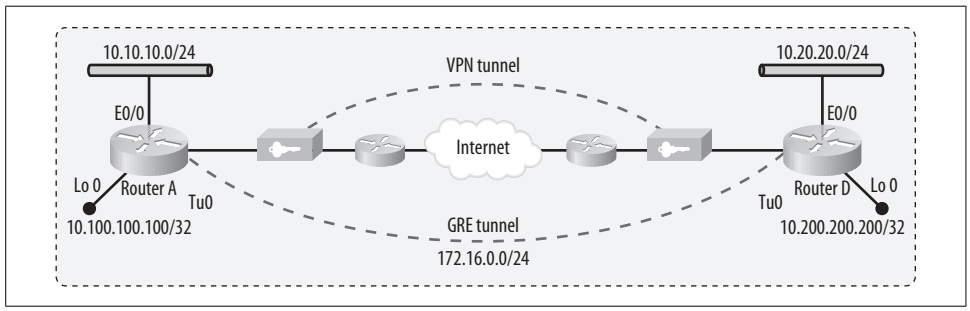


Figure 12-5. GRE through VPN

The configuration for this example is identical with regard to EIGRP and the tunnel. The difference is that in this case, there is no RIP in the middle. Routes to the remote end of the VPN tunnel are default routes for the VPN concentrators because they will have public IP addresses. The VPN concentrators will provide the ability to see the remote router's loopback address through static routes.

This is a relatively common application of GRE, which is necessary when running routing protocols over a VPN. The risk of recursive routing is still present, though, so care must be taken to prevent the remote loopback networks from being included in the EIGRP routing processes.

GRE and Access Lists

GRE is a protocol on the same level as TCP and UDP. When configuring a firewall to allow GRE, you do not configure a port like you would for telnet or SSH. Instead, you must configure the firewall to allow protocol 47. Cisco routers offer the keyword gre when configuring access lists:

```
R1(config)# access-list 101 permit ?
  <0-255>  An IP protocol number
  ahp      Authentication Header Protocol
  eigrp    Cisco's EIGRP routing protocol
  esp      Encapsulation Security Payload
  gre      Cisco's GRE tunneling
  icmp     Internet Control Message Protocol
  igmp     Internet Gateway Message Protocol
  igrp     Cisco's IGRP routing protocol
  ip       Any Internet Protocol
  ipinip   IP in IP tunneling
  nos      KA9Q NOS compatible IP over IP tunneling
  ospf     OSPF routing protocol
  pcp      Payload Compression Protocol
```

```
pim      Protocol Independent Multicast
tcp      Transmission Control Protocol
udp      User Datagram Protocol
```

PIX firewalls also support the keyword gre:

```
PIX(config)# access-list In permit gre host 10.10.10.10 host 20.20.20.20
```

The Point-to-Point Tunneling Protocol (PPTP) uses GRE, so if you're using this protocol for VPN access, you will need to allow GRE on your firewall.

CHAPTER 13

Resilient Ethernet

When designing a network, eliminating single points of failure should be a priority for any network engineer or architect. While it may be easy to assume that having two of every device will provide redundancy, how does one go about truly making the devices redundant?

Devices like PIX firewalls and CSM load balancers have redundancy and fault-tolerance features built into their operating systems, which even go so far as to transfer configuration changes from the primary to the secondary devices. Cisco routers don't really have that level of functionality, though, and with good reason. While you may wish to have two routers be a failover default gateway for a LAN, those two routers may have different serial links connected to them, or perhaps a link from one Internet provider connects to one router, while a link from a different provider connects to the other. The router configurations will not be the same, so configuration sync will not be practical.

Usually, on routers we're looking for the ability for one device to take over for another device on a specific network. Routers generally support multiple protocols, and connect many types of technologies, and each technology can be configured with the failover method preferred for that technology. In the case of Ethernet, the methods most often used are the Hot Standby Router Protocol (HSRP) and the Virtual Router Redundancy Protocol (VRRP). HSRP is Cisco-specific, while VRRP is nonproprietary, and thus available on other vendors' equipment as well. I will cover HSRP, as it is the most commonly used solution on Cisco routers.

HSRP

HSRP works by configuring one or more routers with `ip standby` commands on the interfaces that are to be part of an *HSRP group*. In its simplest form, two routers will each have one interface on a network. One of the routers will be the primary, and one will be the secondary. If the primary fails, the secondary will take over.

The details of Cisco's HSRP can be found in RFC 2281, which is titled "Cisco Hot Standby Router Protocol (HSRP)."

Figure 13-1 shows a simple design with a redundant pair of routers acting as a default gateway. The normal design in such a case would dictate that one router is a primary, and one is a secondary (or, in HSRP terms, one is *active* and one is *standby*). However, this diagram does not contain enough information to determine how the network is behaving. Which router is actually forwarding packets? Are they both forwarding packets? Is one a primary and the other a secondary router?

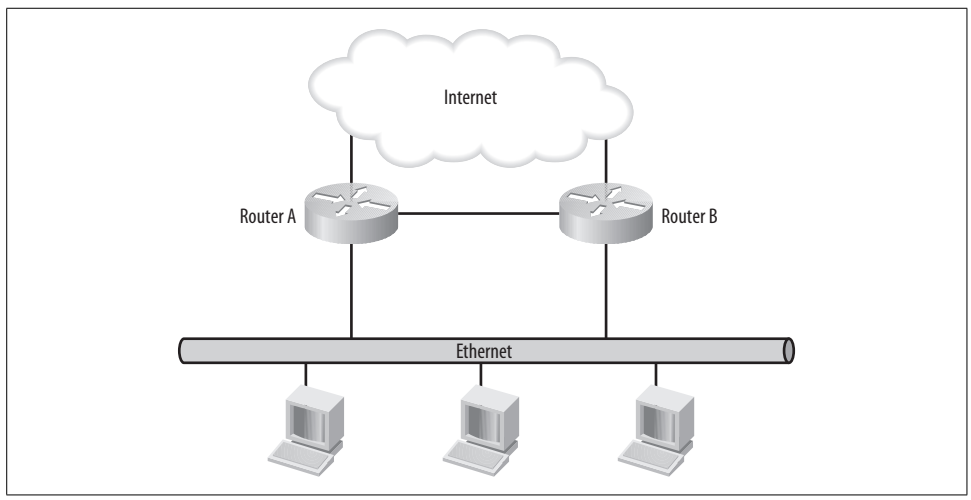


Figure 13-1. Simple HSRP design

To configure HSRP for this network, we'll need to determine three things ahead of time: the IP address of Router A's Ethernet interface, the IP address of Router B's Ethernet interface, and a Virtual IP address (VIP) that will act as the gateway for the devices on the network.

The IP addresses of the router interfaces never change, and neither does the VIP. The only thing that changes in the event of a failure is who owns the VIP. The VIP is active on whichever router has the highest priority. The priority defaults to a value of 100, and can be configured to any value between 0 and 255.

All routers that are in the same HSRP group (the default group is 0) send out HSRP packets to the multicast address 224.0.0.2 using UDP port 1985. All HSRP packets have a time-to-live (TTL) of 1, so they will not escape the local Ethernet segment.

When a router with an interface running HSRP starts that interface (or at any time the interface comes up), HSRP sends out hello packets and waits to see if any other HSRP routers are found. If more than one HSRP router is found, the routers

negotiate to determine who should become the active router. The router with the highest priority becomes the active router, unless there is a tie, in which case the router with the highest configured IP address becomes the active router.

In our example, we'll apply the three IP addresses needed as shown in Figure 13-2.

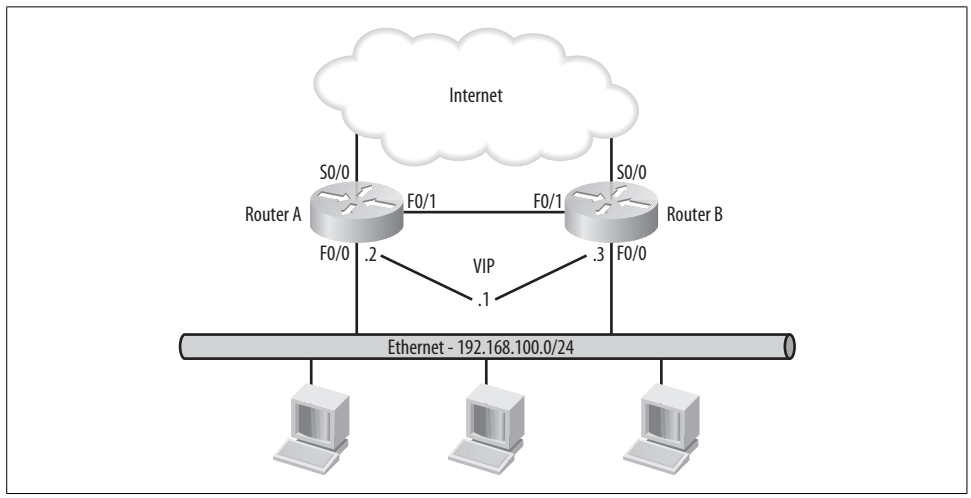


Figure 13-2. IP addresses assigned

We can now create the simplest of HSRP configurations:

- Router A:

```
interface f0/0
 ip address 192.168.100.2 255.255.255.0
 standby ip 192.168.100.1
 standby preempt
```

- Router B:

```
interface f0/0
 ip address 192.168.100.3 255.255.255.0
 standby ip 192.168.100.1
 standby priority 95
 standby preempt
```

On each router, we assign the IP address to the interface as usual. We also assign the same standby IP address to both—this is the VIP.

Notice that only Router B has a `standby priority` statement. Remember that the default priority is 100, so by setting Router B to a priority of 95, we have made Router B the standby (since 95 is lower than 100).

Lastly, both router configurations contain the command `standby preempt`. By default, HSRP does not reinstate the primary as the active router when it comes back online. To enable this behavior, you must configure the routers to preempt. This means that when a router with a higher priority than the active router comes online, the active router will allow the higher-priority router to become active.

To view the status of HSRP on the routers, we can execute the `show standby` command:

- Router A:

```
Router-A> sho standby
FastEthernet0/0 - Group 0
  Local state is Active, priority 100, may preempt
  Hellotime 3 sec, holdtime 10 sec
  Next hello sent in 0.412
  Virtual IP address is 192.168.100.1 configured
  Active router is local
  Standby router is 192.168.100.3 expires in 7.484
  Virtual mac address is 0000.0c07.ac00
  2 state changes, last state change 23w3d
```

- Router B:

```
Router-B> sho standby
FastEthernet0/0 - Group 0
  Local state is Standby, priority 95, may preempt
  Hellotime 3 sec, holdtime 10 sec
  Next hello sent in 1.398
  Virtual IP address is 192.168.100.1 configured
  Active router is 192.168.100.2 priority 100 expires in 9.540
  Standby router is local
  2 state changes, last state change 23w3d
```

Router A's output reflects that Router A is active with a priority of 100 and may preempt. We can also see that the VIP is 192.168.100.1, and that the standby router is 192.168.100.3. With more than two routers participating, it may not be obvious without this information which router is the standby router.

One important aspect of HSRP that many people miss is that if more than two routers are participating, once an election for active and standby routers has completed, the remaining routers are neither active nor standby until the standby router becomes active. RFC 2281 states:

> To minimize network traffic, only the active and the standby routers send periodic HSRP messages once the protocol has completed the election process. If the active router fails, the standby router takes over as the active router. If the standby router fails or becomes the active router, another router is elected as the standby router.

HSRP Interface Tracking

While HSRP is a wonderful solution that enables recovery from router or Ethernet interface failures, its basic functionality falls short in another scenario. Figure 13-3 depicts a more complex problem than the one considered previously. In this scenario, the serial link connecting Router A to the Internet has failed. Because the router and Ethernet interfaces are still up, HSRP is still able to send and receive hello packets, and Router A remains the active router.

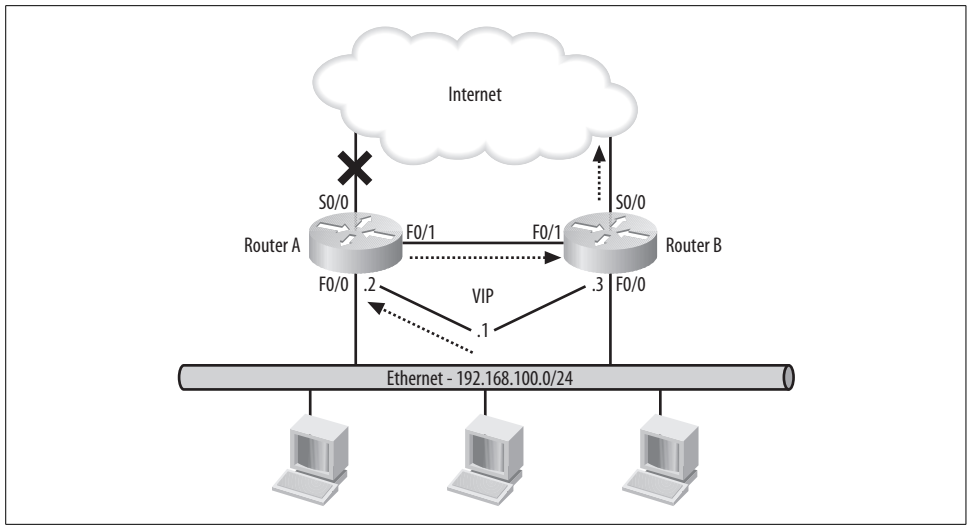


Figure 13-3. Primary Internet link failure without interface tracking

The network is resilient, so the packets will still get to the Internet via the F0/1 interfaces—but why add another hop when we don't need to? If we could somehow influence the HSRP priority based on the status of another interface, we could fail the VIP from Router A over to Router B based on the status of S0/0. *HSRP interface tracking* allows us to do exactly that.

By adding a couple of simple commands to our HSRP configurations, we can create a design that will allow the Ethernet interfaces to failover in the result of a serial interface failure:

- Router A:

```
interface f0/0
 ip address 192.168.100.2 255.255.255.0
 standby ip 192.168.100.1
 standby preempt
 standby track Serial0/0 10
```

- Router B:

```
interface f0/0
 ip address 192.168.100.3 255.255.255.0
 standby ip 192.168.100.1
 standby priority 95
 standby preempt
 standby track Serial0/0 10
```

On each router, we have added the `standby track Serial0/0 10` command to the Ethernet interface. This command tells HSRP to decrement the Ethernet interface's priority by 10 if the Serial0/0 interface goes down.

I've seen many networks where one router has a priority of 100, and the other has a priority of 90. When a tracked interface on the primary fails, this will result in a tie, which will cause IOS to assign the router with the highest configured IP address as the active router. While this may not seem like a problem with only two routers, traffic may not flow where you expect it to in this situation.

Adding a priority decrement value is a very handy feature. If each router had three links to the Internet, for instance, you could decrement the priority by 3 for each tracked interface. In our example, if one link went down, Router A would remain active, but if two serial links went down, we would decrement its priority by a total of 6, bringing it down to 94; this would be lower than Router B's priority of 95, so Router B would become the active router. In other words, with two routers, each containing three links to the Internet, the one with the most serial links up would become the active router. (Of course, a router or Ethernet interface failure would still affect the routers in the same way as the basic example.)

When HSRP Isn't Enough

HSRP is an awesome tool, and with the addition of interface tracking, it can be the means for near total redundancy. There are situations, however, where HSRP is not enough. The example I will show here is one of my favorite interview questions, because usually only someone with real-world experience in complex networks has seen it.

Figure 13-4 shows a deceptively simple HSRP setup. Two locations, New York and Los Angeles, are connected via two T1s. The routers on either side are connected via the F0/1 interfaces, and HSRP is implemented with interface tracking on the F0/0 interfaces. The idea here is that if either of the primary routers should fail, the secondary routers will take over for them. Additionally, should the primary T1 link fail, the secondary link should take over because interface tracking is enabled.

Here are the primary Ethernet configurations for each router:

- NY-Primary:

```
interface f0/0
 ip address 10.10.10.2 255.255.255.0
 standby ip 10.10.10.1
 standby preempt
 standby track Serial0/0 10
```

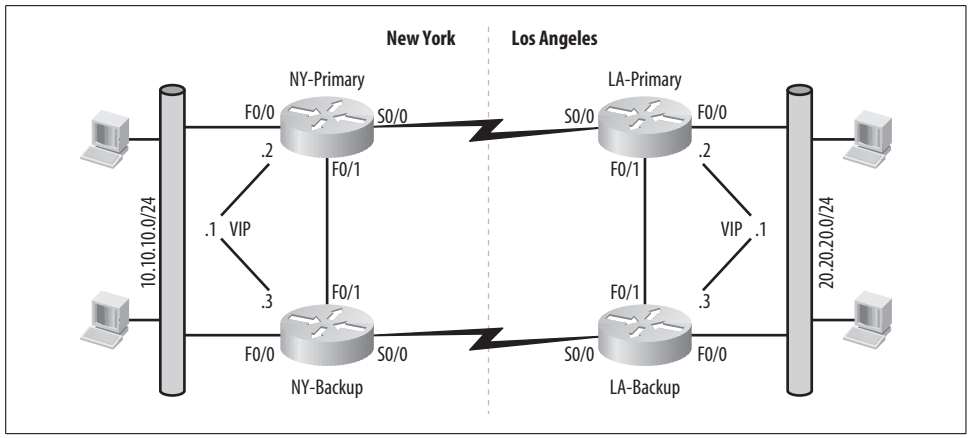


Figure 13-4. Two-link failover scenario using HSRP

- NY-Secondary:

```
interface f0/0
 ip address 10.10.10.3 255.255.255.0
 standby ip 10.10.10.1
 standby priority 95
 standby preempt
 standby track Serial0/0 10
```

- LA-Primary:

```
interface f0/0
 ip address 20.20.20.2 255.255.255.0
 standby ip 20.20.20.1
 standby preempt
 standby track Serial0/0 10
```

- LA-Secondary:

```
interface f0/0
 ip address 20.20.20.3 255.255.255.0
 standby ip 20.20.20.1
 standby priority 95
 standby preempt
 standby track Serial0/0 10
```

Should the T1 connecting NY-Primary with LA-Primary go down completely, the NY and LA routers will recognize the failure, and the secondary routers will take over. But real-world problems tend to be more complex than theoretical ones, and this design doesn't work as well as we'd like. Figure 13-5 shows what can go wrong.

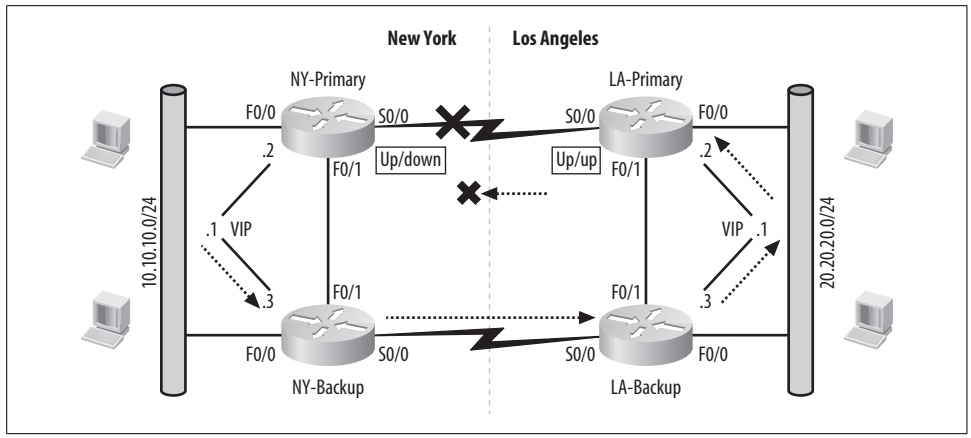


Figure 13-5. HSRP limitations

Assume that the link between New York and Los Angeles suffers a partial outage. Something has happened to cause the serial interface on NY-Primary to enter a state of up/down, but the serial interface on LA-Primary has stayed up/up. I've seen this more than once on different kinds of circuits.

Metropolitan Area Ethernet (Metro-E) is susceptible to this condition. Because the Metro-E link is usually a SONET transport that's converted to Ethernet, link integrity is local to each side. If you unplug one side of a Metro-E circuit, the far side will not go down with most installations.

HSRP responds to the down interface on the New York side by making the NY-Backup router active because we're tracking the serial interface on NY-Primary. Packets are forwarded to NY-Backup, and then across the T1 to LA-Backup, which forwards them to their destinations. The return packets have a problem, though. As the LA-Primary router does not recognize the link failure on the primary T1, it has remained the active router. Return packets are sent to the LA-Primary router, and because it believes the link is still up, it forwards the packets out the S0/0 interface, where they die because the other side of the link is down.

A more robust solution to a link-failover scenario is to incorporate an interior gateway protocol running on all of the routers. A protocol like OSPF or EIGRP establishes neighbor adjacencies across links. When a link fails, the routing protocol knows that the remote neighbor is unavailable, and removes the link from the routing table.

The solution therefore includes a routing protocol in addition to HSRP. Figure 13-6 shows the same network, but with EIGRP included. Now when the NY-Primary side of the primary links fails, EIGRP loses the neighbor adjacency between NY-Primary and LA-Primary, and removes the link from each router's routing table. Because EIGRP alters the routing tables, it can route around the failed link, even though one side reports an up/up condition. HSRP is still required in this design, as EIGRP has no means of making two routers appear to be a single gateway on the local Ethernet segments.

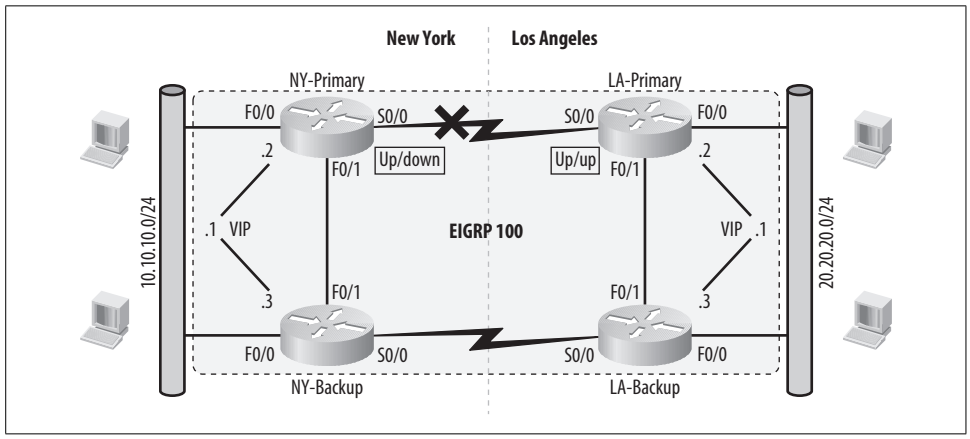


Figure 13-6. Better failover design using EIGRP

CHAPTER 14
Route Maps

Route maps are the bane of many people studying for certification exams. I think the reason for this lies in the way route maps are designed. They're a little bit backward when compared with more common features, like access lists. Why do I consider them backward? Let's take a look.

An access list lists the function of each entry in the entry itself. For example, this line permits any IP packet from any source to any destination:

```
access-list 101 permit ip any any
```

The syntax is pretty straightforward and self-documenting. Access list 101 permits IP packets from anywhere to anywhere. Simple!

In contrast, a route map written to accomplish the same thing might look like this:

```
route-map GAD permit 10
 match ip address 101
```

To determine what the route map is for, you have to see what access list 101 is doing, then figure out how the route map is applying it. This route map also permits any IP packet from any source to any destination, but unlike with the access list above, its purpose is not obvious.

Why add a route map to an already simple access list? First, there are instances where an access list is not directly available for use. BGP, for example, makes use of route maps, and, in many cases, does not support direct application of access lists. Second, route maps are far more flexible than access lists. They allow you to match on a whole list of things that access lists cannot:

```
R1(config)# route-map GAD permit 10
R1(config-route-map)# match ?
  as-path       Match BGP AS path list
  clns          CLNS information
  community     Match BGP community list
  extcommunity  Match BGP/VPN extended community list
  interface     Match first hop interface of route
  ip            IP specific information
  length        Packet length
```

```
  metric        Match metric of route
  route-type    Match route-type of route
  tag           Match tag of route
```

Route maps are particularly useful in routing protocols. Using route maps, you can filter based on route types, route tags, prefixes, packet size, and even the source or next hop of the packet.

Route maps can also alter packets, while access lists cannot. The `set` route map command allows you to change all sorts of things in a packet as it's being sent. You can change the interface to which the packet is being sent, the next hop of the packet, and Quality of Service (QoS) values such as IP precedence:

```
R1(config-route-map)# set ?
  as-path           Prepend string for a BGP AS-path attribute
  automatic-tag     Automatically compute TAG value
  clns              OSI summary address
  comm-list         set BGP community list (for deletion)
  community         BGP community attribute
  dampening         Set BGP route flap dampening parameters
  default           Set default information
  extcommunity      BGP extended community attribute
  interface         Output interface
  ip                IP specific information
  level             Where to import route
  local-preference  BGP local preference path attribute
  metric            Metric value for destination routing protocol
  metric-type       Type of metric for destination routing protocol
  origin            BGP origin code
  tag               Tag value for destination routing protocol
  weight            BGP weight for routing table
```

The IP-specific items that can be changed are accessed under the `ip` category:

```
R1(config-route-map)# set ip ?
  default     Set default information
  df          Set DF bit
  next-hop    Next hop address
  precedence  Set precedence field
  qos-group   Set QOS Group ID
  tos         Set type of service field
```

Policy routing is the term used to describe using a route map to change information regarding where a packet is routed. Care should be used when policy routing, as policy-routing scenarios can involve process switching (which can put a serious strain on the router's CPU). Process switching is discussed in Chapter 15.

Building a Route Map

Route maps are named and are built from *clauses*. The name is included in each clause, and the clauses are numbered to determine the order in which they should be evaluated and to allow you to include/omit only certain steps without having to re-enter the entire route map. The default clause number is 10, and a good standard to use

is to number your clauses in intervals of 10. This allows you to insert multiple clauses without needing to redesign the entire route map. Individual clauses can be entered at any time. The parser will insert them in the proper order within the configuration.

Each clause can either be a `permit` or a `deny` clause, with `permit` being the default. How the `permit` and `deny` values affect the processing of the route map depends on the route map's application. The next section presents an example of policy routing using route maps.

Within each clause, there are two basic types of commands:

`match`
: Selects routes or packets based on the criteria listed

`set`
: Modifies information in either packets or routing protocol tables based on the criteria listed

`match` commands are evaluated in order of entry within the clause. `match` entries can be evaluated in two ways: multiple `match` entries on a single line will be considered logical OR tests, while multiple `match` entries on separate lines will be considered logical AND tests.

This code configures the route map GAD with the default clause values of `permit` and `10`. The first `match` tests for the IP address matching access list 101 OR 102 OR 103. The second `match` tests for the packet to be between 200 and 230 bytes in length:

```
route-map GAD permit 10
 match ip address 101 102 103
 match length 200 230
 set ip next-hop 10.10.10.10
```

By nature of the way route maps operate, any of the three access lists can be matched, AND the packet length must match the second test for the `set` command to be executed. If no match is made, no action is taken for this clause, and the next higher-numbered clause in the route map is evaluated.

If no `match` command is present in the clause, all packets or routes match. In this case, the `set` command is executed on every packet or route.

If no `set` command is present, no action is taken beyond the scope of the clause itself. This is useful when limiting redistribution in routing protocols—because you don't want to change anything, there doesn't need to be a `set` command. The route map will permit route redistribution until a `deny` clause is encountered.

The following route map would be applied to a `redistribute` statement. It would permit any routes that match access list 101 AND access list 102, while denying all others:

```
route-map GAD permit 10
 match ip address 101
 match ip address 102
route-map GAD deny 20
```

Be careful of the order in which you configure entries like these—in this case, access list 102 will be evaluated only if access list 101 matches. Combining the access lists into a single `match` statement changes the behavior of the route map. Because including the matches on a single line is considered an OR test, in this case, access list 102 will be evaluated regardless of whether access list 101 matches:

```
route-map GAD permit 10
 match ip address 101 102
route-map GAD deny 20
```

Similarly, we could use a route map to deny only certain routes while permitting all others by simply reversing the `permit` and `deny` clauses:

```
route-map GAD deny 10
 match ip address 101 102
route-map GAD permit 20
```

Policy-Routing Example

Policy routing is the act of routing packets using some intelligence other than normal routing. For example, with policy routing, you can send packets to a different destination than the one determined by the routing protocol running on the router. It does have some limitations, but this feature can get you out of some interesting jams. Figure 14-1 illustrates an example that comes from a real problem I encountered.

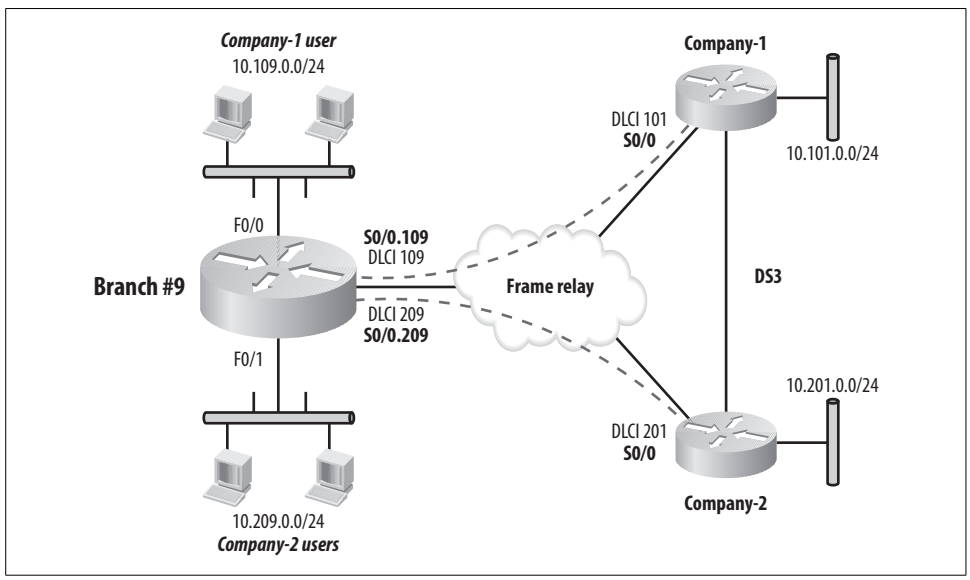


Figure 14-1. Policy-routing example

Two companies, Company 1 and Company 2, partnered together. To save money, they decided they would build each branch such that it would be a single office that connected directly to both companies' headquarters. To save more money, they

decided they would split the cost of a single router for each branch. One Ethernet interface connected the workers from Company 1, while another Ethernet interface connected the workers from Company 2. The workers from each company, while sitting in the same office, could not interact with workers from the other company using the network. We had to put access lists on the Ethernet interfaces to prevent interaction between the two networks.

This design is an excellent example of politics and money trumping best-practice engineering. Still, our job was not to judge, but rather to make the network function the way the client wanted it to function.

To further complicate the issue, each company insisted that its employees should only use the frame-relay link that that company had purchased. The problem with this mandate was that each company's branch employees used servers at both companies' headquarters. In other words, if a Company 1 user at Branch #9 needed to use a server at Company 2's headquarters, that user was not allowed to use the link that connected Branch #9 to Company 2. Instead, he was to use the link provided by Company 1, so that Company 1 could route (and attempt to secure) the request across the DS3 link between the two companies' headquarters. This needed to be done across more than 300 branches, all of which were configured the same way. We were not allowed to add hardware.

Here is the routing table from the Branch-9 router:

```
Branch-9# sho ip route
[- text removed -]

     172.16.0.0/24 is subnetted, 2 subnets
C       172.16.201.0 is directly connected, Serial0/0.209
C       172.16.101.0 is directly connected, Serial0/0.109
     10.0.0.0/24 is subnetted, 4 subnets
C       10.109.0.0 is directly connected, FastEthernet0/0
D       10.101.0.0 [90/2172416] via 172.16.101.1, 00:16:02, Serial0/0.109
D       10.201.0.0 [90/2172416] via 172.16.201.1, 00:16:06, Serial0/0.209
C       10.209.0.0 is directly connected, FastEthernet1/0
```

As you can see, the routing protocol (EIGRP, in this case) is doing what it is designed to do. The network 10.101.0.0/24 is available through the shortest path—the direct link on S0/0.109. The 10.201.0.0/24 network is similarly available through the shortest path, which is found on S0/0.209.

The problem is that routing is based on destination addresses. When a user on either of the locally connected Ethernet networks wants to get to either of the companies' HQ networks, the router doesn't care where the packets originate from; the routing protocol simply provides the best path to the destination networks. With route maps and policy routing, we were able to change that.

The configuration for the Ethernet links on Branch-9 is simple (I've removed the access lists that prevent inter-Ethernet communication to avoid any confusion, as these ACLs are not germane to this discussion):

```
interface FastEthernet0/0
 ip address 10.109.0.1 255.255.255.0

interface FastEthernet0/1
 ip address 10.209.0.1 255.255.255.0
```

What we had to do was add a policy map on each Ethernet interface that told the router to alter the next hop of each packet sent to the HQ offices, based on the source address contained in the packet.

First, we had to define our route maps. The logic for the route maps is shown in this snippet of pseudocode:

```
If the source network is 10.109.0.0/24 and the destination is 10.101.0.0/24
    Then send the packet out interface S0/0.109

If the source network is 10.209.0.0/24 and the destination is 10.201.0.0/24
    Then send the packet out interface S0/0.209
```

In route map terms, we needed to *match* the destination address of the packet and then *set* the next hop. These route maps would then be applied to the input interfaces to accomplish our goal.

To match IP addresses in route maps, you need to specify and include access lists. We made two access lists to match the destination networks. Access list 101 matched the 10.101.0.0/24 network (Company 1), and access list 102 matched the 10.201.0.0/24 network (Company 2):

```
access-list 101 permit ip any 10.101.0.0 0.0.0.255
access-list 101 remark <[ Company-1 Network ]>
!
access-list 102 permit ip any 10.201.0.0 0.0.0.255
access-list 102 remark <[ Company-2 Network >]
```

With the destination networks defined, we were able to create route map clauses to match against them. After matching the destination network, we needed to change the interface to which the packet would switch within the router. The first route map forced packets destined for Company 1's HQ network to go across Company 1's link:

```
route-map Company-1 permit 10
 match ip address 101
 set interface Serial0/0.109
```

The second forced traffic destined for Company 2's HQ network over Company 2's link:

```
route-map Company-2 permit 10
 match ip address 102
 set interface Serial0/0.209
```

With the route maps created, we needed to apply them to the interfaces. This was done with the `ip policy` interface command.

Some fall-through logic was used here. We knew that Company 1's users' packets headed to Company 1's headquarters would route properly without alteration, as would Company 2's users' packets headed to Company 2's headquarters. The only time we needed to interfere was when either company's users accessed a server at the other company's headquarters—then, and only then, we needed to force the packets to take the other link. To accomplish this, we applied the Company-2 route map to the Company 1 Ethernet interface, and vice versa. This was done on the Branch-9 router:

```
interface FastEthernet0/0
 description <[ Company-1 Users ]>
 ip address 10.109.0.1 255.255.255.0
 ip policy route-map Company-2
 half-duplex

interface FastEthernet0/1
 description <[ Company-2 Users ]>
 ip address 10.209.0.1 255.255.255.0
 ip policy route-map Company-1
 half-duplex
```

Policy routing takes place when a packet is received on an interface. For this reason, the policy must be placed on the Ethernet interfaces.

Monitoring Policy Routing

Once policy routing is configured, how do you know it's working? In the preceding source-routing example, we altered the way that packets routed, but the IP routing table didn't change. Normally, we would look at the routing table to determine where a router is sending packets, but with policy routing in place, the routing table is no longer the most reliable source of information.

Policy routing overrides the routing table, which can be confusing when troubleshooting routing problems. If your packets are not routing the way you expect them to, check for policy routing on your interfaces.

To see what policies are applied to what interfaces, use the `show ip policy` command:

```
Branch-9# sho ip policy
Interface          Route map
FastEthernet0/0    Company-1
FastEthernet0/1    Company-2
```

Another method for determining whether policy routing is enabled is with the `show ip interface` command. This command creates a lot of output, so filtering with `include Policy` is useful for our purposes:

```
Branch-9# sho ip int f0/0 | include Policy
  Policy routing is enabled, using route map Company-1
  BGP Policy Mapping is disabled
```

The problem with these commands is that they only show that a policy is applied, not how it's working. The command `show route-map` will show you all the route maps configured on the router, as well as some useful statistics regarding how many times the route map has been used for policy routing. This information is accumulative, so you can only assume the route map is working the way you want it to if the counters are incrementing:

```
Branch-9# sho route-map
route-map Company-2, permit, sequence 10
  Match clauses:
    ip address (access-lists): 102
  Set clauses:
    interface Serial0/0.209
  Policy routing matches: 656 packets, 68624 bytes
route-map Company-1, permit, sequence 10
  Match clauses:
    ip address (access-lists): 101
  Set clauses:
    interface Serial0/0.109
    ip next-hop 172.16.101.1
  Policy routing matches: 626 packets, 65304 bytes
```

Another option you can use to determine whether a router is acting on enabled policies is the `debug ip policy` command.

Take care when using debug, as it can impact the operation of the router. Remember that policy routing is applied to every packet that comes into an interface. This should be tested in a lab before you try it in a production network.

Here, a workstation on Company 2's user network in Branch #9 is pinging Company 1's HQ network (10.101.0.1):

```
D       10.101.0.0 [90/2172416] via 172.16.101.1, 03:21:29, Serial0/0.109
```

According to the routing table, these packets should route through S0/0.109. But, when the user pings 10.101.0.1, the debug output tells a different story:

```
04:49:24: IP: s=10.209.0.2 (FastEthernet0/1), d=10.101.0.1, len 100, FIB policy match
04:49:24: IP: s=10.209.0.2 (FastEthernet0/1), d=10.101.0.1, g=172.16.101.1, len 100,
FIB policy routed
04:49:25: IP: s=10.209.0.2 (FastEthernet0/1), d=10.101.0.1, len 100, FIB policy match
04:49:25: IP: s=10.209.0.2 (FastEthernet0/1), d=10.101.0.1, g=172.16.101.1, len 100,
FIB policy routed
```

```
04:49:25: IP: s=10.209.0.2 (FastEthernet0/1), d=10.101.0.1, len 100, FIB policy match
04:49:25: IP: s=10.209.0.2 (FastEthernet0/1), d=10.101.0.1, g=172.16.101.1, len 100,
FIB policy routed
04:49:25: IP: s=10.209.0.2 (FastEthernet0/1), d=10.101.0.1, len 100, FIB policy match
04:49:25: IP: s=10.209.0.2 (FastEthernet0/1), d=10.101.0.1
Branch-9#, g=172.16.101.1, len 100, FIB policy routed
04:49:25: IP: s=10.209.0.2 (FastEthernet0/1), d=10.101.0.1, len 100, FIB policy match
04:49:25: IP: s=10.209.0.2 (FastEthernet0/1), d=10.101.0.1, g=172.16.101.1, len 100,
FIB policy routed
```

Notice the entries in bold, each of which corresponds to a single ping packet. First, we see an `FIB policy match` entry. This indicates that one of the `match` statements in our route map was successful. The following line contains the phrase `FIB policy routed`. This indicates that the packet was policy-routed instead of being routed as it would normally.

Here's an example of packets that did not match the route map, and as such were routed normally:

```
04:52:35: IP: s=10.209.0.2 (FastEthernet0/1), d=10.201.0.1, len 100, FIB policy
rejected(no match) - normal forwarding
04:52:35: IP: s=10.209.0.2 (FastEthernet0/1), d=10.201.0.1, len 100, FIB policy
rejected(no match) - normal forwarding
04:52:35: IP: s=10.209.0.2 (FastEthernet0/1), d=10.201.0.1, len 100, FIB policy
rejected(no match) - normal forwarding
04:52:35: IP: s=10.209.0.2 (FastEthernet0/1), d=10.201.0.1, len 100, FIB policy
rejected(no match) - normal forwarding
```

Again, I've highlighted the entry for a single packet. This time we see the phrase `FIB policy rejected(no match) - normal forwarding`. This indicates that the packet did not match any clauses in the route map, and was forwarded by normal means.

See Chapter 11 for another example of route maps in action.

CHAPTER 15

Switching Algorithms in Cisco Routers

The term "switching," when used in the context of routers, describes the process of moving packets from one interface to another within a router. Packets in transit arrive at one interface, and must be moved to another, based on routing information stored in the router.

Routing is the process of choosing paths and forwarding packets to destinations outside of the physical router. *Switching* is the internal forwarding of packets between interfaces.

Just as there are different routing protocols for determining the external paths for packets, there are different internal methods for switching. These switching algorithms, or *paths*, are a valuable way to increase (or decrease) a router's performance.

One of the biggest impacts on how fast a packet gets from its source to its destination is the processing delay present in each router along the way. Different switching methods have vastly different impacts on a router's performance. Choosing the right one—and knowing what to look for when there's a problem—will help the savvy administrator keep a network running at peak performance.

A router must move packets from one interface to another, just like a switch. The decisions about how to move packets from one interface to another are based on the *routing information base* (RIB), which is built manually, or by layer-3 routing protocols. The RIB is essentially the routing table (see Chapter 9 for details).

There are many types of switching within a Cisco router. They are divided into two categories: *process switching* and *interrupt context switching*. Process switching involves the processor calling a process that accesses the RIB, and waiting for the next scheduled execution of that process to run. Interrupt context switching involves the processor interrupting the current process to switch the packet. Interrupt context switching is further divided into three types:

- Fast switching
- Optimum switching
- Cisco Express Forwarding (CEF)

Each method uses different techniques to determine the destination interface. Generally speaking, the process of switching a packet involves the following steps:

1. Determine whether the packet's destination is reachable.
2. Determine the next hop to the destination, and to which interface the packet should be switched to get there.
3. Rewrite the MAC header on the packet so it can reach its destination.

While the information in this chapter may not benefit you in the day-to-day operation of your network, understanding how the different choices work will help you understand why you should choose one path over another. Knowledge of your router's switching internals will also help you understand why your router behaves differently when you choose different switching paths.

Figure 15-1 shows a simplified view of the inside of a router. There are three interfaces, all of which have access to input/output memory. When a packet comes into an interface, the router must decide to which interface the packet should be sent. Once that decision is made, the packet's MAC headers are rewritten, and the packet is sent on its way.

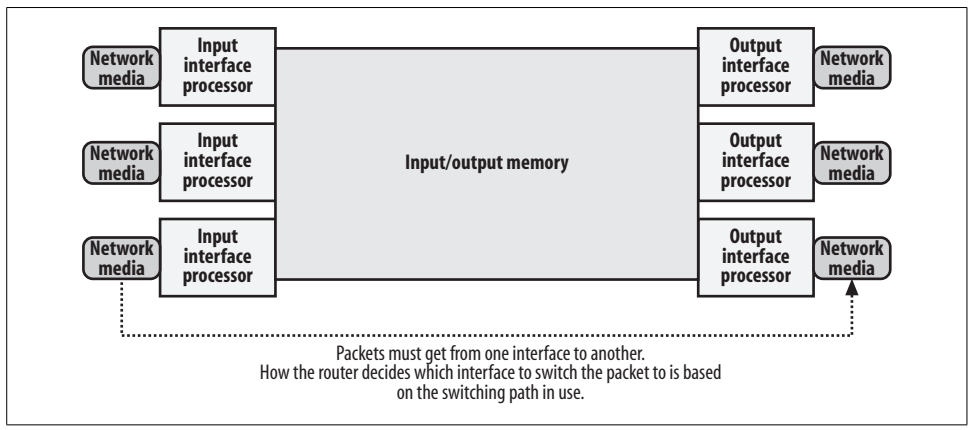


Figure 15-1. Router switching requirements

The routing table contains all the information necessary to determine the correct interface, but process switching must be used to retrieve data from the routing table, and this is inefficient. Interrupt context switching is typically preferred.

The number of steps involved in forwarding a packet varies with the switching path used. The method of storing and retrieving next-hop and interface information also differs in each of the switching paths. Additionally, various router models operate differently in terms of memory and where the decisions are made.

Process Switching

The original method of determining which interface to forward a packet to is called process switching. This may be the easiest method to understand because it behaves in a way you'd probably expect.

With process switching, when a packet comes in, the processor calls a process that examines the routing table, determines what interface the packet should be switched to, and then switches the packet. This happens for every packet seen on every interface. Figure 15-2 shows the steps involved.

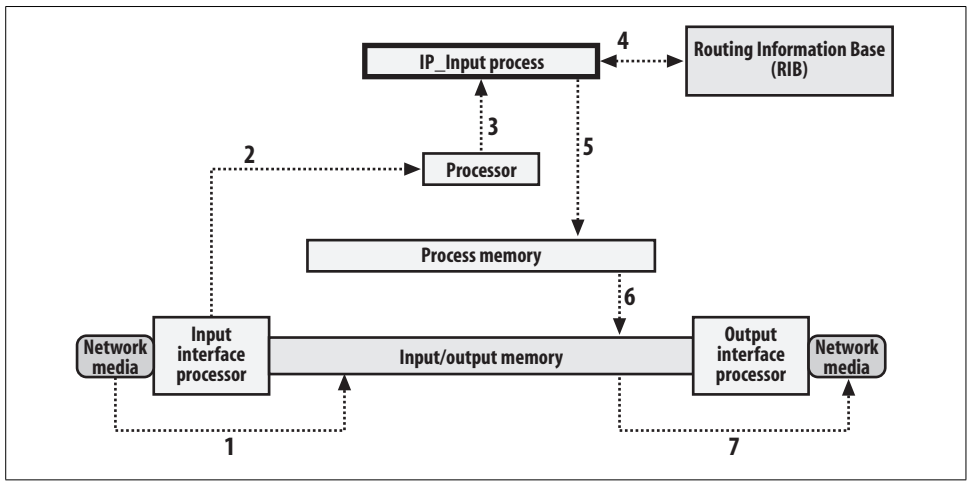


Figure 15-2. Process switching

These are the steps for process switching:

1. The interface processor detects a packet and moves the packet to the input/output memory.
2. The interface processor generates a *receive interrupt*. During this time, the central processor (CPU) determines the packet type (IP), and copies it to the processor memory, if necessary (this is platform-dependent). The processor then places the packet on the appropriate process's input queue and releases the interrupt. The process for IP packets is titled *ip_input*.
3. When the scheduler next runs, it notices the presence of a packet in the input queue for the *ip_input* process, and schedules the process for execution.
4. When the *ip_input* process runs, it looks up the next hop and output interface information in the RIB. The *ip_input* process then consults the ARP cache to retrieve the layer-2 address for the next hop.
5. The process rewrites the packet's MAC header with the appropriate addresses, then places the packet on the output queue of the appropriate interface.

6. The packet is moved from the output queue of the outbound interface to the transmit queue of the outbound interface. Outbound QoS happens in this step.
7. The output interface processor notices the packet in its queue, and transfers the packet to the network media.

There are a couple of key points in this process that make it particularly slow. First, the processor waits for the next scheduled execution of the *ip_input* process. Second, when the *ip_input* process finally runs, it references the RIB, which is a very slow process. The *ip_input* process is run at the same priority level as other processes on the router, such as routing protocols, and the HTTP web server interface.

The benefit of process switching is that it is available on every Cisco router platform, regardless of size or age. Packets sourced from or destined to the router itself, such as SNMP traps from the router and telnet packets destined for the router, are always process-switched.

As you can imagine, on large routers or routers that move a lot of packets, process switching can be very taxing. Even on smaller routers, process switching can cause performance problems. I've seen 2600 routers serving only a single T1 average 60–80 percent CPU utilization while using process switching.

Process switching should never be used as the switching method of choice. Any of the other methods will produce significantly better performance.

Interrupt Context Switching

Interrupt context switching is much faster than process switching. The increase in speed is largely due to the fact that the *ip_input* process is rarely called. Interrupt context switching instead interrupts the process currently running on the router to switch the packet. Interrupt context switching usually bypasses the RIB, and works with parallel tables, which are built more efficiently (the details of these tables differ according to the switching path in use). A considerable amount of time is also saved because the processor no longer has to wait for a process to complete.

The general steps for interrupt context switching are shown in Figure 15-3.

Interrupt context switching is a broad description that encompasses various switching paths: fast switching, optimum switching, and Cisco Express Forwarding, and includes the following steps:

1. The interface processor detects a packet and moves the packet into input/output memory.
2. The interface processor generates a receive interrupt. During this time, the central processor determines the packet type (IP) and begins to switch the packet.
3. The processor searches the route cache for the following information:
   a. Is the destination reachable?
   b. What should the output interface be?

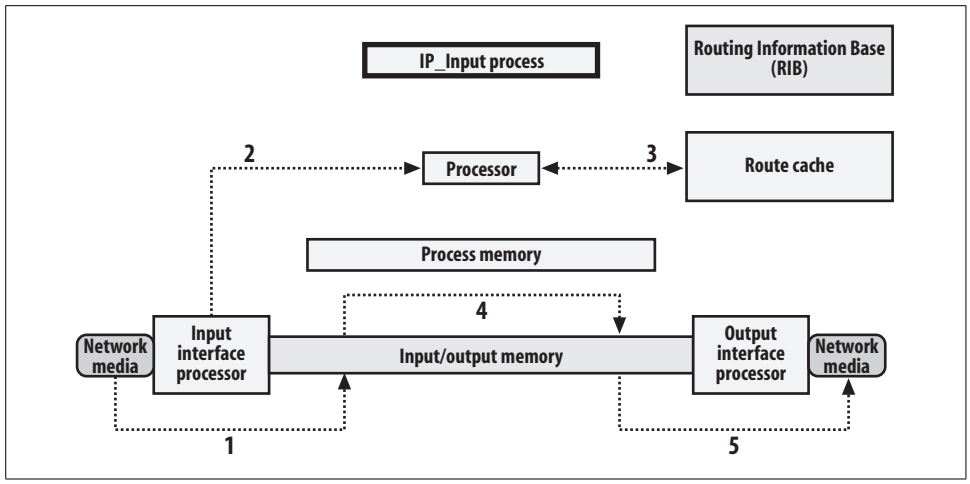


Figure 15-3. Interrupt context switching

c. What is the next hop?

d. What should the MAC addresses be converted to?

e. The processor then uses this information to rewrite the packet's MAC header.

4. The packet is copied to either the transmit or the output queue of the outbound interface. The receive interrupt is ended, and the originally running process continues.
5. The output interface processor notices the packet in its queue, and transfers the packet to the network media.

The obvious difference is that there are only five steps, as opposed to seven for process switching. The big impact comes from the fact that the currently running process on the router is interrupted, as opposed to waiting for the next scheduled execution of the *ip_input* process.

The RIB is also bypassed entirely in this model, and the necessary information is retrieved from other sources. In the example shown in Figure 15-3, the source is called the *route cache*. As we'll see, each switching path has its own means of determining, storing, and retrieving this information. The different methods are what separate the individual switching paths within the interrupt context switching group.

Fast Switching

Fast switching process-switches the first packet in a conversation, then stores the information learned about the destination for that packet in a table called the *route cache*. Fast switching uses the *binary tree* format for recording and retrieving information in the route cache.

Figure 15-4 shows an example of a binary tree as it might be viewed for fast switching. Each branch of the tree appends a 0 or a 1 to the previous level's value. Starting at 0 (the root), the next branch to the right contains a 0 on the bottom, and a 1 on the top. Each of those nodes again branches, appending a 0 on the bottom branch, and a 1 on the top branch. Eight levels of the tree are shown in Figure 15-4, but an actual tree used for IP addresses would have 32 levels, corresponding to the bits within an IP address.

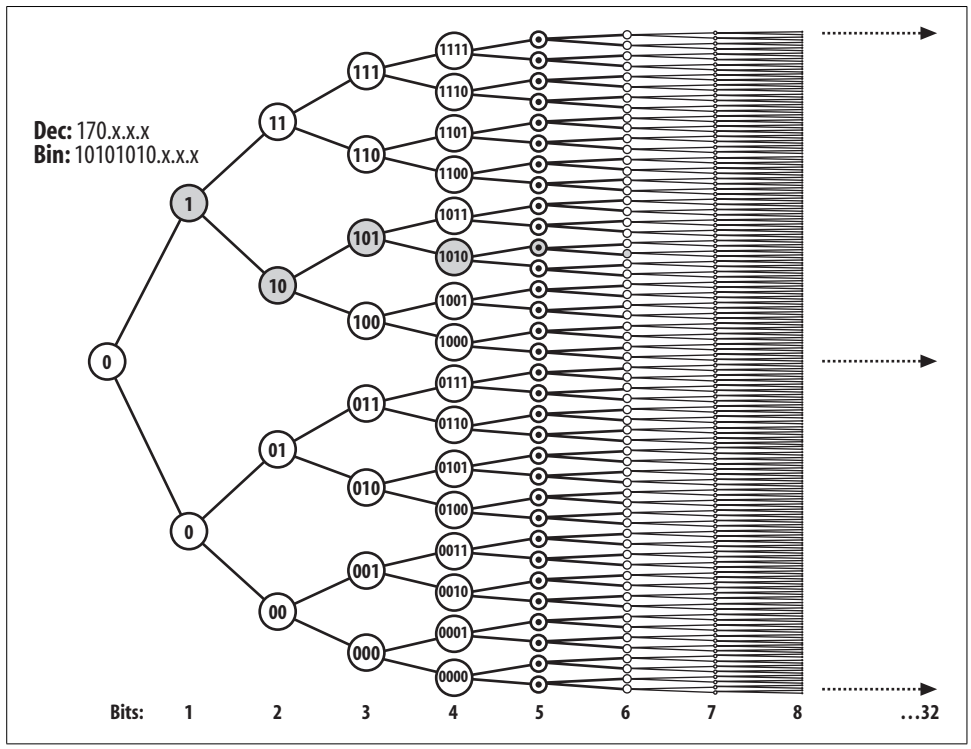


Figure 15-4. Fast-switching binary tree

The nodes marked in gray in Figure 15-4 match an IP address of 170.x.x.x. (The binary value of 170 is 10101010—this should make the example easier to visualize.) The IP address is only 170.x.x.x because the beyond eight bits I couldn't fit any more visible nodes in the drawing.

The benefit of this design is speed when compared with searching the RIB. Information regarding the next hop and MAC address changes is stored within each node. Since the tree is very deterministic, finding specific entries is very quick.

The drawbacks of this implementation include the sheer size of the table, and the fact that while the data for each address is stored within the nodes, the size of the data is not static. Because each node may be a different size, the table can be inefficient.

The route cache is not directly related to the routing table, and it is updated only when packets are process-switched. In other words, the route cache is updated only when the first packet to a destination is switched. From that point, the route cache is used, and the remaining packets are fast-switched. To keep the data in the route cache current, 1/20th of the entire route cache is *aged out* (discarded) every minute. This information must be rebuilt using process switching.

Because the ARP table is not directly related to the contents of the route cache, changes to the ARP table result in parts of the route cache being invalidated. Process switching must also be used to resolve these differences.

Optimum Switching

Optimum switching uses a *multiway tree* instead of a binary tree to record and retrieve information in the route cache. For the purposes of IP addresses, a multiway tree is much faster because each octet can only be one of 256 values. A binary tree is designed to support any type of value, so there is no limit to its potential size.

Figure 15-5 shows a multiway tree as it might appear for optimum switching. The root branches into 256 nodes numbered 0–255. Each node then branches into an additional 256 nodes. This pattern continues for four levels—one for each octet. The nodes in grey show how an IP address would be matched. The example used here is the address 3.253.7.5.

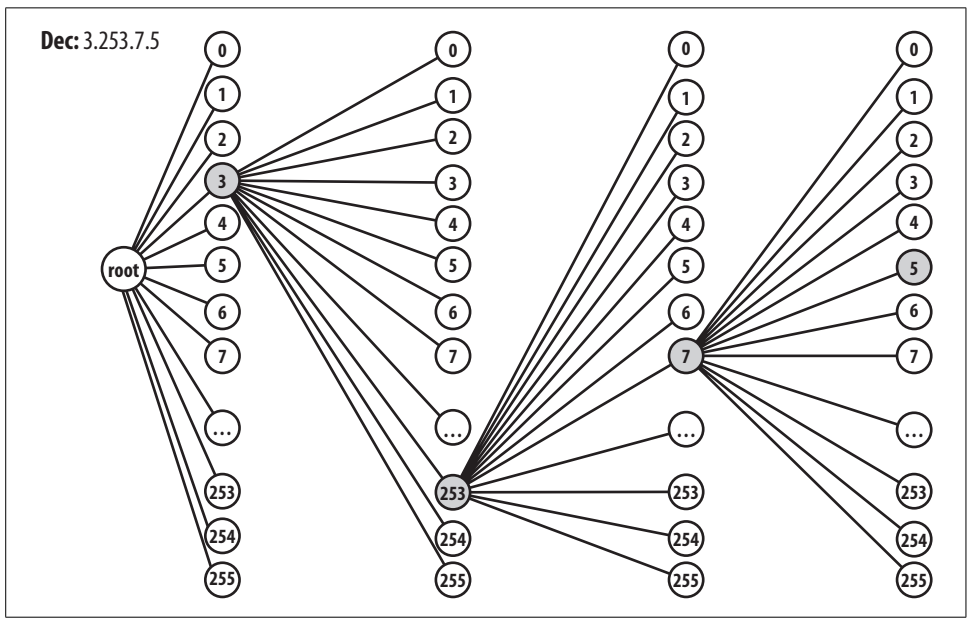


Figure 15-5. Optimum-switching multiway tree

This type of tree is much faster for IP address lookups than a binary tree because the table is optimized for the IP address format. Information for each route (prefix) or IP address is stored within the final node, as shown in Figure 15-6. Because the size of this information can be variable, and each node may or may not contain information, the overall table size is also variable. This is a drawback: searching the tree is not as efficient as it might be if every node were of a known static size.

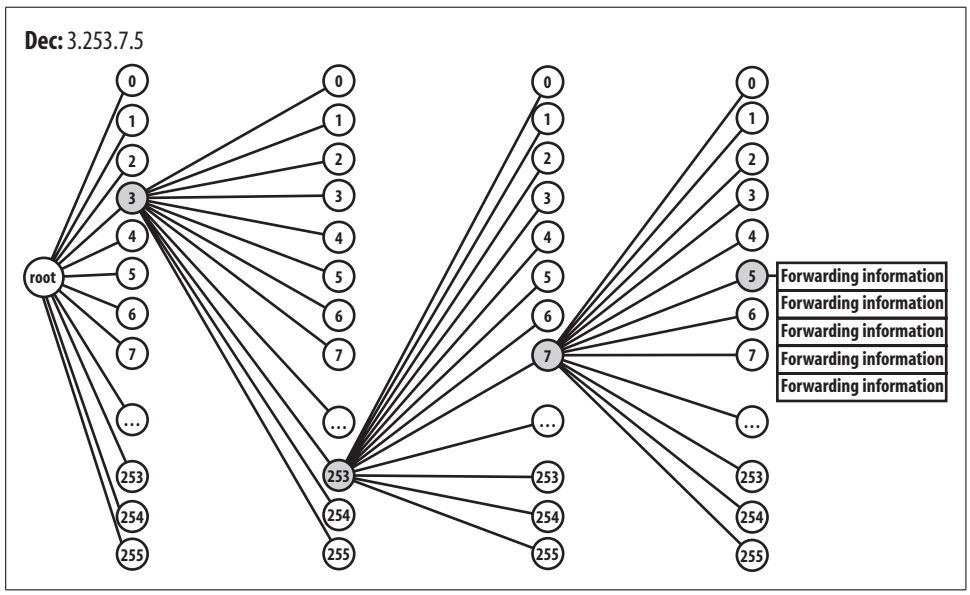


Figure 15-6. Data stored in optimum-switching multiway tree

Because the relevant data is stored in the nodes and has no direct relationship to the RIB or the ARP cache, entries are aged and rebuilt through process switching in the same manner used with fast switching.

Optimum switching is available only on high-end routers, such as models in the 7500 and 7600 series.

Cisco Express Forwarding

Cisco Express Forwarding (CEF) is the switching path of choice on any router that supports it. CEF is the default switching path on all modern routers.

CEF takes the ideas behind the optimum-switching multiway tree a step further by introducing the concept of a *trie*. The initial concept is the same as for a multiway or binary tree, but the data is not stored within the nodes. Rather, each node becomes a pointer to another table, which contains the data.

The term *trie* comes from the word *retrieve,* and is pronounced like *tree*. Some prefer the pronunciation *try* to differentiate the term from the word "tree."

In CEF, this trie is called the *forwarding table*. Each node is the same static size, and contains no data. Instead, the node's position in the trie is itself a reference to another table, called the *adjacency table*. This table stores the pertinent data, such as MAC header substitution and next hop information for the nodes. Figure 15-7 shows a representation of the CEF tables.

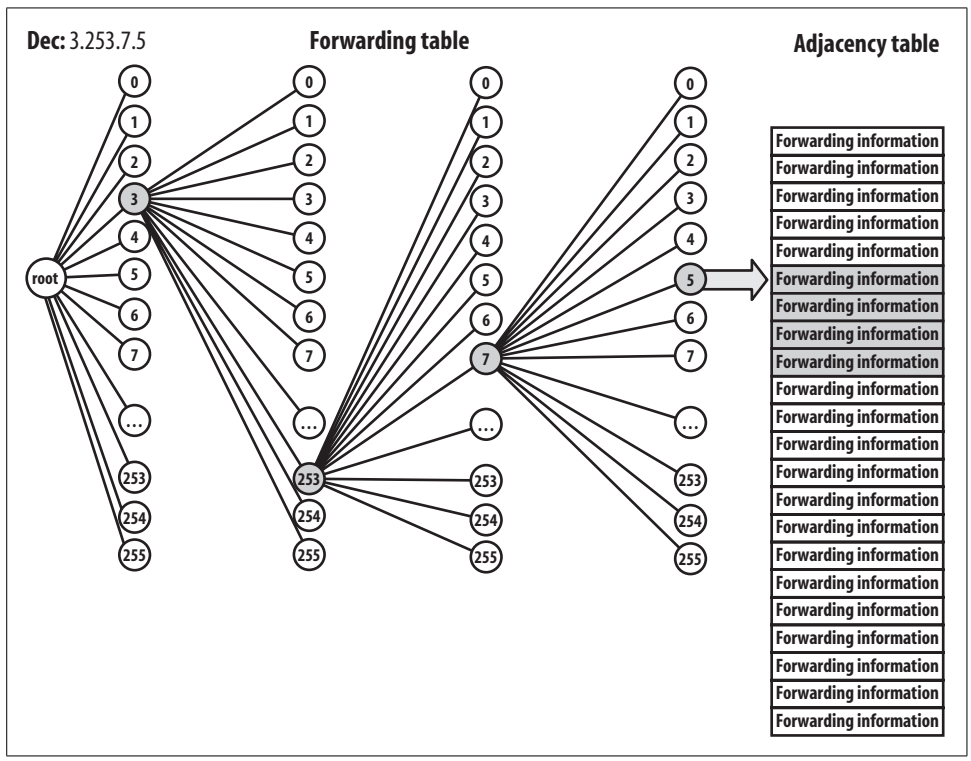


Figure 15-7. CEF forwarding and adjacency tables

One of the biggest advantages of CEF is the fact that the tables are built without process switching. Both tables can be built without waiting for a packet to be sent to a destination. Also, as the forwarding table is built separately from the adjacency table, an error in one table does not cause the other to become stale. When the ARP cache changes, only the adjacency table changes, so aging or invalidation of the forwarding table is not required.

CEF supports load balancing over equal-cost paths, as you'll see in the next section. Load balancing at the switching level is far superior to load balancing by routing protocols, as routing protocols operate at a much higher level. The latest versions of IOS incorporate CEF switching into routing protocol load balancing, so this has become less of an issue.

Configuring and Managing Switching Paths

Configuring switching paths is done both globally and at the interface level. This allows the flexibility of configuring different switching paths on each interface. For example, you may want to disable CEF on an interface to see whether it's causing problems.

Process Switching

To force a router to use process switching, turn off all other switching methods.

Here, I'm showing the performance of a Cisco 2621XM router with about 600k of traffic running over serial interface s0/1:

```
R1# sho int s0/1 | include minute
  5 minute input rate 630000 bits/sec, 391 packets/sec
  5 minute output rate 627000 bits/sec, 391 packets/sec
```

The normal switching method for this interface on this router is CEF. To show what switching path is running on interface s0/1, use the `show ip interface s0/1 | include switching` command:

```
R1# sho ip interface s0/1 | include switching
  IP fast switching is enabled
  IP fast switching on the same interface is enabled
  IP Flow switching is disabled
  IP CEF switching is enabled
  IP CEF Fast switching turbo vector
  IP multicast fast switching is enabled
  IP multicast distributed fast switching is disabled
```

Notice that fast switching and CEF are both enabled. CEF will try to switch the packet first. If CEF cannot switch the packet, it will *punt* the packet to the next best available switching path—fast switching. If fast switching cannot process the packet, the router will process-switch the packet. If all the other switching paths are turned off, the router must process-switch all packets.

To disable all interrupt context switching paths, use the interface command `no ip route-cache`:

```
R1(config-if)# no ip route-cache
R1(config-if)# ^Z
R1# sho ip interface s0/1 | include switching
  IP fast switching is disabled
```

```
  IP fast switching on the same interface is disabled
  IP Flow switching is disabled
  IP CEF switching is disabled
  IP Fast switching turbo vector
  IP multicast fast switching is disabled
  IP multicast distributed fast switching is disabled
```

I've done this for all interfaces on the router. Now let's look at the CPU utilization on the router with the command `show processes cpu history`:

```
R1# sho proc cpu hist


     444444444444444444444411111
     444422222333332222200000555556666644444444444444444444444
100
 90
 80
 70
 60
 50
 40  ************************
 30  ************************
 20  ******************************
 10  ***********************************
   0....5....1....1....2....2....3....3....4....4....5....5....
            0    5    0    5    0    5    0    5    0    5

               CPU% per second (last 60 seconds)
```

The router was running at an average of 4 percent CPU utilization with CEF running. When I disabled all route caching, the router reverted to process switching for all packets. For just 600k of traffic, the router is now using 40 percent of its CPU cycles to forward the packets.

When a router is process switching most of its IP packets, the top process will always be *ip_input*. You can verify this by executing the command `show processes cpu sorted`:

```
R1#sho proc cpu sort
CPU utilization for five seconds: 48%/20%; one minute: 44%; five minutes: 40%
 PID Runtime(ms)   Invoked      uSecs   5Sec   1Min   5Min TTY Process
  31   177614152 132817658       1337 26.60% 24.06% 21.94%   0 IP Input
   3       60540     20255       2988  1.47%  0.22%  0.05%   0 Exec
   4     1496508    210836       7097  0.16%  0.06%  0.05%   0 Check heaps
   1           0         1          0  0.00%  0.00%  0.00%   0 Chunk Manager
   2       12488    328220         38  0.00%  0.00%  0.00%   0 Load Meter
   5         220       332        662  0.00%  0.00%  0.00%   0 Pool Manager
   6           0         2          0  0.00%  0.00%  0.00%   0 Timers
   7           0         8          0  0.00%  0.00%  0.00%   0 Serial Backgroun
   8       12108    327897         36  0.00%  0.00%  0.00%   0 ALARM_TRIGGER_SC
```

This is a good troubleshooting step if you have a router or multilayer switch that is running slowly. If you see *ip_input* taking up the majority of available CPU time (as shown above), you need to ask yourself why the router is process switching. Check your config and look for problems.

Packets that are malformed can cause a router to process switch. I've seen Metropolitan Area Ethernet carrier equipment muck up Ethernet packets just enough that the router tries to process switch them. I've seen the same thing when an ASIC in a switch fails. My point is that external devices or events, not just events internal to the router, can cause the router to fall back on process switching.

Fast Switching

On an interface that is currently process switching, the interface command `ip route-cache` enables fast switching:

```
R1(config)# int s0/1
R1(config-if)# ip route-cache
```

This results in fast switching being turned on while not enabling CEF:

```
R1# sho ip int s0/1 | include swi
  IP fast switching is enabled
  IP fast switching on the same interface is enabled
  IP Flow switching is disabled
  IP CEF switching is disabled
  IP Fast switching turbo vector
  IP multicast fast switching is enabled
  IP multicast distributed fast switching is disabled
```

On an interface that is running CEF, turning off CEF will result in the interface running fast switching:

```
R1(config)# int s0/1
R1(config-if)# no ip route-cache cef
```

You cannot disable fast switching without also disabling CEF, but you can enable fast switching without enabling CEF.

Continuing with the example in the previous section, as soon as I reenable fast switching on the process-switching router, the CPU utilization drops back down to normal levels:

```
R1# sho proc cpu hist

                 222224444444444444444444444444444444444444444444
    4444444444400000777771111188888999998888899999333334444446
100
 90
 80
 70
 60
 50                   *****      ********************          *
```

```
40                  ****************************************
30                  ****************************************
20              ********************************************
10              ********************************************
   0....5....1....1....2....2....3....3....4....4....5....5....
             0    5    0    5    0    5    0    5    0    5

              CPU% per second (last 60 seconds)
```

Cisco Express Forwarding

CEF is enabled by default on all modern Cisco routers, but in the event that you need to enable it on an older router, or if it has been disabled, there are two places that it can be configured. First, using the global command `ip cef`, you can enable CEF on every interface that supports it:

```
R1(config)# ip cef
```

Negating the command disables CEF globally:

```
R1(config)# no ip cef
```

To enable or disable CEF on a single interface, use the interface command `ip route-cache cef`:

```
R1(config)# int s0/1
R1(config-if)# ip route-cache cef
```

Negating the command disables CEF on the interface.

CEF will load-balance packets across equal-cost links. By default, load balancing will be done on a per-destination basis. This means that every packet for a single destination will use the same link. CEF also allows you to configure load balancing on a per-packet basis. This can be beneficial if, for example, there is only one host on the far end of the links, or there is a large server that consumes the majority of the bandwidth for any single link.

Certain protocols, such as VoIP, cannot tolerate per-packet load balancing because packets may arrive out of order. When using such protocols, always ensure that load balancing is performed per-destination, or use a higher-level protocol such as Multilink-PPP.

To change an interface to load-balance using the per-packet method, use the `ip load-sharing per-packet` interface command:

```
R1(config-if)# ip load-sharing per-packet
```

To reconfigure an interface for per-destination load balancing, use the `ip load-sharing per-destination` interface command:

```
R1(config-if)# ip load-sharing per-destination
```

To show the CEF tables in an easy-to-read format, use the `show ip cef` command:

```
R1# sho ip cef
Prefix              Next Hop            Interface
0.0.0.0/32          receive
10.1.1.0/24         attached            Loopback1
10.1.1.0/32         receive
10.1.1.1/32         receive
10.1.1.255/32       receive
10.2.2.0/24         attached            Loopback2
10.2.2.0/32         receive
10.2.2.2/32         receive
10.2.2.255/32       receive
10.3.3.0/24         192.168.1.2         Serial0/1
10.4.4.0/24         192.168.1.2         Serial0/1
10.5.5.0/24         attached            FastEthernet0/1
10.5.5.0/32         receive
10.5.5.1/32         10.5.5.1            FastEthernet0/1
10.5.5.5/32         receive
10.5.5.255/32       receive
10.10.10.0/24       192.168.1.2         Serial0/1
192.168.1.0/24      attached            Serial0/1
192.168.1.0/32      receive
192.168.1.1/32      receive
192.168.1.255/32    receive
224.0.0.0/4         drop
224.0.0.0/24        receive
255.255.255.255/32  receive
```

Usually, you won't need to look into what CEF is doing unless Cisco TAC tells you to. About the only time I've needed this command was when we had a CEF bug that caused packets to be sent out interfaces other than the ones indicated in the routing table.

PART III
Multilayer Switches

This section focuses on multilayer switching. Explanations and examples from the Cisco 6500 and Cisco 3750 Catalyst switches are included.

This section is composed of the following chapters:

Chapter 16, *Multilayer Switches*
Chapter 17, *Cisco 6500 Multilayer Switches*
Chapter 18, *Catalyst 3750 Features*

CHAPTER 16
Multilayer Switches

Switches, in the traditional sense, operate at layer two of the OSI stack. The first multilayer switches were called *layer-3 switches* because they added the ability to route between VLANs. These days, switches can do just about anything a router can do, including protocol testing, and manipulation all the way up to layer seven. Thus, we now refer to switches that operate above layer two as *multilayer switches*.

The core benefit of the multilayer switch is the ability to route between VLANs. This is possible through the addition of virtual interfaces within the switch. These virtual interfaces are tied to VLANs, and are called *switched virtual interfaces* (SVIs).

Figure 16-1 shows an illustration of the principles behind routing within a switch. First, you assign ports to VLANs. Then, you create SVIs, which allow IP addresses to be assigned to the VLANs. The virtual interface becomes a virtual router interface, thus allowing the VLANs to be routed.

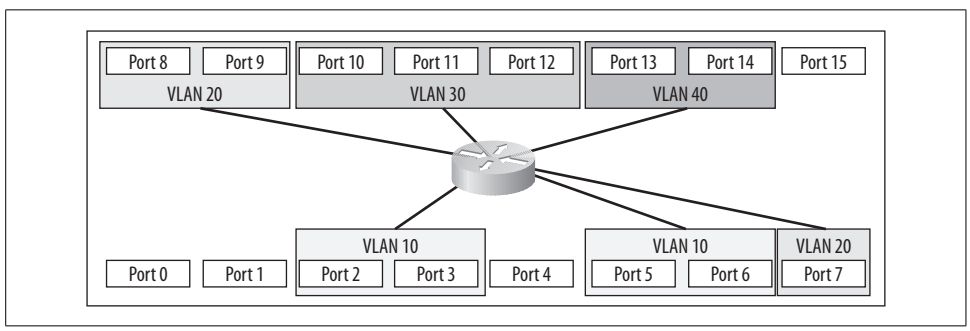


Figure 16-1. VLANs routed from within a switch

Most multilayer switches today do not have visible routers. The router is contained within the circuitry of the switch itself, or in the supervisor (i.e., the CPU) of a modular switch. Older switch designs, like the Cisco 4000 chassis switch, had a layer-3 module that was added to make the switch multilayer-capable. Such modules are no longer needed, since layer-3 functionality is included with most supervisors.

On chassis-based switches with older supervisor modules, the router is a separate device with its own operating system. The router in these switches is a daughter card on the supervisor called the *multilayer switch function card* (MSFC). On these devices, layer-2 operations are controlled by the CatOS operating system, while layer-3 routing operations are controlled by the IOS operating system. This configuration, called *hybrid mode*, can be a bit confusing at first. To some people, having a separate OS for each function makes more sense than combining them. For most people, however, the single-OS model—called *native mode* on the chassis switches—is probably easier.

On a chassis-based switch in hybrid mode, physical interfaces and VLANs must be configured in CatOS. To route between them, you must move to IOS, and create the SVIs for the VLANs you created in CatOS.

Another option for some switches is to change a switch port into a router port—that is, to make a port directly addressable with a layer-3 protocol such as IP. To do this, you must have a switch that is natively IOS, or is running in IOS native mode.

Sometimes, I need to put a layer-3 link between two multilayer switches. Configuring a port, a VLAN, and an SVI involves a lot of steps, especially when you consider that the VLAN will never have any other ports included. In such cases, converting a switch port to a router port is simpler.

To convert a multilayer switch port to a router port, configure the port with the command `no switchport`:

```
Cat-3550(config)# int f0/17
Cat-3550(config-if)# no switchport
```

Once you've done this, you can assign an IP address to the physical interface:

```
Cat-3550(config-if)# int f0/17
Cat-3550(config-if)# ip address 10.10.10.1 255.255.255.0
```

You cannot assign an IP address to a physical interface when it is configured as a switch port (the default state):

```
Cat-3550(config)# int f0/16
Cat-3550(config-if)# ip address 10.10.10.1 255.255.255.0
                             ^
% Invalid input detected at '^' marker.
```

Ethernet ports on routers tend to be expensive, and they don't offer very good port density. The addition of switch modules, which provide a few interfaces, has improved their port densities, but nothing beats the flexibility of a multilayer switch when it comes to Ethernet.

Configuring SVIs

Switch virtual interfaces are configured differently depending on the switch platform and operating systems installed.

Native Mode (4500, 6500, 3550, 3750)

Here is the output of the command `show ip interface brief` from a 3550:

```
Cat-3550# sho ip int brief
Interface              IP-Address      OK? Method Status                Protocol
Vlan1                  192.168.134.22  YES DHCP   up                    up
FastEthernet0/1        unassigned      YES unset  down                  down
FastEthernet0/2        unassigned      YES unset  down                  down
FastEthernet0/3        unassigned      YES unset  down                  down
FastEthernet0/4        unassigned      YES unset  down                  down

[-Text Removed-]

FastEthernet0/23       unassigned      YES unset  up                    up
FastEthernet0/24       unassigned      YES unset  down                  down
GigabitEthernet0/1     unassigned      YES unset  down                  down
GigabitEthernet0/2     unassigned      YES unset  down                  down
```

The first interface is a switched virtual interface for VLAN 1. This SVI cannot be removed on a 3550, as it is used for management. Looking at the VLAN table with the `show vlan` command, we can see there are five VLANs configured in addition to the default VLAN 1:

```
Cat-3550# sho vlan

VLAN Name                             Status    Ports
---- -------------------------------- --------- -------------------------------
1    default                          active    Fa0/1, Fa0/2, Fa0/3, Fa0/4
                                                Fa0/5, Fa0/6, Fa0/7, Fa0/8
                                                Fa0/9, Fa0/10, Fa0/11, Fa0/16
                                                Fa0/17, Fa0/18, Fa0/19, Fa0/21
                                                Fa0/22, Fa0/23, Fa0/24, Gi0/1
                                                Gi0/2
2    VLAN0002                         active    Fa0/12
3    VLAN0003                         active
4    VLAN0004                         active
10   VLAN0010                         active
100  VLAN0100                         active
1002 fddi-default                     act/unsup
1003 token-ring-default               act/unsup
1004 fddinet-default                  act/unsup
1005 trnet-default                    act/unsup

[-Text Removed-]
```

These VLANs are strictly layer-2, in that the switch will not route between them. For the switch to be able to access these VLANs at higher layers, we need to create an SVI for each one.

The only step required to create an SVI is to define it. Simply by entering the global configuration command `interface vlan` *`vlan#`*, you will create the SVI. The *`vlan#`* does not need to match an existing VLAN. For example, defining an SVI for VLAN

200, which does not exist on our switch, will still result in the SVI being created. We can even assign an IP address to the interface, and enable it with the `no shutdown` command:

```
Cat-3550(config)# interface Vlan200
1w5d: %LINEPROTO-5-UPDOWN: Line protocol on Interface Vlan200, changed state to down
Cat-3550(config-if)# ip address 10.200.0.1 255.255.255.0
Cat-3550(config-if)# no shut
```

The interface will initially be down/down because there is no VLAN at layer two to support it. The hardware type is *EtherSVI*, indicating that this is a logical SVI:

```
Cat-3550# sho int vlan 200
Vlan200 is down, line protocol is down
  Hardware is EtherSVI, address is 000f.8f5c.5a00 (bia 000f.8f5c.5a00)
  Internet address is 10.200.0.1/24
  MTU 1500 bytes, BW 1000000 Kbit, DLY 10 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation ARPA, loopback not set
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input never, output never, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: fifo
  Output queue: 0/40 (size/max)
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     0 packets input, 0 bytes, 0 no buffer
     Received 0 broadcasts (0 IP multicast)
     0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored
     0 packets output, 0 bytes, 0 underruns
     0 output errors, 0 interface resets
     0 output buffer failures, 0 output buffers swapped out
```

Once we add VLAN 200 to the switch, the interface comes up:

```
Cat-3550(config)# vlan 200
Cat-3550(config)#
1w5d: %LINEPROTO-5-UPDOWN: Line protocol on Interface Vlan200, changed state to up

Cat-3550# sho ip int brie | include Vlan200
Vlan200                10.200.0.1      YES manual up                    up
```

We have not assigned any ports to this VLAN, but the SVI for the VLAN is up and operating at layer three. We can even ping the new interface:

```
Cat-3550# ping 10.200.0.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.200.0.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms
```

If we were so inclined, we could also add routing protocols, and do anything else we would normally do with an interface operating at layer three.

What's the point of having an SVI without any physical ports assigned to it? One example might be to create a management network other than VLAN 1 for all your devices. You wouldn't need any physical ports assigned to the VLAN, except for a trunk port to your other switches. This way, you can keep your management traffic on a separate VLAN from your production traffic.

Hybrid Mode (4500, 6500)

On a switch supporting hybrid IOS mode, IOS is not an integral part of the switch's function. CatOS is used for switching, which is integral to the operation of layer two. IOS is used only to manage the MSFC.

Creating a VLAN is done in CatOS, but creating an SVI for the VLAN is done in IOS. Naming the VLAN is done in CatOS, but adding a description to the SVI is done in IOS. Anything that is related to layer-2 functionality must be configured in CatOS. Layer-3 and above functions must be configured in IOS.

On an IOS-only switch, there is always a VLAN 1 virtual interface. This is not the case in hybrid-mode switches because VLAN 1 does not, by default, require layer-3 functionality.

I've created two VLANs on a 6509 running in hybrid mode here:

```
CatOS-6509: (enable) sho vlan
VLAN Name                             Status    IfIndex Mod/Ports, Vlans
---- -------------------------------- --------- ------- ------------------------
1    default                          active    9       1/1-2
                                                        2/1-2
                                                        3/5-48
                                                        4/1-48
                                                        5/1-48
10   Lab-VLAN                         active    161
20   VLAN0020                         active    162     3/1-4
1002 fddi-default                     active    10
1003 token-ring-default               active    13
1004 fddinet-default                  active    11
1005 trnet-default                    active    12
```

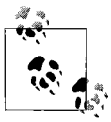
The first, VLAN 10, I have named `Lab-VLAN`. The second, VLAN 20, has the default name of `VLAN0020`. To configure these VLANs in IOS, we must first connect to the MSFC using the `session` command. This command must be followed by the number of the module to which we'd like to connect. The number can be determined with the `show module` command:

```
CatOS-6509: (enable) sho mod
Mod Slot Ports Module-Type               Model               Sub Status
--- ---- ----- ------------------------- ------------------- --- --------
1   1    2     1000BaseX Supervisor      WS-X6K-SUP2-2GE     yes ok
15  1    1     Multilayer Switch Feature WS-F6K-MSFC2        no  ok
2   2    2     1000BaseX Supervisor      WS-X6K-SUP2-2GE     yes standby
16  2    1     Multilayer Switch Feature WS-F6K-MSFC2        no  ok
3   3    48    10/100BaseTX Ethernet     WS-X6348-RJ-45      no  ok
```

```
4   4    48    10/100BaseTX Ethernet     WS-X6348-RJ-45      no  ok
5   5    48    10/100BaseTX Ethernet     WS-X6348-RJ-45      no  ok
```

The first MSFC is reported as being in slot 15. This is normal when the supervisor is in slot 1, as the MSFC is a daughter card on the supervisor. The switch assigns an internal slot number to the MSFC. We can now connect to the MSFC:

```
CatOS-6509: (enable) session 15
Trying Router-15...
Connected to Router-15.
Escape character is '^]'.

MSFC-6509> en
MSFC-6509#
```



Another way to get to the MSFC is with the `switch console` command.

Now that we're in IOS, let's see what the MSFC thinks about the two VLANs:

```
MSFC-6509# sho ip int brief
Interface                  IP-Address      OK? Method Status
Protocol
```

There are no SVIs active on the MSFC—not even VLAN 1. Let's add an SVI for VLAN 20 and see what happens:

```
MSFC-6509# conf t
Enter configuration commands, one per line. End with CNTL/Z.
MSFC-6509(config)# int vlan 20
MSFC-6509(config-if)# ip address 10.20.20.1 255.255.255.0
MSFC-6509(config-if)# no shut
MSFC-6509(config-if)# ^Z
MSFC-6509#
17w2d: %LINK-3-UPDOWN: Interface Vlan20, changed state to down
17w2d: %LINEPROTO-5-UPDOWN: Line protocol on Interface Vlan20, changed state to down
MSFC-6509#
MSFC-6509# sho ip int brief
Interface                  IP-Address      OK? Method Status            Protocol
Vlan20                     10.20.20.1      YES manual down              down
```

The SVI is now there, but it won't come up. The SVI will not come up unless there is an active port in the VLAN in layer two. I often forget this fact and, after adding the SVIs, go off to create my VLANs only to find that none of them will come up. To illustrate the point, I'll assign an IP address to the CatOS management interface SC0, and place it in VLAN 20. This will put an active device in the VLAN:

```
CatOS-6509: (enable) set int sc0 20 10.20.20.20 255.255.255.0
Interface sc0 vlan set, IP address and netmask set.
```

Now, with something active in VLAN 20, the VLAN 20 SVI comes up in the MSFC:

```
MSFC-6509# sho ip int brief
Interface                  IP-Address      OK? Method Status           Protocol
Vlan20                     10.20.20.1      YES manual up               up
```

Multilayer Switch Models

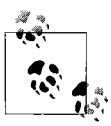
Cisco offers a variety of multilayer switch models. The line has become fuzzy, though, because routers like the 7600 series can now take some of the 6500-series switching modules. The 3800 series of routers also supports a small switching module capable of supporting multiple Ethernet interfaces.

Still, there is no magic all-in-one device. You must choose either a switch with limited routing capabilities, or a router with limited switching capabilities. The difference is primarily in how the system internals are designed, and what modules are supported. A router is designed differently from a switch, though this is also becoming less true if you consider devices like the Gigabit Switch Router (GSR). A router is generally more WAN-centric, whereas a switch is usually more LAN-centric. There are no modules that allow T1 WAN connectivity for the 6500 switches. While you can put 6500 Ethernet modules in a 7500 router, the backplane capacity is not as high in the router as it is in the switch.

Multilayer switches are divided by chassis type. On the lower end are the single rack unit (1-RU) models that are designed for wiring closets and small installations. Some of these switches can be stacked in a number of ways, depending on the model. Some 1-RU models have increased backplane speeds, and even support 10 Gbps uplinks.

Next in the hierarchy are the small chassis-based switches. This group is composed of the models in the 4500 range. These switches are designed for larger wiring closets, or even small core functions. They can support multiple power supplies and supervisors, and are designed for high-availability installations.

The 6500 switches occupy the high end of the spectrum. Available in multiple chassis sizes, these switches are very popular due to their expandability, flexibility, and performance.



For more information on switch types, refer back to Chapter 2. Cisco 6500-series switches are discussed in more detail in Chapter 17, and the 3750 is the focus of Chapter 18.

CHAPTER 17
Cisco 6500 Multilayer Switches

The Cisco 6500 is possibly the most widely deployed enterprise-class switch in the world. The balance of expandability, power, capabilities, and port density offered by this switch is hard to beat. The 6500 series comes in sizes ranging from 3 slots up to 13 slots. There are even versions that are Network Equipment Building System (NEBS) compliant for use in telecom infrastructures that must meet these stringent specifications.

The 6500 architecture has been around for some time, but because it was developed with expandability in mind, the switch that originally supported only 32 Gbps on the backplane now routinely supports 720 Gbps, with 1,440 Gbps on the horizon!

The versatility of the 6500 platform has been a prime driver for this series' placement in a wide variety of positions in an even wider array of solutions. 6509 switches are often seen at the center of enterprise networks, at the access layer of large companies, as the core of e-commerce web sites, and even as the telecom gateways for large VoIP implementations.

Likewise, the flexibility of the 6500 platform has made it prevalent even in smaller companies. The 6500 series includes Firewall Services Modules (FWSMs), Content Switching Modules (CSMs), and Network Analysis Modules (NAMs). The entire network infrastructure, as well as all security, load-balancing, and monitoring hardware, can be contained in a single chassis.

With the addition of a multilayer-switch feature card (MSFC), the 6500 becomes a multilayer switch. With the addition of IOS running natively, the 6500 becomes a router with the potential for more than 300 Ethernet interfaces, while retaining the functionality and speed of a switch.

When running in native IOS mode, a 6500 operates very similarly to the smaller 3550 and 3750 switches, but with more flexibility and power.

Figure 17-1 shows how a typical multitiered e-commerce web site can benefit from using a 6509 chassis-based solution rather than a series of individual components.

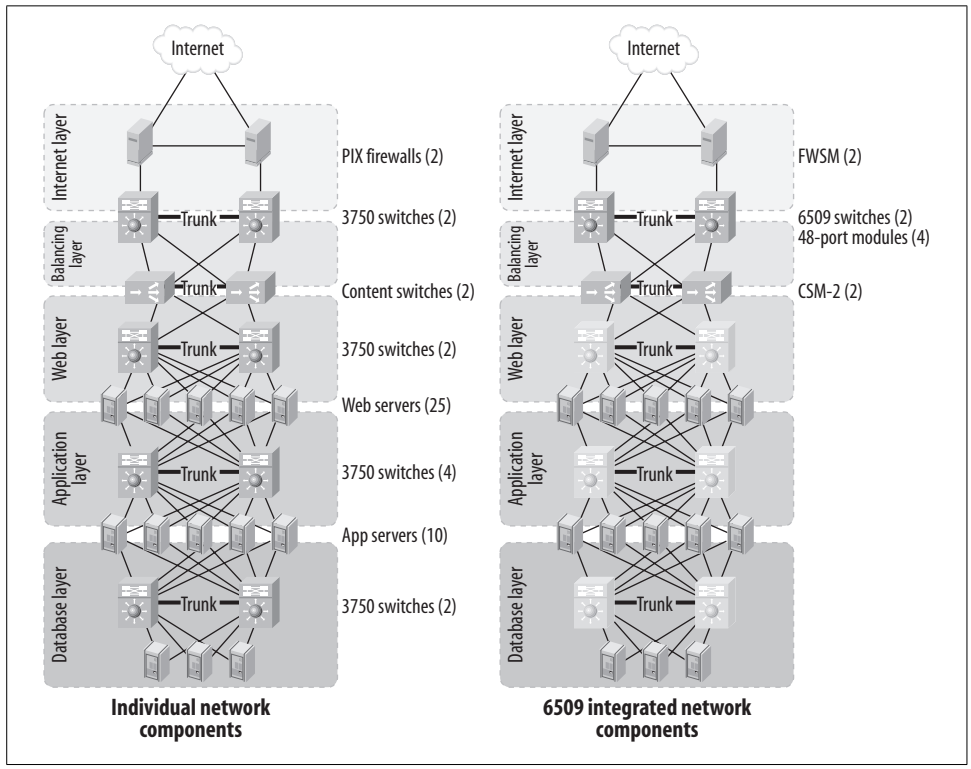


Figure 17-1. Individual versus integrated network components

First, because all of the layers can be consolidated into a single device using VLANs, many of the switches are no longer needed, and are replaced with Ethernet modules in the 6509 chassis.

Second, because some layers do not need a lot of ports, a better utilization of ports can be realized. A module is not dedicated to a specific layer, but can be divided in any way needed. If we allocated a physical switch to each layer, there would be many unused ports on each switch, especially at the upper layers.

Another benefit is that because the components are included in a single chassis, there are fewer maintenance contracts to manage (though modules like the FWSM and CSM require their own contracts). Additionally, because all of the devices are now centralized, there only needs to be one pair of power outlets for each switch. The tradeoff here is that the power will no doubt be 220V 20A, or more.

Some of the new power supplies for the 6500e chassis require multiple power feeds per supply. The 6000-watt AC power supply requires two power outlets per supply. The 8700-watt AC power supply requires three outlets per supply, resulting in a total of six outlets per chassis!

The main advantages are supportability and speed. Each of the modules is available through the switch itself in addition to its own accessibility, and in the case of the CSM, the configuration is a part of the Cisco IOS for the switch itself (assuming native IOS). Each of the modules is hot-swappable, with the only limitation being that some modules must be shut down before being removed. Also, because the modules communicate with each other over the backplane, they offer substantial increases in speed over their standalone counterparts. The FWSM, for example, is capable of more than 4 Gbps of throughput, while the fastest standalone PIX firewall at the time of this writing is capable of only 1.8 Gbps. While this processing difference has a lot to do with the design of the FWSM and PIX firewalls, the fact remains that standalone devices must communicate through Gigabit Ethernet interfaces, while service modules communicate directly over the backplane of the switch.

6500 switches are designed to be highly redundant. They support dual power supplies and dual supervisors. The supervisor MSFCs can run as individual routers or in single-router mode. The power supplies can be configured in a failover mode or a combined mode to allow more power for hungry modules.

Combine this switch's features, scalability, and resilience with the additional fault-tolerance of a high-availability network design, and you've got a world-class architecture at your fingertips.

Architecture

The 6500-series switches are an evolution of the 6000 series. The 6000-series switch contained only a 32-Gbps backplane bus, whereas the 6500 series contains an additional bus called the *fabric bus* or *crossbar switching bus*. This bus allows backplane speeds to be boosted up to 720 Gbps and beyond.

The addition of the crossbar switching fabric in the 6500 series also provides an amazing amount of flexibility in the new chassis. Legacy modules from the 6000 chassis could still be used, as the 32 Gbps bus in the 6500 is identical to one found in the 6000 series. However, with the addition of a required *Switch Fabric Module* (SFM), newer fabric-enabled modules are able to take advantage of the new bus.

The SFM is essentially a 16-port switch that connects each of the fabric-enabled modules via the fabric bus. Because the SFM is a switch unto itself, modules communicate concurrently, much the same way multiple computers can communicate on a

switched Ethernet network. By contrast, the 32 Gbps bus operated in such a way that all modules received all packets, regardless of the destination module (similar to the way computers communicate on an Ethernet network connected with a hub).

Because it controlled the crossbar fabric bus, the SFM could only reside in certain slots. One of the major downsides to this design was that a highly redundant installation required two slots for the supervisors and an additional two slots for the SFMs. In a nine-slot chassis, this left only five slots for line cards or service modules.

The Supervisor-720 solved the slot-density problem by incorporating the SFM into the supervisor module. Now, a highly resilient installation requires only two slots for supervisors. However, note that because the Supervisor-720 includes the SFM's functionality, it must reside in the SFM's slots. For example, on a redundant 6509, the Supervisor-2 modules reside in slots 1 and 2, while the SFMs reside in slots 5 and 6. Supervisor-720 modules must reside in slots 5 and 6, which frees up slots 1 and 2 for line cards or service modules.

Buses

The 6500 series switch backplane is composed of the following four buses:

D bus
: The data bus, or D bus, is used by the EARL chipset to transfer data between modules. The speed of the D bus is 32 Gbps. The D bus is shared much like a traditional Ethernet network, in that all modules receive all frames that are placed on the bus. When frames need to be forwarded from a port on one module to a port on another module, assuming the crossbar fabric bus is not in use or available to the modules, they will traverse this bus.

R bus
: The results bus, or R bus, is used to handle communication between the modules and the switching logic on the supervisors. The speed of the R bus is 4 Gbps.

C bus
: The control bus, or C bus, is also sometimes referred to as the *Ethernet Out-of-Band Channel* (EOBC). The C bus is used for communication between the line cards and the network management processors on the supervisors. The C bus is actually a 100 Mbps half-duplex network. When line-control code is downloaded to the line cards, it is done on this bus.

Crossbar fabric bus
: *Crossbar* is a type of switching technology where each node is connected to every other node by means of intersecting paths. An alternative switching fabric is the fully interconnected model, where each port is directly connected to every other port.

Figure 17-2 shows visual representations of such switching fabrics. The term *fabric* is used to describe the mesh of connections in such designs, as logically, the connections resemble interwoven strands of fabric.

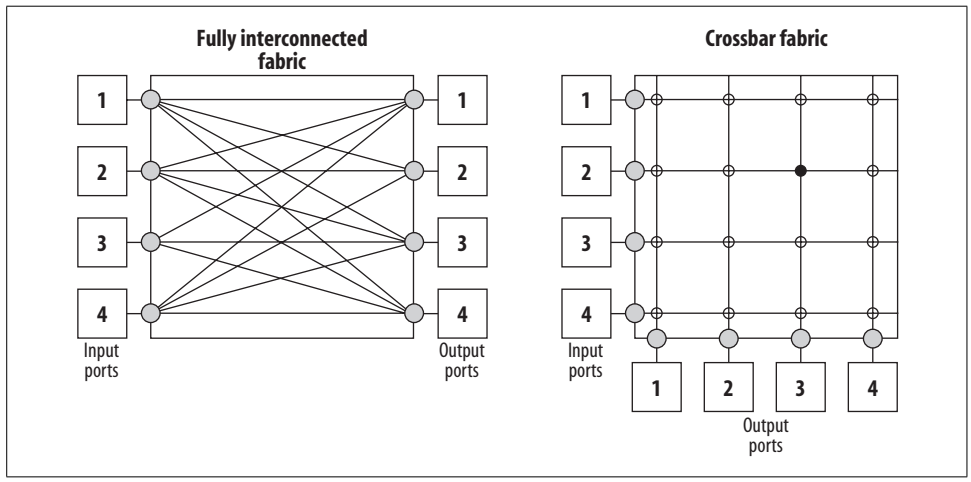


Figure 17-2. Switch fabric examples

The crossbar fabric shown in Figure 17-2 shows one black intersection. This is an active connection, whereas the others are inactive connections. The active connection shown here indicates that port 2 is in communication with port 3.

The crossbar fabric bus, in combination with a Supervisor-2 and a Switch Fabric Module, is capable of 256 Gbps and 30 million packets per second (Mpps). With the addition of a distributed forwarding card, this combination is capable of 210 Mpps. With a Supervisor-720 module, the crossbar fabric supports up to 720 Gbps. When using distributed Cisco Express Forwarding (dCEF) interface modules, a Sup-720-equipped 6500 is capable of 400 Mpps.

The SFM is what provides the actual switch fabric between all the fabric-enabled modules (recall that the SFM's functionality is included in the Supervisor-720, so in this case, a separate module is not required). The module is actually a switch in and of itself that uses the backplane fabric bus as a communication channel between the modules. It is for this reason that the speed of a 6500 can change with a new supervisor module.

Figure 17-3 shows a visual representation of the backplane in a 6509 chassis. Looking at the chassis from the front, you would see each slot's connectors as shown. There are two backplane circuit boards separated by a vertical space. To the left of the space (at the back of the chassis) are the power connectors and the crossbar fabric bus. To the right of the space are the D, R, and C buses. A 6000 chassis would look the same to the right of the space, but have no crossbar fabric bus to the left.

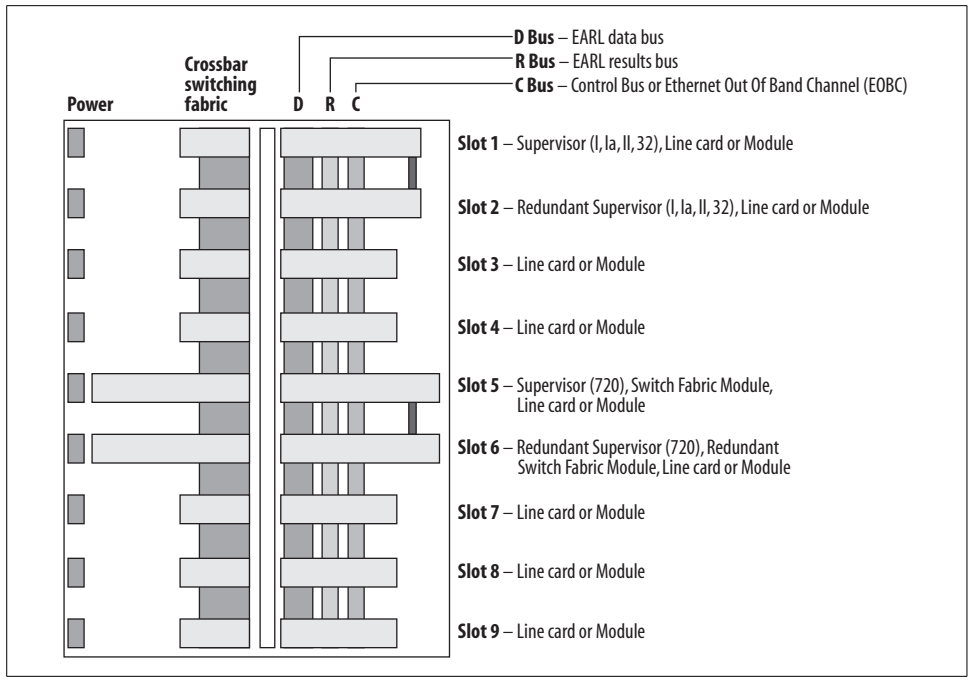


Figure 17-3. Cisco 6509 backplanes

Certain slots are capable of taking specific modules, while other slots are not. The breakdown of slots in a 6509 is as follows:

Slot 1
: Slot 1 is capable of housing supervisor modules 1, 1A, and 2; line cards; or service modules. If there is only one Sup-1, Sup-1A, or Sup-2 in the chassis, it should reside in slot 1.

Slot 2
: Slot 2 is capable of housing supervisor modules 1, 1A, and 2; line cards; or service modules. This slot is used for the redundant supervisor module if a failover pair is installed. Though a single supervisor can be installed in this slot, the first slot is generally used for single-supervisor installations.

Slot 3
: Slot 3 is capable of housing any line card or module, with the exception of supervisors or SFMs.

Slot 4
: Slot 4 is capable of housing any line card or module, with the exception of supervisors or SFMs.

Slot 5
: Slot 5 is capable of housing an SFM or a supervisor incorporating an SFM, such as the Supervisor-720. This slot may also support any line card or module, with the exception of supervisors that would normally be placed in slot 1 or 2.

Slot 6
: Slot 6 is capable of housing an SFM or a supervisor incorporating an SFM, such as the Supervisor-720. This slot may also support any line card or module, with the exception of supervisors that would normally be placed in slot 1 or 2. This slot is used for the redundant fabric module or supervisor module if a failover pair is installed. Though a single fabric module or supervisor can be installed in this slot, slot 5 is generally used for single-supervisor/SFM installations.

Slot 7
: Slot 7 is capable of housing any line card or module, with the exception of supervisors or SFMs.

Slot 8
: Slot 8 is capable of housing any line card or module, with the exception of supervisors or SFMs.

Slot 9
: Slot 9 is capable of housing any line card or module, with the exception of supervisors or SFMs.

The 6506-chassis slots are allocated the same way, with the obvious difference that there are no slots 7, 8, and 9; the 6513-chassis slots are allocated the same way, but with the addition of slots 10–13, which can house any line card or service module apart from supervisors or SFMs. These last four slots in the 6513 chassis cannot support certain fabric-only blades. Consult the Cisco documentation for specifics when ordering cards for this chassis.

Enhanced Chassis

A series of enhanced 6500 chassis, identified by an *e* at the end of the chassis part number, are also available. An example of an enhanced chassis is the 6500e. The enhanced chassis are designed to allow more power to be drawn to the line cards. The advent of Power over Ethernet (PoE) line cards for Voice-over-IP applications was one of the key drivers for this evolution. The enhanced chassis use high-speed fans to cool these power-hungry modules.

The e-series chassis also provide a redesigned backplane that allows for a total of 80 Gbps of throughput per slot. This represents a theoretical doubling of the capacity of the standard 6500 chassis (40 Gbps of throughput per slot), though at the time of this writing, there are no line cards or supervisors that support this speed. The new

architecture will allow eight 10 Gbps ports per blade with no oversubscription. Cisco now only produces the enhanced chassis models, though the standard chassis models are still available though the secondhand market.

Supervisors

Chassis-based switches do not have processors built into them like smaller switches do. Instead, the processor is on a module, which allows the hardware to be swapped and upgraded with ease. The processor for a Cisco chassis-based switch is called a supervisor. Supervisors are also commonly referred to as *sups* (pronounced like "soups").

Over the years, different supervisor models have been introduced to offer greater speed and versatility. Increased functionality has also been made available via add-on daughter cards, which are built into the later supervisor models.

MSFC

Supervisors offer layer-2 processing capabilities, while an add-on daughter card—called a multilayer switch feature card—supports layer-3 and higher functionality. Supervisor models 1 and 2 offer the MSFC as an add-on, while later models include the MSFC as an integral part of the supervisor.

When running hybrid-mode IOS on the 6500 chassis, the MSFC is considered a separate device regardless of the supervisor model. In CiscoView, the MSFC appears as a small router icon to the left of the supervisor, where the fan tray resides. Figure 17-4 shows the CiscoView representation of a Supervisor-720 with the MSFC on the left.

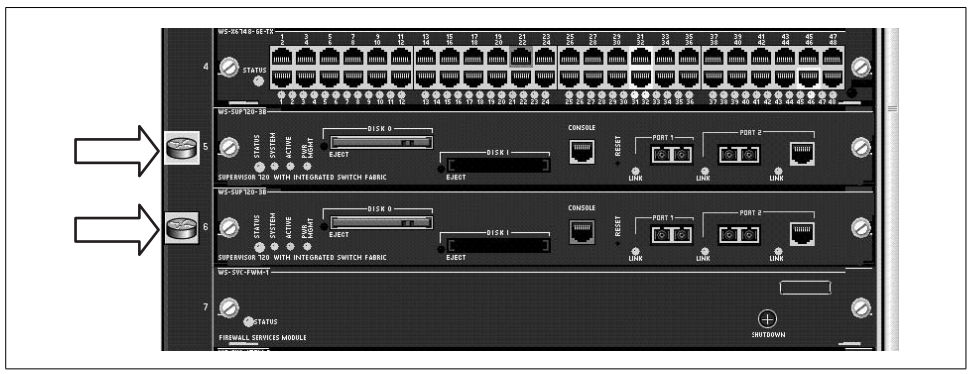


Figure 17-4. CiscoView representation of Supervisor-720 and MSFC

Different versions of the MSFC are referenced as MSFC1, MSFC2, and MSFC3. The MSFC2 is paired with the Supervisor-2, while the MSFC3 is part of the Supervisor-720.

PFC

The *policy feature card* (PFC) is a daughter card that supports Quality of Service functions in hardware, drastically improving performance where QoS is needed. No direct configuration of the PFC is required. The three generations of the PFC are named PFC1, PFC2, and PFC3. The PFC2 is paired with the Supervisor-2, and the PFC3 is an integral part of the Supervisor-720.

Models

Supervisor models most commonly seen today include:

Supervisor-1A
: This is the slowest and oldest of the supervisor modules, capable of supporting 32 Gbps and 15 Mpps. (The Supervisor-1A replaced the original Supervisor Engine, also called the Supervisor-1.) The Supervisor-1A is end-of-life, but may still be seen in older installations. When coupled with a PFC and an MSFC, the Sup-1A is capable of layer 2–4 forwarding, as well as enhanced security and QoS. The Supervisor-1A was an excellent solution for wiring closets or networks that did not require the throughput and speed of the Supervisor-2.

Supervisor-2
: This model was the standard in backbone and e-commerce web site switching until the Supervisor-720 was released. The Supervisor-2 is capable of 30 Mpps and 256 Gbps when paired with a Switch Fabric Module. When coupled with a PFC2 and an MSFC2, the Supervisor-2's forwarding capability increases to 210 Mpps, and it is capable of layer 2–4 forwarding as well as enhanced security and QoS.

Supervisor-32
: This is the latest replacement for the Supervisor-1A. Any existing Supervisor-1As should be replaced with Supervisor-32s. This model differs from the other supervisors in that it includes eight 1-Gbps small-form-factor GBIC ports, or two 10 Gbps Ethernet XENPAK-based ports. Other supervisors will offer, at most, two 1 Gbps ports.

Supervisor-720
: This model represents a major upgrade to the aging Supervisor-2 architecture. Capable of 400 Mpps and a blazing 720 Gbps, this supervisor is designed for bandwidth-hungry installations and also for critical core implementations. The

Supervisor-720 includes the PFC3 and MSFC3 as well as new accelerated Cisco Express Forwarding and distributed Cisco Express Forwarding capabilities. Fabric-only modules are capable of 40 Gbps throughput when coupled with a Sup-720.

Modules

Modules for the 6500 chassis are designed to support one or both of the chassis backplanes. A module that does not support the crossbar fabric is considered *nonfabric-enabled*. One that supports the 32 Gbps D bus and the fabric bus is considered to be *fabric-enabled*. A module that uses only the fabric bus and has no connection to the D bus is considered to be *fabric-only*.

Supervisors do not have the same connectors for insertion into the backplane as SFMs. Supervisor-720 modules that include the SFM's functionality have large connectors that can mate only with the receptacles in slots 5 and 6. The connectors for a Sup-720 are shown in Figure 17-5.

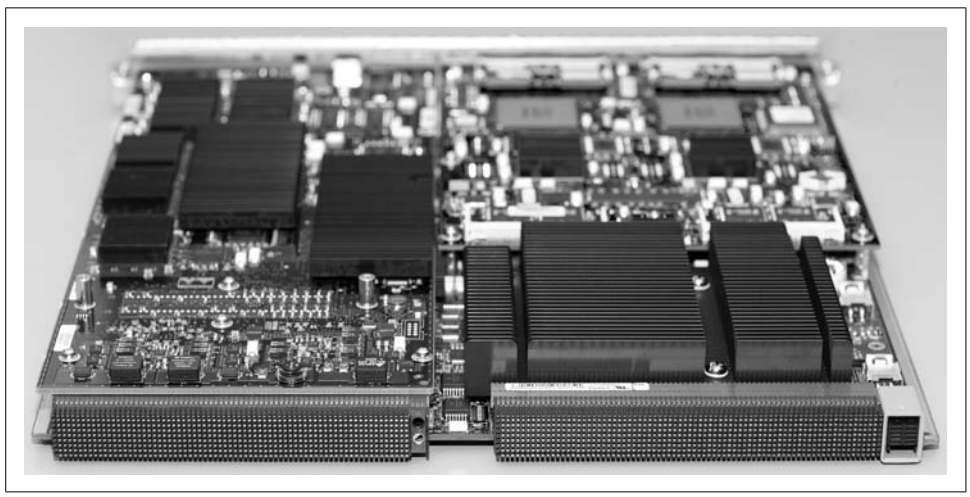


Figure 17-5. Supervisor-720 connectors (photo by Gary A. Donahue)

Nonfabric-enabled modules only have connectors on one side, for connection to the D bus. Modules from a 6000 chassis are nonfabric-enabled, since there is no crossbar fabric bus in the 6000 series.

A fabric-enabled module has two connectors on the back of the blade: one for the D bus, and one for the crossbar fabric bus.

An example of such a blade (in this case, a 16-port gigabit fiber module) is shown in Figure 17-6.

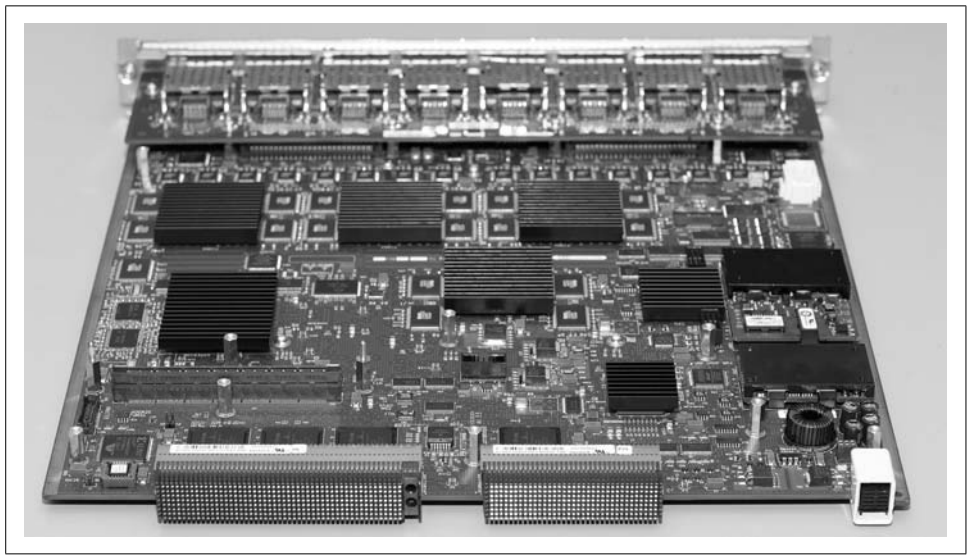


Figure 17-6. Fabric-enabled blade connectors (photo by Gary A. Donahue)

Modules that are fabric-only have a single connector on the fabric side, with no connector on the D bus side.

Be very careful when inserting modules into chassis-based switches such as the 6500 series. Many of the components on the modules are quite tall. As a result, they can impact the chassis, and be damaged by improper or forced insertion. Supervisor modules and service modules such as CSMs and the FWSMs are particularly susceptible to this problem, due to the large quantity of components incorporated into these devices. Some of these modules retail for more than $50,000, and you probably don't want to be the one who has to admit to breaking them.

Module interaction

When fabric-enabled or fabric-only blades are placed in a chassis with nonfabric-enabled blades, the supervisor must make compromises to facilitate the interaction between the different buses. Specifically, if there is a nonfabric-enabled module in the chassis, the Supervisor-720 will not be able to run at 720 Gbps speeds.

Here is an example of a 6509 that is filled with fabric-only 10/100/1000-Mb model 6748 Ethernet modules and two Sup-720 supervisors:

```
6509-1# sho mod
Mod Ports Card Type                              Model              Serial No.
```

```
--- ----- ------------------------------------ ------------------ -----------
  1   48  CEF720 48 port 10/100/1000mb Ethernet  WS-X6748-GE-TX     SAL05340V5X
  2   48  CEF720 48 port 10/100/1000mb Ethernet  WS-X6748-GE-TX     SAL09347ZXK
  3   48  CEF720 48 port 10/100/1000mb Ethernet  WS-X6748-GE-TX     SAL05380V5Y
  4   48  CEF720 48 port 10/100/1000mb Ethernet  WS-X6748-GE-TX     SAL092644CJ
  5    2  Supervisor Engine 720 (Active)         WS-SUP720-3B       SAL05304AZV
  6    2  Supervisor Engine 720 (Hot)            WS-SUP720-3B       SAL09295RWB
  7   48  CEF720 48 port 10/100/1000mb Ethernet  WS-X6748-GE-TX     SAL05340Z9H
  8   48  CEF720 48 port 10/100/1000mb Ethernet  WS-X6748-GE-TX     SAL0938145M
  9   48  CEF720 48 port 10/100/1000mb Ethernet  WS-X6748-GE-TX     SAL053415EC
```

The command `show fabric switching-mode` demonstrates how each of the modules is communicating with the system. The output shows that all of the modules are using the crossbar switching bus, and the Sup-720 is operating in dCEF mode, which allows forwarding at up to 720 Gbps:

```
6509-1# sho fabric switching-mode
Fabric module is not required for system to operate
Modules are allowed to operate in bus mode
Truncated mode allowed, due to presence of aCEF720  module

Module Slot     Switching Mode
    1                Crossbar
    2                Crossbar
    3                Crossbar
    4                Crossbar
    5                    dCEF
    6                Crossbar
    7                Crossbar
    8                Crossbar
    9                Crossbar
```

Each of the fabric-only modules has two 20 Gbps connections to the crossbar fabric bus, as we can see with the `show fabric status` or `show fabric utilization` command. Notice that the supervisors each only have one 20 Gbps connection to the fabric bus:

```
6509-1# sho fabric util
 slot    channel      speed    Ingress %     Egress %
    1          0        20G            1            0
    1          1        20G            0            2
    2          0        20G            1            0
    2          1        20G            0            0
    3          0        20G            1            0
    3          1        20G            0            0
    4          0        20G            0            0
    4          1        20G            0            0
    5          0        20G            0            0
    6          0        20G            0            0
    7          0        20G            0            0
    7          1        20G            0            0
    8          0        20G            0            0
    8          1        20G            0            0
    9          0        20G            0            0
    9          1        20G            0            0
```

For comparison, here is a 6509 that is operating with two Supervisor-720s, one fabric-only module, a couple of fabric-enabled modules, and one nonfabric-enabled module:

```
6509-2# sho mod
Mod Ports Card Type                              Model              Serial No.
--- ----- -------------------------------------- ------------------ -----------
  1   48  CEF720 48 port 10/100/1000mb Ethernet  WS-X6748-GE-TX     SAL04654F2K
  4    8  Network Analysis Module                WS-SVC-NAM-2       SAD093002B6
  5    2  Supervisor Engine 720 (Active)         WS-SUP720-3B       SAL0485498A
  6    2  Supervisor Engine 720 (Hot)            WS-SUP720-3B       SAL09358NE6
  7    6  Firewall Module                        WS-SVC-FWM-1       SAD042408DF
  8    4  CSM with SSL                           WS-X6066-SLB-S-K9  SAD094107YN
  9    8  Intrusion Detection System             WS-SVC-IDSM-2      SAD048102CG
```

The module in slot 1 is the same as the Ethernet modules in the previous example. This module is fabric-only. Modules 4, 7, and 9 are all fabric-enabled, while module 8 is nonfabric-enabled. The output from the `show fabric switching-mode` command reveals that the single nonfabric-enabled blade has caused the supervisor to revert to a slower operating mode:

```
6509-2# sho fabric switching-mode
Global switching mode is Truncated
dCEF mode is not enforced for system to operate
Fabric module is not required for system to operate
Modules are allowed to operate in bus mode
Truncated mode is allowed, due to presence of aCEF720, Standby supervisor module

Module Slot     Switching Mode
    1                Crossbar
    4                Crossbar
    5                     Bus
    6                Crossbar
    7                Crossbar
    8                     Bus
    9                Crossbar
```

In this case, the module in question is a CSM, and is one of the more expensive modules available. Remember that cost does not equate to speed. The CSM is an excellent device, and I highly recommend it for situations where load balancing is a necessity, but in the case of extremely high throughput requirements, the service module may become a bottleneck. In the case of web site architecture, it would be extremely rare for more than 32 Gbps to be flowing through the frontend. Such throughput would be possible in the case of balancing large application server farms or databases on the backend.

Using the `show fabric status` command on this switch indicates that not all fabric-enabled modules are created equal. The fabric-only module in slot 1 has two 20 Gbps channels to the fiber bus. The NAM in slot 4 is fabric-enabled, but only connects with one 8 Gbps channel, as do the FWSM and IDS modules in slots 7 and 9:

```
6509-2# sho fabric status
 slot    channel      speed    module              fabric
                               status              status
    1          0        20G        OK                  OK
    1          1        20G        OK                  OK
    4          0         8G        OK                  OK
    5          0        20G        OK                  OK
    6          0        20G        OK                  OK
    7          0         8G        OK                  OK
    9          0         8G        OK                  OK
```

The lesson here is that it's important to understand how your modules interoperate. Even though a module may be a "fabric blade," it may not perform the same way as another fabric-enabled module. Knowing how the different modules operate can help you understand your current setup and design future solutions.

Module types

Modules are generally divided into line cards and service modules. A line card offers connectivity, such as copper or fiber Ethernet. Service modules offer functionality. Examples of service modules include Firewall Services Modules and Content Switch Modules.

Service modules dramatically enhance the usefulness of the 6500 switch. In one chassis, you can have a complete web server architecture, including Ethernet ports, DS3 Internet feeds, firewalls, IDSs, and load balancing. All devices will be configurable from the single chassis, and all will be powered from the same source. For redundancy, two identically configured chassis could be deployed with complete failover functionality that would provide no single point of failure.

Ethernet modules. Ethernet modules are available in many flavors and speeds. Some offer simple connectivity, while others offer extreme speed with 40 Gbps connections to the crossbar fabric bus.

Connectivity options for Ethernet modules include RJ-45, GBIC, small-form-factor GBIC, and Amphenol connectors for direct connection to patch panels. Port density ranges from 4-port 10 Gbps XENPAK-based modules to 48-port 1000 Mbps RJ-45 modules, and even 96-port RJ-21 connector modules supporting 10/100 Mbps. Options include PoE and dCEF capabilities.

Firewall Services Modules. Firewall Services Modules provide firewall services, just as a PIX firewall appliance would. The difference is that all connections are internal to the switch, resulting in very high throughput. Because the interfaces are switched virtual interfaces (SVIs), the FWSM is not limited to physical connections like an appliance is. There can be hundreds of interfaces on an FWSM, each corresponding to a VLAN in the switch. The FWSM is also capable of over 4 Gbps of throughput, as compared with 1.7 Gbps on the PIX 535.

The FWSM supports multiple contexts, which allows for virtual firewalls that can serve different functions, be supported by different parties, or both. One example where this might be useful would be for a service provider who wishes to provide individual firewalls to customers while having only a single physical device.

The FWSM is a separate device in the chassis. To administer the FWSM, you must first connect to it. Here, I'm connecting to an FWSM in slot 8:

```
Switch-IOS# session slot 8 proc 1
The default escape character is Ctrl-^, then x.
You can also type 'exit' at the remote prompt to end the session
Trying 127.0.0.71 ... Open

User Access Verification

Password:
Type help or '?' for a list of available commands.
Switch-IOS-FWSM > en
Password: ********
Switch-IOS-FWSM #
```

If the FWSM is running in single-context mode, you will be able to run all PIX commands as if you were in any other PIX firewall. If the FWSM is running in multiple-context mode, you will be in the system context and will need to change to the proper context to make your changes. This is done with the changeto context command:

```
Switch-IOS-FWSM# sho context
Context Name      Class      Interfaces          URL
 admin            default                        disk:/admin.cfg
*EComm            default    vlan20,30           disk:/Ecomm.cfg
Switch-IOS-FWSM# changeto context EComm
Switch-IOS-FWSM/EComm#
```

At this point, you will be in the EComm context, and assuming you're used to PIX firewalls, everything should look very familiar:

```
Switch-IOS-FWSM/EComm# sho int
Interface Vlan20 "outside", is up, line protocol is up
        MAC address 0008.4cff.b403, MTU 1500
        IP address 10.1.1.1, subnet mask 255.255.255.0
                Received 90083941155 packets, 6909049206185 bytes
                Transmitted 3710031826 packets, 1371444635 bytes
                Dropped 156162887 packets
Interface Vlan30 "inside", is up, line protocol is up
        MAC address 0008.4cff.b403, MTU 1500
        IP address 10.10.10.1, subnet mask 255.255.255.0
                Received 156247369908 packets, 214566399699153 bytes
                Transmitted 2954364369 packets, 7023125736 bytes
                Dropped 14255735 packets
```

Content Switch Modules. The Content Switch Modules from Cisco are an excellent alternative to standalone content switches. The CSM is capable of 4 Gbps of throughput, and is available with an SSL accelerator daughter card.

CSM integration with the 6500 running native IOS is very smooth. All of the CSM commands are included in the switch's CLI. The commands for the CSM are included under the `module CSM` *`module#`* command. The command expands to the full `module contentswitchingmodule` *`module#`* in the configuration:

```
Switch-IOS (config)# mod csm 9
Switch-IOS (config-module-csm)#
```

One big drawback of CSM modules is that they are not fabric-enabled. While this is not an issue in terms of the throughput of the blade itself, it becomes an issue if the switch containing the CSM will also be serving the servers being balanced. The CSM is a 32 Gbps blade. Inserting it into a switch that is using the fabric backplane will cause the supervisor to revert to bus mode instead of faster modes such as dCEF. A switch with a Supervisor-720, fabric-only Ethernet modules, and a CSM will not run at 720 Gbps because of the CSM's limited backplane connections.

CSM blades will operate in a stateful failover design. The pair of CSMs can sync their configurations, provided they are running Version 4.2(1) or later. They can be synced with the `hw-module csm` *`module#`* `standby config-sync` command:

```
Switch-IOS# hw-module csm 9 standby config-sync
Switch-IOS#
May  5 17:12:14: %CSM_SLB-6-REDUNDANCY_INFO: Module 9 FT info: Active: Bulk sync
started
May  5 17:12:17: %CSM_SLB-4-REDUNDANCY_WARN: Module 9 FT warning: FT configuration
might be out of sync.
May  5 17:12:24: %CSM_SLB-4-REDUNDANCY_WARN: Module 9 FT warning: FT configuration
back in sync
May  5 17:12:26: %CSM_SLB-6-REDUNDANCY_INFO: Module 9 FT info: Active: Manual bulk
sync completed
```

Network Analysis Modules. Cisco's NAM is essentially a remote monitoring (RMON) probe and packet-capture device that allows you to monitor any port, VLAN, or combination of the two as if you were using an external packet-capture device.

The NAM is controlled through a web browser, which can be tedious when you're looking at large capture files. The benefit of the web-based implementation is that no extra software is required. The NAM may also be used from anywhere that the network design allows.

The interface of the packet-capture screen should be familiar to anyone who has used products such as Ethereal. Each packet is broken down as far as possible, and there is an additional window showing the ASCII contents of the packets.

One of the limitations of the packet capture is the lack of smart alarm indications such as those found in high-end packet-capture utilities. Many other features are available on the NAM, as it operates as an RMON probe.

The NAM is an excellent troubleshooting tool, and because it's always there, it can be invaluable during a crisis. (Chances are someone won't borrow the blade out of your production 6509, though stranger things have happened.) The additional feature of being able to capture more than one session at a time makes the NAM blade an excellent addition to your arsenal of tools. With the ability to capture from RSPAN sources (see Chapter 18), the NAM blade can be used to analyze traffic on any switch on your network.

A sample screen from the NAM interface is shown in Figure 17-7.

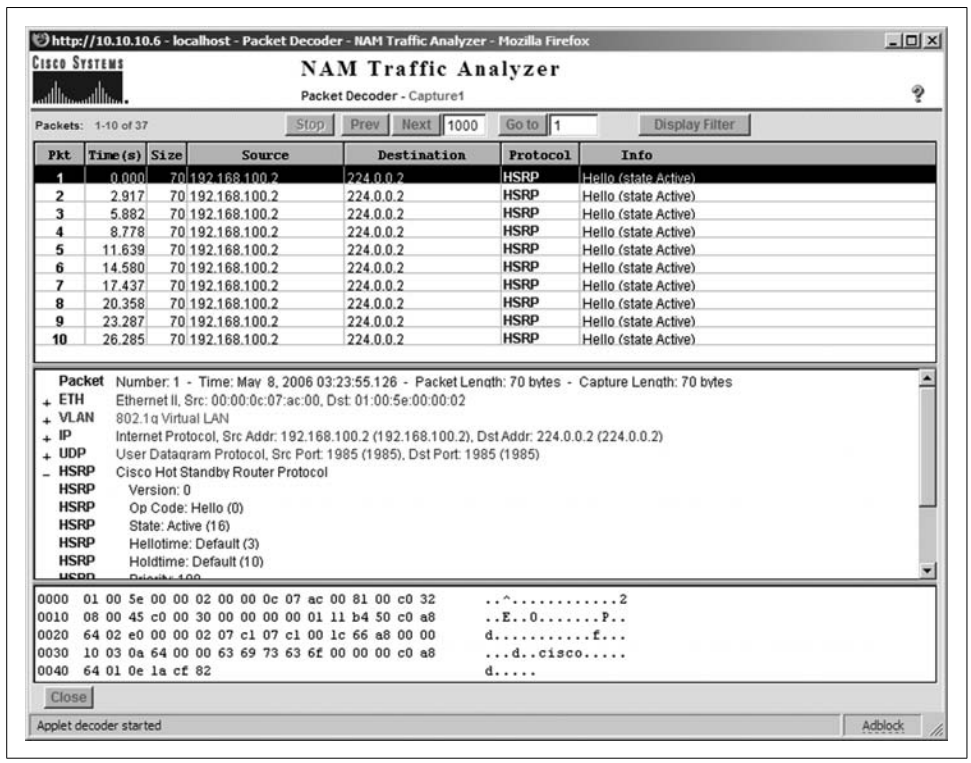


Figure 17-7. Network Analysis Module packet capture detail

Intrusion Detection System Modules. Intrusion detection functionality can be added to the 6500-series chassis with the introduction of an IDSM. These modules are actually preconfigured Linux servers that reside on a blade. They act like IDS appliances, but have the added ability of sampling data streams at wire speed because they are connected to the crossbar fabric bus.

These modules can be managed through an onboard secure web interface, which is shown in Figure 17-8, though Cisco recommends that they be managed through another application such as VPN/Security Management Solution (VMS), Cisco Security Manager, or Cisco Security Monitoring, Analysis, and Response System (MARS).

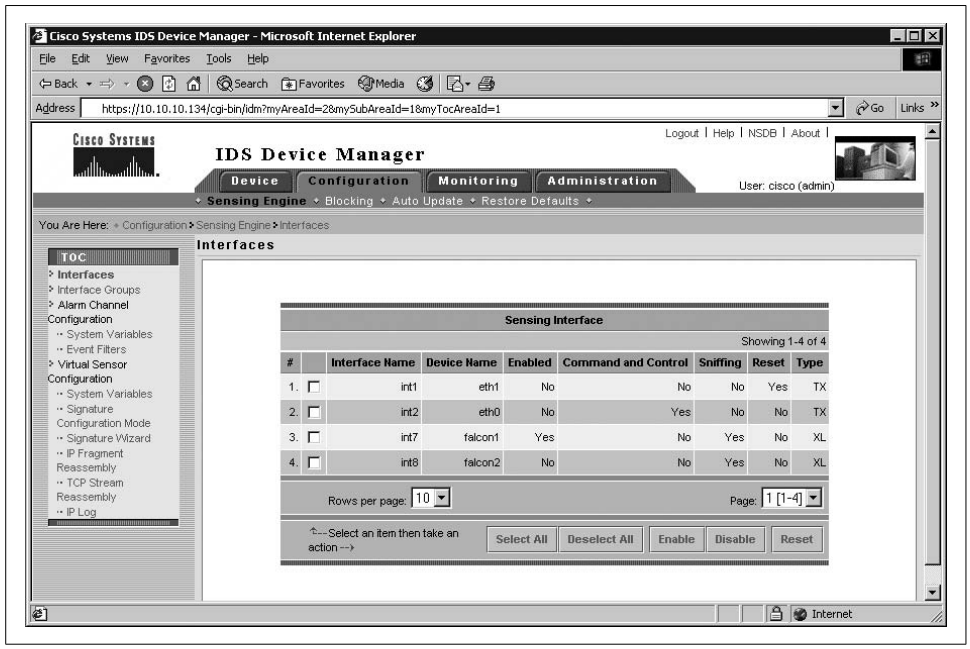


Figure 17-8. IDS module on-board configuration

Basic configuration of the module is done via the switch itself by connecting to the module with the `session slot` *`module#`* `processor` *`processor#`* command. The *`processor#`* is usually 1:

```
Switch-IOS-1# session slot 9 proc 1
The default escape character is Ctrl-^, then x.
You can also type 'exit' at the remote prompt to end the session
Trying 127.0.0.91 ... Open
login: cisco
Password:
***NOTICE***
This product contains cryptographic features and is subject to United States
and local country laws governing import, export, transfer and use. Delivery
of Cisco cryptographic products does not imply third-party authority to import,
export, distribute or use encryption. Importers, exporters, distributors and
users are responsible for compliance with U.S. and local country laws. By using
this product you agree to comply with applicable laws and regulations. If you
are unable to comply with U.S. and local laws, return this product immediately.

A summary of U.S. laws governing Cisco cryptographic products may be found at:
http://www.cisco.com/wwl/export/crypto
```

```
If you require further assistance please contact us by sending email to
export@cisco.com.
Switch-IOS-1-IDS#
```

Configuration of the IDSM is quite different from that of other devices, and is a topic for a book unto itself.

FlexWAN modules. FlexWAN modules allow the connection of WAN links such as T1s as well as high-speed links such as DS3s up to OC3s.

There are two types of FlexWAN modules: FlexWAN and Enhanced FlexWAN. The primary differences between the two versions are CPU speed, memory capacity, and connection to the crossbar fabric bus.

Enhanced FlexWAN modules use the same WAN port adapters used in the Cisco 7600-series routers. The module is layer-3-specific, and requires either a Supervisor-2 with an MSFC, or a Supervisor-720 to operate. When running in hybrid IOS mode, the FlexWAN interfaces are not visible to layer two with CatOS.

Communication Media Modules. The Communication Media Module (CMM) provides telephony integration into 6500-series switches. This fabric-enabled module has thee slots within it, which accept a variety of *port adapters*.

The port adapters available for the CMM include Foreign eXchange Service (FXS) modules for connection to analog phones, modems, and fax machines; T1/E1 CAS and PRI gateway modules; conferencing and transcoding port adapters that allow conferencing services; and Unified Survivable Remote Site Telephony (SRST) modules that will manage phones and connections should the connection to a Unified Call Manager become unavailable.

The port adapters can be mixed and matched in each of the CMMs installed. A 6500-series chassis can be filled with CMMs and a supervisor, providing large port density for VoIP connectivity.

CatOS Versus IOS

Cisco Catalyst switches originally did not run IOS—the early chassis-based switches were CatOS-based. The reason for this was that the technology for these switches came from other companies that Cisco acquired, such as Crescendo, Kalpana, and Grand Junction.

CatOS may appear clunky to those who have used only IOS, but there are some distinct advantages to using CatOS in a switching environment. One of these advantages can also be considered a disadvantage: when a Catalyst 6500 runs CatOS, and also has an MSFC for layer-3 functionality, the MSFC is treated like a separate device. The switch runs CatOS for layer-2 functionality, and the MSFC runs IOS for

layer-3 and above functionality. This separation can be easier to understand for people who do not have experience with IOS layer-3 switches, but for those who are used to IOS-based switches like Catalyst 3550s and 3750s, the need to switch between operating systems can be burdensome and confusing.

Because all of the new Cisco layer-3 switches (such as the 3550 and the 3750) run only IOS, learning the native IOS way of thinking is a smart move, as that's clearly the direction Cisco is taking. At one point, Cisco actually announced plans to discontinue CatOS, but there was such an uproar from die-hard CatOS users that the plans were scrubbed. As a result, CatOS is still alive and well.

Another advantage of CatOS over IOS is the concise way in which it organizes information. An excellent example is the `show port` command in CatOS:

```
Switch-CatOS# sho port

Port  Name                 Status     Vlan       Duplex Speed Type
----- -------------------- ---------- ---------- ------ ----- ------------
 1/1  Trunk                connected  trunk        full  1000 1000BaseSX
 1/2  Trunk                connected  trunk        full  1000 1000BaseSX
 2/1  Trunk                connected  trunk        full  1000 1000BaseSX
 2/2  Trunk                connected  trunk        full  1000 1000BaseSX
 3/1  Web-1-E1             connected  20         a-full a-100 10/100BaseTX
 3/2  Web-2-E1             connected  20         a-full a-100 10/100BaseTX
 3/3  Web-3-E1             connected  20           full   100 10/100BaseTX
 3/4  Web-4-E1             connected  20           full   100 10/100BaseTX
 3/5  Web-5-E1                 connected  20         a-full a-100 10/100BaseTX
 3/6  Web-6-E1               connected  20         a-full a-100 10/100BaseTX
 3/7  Web-7-E1               connected  20         a-full a-100 10/100BaseTX
 3/8  App-1-E1             connected  40         a-full a-100 10/100BaseTX
 3/9  App-2-E1             connected  40         a-full a-100 10/100BaseTX
 3/10 App-3-E1             connected  40         a-full a-100 10/100BaseTX
 3/11 App-4-E1             connected  40         a-full a-100 10/100BaseTX
 3/12                      notconnect              full   100 10/100BaseTX
 3/13                      notconnect              full   100 10/100BaseTX
 3/14 DB-1-E1              connected  50           full   100 10/100BaseTX
 3/15 DB-2-E1              connected  50         a-full a-100 10/100BaseTX
 3/16 DB-3-E1              connected  50         a-full a-100 10/100BaseTX
```

Here, on one screen, we can see the port, the port's name (if any), its status, what VLAN it is associated with, the speed and duplex mode, the auto-negotiation status, and the port type.

IOS has nothing that directly compares to this command. Instead, the user must piece together the information from multiple sources. One of the best commands to start with is `show ip interface brief`:

```
Switch-IOS# sho ip int brief
Interface                  IP-Address      OK? Method Status                Protocol
Vlan1                      unassigned      YES NVRAM  administratively down down
```

```
Vlan20                 10.10.20.2      YES manual up                    up
Vlan40                 10.10.40.2      YES manual up                    up
Vlan50                 10.10.50.2      YES manual up                    up
GigabitEthernet1/1     unassigned      YES unset  up                    up
GigabitEthernet1/2     unassigned      YES unset  up                    up
GigabitEthernet1/3     unassigned      YES unset  up                    up
GigabitEthernet1/4     unassigned      YES unset  up                    up
```

Unfortunately, this command, while useful, does not show you the port names. You need the `show interface description` command for that:

```
Switch-IOS# sho int desc
Interface                      Status         Protocol Description
Vl1                            admin down     down
Vl20                           up             up       Web-VLAN
Vl40                           up             up       App-VLAN
Vl50                           up             up       DB-VLAN
Gi1/1                          up             up       Web-1-E1
Gi1/2                          up             up       Web-2-E1
Gi1/3                          up             up       Web-3-E1
Gi1/4                          up             up       Web-4-E1
```

Even with the use of both of these commands, you still don't know the VLANs to which the ports are assigned. For VLAN assignments, you need the `show vlan` command:

```
Switch-IOS# sho vlan

VLAN Name                             Status    Ports
---- -------------------------------- --------- -------------------------------
1    default                          active
20   WEB-VLAN                         active    Gi1/1, Gi1/2, Gi1/3, Gi1/4
                                                Gi1/5, Gi1/6, Gi1/7
40   APP-VLAN                         active    Gi1/8, Gi1/9, Gi1/10, Gi1/11
50   DB-VLAN                          active    Gi1/14, Gi1/15, Gi1/16
1002 fddi-default                     act/unsup
1003 token-ring-default               act/unsup
1004 fddinet-default                  act/unsup
```

IOS tends to be a bit wordy. For example, the output of the IOS `show interface` *`interface#`* command, which shows the pertinent information for interfaces, looks like this:

```
Switch-IOS# sho int g3/1
GigabitEthernet3/1 is up, line protocol is up (connected)
  Hardware is C6k 1000Mb 802.3, address is 0015.6356.62bc (bia 0015.6356.62bc)
  Description: Web-1-E1
  MTU 1500 bytes, BW 1000000 Kbit, DLY 10 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation ARPA, loopback not set
  Full-duplex, 1000Mb/s
  input flow-control is off, output flow-control is on
  Clock mode is auto
```

```
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input never, output 00:00:47, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/2000/2/0 (size/max/drops/flushes); Total output drops: 2
  Queueing strategy: fifo
  Output queue: 0/40 (size/max)
  5 minute input rate 456000 bits/sec, 91 packets/sec
  5 minute output rate 110000 bits/sec, 81 packets/sec
     714351663 packets input, 405552413403 bytes, 0 no buffer
     Received 15294 broadcasts, 0 runts, 0 giants, 0 throttles
     2 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored
     0 input packets with dribble condition detected
     656796418 packets output, 97781644875 bytes, 0 underruns
     2 output errors, 0 collisions, 2 interface resets
     0 babbles, 0 late collision, 0 deferred
     0 lost carrier, 0 no carrier
     0 output buffer failures, 0 output buffers swapped out
```

The output from the CatOS command `show port` *port#* is much easier to read, especially when you're glancing quickly for a specific tidbit of information. The tradeoff is that the command provides less information than the IOS version:

```
Switch-CatOS: (enable) sho port 3/1
* = Configured MAC Address

Port  Name                 Status     Vlan       Duplex Speed Type
----- -------------------- ---------- ---------- ------ ----- ------------
 3/1  Web-1-E1             connected  20           auto  auto 10/100BaseTX

Port  AuxiliaryVlan AuxVlan-Status     InlinePowered     PowerAllocated
                                    Admin Oper   Detected mWatt mA @42V
----- ------------- -------------- ----- ------ -------- ----- --------
 3/1  none          none           -     -      -        -     -


Port  Security Violation Shutdown-Time Age-Time Max-Addr Trap     IfIndex
----- -------- --------- ------------- -------- -------- -------- -------
 3/1  disabled  shutdown              0        0        1 disabled       5

Port  Num-Addr Secure-Src-Addr     Age-Left Last-Src-Addr     Shutdown/Time-Left
----- -------- -----------------   -------- ----------------- ------------------
 3/1         0                 -          -                 -        -         -

Port  Flooding on Address Limit
----- -------------------------
 3/1                    Enabled
--More–
```

Many people prefer the output of the commands on CatOS switches, though as a consultant, I have no real preference, and must work with whatever the client has at the time.

One of the big features found in CatOS that was not available in IOS until very recently is the `show top` feature. Executing the command `show top 5 util all back interval 60` instructs the switch to run a Top-N report in the background for the five most utilized ports, and save the report for viewing. When the report is done, a message is displayed indicating that it is ready to be viewed:

```
Switch-CatOS: (enable) show top 5 util all back interval 60
Switch-CatOS: (enable) 2006 May 07 12:47:00 EST +00:00 %MGMT-5-TOPN_START:Report 3
started by telnet/20.20.20.100/GAD
```

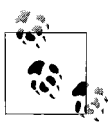
Notice that because I specified the background option, I can do other things while the report is running:

```
Switch-CatOS: (enable)
Switch-CatOS: (enable)
Sec-6505-1-TOP: (enable) dir
-#- -length- -----date/time------ name
  2 10518855 May 2  2006 02:27:09 cat6000-supk8.7-7-9.bin
 15    82230 May 2  2006 08:21:55 switch.cfg

4604208 bytes available (11386576 bytes used)
Switch-CatOS: (enable)
Switch-CatOS: (enable)
Switch-CatOS: (enable) 2006 May 07 12:48:01 EST +00:00 %MGMT-5-TOPN_AVAILABLE:Report
3 available: (enable) 2006 May 07 12:48:01 EST +00:00 %MGMT-5-TOPN_AVAILABLE:Report 3
available
```

While I was looking at the flash directory, my report finished. The switch told me that the report generated was report #3. I can view it using this command:

```
Switch-CatOS: (enable) sho top report 3
Start Time:     May 07 2006 12:47:00
End Time:       May 07 2006 12:48:01
PortType:       all
Metric:         util
Port  Band- Uti Bytes               Pkts       Bcst       Mcst       Error Over
      width  %  (Tx + Rx)           (Tx + Rx)  (Tx + Rx)  (Tx + Rx)  (Rx)  flow
----- ----- --- -------------------- ---------- ---------- ---------- ----- ----
 3/14   100   0               624014       1126         89         89     0    0
 3/15   100   0               105347        590          6         32     0    0
 3/16   100   0               889310       2319         89         99     0    0
 3/8    100   0               536246       3422         97         41     0    0
 3/9    100   0               315228       2094          0        405     0    0
```

The `show top` feature also provides the ability to run a report showing the Top-N error-producing ports, which is a tremendously useful tool when suspected auto-negotiation issues exist. To run an error-based Top-N report on CatOS, execute the command `show top error all back interval 60`.



IOS Versions 12.2(18)SXE and later for the Catalyst 6500 allow Top-N reports to be generated. The results are very similar to those generated by CatOS, but the commands to run the reports are different. To run a Top-N report on IOS, execute the `collect top` command.

CHAPTER 18

Catalyst 3750 Features

The Catalyst 3750 switch is the next step in the evolution of the very popular 3550 fixed-configuration switch. The 3550 was the first multilayer switch offered at its price point to boast such a vast array of features. It was later succeeded by the 3560. The 3750 is a more powerful switch that introduced, among other things, a true stacking feature, which the 3560 lacks.

There is not enough room to cover all of the capabilities of the 3750 in one chapter, so I've focused on those features that I have found most useful in the field. I've purposely not included all the gory details of each feature discussed. Instead, I've covered what I believe you'll need to know to take advantage of these features.

Not all of the features I'll discuss are specific to the 3750, though the commands are. (The commands may be identical on other models, but this chapter specifically includes examples taken from the 3750.) As always, Cisco's documentation covers all the features in detail.

Stacking

One of the major shortcomings of the 3550 and 3560 switches was the way they were stacked. *Stacking* refers to the ability to link together multiple switches, usually of the same type, to form a single logical switch with a single management IP address. Once you telnet or SSH to the IP address, you can control the stack as if it were a single device.

The 3550 used a stacking design that required modules called *stacking GBICs* to be used in one of the gigabit GBIC slots. This not only limited the stacking backplane speed to 1 Gbps, but also tied up the only gigabit slots on the otherwise 100 Mbps switches for stacking. So, when you stacked your switches, you could no longer

connect your uplinks at gigabit speeds. The 3560 uses a special SFP interconnect cable. It is available with 10/100/1000 RJ-45 ports, but using the stacking cables still occupies one of the fiber uplink ports.

The 3750 uses a more traditional approach, incorporating special stacking cables that connect to the back of the switch chassis. This backplane connection is 32 Gbps, and does not tie up any of the ports on the front of the switch. I won't go into the physical connection of a switch stack, as the Cisco documentation is more than adequate.

On a single 24-port 3750, the interfaces are numbered Gi1/0/1–Gi1/0/23. The interfaces are described as *interface-type stack-member#/module#/port#*. The *interface-type* is usually Gigabit Ethernet on a 3750, though some models support 10 Gbps ports. The *stack-member#* is 1 for a standalone switch, and for a stacked switch reflects the position the switch occupies in the stack. The *module#* on a 3750 is always 0. The *port#* is the physical port number on the switch. Thus, port 14 on the third switch in a stack would be numbered Gi3/0/14.

For the most part, any feature can be used on a single switch or a stack. For example, EtherChannels and SPAN sessions can be configured between switches within a stack.

Interface Ranges

Interface ranges are a very useful addition to IOS. Instead of entering the same commands on multiple interfaces, you can specify a range of interfaces, and then enter the commands once. When you apply commands to an interface range, the parser will replicate the commands on each interface within the range. On a switch with 96 interfaces, this can save hours of time—especially during initial configurations.

Interface ranges are composed of lists of interfaces. Interfaces can be specified individually or grouped in concurrent ranges. Individual interfaces are separated by commas, while ranges are shown as the starting and ending interfaces separated by hyphens. Here, I'm accessing the two interfaces g1/0/10 and g1/0/12:

```
3750(config)# interface range g1/0/10 , g1/0/12
3750(config-if-range)#
```

Once in *config-if-range* configuration mode, all commands you enter will be applied to every interface you've included in the specified range.

Here, I'm accessing the range of ports g1/0/10 through (and including) g1/0/12. When specifying a range, the second value only needs to include the significant value from the interface. That is, I don't need to type `g/1/0/10 - g1/0/12`, but only `g1/0/10 - 12`:

```
3750(config)# interface range g1/0/10 - 12
3750(config-if-range)#
```

To reference multiple ranges, separate them with commas. Here, I'm referencing the ranges g1/0/10 through g1/0/12 and g1/0/18 through g1/0/20:

```
3750(config)# interface range g1/0/10 - 12 , g1/0/18 - 20
3750(config-if-range)#
```

Not only can you specify lists of interfaces, but you can save lists of interfaces for future reference. To do this, define the interface range with the `define interface-range` *`macro-name`* command. Here, I've created a macro called `Servers`:

```
3750(config)# define interface-range Servers g1/0/10 , g1/0/12 - 14
3750(config)#
```

Don't confuse these macros with another feature, called *smartport macros* (covered next). The two features are unrelated. To make matters even more confusing, you can apply a smartport macro to an interface-range macro. For the programmers out there, smartport macros are probably closer to the macros you're used to.

Once you've defined an interface-range macro, you can reference it with the `interface range macro` *`macro-name`* command:

```
3750(config)# interface range macro Servers
3750(config-if-range)#
```

Macros

Macros, called *smartport macros* by Cisco, are groups of commands saved with reference names. Macros are useful when you find yourself entering the same group of commands repeatedly. For example, say you're adding a lot of servers. Every time you add a server, you execute the same configuration commands for the switch interface to be used for that server. You could create a macro that would execute all of the commands automatically, and then simply reference this macro every time you add a new server to the switch.

Macros are created with the `macro` command. There are two types of macros: *global* and *interface*. An interface macro (the default type) is applied to one or more interfaces. To make a macro global, include the `global` keyword when creating it.

The way macros are created is a little strange, because the commands are not parsed as you enter them. As a result, you can enter invalid commands without causing errors. First, enter the `macro name` *`macroname`* command. Because you're not including the `global` keyword, this will be an interface macro. Then, enter the commands you'd like to include in the macro, one by one. These commands are not checked for syntax. When you're done entering commands, put an at sign (`@`) on a line by itself.

Here, I've created a macro named SetServerPorts. The commands included are `spanning-tree portfast`, `hhhhhh` (an invalid command), and `description <[ Server ]>`:

```
3750(config)# macro name SetServerPorts
Enter macro commands one per line. End with the character '@'.
spanning-tree portfast
hhhhhh
description <[ Server ]>
@
3750(config)#
```

Inserting the description within <[and]> brackets does not accomplish anything special in IOS. This is just something I've done for years to make the descriptions stand out, both in the running config and in the output of `show` commands, such as `show interface`.

As you can see, the switch accepted all of the commands without a problem. The macro, including the bogus command, now appears in the running config:

```
!
macro name SetServerPorts
spanning-tree portfast
hhhhhh
description <[ Server ]>
@
!
```

When you apply a macro, the parser gets involved, and applies syntax and validity checking to your commands. This is when you'll see errors if you've entered invalid commands. The macro will not terminate on errors, so be sure to watch out for them:

```
3750(config-if)# macro apply SetServerPorts
%Warning: portfast should only be enabled on ports connected to a single
 host. Connecting hubs, concentrators, switches, bridges, etc... to this
 interface  when portfast is enabled, can cause temporary bridging loops.
 Use with CAUTION

%Portfast has been configured on GigabitEthernet1/0/20 but will only
 have effect when the interface is in a non-trunking mode.
hhhhhh
 ^
% Invalid input detected at '^' marker.

3750(config-if)#
```

Notice also that commands that do not generate output show no indication of being completed. This is because a macro is just a group of commands that are run when the macro is invoked. If you'd like more information about what your macro is doing when you run it, you can include the `trace` keyword. This will add a line indicating when each command in the macro is run:

```
3750(config-if)# macro trace SetServerPorts
Applying command... 'spanning-tree portfast'
```

```
%Warning: portfast should only be enabled on ports connected to a single
 host. Connecting hubs, concentrators, switches, bridges, etc... to this
 interface  when portfast is enabled, can cause temporary bridging loops.
 Use with CAUTION

%Portfast has been configured on GigabitEthernet1/0/20 but will only
 have effect when the interface is in a non-trunking mode.
Applying command... 'hhhhhh'
hhhhhh
 ^
% Invalid input detected at '^' marker.

Applying command... 'description <[ Server ]>'
3750(config-if)#
```

When you run a macro, a *macro description* is added to the interface or interfaces to which the macro has been applied. The configuration for the interface is altered to include the command `macro description`, followed by the name of the macro:

```
interface GigabitEthernet1/0/20
 description <[ Server ]>
 switchport mode access
 macro description SetServerPorts
 storm-control broadcast level bps 1g 900m
 spanning-tree portfast
```

You can add your own macro description with the `macro description` command, from within the macro, or from the command line:

```
3750(config-if)# macro description [- Macro Description -]

interface GigabitEthernet1/0/20
 description <[ Server ]>
 switchport mode access
 macro description SetServerPorts | [- Macro Description -]
 storm-control broadcast level bps 1g 900m
 spanning-tree portfast
```

As you can see, every time you run a macro or execute the `macro description` command, the description specified (or the macro name) is appended to the `macro description` command in the configuration of the interface to which it's applied. Iterations are separated with vertical bars.

An easier way to see where macros have been applied is with the `show parser macro description` command. Here, you can see that I ran the same macro repeatedly on the Gi1/0/20 interface:

```
SW2# sho parser macro description
Interface    Macro Description(s)
-------------------------------------------------------------
Gi1/0/20     SetServerPorts | SetServerPorts | SetServerPorts | [- Macro
Description -]
-------------------------------------------------------------
```

To see all the macros on the switch, use the `show parser macro brief` command:

```
3750# sho parser macro brief
    default global   : cisco-global
    default interface: cisco-desktop
    default interface: cisco-phone
    default interface: cisco-switch
    default interface: cisco-router
    default interface: cisco-wireless
    customizable     : SetServerPorts
```

Six macros are included in IOS by default, as you can see in the preceding output (the six listed `default` macros). You can use the `show parser macro name` *`macroname`* to view the details of any of the macros:

```
SW2# sho parser macro name cisco-desktop
Macro name : cisco-desktop
Macro type : default interface
# macro keywords $access_vlan
# Basic interface - Enable data VLAN only
# Recommended value for access vlan should not be 1
switchport access vlan $access_vlan
switchport mode access

# Enable port security limiting port to a single
# MAC address -- that of desktop
switchport port-security
switchport port-security maximum 1

# Ensure port-security age is greater than one minute
# and use inactivity timer
switchport port-security violation restrict
switchport port-security aging time 2
switchport port-security aging type inactivity

# Configure port as an edge network port
spanning-tree portfast
spanning-tree bpduguard enable
```

This macro contains some advanced features, such as variables and comments. See the Cisco documentation for further details on the macro feature.

If you've been wondering how to apply a smartport macro to an interface-range macro, here is your answer (assuming an interface-range macro named `Workstations`, and a smartport macro named `SetPortsPortfast`):

```
SW2(config)# interface range macro Workstations
SW2(config-if-range)# macro apply SetPortsPortfast
```

Flex Links

Flex links are layer-2 interfaces manually configured in primary/failover pairs. The Spanning Tree Protocol (STP, discussed in Chapter 8) normally provides primary/failover functionality, but it was designed for the sole purpose of preventing loops. Flex links are used to ensure that there are backup links for primary links. Only one of the links in a flex-link pair will be forwarding traffic at any time.

Flex links are designed for switches where you do not wish to run spanning tree, and should be used only on switches that do not run spanning tree. Should flex links be configured on a switch running spanning tree, the flex links will not participate in STP.

Flex links are configured on the primary interface by specifying the backup interface with the `switchport backup interface` command:

```
interface GigabitEthernet1/0/20
 switchport access vlan 10
 switchport backup interface Gi1/0/21
!
interface GigabitEthernet1/0/21
 switchport access vlan 10
```

No configuration is necessary on the backup interface.

Neither of the links can be an interface that is a member of an EtherChannel. An EtherChannel *can* be a flex-link backup for another port channel. A single physical interface can be a backup to an EtherChannel as well. The backup link does not need to be the same type of interface as the primary. For example, a 100 Mbps interface can be a backup for a 1 Gbps interface.

Monitoring flex links is done with the `show interface switchport backup` command:

```
3750# sho int switchport backup

Switch Backup Interface Pairs:

Active Interface        Backup Interface        State
------------------------------------------------------------------------
GigabitEthernet1/0/20   GigabitEthernet1/0/21   Active Down/Backup Down
```

Storm Control

Storm control prevents broadcast, multicast, and unicast storms from overwhelming a network. Storms can be the result of a number of issues, from bridging loops to virus outbreaks. With storm control, you can limit the amount of storm traffic that can come into a switch port. Outbound traffic is not limited.

With storm control enabled, the switch monitors the packets coming into the configured interface. It determines the amount of unicast, multicast, or broadcast traffic every 200 milliseconds, then compares that amount with a configured threshold. Packets that exceed the threshold are dropped.

This sounds straightforward, but the feature actually works differently from how many people expect. When I first learned of it, I assumed that the preceding description was accurate—that is, that at any given time, traffic of the type I'd configured for monitoring would be allowed to come into the switch until the threshold was met (similar to what is shown in Figure 18-1). The reality, however, is more complicated.

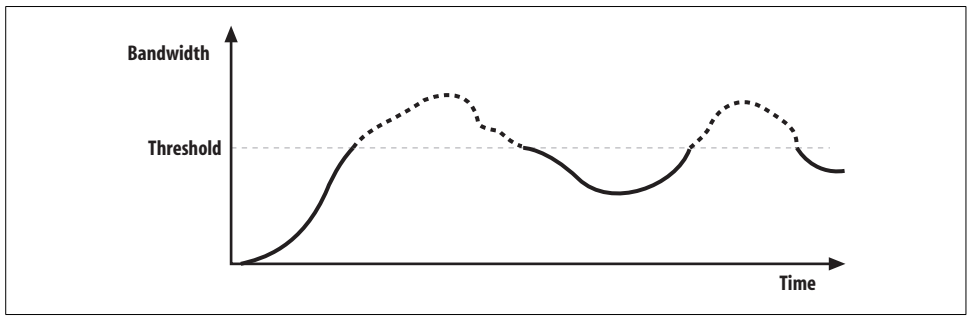


Figure 18-1. Incorrect storm-control model

In reality, the switch monitors the interface, accumulating statistics in 200 ms increments. If, at the end of 200 ms, the threshold has been exceeded, the configured (or default) action is taken for the next 200 ms increment.

Figure 18-2 shows how storm-control actually functions. Traffic is measured in 200 ms increments, shown on the graph as T_0, T_1, and so on. If the type of traffic being monitored does not surpass the configured threshold during a given interval, the next 200 ms interval is unaffected. In this example, when T_1 is reached, the threshold has not been exceeded, so the next interval (ending with T_2) is unaffected. However, the configured threshold is exceeded during the T_2 interval, so the packets received during the next interval (T_3) are dropped. The important distinction here is that during each interval, received packets or bytes are counted from the start of that interval only.

Traffic is still monitored on the interface during the interval in which packets are being dropped (packets are received, but not passed on to other interfaces within the switch). If the packet rate again exceeds the configured threshold, packets for the next interval are again dropped. If, however, the number of packets received is below the configured threshold, as is the case in the interval T_3 in Figure 18-2, packets are allowed for the next interval.

Because packets are not received in a smooth pattern, understanding how storm control works will help you understand why it may cause intermittent communication failures in normal applications on a healthy network. For example, if you were to

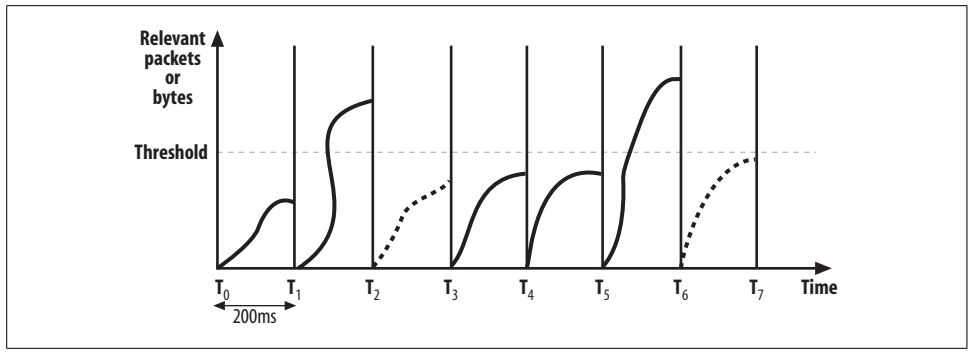


Figure 18-2. Actual storm-control function

encounter a virus that sent enough broadcasts to trigger your configured storm control, chances are that the port would stop forwarding all broadcasts because the threshold would constantly be exceeded. On the other hand, if you had a healthy network, but your normal broadcast traffic was hovering around the threshold, you would probably end up missing some broadcasts while passing others. Only by closely monitoring your switches can you be sure that you're not impeding normal traffic. If storm control is causing problems in an otherwise healthy network, you probably need to tune the storm-control parameters.

Storm control is configured using the `storm-control` interface command:

```
3750(config-if)# storm-control ?
  action     Action to take for storm-control
  broadcast  Broadcast address storm control
  multicast  Multicast address storm control
  unicast    Unicast address storm control
```

Storm control is available only on physical interfaces. While the commands are available on EtherChannel (port channel) interfaces, they are ignored if configured.

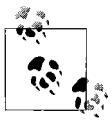
Storm control has changed over the past few versions of IOS. Originally, the commands to implement this feature were `switchport broadcast`, `switchport multicast`, and `switchport unicast`. Older IOS versions and 3550 switches may still use these commands.

Additionally, the latest releases of IOS allow for some newer features. One of the newer features is the ability to send an SNMP trap instead of shutting down the port. This can be configured with the `storm-control action` interface command:

```
3750(config-if)# storm-control action ?
  shutdown  Shutdown this interface if a storm occurs
  trap      Send SNMP trap if a storm occurs
```

Also, thresholds could originally only be set as percentages of the overall available bandwidth. Now, you have the option of configuring percentages, actual bits per second, or packets per second values. Each storm-control type (broadcast, multicast, and unicast) can be configured with any of these threshold types:

```
3750(config-if)# storm-control broadcast level ?
  <0.00 - 100.00>  Enter rising threshold
  bps              Enter suppression level in bits per second
  pps              Enter suppression level in packets per second
```



A threshold (any type) of 0 indicates that no traffic of the configured type is allowed. A percentage threshold of 100 indicates that the configured type should never be blocked.

When configuring bits per second or packets per second thresholds, you can specify a value either alone or with a metric suffix. The suffixes allowed are `k`, `m`, and `g`, for *kilo*, *mega*, and *giga*:

```
3750(config-if)# storm-control broadcast level bps ?
  <0.0 - 10000000000.0>[k|m|g]  Enter rising threshold
```

Another new feature is the ability to specify a rising threshold and a falling threshold. When the rising threshold is passed, the configured type of packets will be dropped for the next interval. When the falling threshold is passed, the next interval will be allowed to pass the configured type of packets again.

Figure 18-3 shows an example of the effects of configuring rising and falling thresholds. The rising threshold is set higher than the falling threshold. This has a dramatic impact on the number of intervals dropping packets. When T_2 exceeds the rising threshold, T_3 drops packets, just as it did when only one threshold was configured. T_3 does not exceed the rising threshold, but because it exceeds the falling threshold, packets are again dropped in T_4. Once the rising threshold has been exceeded, traffic of the configured type will continue to be dropped as long as the falling threshold is exceeded. It is not until the interval ending at T_5 that the level finally falls below the falling threshold, thus allowing packets of the configured type to be forwarded again during the next interval.

The falling threshold is configured after the rising threshold. If no value is entered, the falling threshold is the same as the rising threshold:

```
3750(config-if)# storm-control broadcast level bps 100 ?
  <0.0 - 10000000000.0>[k|m|g]  Enter falling threshold
  <cr>
```

Here, I've configured the same thresholds using different forms of the same numbers. Either way is valid:

```
3750(config-if)# storm-control broadcast level bps 100000 90000
3750(config-if)# storm-control broadcast level bps 100m 90k
```

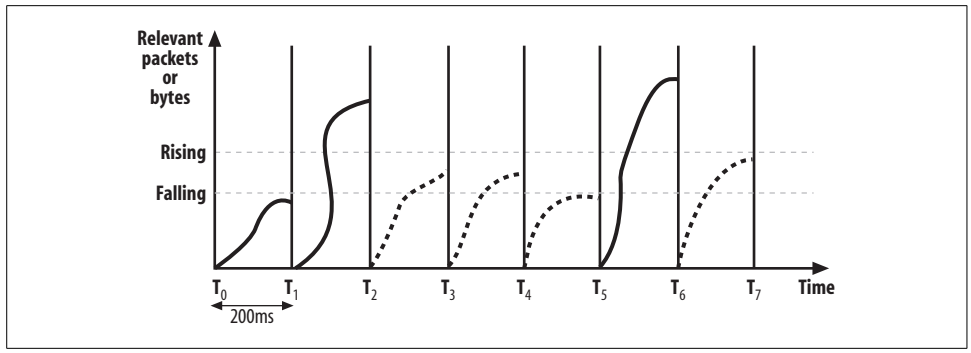


Figure 18-3. Rising and falling thresholds

I think the simplest way to configure storm control is with percentages. This is also the only supported method for older versions of IOS:

```
3750(config-if)# storm-control multicast level 40.5 30
```

Be careful when configuring multicast storm control. When multicast packets are suppressed, routing protocols that use multicasts will be affected. Control traffic such as Cisco Discovery Protocol (CDP) packets will not be affected, though, and having CDP functioning while routing protocols are not can make for some confusion in the field during outages.

To monitor storm control, use the `show storm-control` command. The output is not extensive, but you probably won't need to know anything else:

```
3750# sho storm-control
Interface  Filter State   Upper        Lower        Current
---------  -------------  -----------  -----------  ----------
Gi1/0/20   Link Down          1g bps     900m bps        0 bps
Gi1/0/21   Link Down          50.00%       40.00%        0.00%
Gi1/0/22   Forwarding         1m pps     500k pps        0 pps
```

The Current column shows the current value for the interface. This should be the first place you look if you think you're dropping packets due to storm control. Remember that this is measured every 200 ms, so you may have to execute the command many times to see whether your traffic is spiking.

You can also run the command for specific storm-control types (`broadcast`, `multicast`, or `unicast`). The output is the same, but includes only the type specified:

```
3750# sho storm-control unicast
Interface  Filter State   Upper        Lower        Current
---------  -------------  -----------  -----------  ----------
Gi1/0/19   Link Down       50.00%       40.00%        0.00%
```

Lastly, you can specify a specific interface by including the interface name:

```
3750# sho storm-control g1/0/20
Interface  Filter State   Upper        Lower        Current
---------  -------------  -----------  -----------  ----------
Gi1/0/20   Link Down          1g bps     900m bps        0 bps
```

Port Security

Port security is the means whereby you can prevent network devices from using a port on your switch. At the port level, you can specify certain MAC addresses that you allow or deny the right to use the port. This can be done statically or dynamically. For example, you can tell the switch to allow only the first three stations that connect to use a port, and then deny all the rest. You can also tell the switch that only the device with the specified MAC address can use the switch port, or that any node *except* the one with the specified MAC address can use the switch port.

MAC addresses can be either manually configured or dynamically learned. Addresses that are learned can be saved. Manually configured addresses are called *static secure MAC addresses*; dynamically learned MAC addresses are termed *dynamic secure MAC addresses*, and saved dynamic MAC addresses are called *sticky secure MAC addresses*.

Port security is enabled with the `switchport port-security` interface command. This command can be configured only on an interface that has been set as a switchport. Trunks and interfaces that are dynamic (the default) cannot be configured with port security:

```
3750(config-if)# switchport port-security
Command rejected: GigabitEthernet1/0/20 is a dynamic port.
```

If you get this error, you need to configure the port for switchport mode access before you can continue:

```
3750(config-if)# switchport mode access
3750(config-if)# switchport port-security
```

You cannot configure port security on a port that is configured as a SPAN destination:

```
3750(config-if)# switchport port-security
Command rejected: GigabitEthernet1/0/20 is a SPAN destination.
```

Once you've enabled port security, you can configure your options:

```
3750(config-if)# switchport port-security ?
  aging        Port-security aging commands
  mac-address  Secure mac address
  maximum      Max secure addresses
  violation    Security violation mode
  <cr>
```

Here, I'm configuring the interface to accept packets only from the MAC address 1234.5678.9012:

```
3750(config-if)# switchport port-security mac-address 1234.5678.9012
```

You might think that you can run the same command to add another MAC address to the permitted device list, but when you do this, you'll get an error:

```
3750(config-if)# switchport port-security mac-address 1234.5678.1111
Total secure mac-addresses on interface GigabitEthernet1/0/20 has reached maximum
limit.
```

By default, only one MAC address can be entered. To increase the limit, use the `switchport port-security maximum` command. Once you've increased the maximum, you can add another MAC address:

```
3750(config-if)# switchport port-security maximum 2
3750(config-if)# switchport port-security mac-address 1234.5678.1111
```

If you try to set the maximum to a number less than the number of secure MAC addresses already configured, you will get an error, and the command will be ignored.

You can also enter the `switchport port-security maximum` command without specifying any MAC addresses. By doing this, you will allow a finite number of MAC addresses to use the port. For example, with a maximum of three, the first three learned MAC addresses will be allowed, while all others will be denied.

If you need the switch to discover MAC addresses and save them, use the `sticky` keyword. Sticky addresses are added to the running configuration:

```
3750(config-if)# switchport port-security mac-address sticky
```

In order for the addresses to be retained, you must copy the running configuration to the startup configuration (or use the command `write memory`) before a reboot.

When you have a port configured with port security, and a packet arrives that is outside the scope of your configured limits, a violation is considered to have occurred. There are three actions the switch can perform in the event of a port-security violation:

`protect`
: When a violation occurs, the switch will drop any packets from MAC addresses that do not meet the configured requirements. No notification is given of this occurrence.

`restrict`
: When a violation occurs, the switch will drop any packets from MAC addresses that do not meet the configured requirements. An SNMP trap is generated, the log is appended, and the violation counter is incremented.

`shutdown`
: When a violation occurs, the switch will put the port into the *error-disabled* state. This action stops all traffic from entering and exiting the port. This action is the default behavior for port-security-enabled ports. To recover from this condition, reset the interface using either the `shutdown` and `no shutdown` commands, or the `errdisable recovery cause psecure-violation` command.

To change the port-security violation behavior, use the `switchport port-security violation` command:

```
3750(config-if)# switchport port-security violation ?
  protect   Security violation protect mode
  restrict  Security violation restrict mode
  shutdown  Security violation shutdown mode
```

Secure MAC addresses can be aged out based on either an absolute time, or the time for which the addresses have been inactive. The latter option can be useful in dynamic environments, where there may be many devices connecting and disconnecting repeatedly. Say you have a room full of consultants. The first three to connect in the morning get to use the network, and the rest are out of luck. If one of the original three leaves early, you may want to free up his spot to allow someone else to use the network. Alternately, you may wish to ensure that only the first three consultants to connect can use the network for the entire day, regardless of what time they leave. This would penalize the rest of the consultants for being late. I've worked with execs who would love to be able to implement such a design to get consultants to come in early! The type of aging employed is configured with the `switchport port-security aging type` command:

```
3750(config-if)# switchport port-security aging type ?
  absolute    Absolute aging (default)
  inactivity  Aging based on inactivity time period
```

The aging time is set in minutes with the `time` keyword:

```
3750(config-if)# switchport port-security aging time ?
  <1-1440>  Aging time in minutes. Enter a value between 1 and 1440
3750(config-if)# switchport port-security aging time 30
```

To see the status of port security, use the `show port-security` command. This command shows a nice summary of all ports on which port security is enabled, how many addresses are configured for them, how many have been discovered, and how many violations have occurred:

```
3750# sho port-security
Secure Port  MaxSecureAddr  CurrentAddr  SecurityViolation  Security Action
               (Count)       (Count)          (Count)
---------------------------------------------------------------------------
   Gi1/0/20              2            2                  0         Shutdown
---------------------------------------------------------------------------
Total Addresses in System (excluding one mac per port)     : 1
Max Addresses limit in System (excluding one mac per port) : 6272
```

For more detail, use the `show port-security interface` command for a specific interface:

```
3750# sho port-security interface g1/0/20
Port Security              : Enabled
Port Status                : Secure-down
Violation Mode             : Shutdown
Aging Time                 : 0 mins
Aging Type                 : Absolute
SecureStatic Address Aging : Disabled
Maximum MAC Addresses      : 2
Total MAC Addresses        : 2
Configured MAC Addresses   : 2
Sticky MAC Addresses       : 0
Last Source Address:Vlan   : 0000.0000.0000:0
Security Violation Count   : 0
```

SPAN

Switched Port Analyzer (SPAN) is a feature that allows traffic to be replicated to a port from a specified source. The traffic to be replicated can be from physical ports, virtual ports, or VLANs, but you cannot mix source types within a single SPAN session. The most common reason for SPAN to be employed is for packet capture. If you need to capture the traffic on VLAN 10, for example, you can't just plug a sniffer on a port in that VLAN, as the switch will only forward packets destined for the sniffer. However, enabling SPAN with the VLAN as the source, and the sniffer's port as the destination, will cause all traffic on the VLAN to be sent to the sniffer. SPAN is also commonly deployed when Intrusion Detection Systems (IDSs) are added to a network. IDS devices need to read all packets in one or more VLANs, and SPAN can be used to get the packets to the IDS devices.

Using *Remote Switched Port Analyzer* (RSPAN), you can even send packets to another switch. RSPAN can be useful in data centers where a packet-capture device is permanently installed on one of many interconnected switches. With RSPAN, you can capture packets on switches other than the one with the sniffer attached. (RSPAN configuration details are provided later in this section.)

SPAN is configured with the `monitor` command. You can have more than one SPAN session, each identified with a session number:

```
3750(config)# monitor session 1 ?
  destination  SPAN destination interface or VLAN
  filter       SPAN filter
  source       SPAN source interface, VLAN
```

Having more than one SPAN session is useful when you have an IDS device on your network and you need to do a packet capture. The IDS device will require one SPAN session, while the packet capture will use another.

For a monitor session to be active, you must configure a source port or VLAN and a destination port. Usually, I configure the destination port first because the packet-capture device is already attached. Note that if you have port security set, you must disable it before you can use the port as a SPAN destination:

```
3750(config)# monitor session 1 destination interface g1/0/20
%Secure port can not be dst span port
```

Sessions can be numbered from 1–66, but you can only have two sessions configured at any given time on a 3750 switch. Here, I have two sessions configured (session 1 and session 10):

```
monitor session 1 source vlan 20 rx
monitor session 1 destination interface Gi1/0/10
!
monitor session 10 source vlan 10 rx
monitor session 10 destination interface Gi1/0/20
```

If you try to configure more than two SPAN sessions on a 3750 switch, you will get the following error:

```
3750(config)# monitor session 20 source int g1/0/10
% Platform can support a maximum of 2 source sessions
```

In this example, I've configured two VLANs to be the sources, both of which will have their packets reflected to interface Gi1/0/20:

```
monitor session 10 source vlan 20 rx
monitor session 10 source vlan 10
monitor session 10 destination interface Gi1/0/20
```

You can also monitor one or more interfaces. Multiple interfaces can be configured separately, or on a single configuration line:

```
3750(config)# monitor session 11 source interface g1/0/11
3750(config)# monitor session 11 source interface g1/0/12
```

Entering the two preceding commands results in the following line being added to the configuration:

```
monitor session 11 source interface Gi1/0/11 - 12
```

The sources in a monitor session can be configured as either receive (rx), transmit (tx), or both. The default is both:

```
3750(config)# monitor session 1 source int g1/0/12 ?
  ,     Specify another range of interfaces
  -     Specify a range of interfaces
  both  Monitor received and transmitted traffic
  rx    Monitor received traffic only
  tx    Monitor transmitted traffic only
  <cr>
```

Interfaces should usually be monitored in both directions, while VLANs should be monitored in only one direction.

When capturing VLAN information, be careful if you see double packets. Remember that each packet will come into the VLAN on one port and exit on another. Using the default behavior of `both` when monitoring a VLAN will result in almost every packet being duplicated in your packet capture.

I can't tell you how many times I've been convinced that I'd stumbled onto some rogue device duplicating packets on the network, only to realize that I'd once again burned myself by monitoring a VLAN in both directions. The safest thing to do when monitoring VLANs is to monitor them only in the `rx` direction. Because the default is `both`, I like to think that I'm a victim of some inside joke at Cisco as opposed to being a complete idiot.

To see what SPAN sessions are configured or active, use the `show monitor` command:

```
3750# sho monitor
Session 1
---------
Type              : Local Session
Source VLANs      :
    RX Only       : 20
Destination Ports : Gi1/0/22
    Encapsulation : Native
          Ingress : Disabled


Session 10
----------
Type              : Local Session
Source VLANs      :
    TX Only       : 20
    Both          : 10
Destination Ports : Gi1/0/20
    Encapsulation : Native
          Ingress : Disabled
```

To disable monitoring on a specific SPAN, you can delete the entire monitor session, remove all the sources, or remove the destination. All `monitor` commands can be negated:

```
3750(config)# no monitor session 11 source interface Gi1/0/11 - 12
```

You can remove all local SPAN, all RSPAN, or all SPAN sessions as a group by adding the `local`, `remote`, or `all` keywords:

```
3750(config)# no monitor session ?
  <1-66>  SPAN session number
  all     Remove all SPAN sessions in the box
  local   Remove Local SPAN sessions in the box
  remote  Remove Remote SPAN sessions in the box
```

You should always remove your SPAN sessions when you no longer need them. SPAN takes up system resources, and confusion can be caused if someone plugs a device into the SPAN destination port.

RSPAN works the same way that SPAN does, with the exception that the destination interface is on another switch. The switches must be connected with an *RSPAN VLAN*. To create an RSPAN VLAN, configure a VLAN, and add the `remote-span` command:

```
3750-1(config)# vlan 777
3750-1(config-vlan)# remote-span
```

If you're running VTP, you may not need to create the VLAN, but you will still need to configure it for RSPAN. In either case, the steps are the same. On the source switch, specify the destination as the RSPAN VLAN:

```
3750-1(config)# monitor session 11 destination remote vlan 777
```

You can enter a destination VLAN that has not been configured as an RSPAN VLAN, but alas, it won't work.

Now, on the destination switch, configure the same VLAN as an RSPAN VLAN. Once you've done that, configure a monitor session to receive the RSPAN being sent from the source switch:

```
3750-2(config)# vlan 777
3750-2(config-vlan)# remote-span
3750-2(config)# monitor session 11 source remote vlan 777
```

There is no requirement for the monitor session numbers to be the same, but as I like to say, *simple is good*. If you have not configured the source switch to be the RSPAN source, you will get an error:

```
3750-2(config)# monitor session 11 source remote vlan 777
% Cannot add RSPAN VLAN as source for SPAN session 11 as it is not a RSPAN
Destination session
```

When using RSPAN, don't use an existing trunk for your RSPAN VLAN. SPAN can create a large amount of traffic. When monitoring VLANs composed of multiple gigabit interfaces, the SPAN traffic can easily overwhelm a single gigabit RSPAN link. Whenever possible, set up a dedicated RSPAN VLAN link between the switches.

Voice VLAN

Voice VLAN is a feature that allows the 3750 to configure a Cisco IP phone that's connected to the switch. The switch uses CDP to transfer to the phone configuration information regarding Class of Service (CoS), and the VLANs to be used for voice and data traffic. By default, this feature is disabled, which results in the phone not receiving configuration instructions from the switch. In this case, the phone will send voice and data over the default VLAN (VLAN 0 on the phone).

Cisco IP phones such as the model 7960 have built-in three-port switches. Port 1 on the built-in switch is the connection to the upstream switch (the 3750 we'll configure here). Port 2 is the internal connection to the phone itself. Port 3 is the external port, which usually connects to the user's PC.

By using the `switchport voice vlan` interface command, you can have the switch configure an IP phone that is connected to the interface being configured. You can specify a VLAN for voice calls originating from the phone, or you can have the switch tell the phone to use the regular data VLAN for voice calls (with or without setting CoS values):

```
3750(config-if)# switchport voice vlan ?
  <1-4094>  Vlan for voice traffic
  dot1p     Priority tagged on PVID
  none      Don't tell telephone about voice vlan
  untagged  Untagged on PVID
```

To set the VLAN, specify a VLAN number. The `dot1p` option tells the phone to set CoS bits in voice packets while using the data VLAN. The `untagged` option tells the phone to use the data VLAN without setting any CoS values.

To take advantage of Voice VLANs, you need to tell the switch to trust the CoS values being sent by the phone. This is done with the `mls qos trust cos` interface command.

The `mls qos trust cos` interface command will not take effect unless you globally enable QoS with the `mls qos` command.

Here is a sample interface configured to use VLAN 100 for data and VLAN 10 for voice. The switch will instruct the IP phone to use VLAN 10 for voice, and will trust the CoS values as set by the phone:

```
interface GigabitEthernet1/0/20
 switchport access vlan 100
 switchport voice vlan 10
 mls qos trust cos
```

Another nice aspect of the Voice VLAN feature is that you can have the IP phone alter or trust any CoS values set by the device plugged into its external switch port (usually, the user's PC). This feature is configured with the `switchport priority extend` interface command. The options are `cos` and `trust`. When using the `cos` option, you may set the CoS field to whatever CoS value you like:

```
3750(config-if)# switchport priority extend ?
  cos    Override 802.1p priority of devices on appliance
  trust  Trust 802.1p priorities of devices on appliance
```

I prefer to trust the PC's CoS values, as different software on the PC may have different values. For example, the user may wish to run a soft-phone application on the PC. Overriding the CoS values set by this software might lead to voice quality issues for the soft phone.

Here, I've configured an interface to use VLAN 10 for voice while trusting the CoS values set by the user's PC and phone:

```
interface GigabitEthernet1/0/20
 switchport access vlan 100
 switchport voice vlan 10
 switchport priority extend trust
 mls qos trust cos
```

To see which VLAN is configured as the Voice VLAN, use the `show interface` *`interface-name`* `switchport` command:

```
3750# sho int g1/0/20 switchport
Name: Gi1/0/20
Switchport: Enabled
Administrative Mode: static access
Operational Mode: down
Administrative Trunking Encapsulation: negotiate
Negotiation of Trunking: Off
Access Mode VLAN: 1 (default)
Trunking Native Mode VLAN: 1 (default)
Administrative Native VLAN tagging: enabled
Voice VLAN: 10 (Inactive)
Administrative private-vlan host-association: none
Administrative private-vlan mapping: none
Administrative private-vlan trunk native VLAN: none
Administrative private-vlan trunk Native VLAN tagging: enabled
Administrative private-vlan trunk encapsulation: dot1q
Administrative private-vlan trunk normal VLANs: none
Administrative private-vlan trunk private VLANs: none
Operational private-vlan: none
Trunking VLANs Enabled: ALL
Pruning VLANs Enabled: 2-1001
Capture Mode Disabled
Capture VLANs Allowed: ALL

Protected: false
Unknown unicast blocked: disabled
Unknown multicast blocked: disabled
Appliance trust: none
```

QoS

Quality of Service is covered in detail in Chapter 29, and it's a topic that could easily fill an entire book. In this section, I will focus on some 3750-specific QoS features.

One of the useful features of the 3750 is the ability to enable *AutoQoS*, which makes certain assumptions about your network, and configures the switch accordingly. While I'm not a fan of letting network devices assume anything, in this case, the assumptions are accurate most of the time. I have had no qualms about enabling AutoQoS on the 3750s I've installed in VoIP networks with hundreds of phones supported by Cisco Call Manager. The reason I'm OK with this is that Cisco's assumptions are built around the idea that you're using Call Manager, Cisco IP phones, and low-latency queuing on your network. Chances are, if you need QoS enabled on your switches, it's because you're implementing VoIP.

AutoQoS can be enabled on an interface with the `auto qos voip` command:

```
3750(config-if)# auto qos voip ?
  cisco-phone      Trust the QoS marking of Cisco IP Phone
  cisco-softphone  Trust the QoS marking of Cisco IP SoftPhone
  trust            Trust the DSCP/CoS marking
```

There are three options: `cisco-phone`, `cisco-softphone`, and `trust`. The first two are used for interfaces connected to either hard or soft phones. When configured with these options, the QoS values received in packets will be trusted only if they're sourced from Cisco IP phones. The `trust` option is used to enable QoS while trusting all packets' QoS values.

If you'd like to see what AutoQoS does, enable AutoQoS debugging with the `debug auto qos` command before you configure the interface:

```
3750# debug auto qos
3750# conf t
Enter configuration commands, one per line. End with CNTL/Z.
3750(config)# int g1/0/20
3750(config-if)# auto qos voip cisco-phone
3750(config-if)#
3d04h: mls qos map cos-dscp 0 8 16 26 32 46 48 56
3d04h: mls qos
3d04h: no mls qos srr-queue input cos-map
3d04h: no mls qos srr-queue output cos-map
3d04h: mls qos srr-queue input cos-map queue 1 threshold 3  0
3d04h: mls qos srr-queue input cos-map queue 1 threshold 2  1
3d04h: mls qos srr-queue input cos-map queue 2 threshold 1  2
3d04h: mls qos srr-queue input cos-map queue 2 threshold 2  4 6 7
```

```
3d04h: mls qos srr-queue input cos-map queue 2 threshold 3  3 5
3d04h: mls qos srr-queue output cos-map queue 1 threshold 3  5
3d04h: mls qos srr-queue output cos-map queue 2 threshold 3  3 6 7
3d04h: mls qos srr-queue output cos-map queue 3 threshold 3  2 4
3d04h: mls qos srr-queue output cos-map queue 4 threshold 2  1
3d04h: mls qos srr-queue output cos-map queue 4 threshold 3  0
[-Lots of text removed-]
```

The interface's configuration will look as follows:

```
interface GigabitEthernet1/0/20
 srr-queue bandwidth share 10 10 60 20
 srr-queue bandwidth shape  10  0  0  0
 queue-set 2
 auto qos voip cisco-phone
```

The changes to the switch's global configuration are a bit more extensive. Thankfully, AutoQoS does all the work for you:

```
mls qos map cos-dscp 0 8 16 26 32 46 48 56
mls qos srr-queue input bandwidth 90 10
mls qos srr-queue input threshold 1 8 16
mls qos srr-queue input threshold 2 34 66
mls qos srr-queue input buffers 67 33
mls qos srr-queue input cos-map queue 1 threshold 2  1
mls qos srr-queue input cos-map queue 1 threshold 3  0
mls qos srr-queue input cos-map queue 2 threshold 1  2
mls qos srr-queue input cos-map queue 2 threshold 2  4 6 7
mls qos srr-queue input cos-map queue 2 threshold 3  3 5
mls qos srr-queue input dscp-map queue 1 threshold 2  9 10 11 12 13 14 15
mls qos srr-queue input dscp-map queue 1 threshold 3  0 1 2 3 4 5 6 7
mls qos srr-queue input dscp-map queue 1 threshold 3  32
mls qos srr-queue input dscp-map queue 2 threshold 1  16 17 18 19 20 21 22 23
mls qos srr-queue input dscp-map queue 2 threshold 2  33 34 35 36 37 38 39 48
mls qos srr-queue input dscp-map queue 2 threshold 2  49 50 51 52 53 54 55 56
mls qos srr-queue input dscp-map queue 2 threshold 2  57 58 59 60 61 62 63
mls qos srr-queue input dscp-map queue 2 threshold 3  24 25 26 27 28 29 30 31
mls qos srr-queue input dscp-map queue 2 threshold 3  40 41 42 43 44 45 46 47
mls qos srr-queue output cos-map queue 1 threshold 3  5
mls qos srr-queue output cos-map queue 2 threshold 3  3 6 7
mls qos srr-queue output cos-map queue 3 threshold 3  2 4
mls qos srr-queue output cos-map queue 4 threshold 2  1
mls qos srr-queue output cos-map queue 4 threshold 3  0
mls qos srr-queue output dscp-map queue 1 threshold 3  40 41 42 43 44 45 46 47
mls qos srr-queue output dscp-map queue 2 threshold 3  24 25 26 27 28 29 30 31
mls qos srr-queue output dscp-map queue 2 threshold 3  48 49 50 51 52 53 54 55
mls qos srr-queue output dscp-map queue 2 threshold 3  56 57 58 59 60 61 62 63
mls qos srr-queue output dscp-map queue 3 threshold 3  16 17 18 19 20 21 22 23
mls qos srr-queue output dscp-map queue 3 threshold 3  32 33 34 35 36 37 38 39
mls qos srr-queue output dscp-map queue 4 threshold 1  8
mls qos srr-queue output dscp-map queue 4 threshold 2  9 10 11 12 13 14 15
mls qos srr-queue output dscp-map queue 4 threshold 3  0 1 2 3 4 5 6 7
mls qos queue-set output 1 threshold 1 138 138 92 138
mls qos queue-set output 1 threshold 2 138 138 92 400
mls qos queue-set output 1 threshold 3 36 77 100 318
```

```
mls qos queue-set output 1 threshold 4 20 50 67 400
mls qos queue-set output 2 threshold 1 149 149 100 149
mls qos queue-set output 2 threshold 2 118 118 100 235
mls qos queue-set output 2 threshold 3 41 68 100 272
mls qos queue-set output 2 threshold 4 42 72 100 242
mls qos queue-set output 1 buffers 10 10 26 54
mls qos queue-set output 2 buffers 16 6 17 61
mls qos
```

If you're looking at someone else's router, and you see all this stuff, resist the urge to think he's some sort of QoS genius. Chances are he just ran AutoQoS!

To see what interfaces AutoQoS is enabled on, use the `show auto qos` global command:

```
3750# show auto qos
GigabitEthernet1/0/20
auto qos voip cisco-phone
```

To disable AutoQoS on an interface, use the `no auto qos voip` interface command. To disable AutoQoS globally, use the `no mls qos` command. Beware that this disables *all* QoS on the switch.

PART IV
Telecom

This section covers telecom technologies as they pertain to the data-networking world. A general glossary is presented, followed by detailed information regarding T1s, DS3s, and frame relay.

This section is composed of the following chapters:

Chapter 19, *Telecom Nomenclature*
Chapter 20, *T1*
Chapter 21, *DS3*
Chapter 22, *Frame Relay*

CHAPTER 19

Telecom Nomenclature

Introduction and History

The telecom world is a bit different from the data world, as endless telecom engineers will no doubt tell you. For example, a lot of the telecom infrastructure that exists today is the way it is because of standards that have been in place for upwards of 100 years. Samuel Morse invented the telegraph in 1835. Alexander Graham Bell invented the telephone in 1876. In 1961, Bell Labs invented the T1 as a way to aggregate links between the central offices (COs) of the phone companies. It took almost 100 years to get from the first telephone to the invention of the T1.

In contrast, consider the data world: the Arpanet was started in 1969, Robert Metcalfe and David Boggs built the first Ethernet in 1973, and Vint Cerf and Bob Kahn published the original TCP/IP standard in 1974. Hayes introduced the first modem in 1977 (300 bps, baby!), and 3Com shipped the first 10 Mbps Ethernet card in 1981. The first commercial router was sold in 1983.

Let's think about that for a moment—the first commercial router was sold in 1983. Ask anyone around you if she can remember a time when there weren't phones.

The telecom world is built on standards that work and have worked for a very long time. How often does your phone stop working? The telecom infrastructure is so reliable that we expect reliable phone service even more than we expect reliable electrical service. (Cellular service is a whole different ball game, and does not apply to this discussion.)

As with any technology, the engineers in the trenches (and their bosses behind the desks) like to sling the lingo. If you've spent your professional life around data equipment, telecom lingo might seem pretty foreign to you. To help bridge the gap between the data and telecom worlds, I've put together a list of terms that you might hear when dealing with telecom technologies.

Most telecom words and phrases have standard meanings defined in Federal Standard 1037C, titled "Telecommunications: Glossary of Telecommunication Terms." These definitions are often very simple, and don't go into a lot of detail. The terms I'll cover here are the ones most often encountered in the life of a network administrator or engineer. If you need to know what circuit noise voltage measured with a psophometer that includes a CCIF weighting network is referred to as, Federal Standard 1037C is a good place to look. Another excellent source that should be on the bookshelf of anyone in the telecom world is *Newton's Telecom Dictionary*, by Harry Newton (CMP Books).

The meanings of many widely used telecom terms have changed over time. Through regulation over the years, the functions of entities like IXCs and LECs (both defined below) have changed. I will cover the original intended meanings in this text.

Telecom Glossary

ACD

ACD stands for Automatic Call Distributor. An ACD is usually found in a call center, where calls may come in from anywhere, and need to be directed to the next available operator, or queued until one is available.

Add/Drop

The term Add/Drop is used in telecom to describe the capability of peeling off channels from a circuit and routing them elsewhere. An Add/Drop CSU/DSU has the ability to separate ranges of channels to discrete data ports, thus allowing a T1 to be split into two partial T1s. One could be used for voice, and the other for data, or both could be used for either function and routed differently. You can make an Add/Drop device function like a non-Add/Drop device simply by assigning all of the channels to a single data port. However, Add/Drop functionality adds cost to devices, so it should only be considered if the added functionality is required.

Analog and digital

Would you like to have some fun? Ask someone in the computer field to define the word "analog." You might be surprised at some of the answers you receive.

Analog means, literally, *the same*. When one item is analogous to another, it is the same as the other item. In the telecom and data worlds, "analog" refers to a signal that is continuous in amplitude and time.

An analog signal is not composed of discrete values: any small fluctuation of the signal is important. Radio waves are analog, as are power waves. Sound is also analog. When you speak, you create waves of air that hit people's eardrums. The sound waves are an analog signal.

Digital refers to a signal that has discrete values. If you analyze a sound wave, and then assign a value to each sample of the wave at specific time intervals, you will create a digital representation of the analog wave.

Because digital involves discrete values, and analog does not, converting analog signals to digital will always result in loss of information.

While increasing the rate at which the signal is sampled (among other things) can increase the quality of the final reproduction, technically, the signal cannot be reproduced exactly the same way.

Bandwidth

Bandwidth is one of those terms that's thrown around a lot by people who don't really know what it means. A range of frequencies is called a *band*. The width of the band is referred to as *bandwidth*. For those of you who aren't old enough to remember FM radios with analog dials, I've drawn one in Figure 19-1.

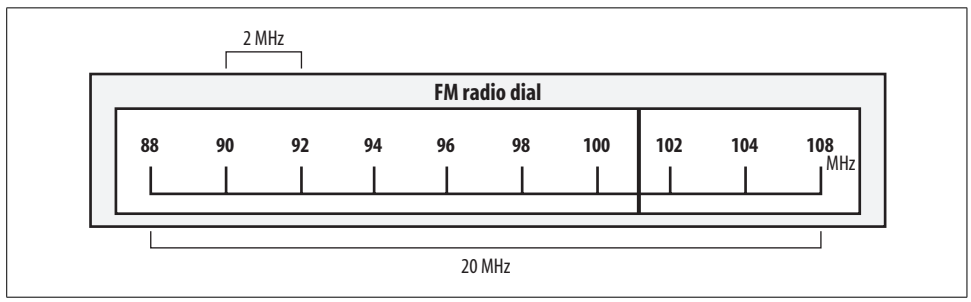


Figure 19-1. True bandwidth example

An FM radio dial displayed the range of frequencies allocated by the U.S. government for stereo radio broadcasts. The exact range of frequencies is 87.9 MHz to 107.9 MHz. The bandwidth of this range is 20 MHz. The frequency range of 90 MHz to 92 MHz inclusive has a bandwidth of 2 MHz.

What we're really referring to when we talk about bandwidth on digital links is the *throughput*. On a digital link, the number of possible state transitions per second is how we measure throughput.

Figure 19-2 shows how the number of state transitions can vary based on link speed. The signal on the left shows six possible state changes in the time depicted. Two concurrent equal values do not require a change of state, so only five state changes occurred, though six were possible. The signal on the right shows 19 possible state changes in the same amount of time (with 17 occurring). The signal on the right would be described as having more bits per second (bps) of throughput.

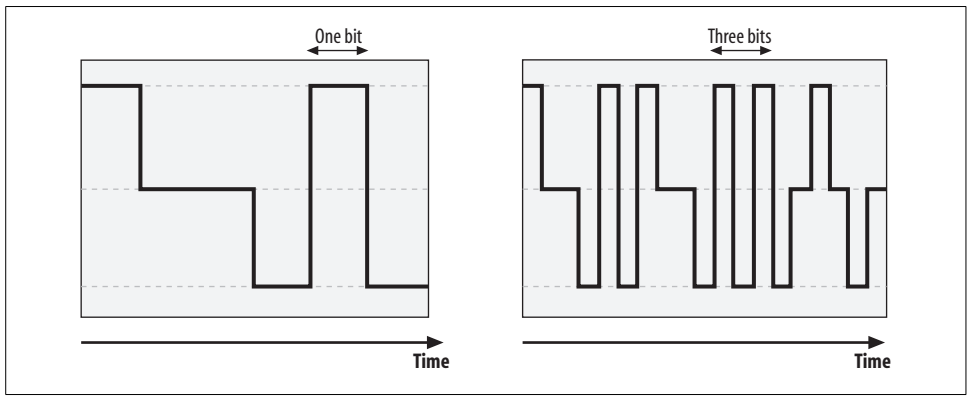


Figure 19-2. Digital state changes over time

When someone says that a DS3 has more bandwidth than a T1, what she's really saying is that the DS3 will deliver higher throughput, in that it is capable of 45 million bits per second (Mbps), as opposed to the T1's paltry (by comparison) 1.54 Mbps. In common usage, the terms bandwidth and throughput are interchangeable. A more accurate term when referring to links might be *data rate*, as a DS3 can deliver the same number of bits in a shorter amount of time than a T1.

BERT

BERT stands for *Bit Error Rate Test*. You'll often hear the term "BERT test," although this is technically redundant because the T in BERT stands for test. Still, saying, "We're going to run some BERTs on the T1" will make everyone look at you funny.

BERT tests are disruptive tests run on a link to validate the data integrity of the circuit. A BERT test is usually run by putting the remote end of the link in loopback and sending out a pattern of ones and zeros. How the data is returned from the far-end loopback can help determine whether certain types of problems exist

in the line. Some of the common types of BERT tests you may hear mentioned include QRSS (Quasi-Random Signal Source), 3 in 24, 1:7 or 1 in 8, Min/Max, All Ones, and All Zeros. Each of these patterns stresses a link in a certain predictable way. A device used to perform BERT testing on a T1 is called a *T-Berd*.

Central Office (CO)

The central office is where phone lines from residences or businesses physically terminate. COs have the necessary switching equipment to route calls locally, or to another carrier, as needed. When you make a call that is destined for somewhere other than your premises, the CO is the first stop.

With technology like T1, the copper connection from your location will probably terminate at a CO, where it may be aggregated into part of a larger SONET system.

Channel bank

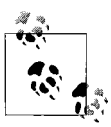
A channel bank is a device that separates a T1 into 24 individual analog phone lines, and vice versa. Today, PBXs usually take care of partitioning T1s into analog lines, thereby eliminating the need for a channel bank.

CSU/DSU

CSU stands for *Channel Service Unit*, and DSU stands for *Data Service Unit*. A CSU is responsible for interfacing with the WAN link, and a DSU is responsible for interfacing with data equipment such as routers. A CSU/DSU combines these functions into a single unit. Typically, an RJ-45-terminated cable will connect the demarc to the CSU/DSU, and a V.35 cable will connect the CSU/DSU to a router. The CSU/DSU is usually configurable to support all the common T1 signaling and framing options. Modern Cisco routers support WAN interface cards (WICs) that have integrated CSU/DSUs.

CPE

CPE is short for *customer premises equipment* (i.e., equipment that is located at the customer's premises). Examples might include a PBX, phones, routers, and even cable modems. Traditionally, the term was used to describe equipment owned by a telephone service provider that resided at customer premises, but it has evolved to include equipment owned by anyone.



Telecom engineers often shorten the word *premises* to *prem* when speaking.

DACCS

DACCS (pronounced *dacks*) stands for *Digital Access Cross-Connect System*. You may also see this as DAX, or DACS®, which is a registered trademark of AT&T.

A DACCS is a device that allows changes to the way voice channels are connected between trunks through the use of software.

Figure 19-3 shows a logical representation of a DACCS in use. T1-A connects to the DACCS, and has 18 of its 24 channels in use. Those channels need to be routed to three different places. With the DACCS, we can link the first six channels (1–6) of T1-A to channels 1–6 on T1-B, the next six channels (7–12) of T1-A to channels 1–6 on T1-C, and the next six channels (13–18) of T1-A to channels 1–6 on T1-D. The channels do not have to be grouped, and may be mapped between links in any way, provided there are available channels on the links.

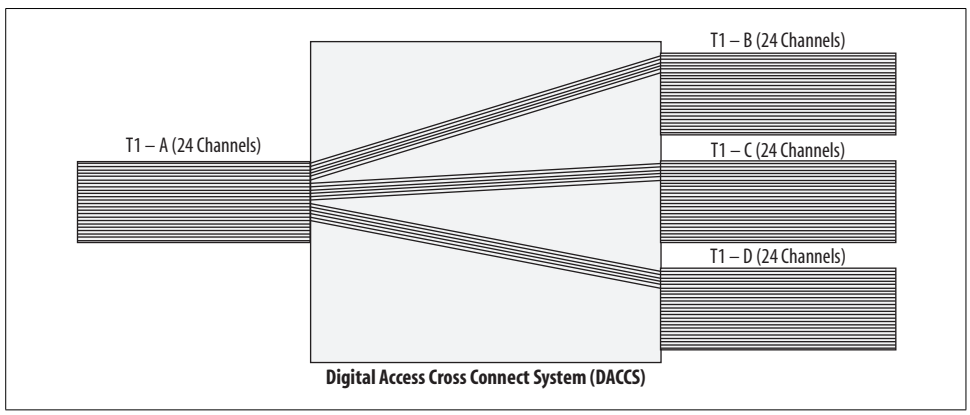


Figure 19-3. DACCS

Demarc

Demarc (pronounced *dee-mark*) is a slang abbreviation for *demarcation point*. The demarcation point is the point where the telecom provider's responsibilities end and yours begin. Demarcs are often telecom closets or similar locations that can be secured to allow access for the telecom provider's engineers.

Digital signal hierarchy

The digital signal (DS) hierarchy describes the signaling rates of trunk links. These links are the physical links on which logical T-carriers are placed.

The carrier numbers grow larger as the number of multiplexed DS0s increases. DS0 is the smallest designation, and is the rate required for a single phone line. The hierarchy is shown in Table 19-1.

Table 19-1. Digital signal hierarchy

Designator	Carrier	Transmission rate	Voice channels
DS0	N/A	64 Kbps	1
DS1	T1	1.544 Mbps	24
DS2	T2	6.312 Mbps	96
DS3	T3	44.736 Mbps	672
DS4	T4	274.176 Mbps	4,032

E-carrier

The E-carrier hierarchy is similar to the U.S. T-carrier hierarchy (described later), though the speeds are slightly different, as is the signaling. The European E-carrier hierarchy is shown in Table 19-2.

Table 19-2. European E-carrier hierarchy

Designator	Transmission rate	Voice channels
E0	64 Kbps	1
E1	2.048 Mbps	30
E2	8.448 Mbps	120
E3	34.368 Mbps	480
E4	139.268 Mbps	1,920
E5	565.148 Mbps	7,680

ISDN

ISDN stands for *Integrated Services Digital Network*. ISDN is a form of digital transmission for voice and data. Unlike normal POTS lines, or channelized T1 services—which use the voice path for signaling—ISDN uses a separate channel called the *data channel* for signaling, so the remaining channels (called *bearer channels*) can be used exclusively for voice. Because a separate channel is used for signaling, greater functionality is possible when using ISDN.

The bearer channels are sometimes referred to as *B-channels*, and the data channel is sometimes referred to as the *D-channel*.

One of the benefits of ISDN is that it can support normal voice calls and ISDN digital calls. In the early 1990s, ISDN was considered to be the next big thing: it was supposed to revolutionize phone and data service. There are two types of ISDN links:

BRI

BRI is short for *Basic Rate Interface*. A BRI is an ISDN link composed of two 64-Kbps bearer channels, and one 16 Kbps data channel.

PRI

PRI is short for *Primary Rate Interface*. A PRI is an ISDN T1 link composed of 23 64-Kbps bearer channels, and one 64-Kbps data channel. PRIs are used a lot when connecting PBX systems, or at ISPs for dial-up lines.

While PRI circuits are used today for voice, BRI circuits, which were commonly used for data, have been widely replaced by cheaper alternatives such as DSL.

IXC

IXC stands for *interexchange carrier*. An IXC is a telephone company that supplies connections between local exchanges provided by local exchange carriers. Connecting between LATAs may involve IXCs.

J-carrier

The J-carrier hierarchy is much closer to the U.S. T-carrier hierarchy (in terms of speed) than the European hierarchy, though the values change as the rates get faster. While J-carrier circuits may still be seen, most of the circuits I've worked on in Japan have actually been E1s or T1s.

The Japanese J-carrier hierarchy is shown in Table 19-3.

Table 19-3. Japanese J-carrier hierarchy

Designator	Transmission rate	Voice channels
J0	64 Kbps	1
J1	1.544 Mbps	24
J2	6.312 Mbps	96
J3	32.064 Mbps	480
J4	397.200 Mbps	5,760

LATA

LATA (pronounced *lat-ah*) is short for *local access and transport area*. LATAs are geographically defined areas in which a telecom provider can provide local service. The Regional Bell Operating Companies (RBOCs), for example, were usually not permitted to provide services between LATAs (inter-LATA), but could provide services within a LATA (intra-LATA).

LATAs come into play when point-to-point circuits like T1s are ordered. When a T1 starts and ends within the same LATA, the cost for the circuit is usually much lower than if the circuit starts in one LATA and ends in another. This is because IXCs must be involved to connect LATAs. To further complicate things, LATAs are geographic, and often do not mirror political boundaries such as county or state lines.

Latency

Latency is the term used to describe the amount of time it takes for data to be processed or moved. Latency has nothing to do with the throughput, bandwidth, or speed of a link. Latency has to do with distance, the speed of light, and the amount of time it takes for hardware to process data.

Latency on networks links is a combination of propagation delay and processing delay:

Propagation delay

Figure 19-4 shows three locations: New York, Cleveland, and Los Angeles. There are two links: one T1 between New York and Cleveland, and one T1 between New York and Los Angeles. Both links have the same speed (1.54 Mbps), but it takes longer for packets to get from New York to Los Angeles than it does for them to get from New York to Cleveland.

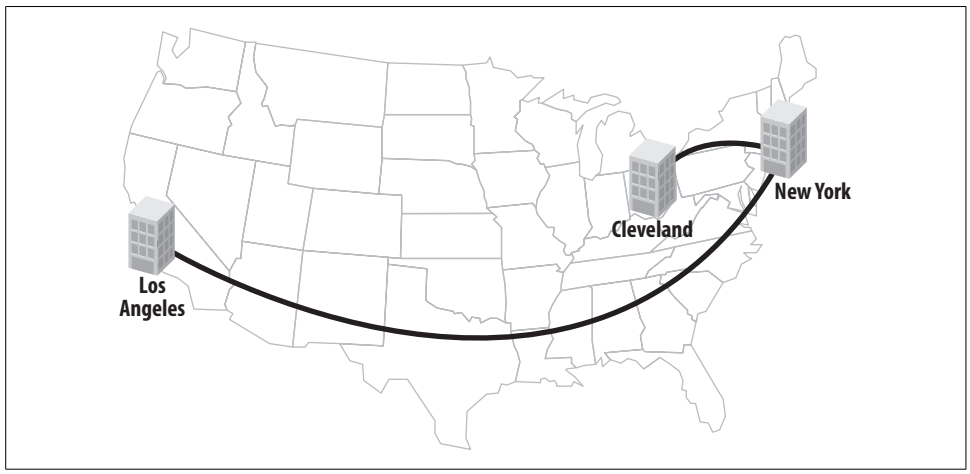


Figure 19-4. Different propagation delays

The discrepancy occurs because Los Angeles is a lot farther away from New York than Cleveland is. This form of latency is called *propagation delay*. Propagation delay is, to a large degree, a function of physics, and as such cannot be fixed, improved, or otherwise changed (no matter what your boss may want). To oversimplify, the speed at which electrons can transmit electrical impulses is limited. The speed at which photons can move in fiber is similarly limited.

Processing delay

Another form of latency is called *processing delay*, which is the time it takes for a device to process information. In contrast to propagation delay, which usually cannot be changed, processing delay is a function of the speed of the equipment in use.

Figure 19-5 shows two links: the top link is a direct connection between two modern Cisco 7609 routers, involving only the latest hardware; the bottom link connects the same two routers, but with a very old Cisco 2501 router in the middle.

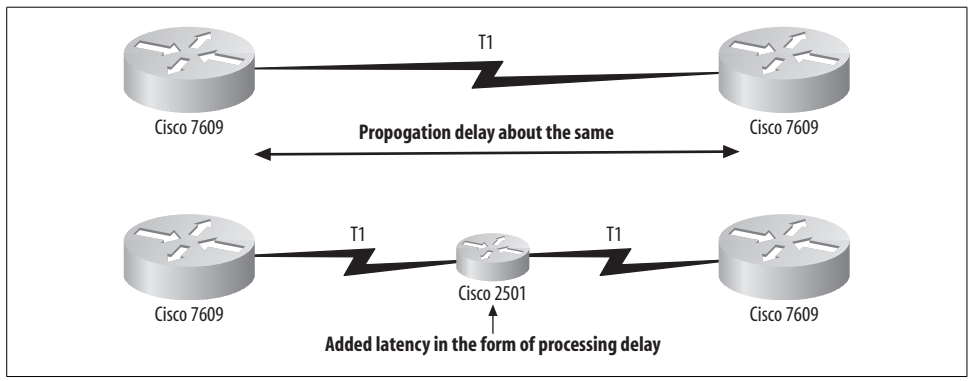


Figure 19-5. Processing delay

Although the total distance between the two Cisco 7609s is the same from point to point in both cases, adding a Cisco 2501 in the middle of the second link increases the processing delay dramatically.

Another example of increasing processing delay occurs when using multilink-PPP. Taking three 1.54 Mbps T1s and bonding them to form one logical 4.5 Mbps link sounds great, and it can be, but the added processing delay when you do so can be enormous.

As an example, notice the delay parameter in this `show interface` output from a T1 interface:

```
Router# sho int s0/1
Serial0/1 is administratively down, line protocol is down
  Hardware is QUICC with integrated T1 CSU/DSU
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation HDLC, loopback not set
  Keepalive set (10 sec)
  Last input never, output never, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
```

```
Queueing strategy: weighted fair
Output queue: 0/1000/64/0 (size/max total/threshold/drops)
   Conversations  0/0/256 (active/max active/max total)
   Reserved Conversations 0/0 (allocated/max allocated)
   Available Bandwidth 1158 kilobits/sec
5 minute input rate 0 bits/sec, 0 packets/sec
5 minute output rate 0 bits/sec, 0 packets/sec
   0 packets input, 0 bytes, 0 no buffer
   Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
   0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
   0 packets output, 0 bytes, 0 underruns
   0 output errors, 0 collisions, 0 interface resets
   0 output buffer failures, 0 output buffers swapped out
   0 carrier transitions
   DCD=down  DSR=up  DTR=down  RTS=down  CTS=down
```

Compare that with the output from the same `show interface` command for a multilink interface:

```
Router# sho int multilink 1
Multilink1 is down, line protocol is down
  Hardware is multilink group interface
  MTU 1500 bytes, BW 100000 Kbit, DLY 100000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation PPP, loopback not set
  Keepalive set (10 sec)
  DTR is pulsed for 2 seconds on reset
  LCP Closed, multilink Closed
  Closed: LEXCP, DECCP, OSICP, BRIDGECP, VINESCP, XNSCP, TAGCP, IPCP, CCP
          CDPCP, LLC2, ATCP, IPXCP, NBFCP, BACP
  Last input never, output never, output hang never
  Last clearing of "show interface" counters 00:00:07
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: fifo
  Output queue: 0/40 (size/max)
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     0 packets input, 0 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     0 packets output, 0 bytes, 0 underruns
     0 output errors, 0 collisions, 1 interface resets
     0 output buffer failures, 0 output buffers swapped out
```

The delay for a multilink interface is five times that of a serial T1 interface.

The bandwidth and delay values shown for an interface are representative of the actual bandwidth and delay, provided they have not been modified. The bandwidth and delay values are configurable in IOS. The default values reflect the propagation delay well enough to illustrate the impact of multilink-PPP.

LEC

LEC (pronounced *leck*) is short for *local exchange carrier*. A LEC is a phone company that provides local service, as opposed to an IXC, which interconnects LECs to provide long-distance service. Most of the largest LECs are RBOCs.

Local loop

The local loop (also referred to as the *last mile*) is the copper handoff for a circuit from the telecom facility to your facility. While a T1 may be converted and multiplexed into a larger circuit like a DS3 or SONET circuit, the last mile is usually copper.

Multiplexing

Multiplexing is the act of taking multiple signals and sharing them on a single signal. The act of converting 24 64-Kbps channels into a single T1 is an example of multiplexing.

PBX

PBX is the abbreviation for *private branch exchange*. A PBX is essentially a phone system as most people know it; it offers a phone network to an entity such as an enterprise. Some of the main features of a PBX are the ability for many phones to share a limited number of public phone lines, and the ability to number individual extensions with, typically, three- or four-digit extension numbers. PBX systems have traditionally been large hardware devices with cryptic control systems and proprietary hardware. VoIP is often controlled by software versions of PBXs. Examples include Cisco's Call Manager and the open source product Asterisk.

POTS

POTS is short for the clever phrase *plain-old telephone service*. A POTS line is one into which you can plug in a normal analog phone or fax machine. Most home phone lines are POTS lines.

RBOC

RBOC (pronounced *are-bock*) is short for *Regional Bell Operating Company*.

In 1875, Alexander Graham Bell (and two others who agreed to finance his inventions) started the Bell Telephone Company. Bell Telephone later became known as the Bell System as it acquired controlling interests in other companies such as Western Electric. In 1885, American Telephone and Telegraph (AT&T) was incorporated to build and operate the U.S.'s original long-distance telephone network. In 1899, AT&T acquired Bell. For almost 100 years, AT&T and its Bell System operated as a legally sanctioned, though regulated, monopoly, building what was by all accounts the best telephone system in the world.

In 1984, however, a judge—citing antitrust monopoly issues—broke up AT&T's Bell System, known as Ma Bell. The resulting seven companies were known as the RBOCs, or, more commonly, the *Baby Bells*. AT&T remained a long-distance carrier, or IXC, during the divestiture. However, the Telecommunications

Deregulation Act of 1996 allowed RBOCs (also called LECs) and long-distance companies to sell local, long-distance, and international services, making these lines very fuzzy.

Each of the original seven companies was given a region in which it was allowed to do business. The regions are shown in Figure 19-6.

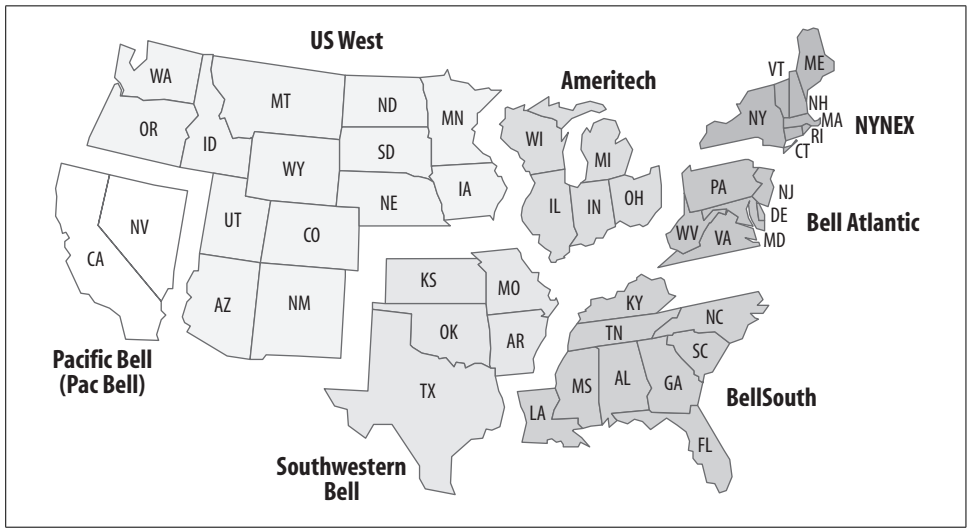


Figure 19-6. RBOC regions

Most of the original RBOCs are now part of SBC, which also acquired AT&T, bringing the entire dizzying affair one step closer to a monopoly again. Here's what's become of the seven original RBOCs:

Bell Atlantic
: Bell Atlantic merged with NYNEX in 1996 to become Verizon.

Southwestern Bell
: Southwestern Bell changed its name to SBC in 1995. SBC acquired PacBell in 1996, and has subsequently acquired Ameritech, Southern New England Telecommunications, and AT&T. SBC adopted the name AT&T following the acquisition of that company.

NYNEX
: NYNEX merged with Bell Atlantic in 1996 to become Verizon.

Pacific Bell (PacBell)
: PacBell was acquired by SBC in 1996.

BellSouth
: BellSouth has been acquired by SBC/AT&T.

Ameritech
: Ameritech was acquired by SBC in 1998.

US West
: US West was acquired by Qwest Communications in 2000.

Smart jack
: A smart jack is a device that terminates a digital circuit. It is considered "smart" because the phone company can control it remotely. Smart jacks also offer test points for equipment such as BERT testers. T1s are usually terminated at smart jacks. Larger installations have racks of smart jacks called smart jack racks.

SONET
: SONET (pronounced like *bonnet*) is short for *synchronous optical network*. SONET is an ANSI standard for fiber-optic transmission systems. The equivalent European standard is called the synchronous digital hierarchy (SDH).

  SONET is strictly optical, as its name suggests, and is very fast. SONET defines certain *optical carrier* levels, as shown in Table 19-4.

Table 19-4. Optical carrier levels

Optical carrier level	Line rate	Payload rate
OC1	51 Mbps	50 Mbps
OC3	155 Mbps	150 Mbps
OC12	622 Mbps	601 Mbps
OC48	2,488 Mbps	2,405 Mbps
OC192	9,953 Mbps	9,621 Mbps
OC768	39,813 Mbps	38,486 Mbps

T-carrier
: T-carrier is the generic name for digital multiplexed carrier systems. The letter T stands for *trunk*, as these links were originally designed to trunk multiple phone lines between central offices. The T-carrier hierarchy is used in the U.S. and Canada. Europe uses a similar scale called the European E-carrier hierarchy, and Japan uses a system titled the Japanese J-carrier hierarchy. The North American T-carrier hierarchy is shown in Table 19-5.

Table 19-5. North American T-carrier hierarchy

Designator	Transmission rate	Voice channels
T1	1.544 Mbps	24
T1C	3.152 Mbps	48
T2	6.312 Mbps	96
T3	44.736 Mbps	672
T4	274.176 Mbps	4,032

T-Berd

A T-Berd is a *T1 Bit Error Rate Detector*. The generic term is used for any device that will perform BERT tests on a T1. If you have a T1 that's misbehaving, the provider will probably send out an engineer with a T-Berd to perform invasive testing.

TDM

TDM stands for *time-division multiplexing*. A T1 link is a TDM link because its 24 channels are divided into time slots. A T1 link is a serial link, so one bit is sent at a time. The channels are cycled through at a high rate of speed, with each channel being dedicated to a slice of time.

CHAPTER 20
T1

In the 1950s, the only method for connecting phone lines was with a pair of copper wires. For each phone line entering a building, there had to be a pair of copper wires. Wire congestion was a huge problem in central offices and under streets in metropolitan areas at the time. Imagine the central office of a large city, where tens of thousands of phone lines terminated, each requiring a pair of wires. These COs also needed to communicate with each other, which required even more wiring.

In 1961, Bell Labs in New Jersey invented the T1 as a means for digitally trunking multiple voice channels together between locations. The T1 delivered a 12:1 factor of relief from the congestion, as it could replace 24 two-wire phone lines with four wires. Back then, this was a major shift in thinking. Remember that at the time, digital technology was practically nonexistent. The first T1 went into service in 1962, linking Illinois Bell's Dearborn office in Chicago with Skokie, Illinois. Today, you would be hard-pressed to find a company that doesn't deploy multiple T1s.

In this chapter, I will go into detail about the design, function, and troubleshooting of T1s. While I usually try to simplify complex engineering topics, I feel that it's important to understand the principles of T1 operation. We live in a connected world, and much of that world is connected with T1s. Knowing how they work can save you countless hours of troubleshooting time when they break.

Understanding T1 Duplex

A full-duplex link can send and receive data at the same time. As anyone who's ever had a fight with his or her significant other over the phone can attest, both parties on a call can talk (or scream) at the same time. This is a full-duplex conversation. If only one person could talk at a time, the conversation would be half duplex. Using walkie-talkies where you have to push a button to talk is an example of a half-duplex conversation. While the button is depressed, typically, you cannot hear the person with whom you are speaking (although some high-end walkie-talkies can use one frequency for transmitting, and another for receiving, thereby allowing full-duplex conversations).

T1s are full-duplex links. Voice T1s transmit and receive audio simultaneously, and data is sent and received simultaneously on WAN-link T1s. Still, I've met many people in the field who don't understand T1 duplex. This may have a lot to do with the way data flow across WAN links is commonly reported.

Figure 20-1 shows a T1 Internet link's bandwidth utilization as monitored by the Multi Router Traffic Grapher (MRTG). The numbers on the bottom of the graph are the hours of the day, with 0 being midnight, 2 being 2:00 a.m., and so on. The solid graph, which looks like an Arizona landscape, is the inbound data flow. The solid line in the foreground is the outbound data flow. This is a typical usage pattern for a heavily used T1 Internet link. The bulk of the data comes from the Internet. The requests to get that data are very small. At about 8:45 a.m., there's a spike in the outbound traffic—perhaps a large email was sent at that time.

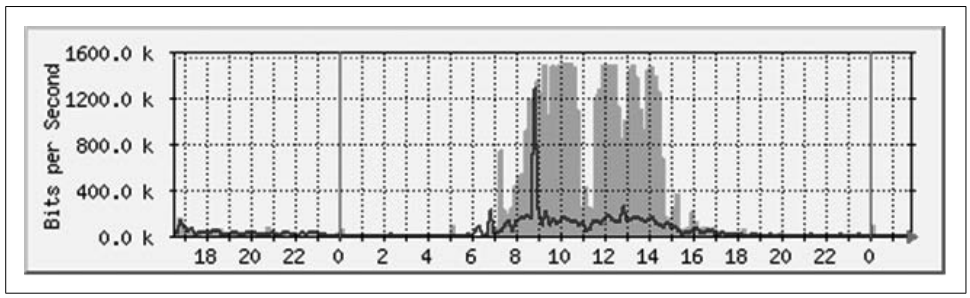


Figure 20-1. MRTG usage graph

The graph does not make obvious the duplex mode of the link. One could easily assume that the T1 was half duplex and switched from transmit to receive very quickly. If someone who didn't understand the technology only saw graphs like this one, he might conclude that a T1 can only send data in one direction at a time. You'd be surprised how common this misconception is.

Types of T1

Terminology is important when discussing any technology, and T1 is no exception. Many terms are commonly misused, even by people who have been in the industry for years. The terms *T1* and *DS1* are often thrown around interchangeably, although doing this can get you into trouble if you're talking with people who have a long history in telecommunications. You may also hear some people refer to a Primary Rate Interface (PRI) as a "digital T1," which is not strictly correct. All T1s are digital. The difference with PRI is that it uses digital signaling within the data channel as opposed to analog signaling within each voice channel. Even with an "analog" T1, each channel's audio must be converted to digital to be sent over the T1.

You may encounter a lot of conflicting information when learning about T1s. While there is a lot to learn, there are only a few basic types of T1s:

Channelized T1
: A channelized T1 is a voice circuit that has 24 voice channels. Each channel contains its own signaling information, which is inserted into the data stream of the digitized voice. This is called *in-band signaling*. Provided the circuit has been provisioned correctly (see the upcoming "Encoding" and "Framing" sections), with the use of an Add/Drop CSU/DSU, a channelized T1 can be used for data.

PRI
: A Primary Rate Interface is a voice circuit that has 24 channels, one of which is dedicated to signaling. Thus, the number of available voice channels is 23. The voice channels are called *bearer channels*,and the signaling channel is called the *data channel*. This type of signaling is called *out-of-band signaling*.

Clear-channel T1
: A clear-channel T1 is one that is not framed in any way. There are no channels, and no organization of the bits flowing through the link. Clear-channel T1s are actually a rarity, as most data links are provisioned with ESF framing.

You can think of in-band signaling like pushing buttons on your phone during a call. The tones are in the voice path of the call. Tones are sent at the beginning of the call (sometimes you can hear them) from switch to switch (when using CAS signaling) to provide signals for the switch to make decisions about how to route your call. There are also signals within the channel that are not audible. These signals are bits embedded in the voice data; they are called the ABCD bits, and are used to report on the status of phones (e.g., off-hook/on-hook).

Out-of-band signaling, in contrast, works similarly to the FTP protocol: a channel is used to set up the call, and then a separate channel is chosen and used to deliver the payload (in this case, a voice call).

Encoding

Encoding refers to the method by which electrical signals are generated and decoded. There are two types of encoding in use on T1 links today: Alternate Mark Inversion (AMI), and Binary Eight Zero Substitution (B8ZS). Generally, AMI is used for voice circuits, and B8ZS is used for data. B8ZS can be used for voice, but AMI should not be used for data (the reason is noted below).

AMI

AMI is a method of encoding that inverts alternate marks. In T1 signaling, there are two possible states: mark and space. Simply put, a *mark* is a one, and a *space* is a zero. On a T1, a space is 0V, and a mark is either +5V or –5V. AMI encodes the signal such that the polarity of each mark is the opposite of the one preceding it.

This allows for some interesting error-detection techniques. For example, Figure 20-2 shows two ones occurring in a row with the same polarity. This is considered an error (a *bipolar violation*, or BPV). If all ones were positive voltage, a voltage spike could be misconstrued as a valid one. As an added benefit, when the alternating marks are flipped, the average voltage of the physical line will always be 0V, making the physical T1 wires safe to handle.

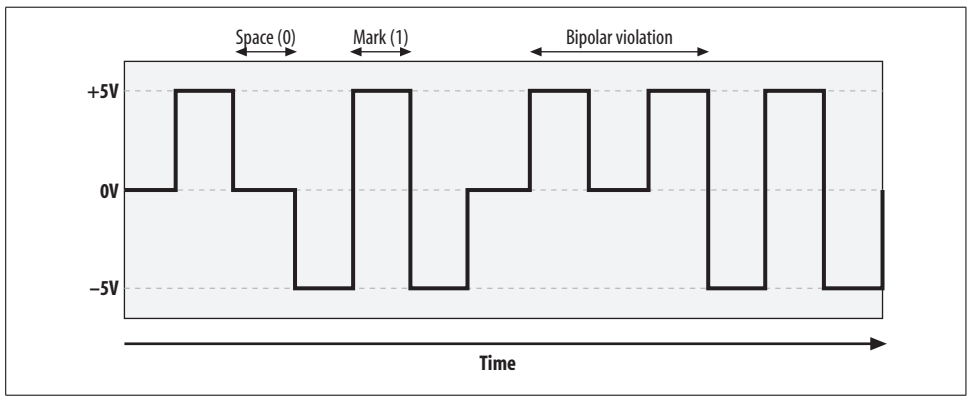


Figure 20-2. T1 AMI signaling

T1s are asynchronous links, meaning that only one side of the link provides clocking. The far side of the link must rely on the signal itself to determine where bits begin and end. Because the duration of a mark is known, synchronization can be achieved simply by receiving marks. When using AMI, a long progression of spaces will result in a loss of synchronization. With no marks in the signal, the receiving end will eventually lose track of where bits begin and end.

The risk of an *all zeros* signal exists, so AMI sets every eighth bit to a 1, regardless of its original value. This ensures there are enough signal transitions in the line (i.e., that the *ones density* of the signal stream is sufficiently high to ensure synchronization). As few as 16 zeros in a row can cause the remote end to lose synchronization.

Voice signals can easily absorb having every eighth bit set to 1. The human ear can't hear the difference if a single bit in the stream is changed. Data signals, though, cannot tolerate having any bits changed. If one bit is different in a TCP packet, the Cyclic Redundancy Check (CRC) will fail, and the packet will be resent (CRCs are not performed for UDP packets). Because of this limitation, AMI is not an acceptable encoding technique for use on data T1s.

B8ZS

B8ZS encoding was introduced to resolve the shortcomings of AMI. The idea behind B8ZS is that if eight zeros in a row are detected in a signal, those eight zeros are converted to a pattern including intentional BPVs. When the remote side sees this well-known pattern, it converts it back to all zeros.

Figure 20-3 shows how long strings of zeros are converted on the wire. The top signal consists of a one, followed by nine zeros, then three ones. B8ZS takes the first eight zeros and converts them to a pattern including two BPVs. This pattern would not be seen on a normal, healthy circuit. When the remote side receives the pattern, it converts it back into eight zeros. This technique allows data streams to contain as many consecutive zeros as necessary while maintaining ones density.

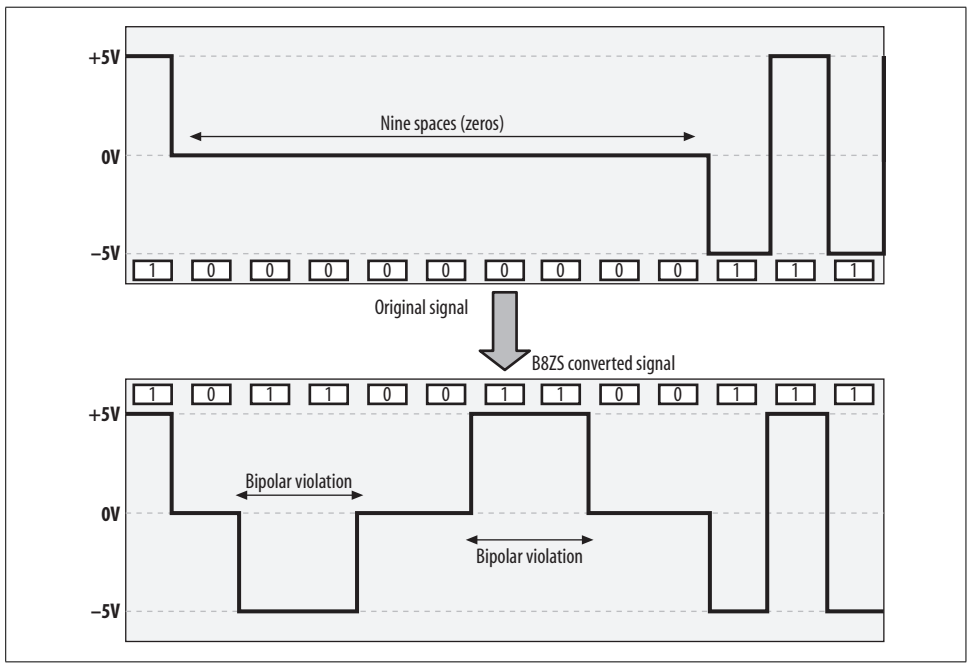


Figure 20-3. B8ZS zero substitution

Framing

Phone audio is sampled 8,000 times per second (i.e., at 8 kHz). Each sample is converted to an 8-bit value, with one of the bits used for signaling.

Figure 20-4 shows a single-channel sample, with one bit used for signaling. This is called *in-band signaling*.

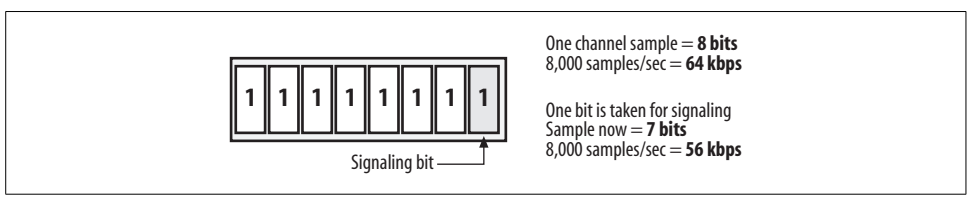


Figure 20-4. One-channel sample

When a T1 is configured as a PRI, all eight bits in each channel may be used for data because one entire channel is reserved for signaling (as opposed to pulling one bit from each channel). This reduces the number of usable channels from 24 to 23, and is called *out-of-band signaling*.

T1s use time-division multiplexing, which means that each channel is actually a group of serial binary values. The channels are relayed in order, but the receiving equipment needs to know when the first channel starts, and when the last channel ends. The way this is done is called *framing*.

D4/Superframe

In standard voice framing, called *D4* or *superframe*, each 8-bit sample is relayed from each channel in order. One sample from channel one is relayed, then one sample from channel two is relayed, and so on, until all 24 channels have relayed one sample. The process then repeats.

For the receiving end to understand which channel is which, framing bits are added after each of the 24 channels has relayed one sample. Because each sample is 8 bits, and there are 24 channels, one iteration for all channels is 192 bits. With the addition of the framing bit, we have 193 bits. These 193-bit chunks are called *frames*. Each set of 12 frames is called a superframe.

The framing scheme is outlined in Figure 20-5. The T1 devices keep track of the frames by inserting the pattern 110111001000 into the framing bits over the span of a superframe.

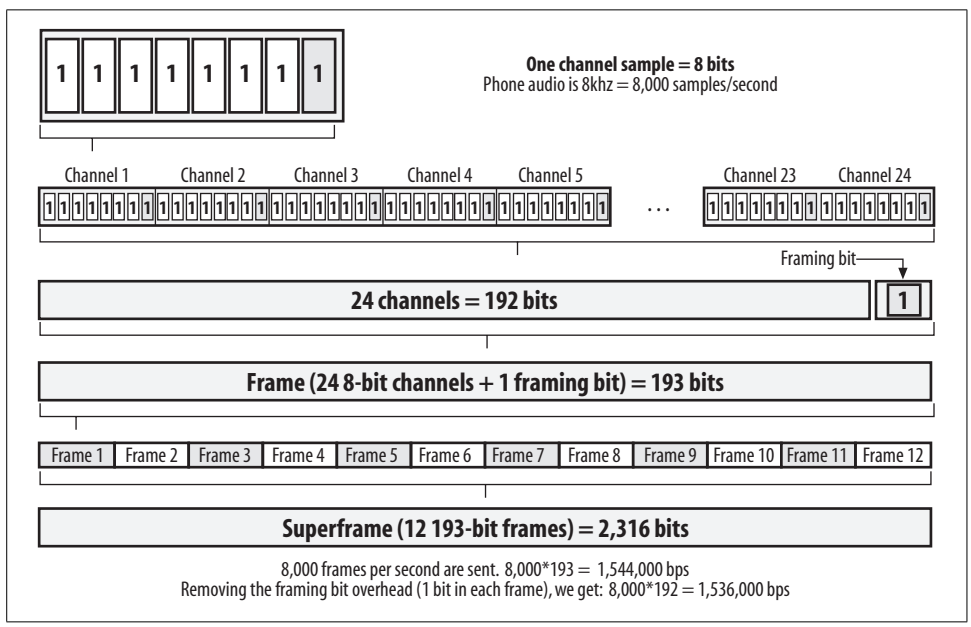


Figure 20-5. DS1 framing

When the framing bits do not match the expected sequence, the receiving equipment logs a *framing error*. When a T1 device reaches a certain threshold of framing errors, an alarm is triggered.

You may have seen a discrepancy in the reported speed of T1 links in your reading. Some texts will show a T1 to be 1.544 Mbps, while others may show 1.536 Mbps. This discrepancy is a result of the framing bits. As the framing bits are used by the T1 hardware, and are not available as data, they are considered overhead. Thus, 1.536 Mbps is the usable speed of a T1 when framing bits are taken into consideration.

Extended Superframe (ESF)

The D4/superframe standard was developed for voice, and is not practical for data transmissions. One of the reasons D4 is unsuitable for data is the lack of error detection. To provide error-detection capabilities, and to better use the framing bits, a newer framing standard called *extended superframe* was developed.

ESF works under the same general principles as D4/superframe, except that an extended superframe is composed of 24 frames instead of 12. The framing bit of each frame is used to much greater effect in ESF than it was in D4. Instead of simply filling the framing bits with an expected pattern throughout the course of the superframe, ESF uses these bits as follows:

Frames 4, 8, 12, 16, 20, and 24 (every fourth frame)
: These frames' framing bits are filled with the pattern 001011.

Frames 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, and 23 (every odd-numbered frame)
: These frames' framing bits are used for a new, 4,000 bps virtual data channel. This channel is used for out-of-band communications between networking devices on the link.

Frames 2, 6, 10, 14, 18, and 22 (the remaining even-numbered frames)
: These frames' framing bits are used to store a six-bit CRC value for each superframe.

Any time a T1 circuit is ordered for use as a WAN link for data networks, the T1 should be provisioned with B8ZS encoding and ESF framing.

Performance Monitoring

CSU/DSUs report on the status of T1 links by reporting the incidence of a set of standard events. Some of the events and the alarms they trigger can be a bit confusing to data professionals who have not been exposed to the technology before. To make matters worse, most CSU/DSUs report errors using not words, but rather the well-known (to telecom engineers) abbreviations of the errors.

Different vendors often define performance events differently, and finding detailed descriptions of these events can be challenging. One place where the event types are outlined is in RFC 1232, titled "Definitions of Managed Objects for the DS1 Interface Type." This RFC defines the standards for use in SNMP traps, and does not describe the electrical properties of these alarms. These descriptions are not binding for manufacturers, and may not be accurate for any given device. Still, the RFC does contain some of the clearest descriptions of these events.

Loss of Signal (LOS)

Loss of signal is the state where no electrical pulses have been detected in a preset amount of time. RFC 1232 describes LOS as:

> This event is declared upon observing 175 +/- 75 contiguous pulse positions with no pulses of either positive or negative polarity (also called keep alive).

In English, that means that the line is dead. There are no alarms, no signals, etc. LOS is equivalent to having no cable in the T1 jack.

Out of Frame (OOF)

An *out-of-frame* condition (also called *loss of frame*, or LOF) indicates that a certain number of frames have been received with framing bits in error. In this case, the data cannot be trusted because the synchronization between the two sides of the T1 is invalid. Excessive OOF errors will trigger a red alarm. An OOF condition is described in RFC 1232 as follows:

> An Out of Frame event is declared when the receiver detects two or more framing-bit errors within a 3 millisecond period, or two or more errors out of five or less consecutive framing-bits. At this time, the framer enters the Out of Frame State, and starts searching for a correct framing pattern. The Out of Frame state ends when reframe occurs.

Bipolar Violation (BPV)

A *bipolar violation* occurs when two mark signals (ones) occur in sequence with the same polarity. DS1 signaling specifies that each mark must be the opposite polarity of the one preceding it. When two marks occur with the same polarity (when not part of a B8ZS substitution), this is considered an error. Excessive BPVs will put the station into alarm. BPVs are described in RFC 1232 as follows:

> A Bipolar Violation, for B8ZS-coded signals, is the occurrence of a received bipolar violation that is not part of a zero-substitution code. It also includes other error patterns such as: eight or more consecutive zeros and incorrect parity.

CRC6

CRC6 is the Cyclic Redundancy Check (6-bit) mechanism for error checking in ESF. This error is a result of ESF reporting data integrity problems. CRC6 events are not described in RFC 1232.

Errored Seconds (ES)

Errored Seconds (ES) is a counter showing the number of seconds in a 15-minute interval during which errors have occurred. This counter provides a quick way to see whether there are problems. It also offers an indication of how "dirty" a T1 might be. If the number is high, there is a consistent problem. If it is low, there is probably a short-term (possibly repetitive) or intermittent problem. Errored seconds are usually incremented when one or more errors occur in the span of one second. Errored seconds are described in RFC 1232 as follows:

> An Errored Second is a second with one or more Code Violation Error Events OR one or more Out of Frame events. In D4 and G.704 section 2.1.3.2 (eg, G.704 which does not implement the CRC), the presence of Bipolar Violations also triggers an Errored Second.

Extreme Errored Seconds (EES)

Sometimes also referred to as severely errored seconds (SES), this counter increments when a certain threshold of errors is passed in the span of one second. The threshold and the errors to be counted depend on the hardware implementation. You should not see extreme errored seconds on a healthy link, but some errored seconds may occur on a normal circuit. SES events are described in RFC 1232 as follows:

> A Severely Errored Second is a second with 320 or more Code Violation Error Events OR one or more Out of Frame events.

Alarms

Alarms are serious conditions that require attention. Excessive errors can trigger alarms, as can hardware problems, and signal disruption. The alarms are coded as colors. As with performance events, different vendors define alarms differently, and finding detailed descriptions of them can be challenging. RFC 1232 also describes most alarms, though again, they are described for use in SNMP, and the descriptions are not intended as a standard for hardware implementation.

Red Alarm

A *red alarm* is defined in RFC 1232 as follows:

> A Red Alarm is declared because of an incoming Loss of Signal, Loss of Framing, Alarm Indication Signal. After a Red Alarm is declared, the device sends a Yellow Signal to the far-end. The far-end, when it receives the Yellow Signal, declares a Yellow Alarm.

A red alarm is triggered when a local failure has been detected, or continuous OOF errors have been detected for more than *x* seconds (vendor-specific). The alarm is cleared after a specific amount of time has elapsed with no OOF errors detected (the amount of time varies by hardware).

When a device has a local red alarm, it sends out a yellow alarm.

Figure 20-6 shows a sample red alarm. Something has failed on Telecom Switch C. The switch triggers a local red alarm, and sends out the yellow alarm signal to alert its neighbors of the problem.

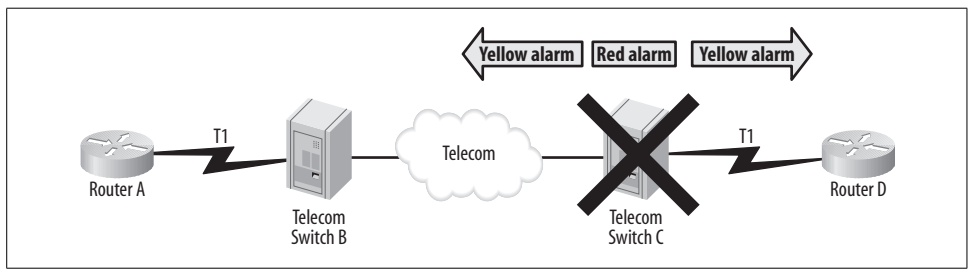


Figure 20-6. Red alarm

Figure 20-7 shows another red alarm scenario. In this example, Telecom Switch C is sending garbled frames to Router D, which sees consecutive OOF problems and declares a red alarm. When Router D declares the red alarm, a yellow alarm signal is sent back out the link.

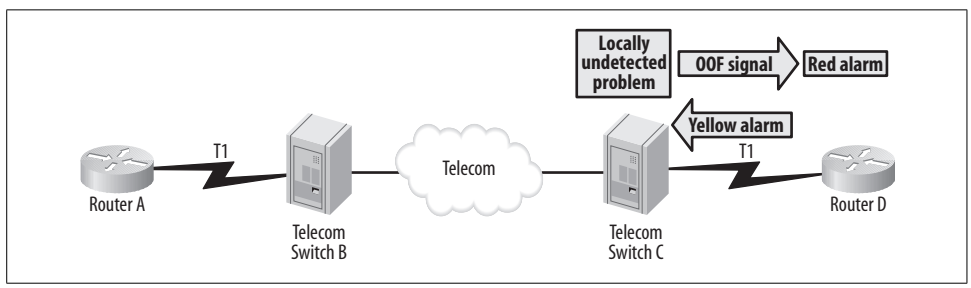


Figure 20-7. Yellow alarm

The way these alarms behave can be a bit confusing. A red alarm generally indicates that something serious is happening on the local equipment, but in the example in Figure 20-7, Router D is receiving so many OOF errors that the signal is useless. Because Router D cannot figure out how to read the frames from the far side, it triggers a local red alarm and sends out a yellow alarm.

Yellow Alarm (RAI)

A yellow alarm is also called a *remote alarm indication* (RAI). A yellow alarm indicates a remote problem. A yellow alarm is defined in RFC 1232 as follows:

```
A Yellow Alarm is declared because of an incoming Yellow Signal from the far-end.
In effect, the circuit is declared to be a one way link.
```

One vendor specifies that a yellow alarm to be declared for SF links when bit six of all the channels has been zero for at least 335 ms, and that it be cleared when bit six of at least one channel is not zero for less than one to five seconds. For an ESF link, a yellow alarm is declared if the signal pattern occurs in at least 7 of 10 contiguous 16-bit pattern intervals, and is cleared if the pattern does not occur in 10 contiguous 16-bit pattern intervals.

Wow, what a mouthful! The simple truth is that unless you're designing T1 CSU/DSUs, you don't need to know all of that. Here's what you need to know: a yellow alarm does not necessarily indicate a problem with your device; rather, it's a problem being reported by the device to which you are connected.

Figure 20-8 shows a simple network. Router A has a T1 link to Router D. The T1 is actually terminated locally at the T1 provider's central office, where it is usually aggregated into a larger circuit, hauled to the remote CO, and then converted back to a T1 for delivery to the remote location. Telecom Switch B is fine, but Telecom Switch C has experienced a failure.

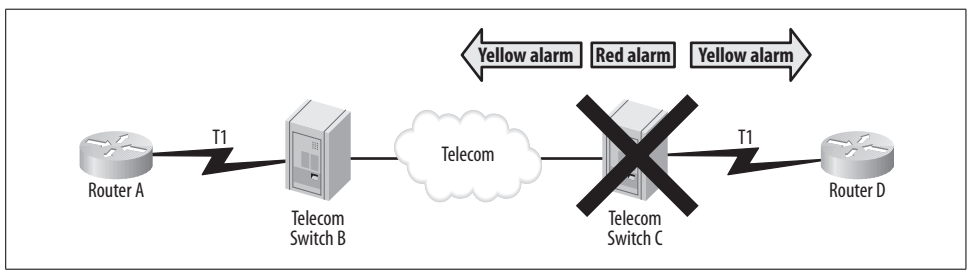


Figure 20-8. Another yellow alarm

Telecom Switch B receives a yellow alarm from Telecom Switch C. This alarm may be forwarded to Router A. When diagnosing the outage, the presence of a yellow alarm usually indicates that some other device is at fault. Here, Telecom Switch C is the cause of the outage.

Watch out for assumptions. Router D will receive a yellow alarm, as might Router A. In this case, the admin for Router A may blame Router D (and vice versa), as he probably has no idea that telecom switches are involved.

Blue Alarm (AIS)

A blue alarm is also called an *alarm indication signal* (AIS). There is no definition in RFC 1232 for this condition. A blue alarm is representative of a complete lack of an incoming signal, and is indicated by a constant stream of unframed ones. You may hear someone say that the interface is receiving "all ones" when a blue alarm is active. If you're receiving a blue alarm, there is a good chance that a cable is disconnected, or a device has failed.

Troubleshooting T1s

The first step in troubleshooting a T1 is to determine where the problem lies. Usually, it's cabling-, hardware-, or telco-related. Running some simple tests can help determine what steps to take. Note that all of these tests are invasive, which means you must take the T1 out of service to perform them.

Loopback Tests

Loopback tests involve setting one piece of equipment to a loopback state and sending data over the link. The data should return to you exactly as you sent it. When the data does not come back as expected, something has happened to alter it. Figure 20-9 shows conceptually how a loopback test might fail.

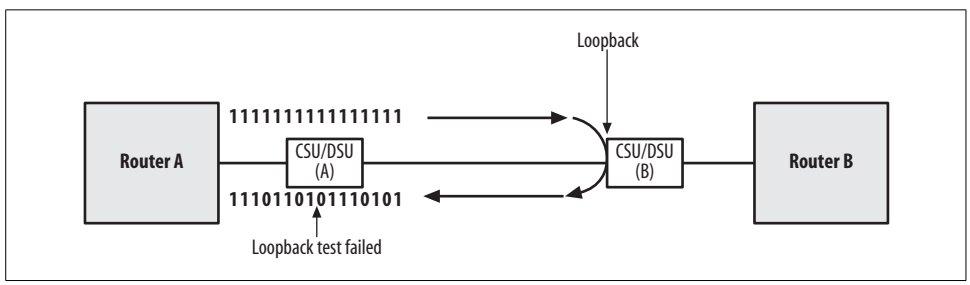


Figure 20-9. Conceptual loopback test failure

When you perform a loopback test, failed results won't typically be as clean as ones being changed into zeros. Usually, the problem is more electrical in nature, like the scenario shown in Figure 20-10, or framing errors cause the data to become entirely unreadable.

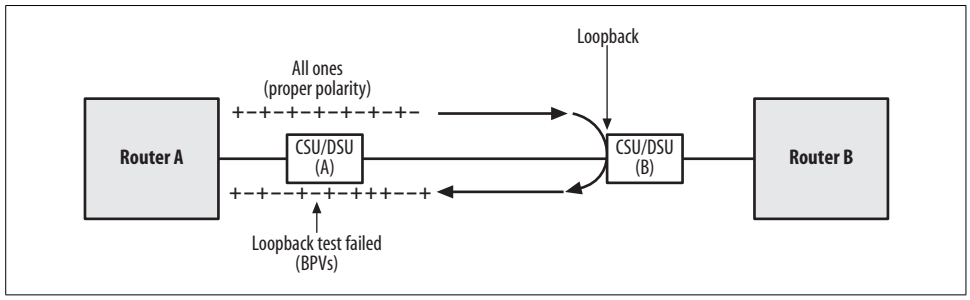


Figure 20-10. BPVs seen during loopback test

When performing loopback tests, the logical way to proceed is to start at one end of the link and move across it until the symptom appears.

CSU/DSUs generally offer the option of setting multiple types of loopback, which greatly assists in this process: you can usually set a loopback at the interface connecting the T1 to telco (called a *line loopback*), or after the signal has passed through the CSU/DSU's logic (called a *payload loopback*).

Many CSU/DSUs allow these loopbacks to be set in either direction, which can further aid in trouble isolation.

Telecom engineers look for *trouble* in a line when they troubleshoot. This may sound pedantic, but when you listen to telecom engineers troubleshooting, they will use very specific terms like *no trouble found* (which is often abbreviated as NTF). Remember, there are more than 100 years of standards at work here.

Let's look at an example. In Figure 20-11, Router A is connected to Router B by a T1 using two CSU/DSUs. From the point of view of CSU/DSU (A), we can see the possible loopback points available on the CSU/DSUs themselves.

Bear in mind that if we were to look from the point of view of CSU/DSU (B), all of the points would be reversed.

Not all models of CSU/DSU have all of these options, and not all CSU/DSUs call these loopback points by the same names.

The following list of terms describes the loopback points that are usually available. Remember that the descriptions are based on the network as seen from Router A in Figure 20-11:

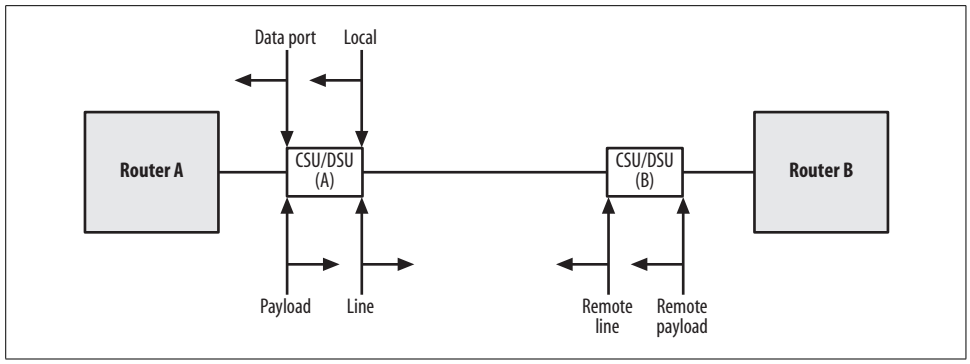


Figure 20-11. Loopback points in CSU/DSUs

Data port/DTE
: This point loops the signal from the directly connected data device (in our case, Router A) back to that device without using the T1 framer logic. This tests the V.35 cable and the router.

Local
: This point loops the signal from the directly connected data device back to that device after it has been processed by the T1 framer logic. This tests the CSU/DSU, in addition to the V.35 cable and the router.

Payload
: This point loops the signal coming from the T1 back onto the T1 after it has been processed by the T1 framer logic. This test would be administered on the (B) side in our example, but the loopback would be set locally on CSU/DSU (A).

Line
: This point loops the signal coming from the T1 back onto the T1 before it has been processed by the T1 framer logic, effectively testing the T1 line without testing the CSU/DSU. In our example, a line loopback on CSU/DSU (A) would be tested from the (B) side, though the line loopback would be set locally on CSU/DSU (A).

Remote line
: Remote line loopback is a feature available on some CSU/DSUs that allows a local CSU/DSU to set a line loopback on the far-end device. In our example, though the loopback would exist on CSU/DSU (B), the command to initiate the loopback would be entered on CSU/DSU (A).

Remote payload
: Remote payload loopback is a feature available on some CSU/DSUs that allows a local CSU/DSU to set a payload loopback on the far-end device. In our example, though the loopback would exist on CSU/DSU (B), the command to initiate the loopback would be entered on CSU/DSU (A).

Say we're having problems with the T1 in Figure 20-11 and we need to troubleshoot. Looking at the error stats in CSU/DSU (A), we see many OOF, BPV, and CRC6 errors. Actual testing of the T1 line would typically proceed as shown in Figure 20-12.

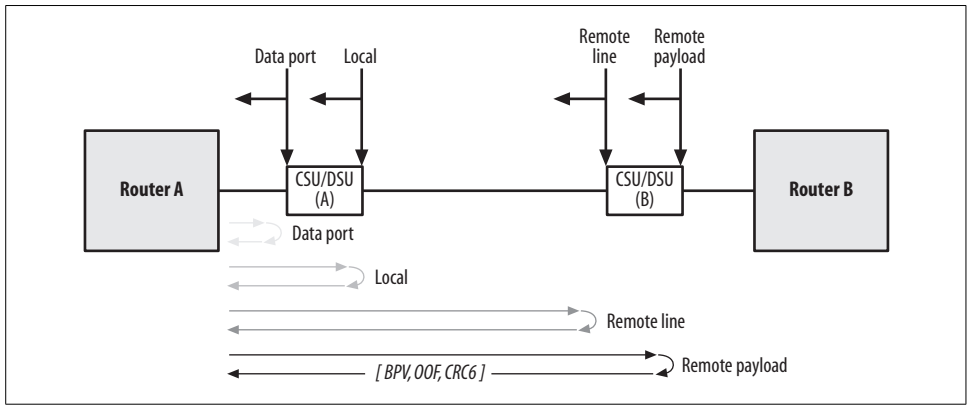


Figure 20-12. Loopback testing progression

First, we set a data port loopback on CSU/DSU (A) and send our tests. The tests all pass without error, indicating that the V.35 cable is good.

Next, we clear the data port loopback on CSU/DSU (A) and set a local loopback. Again, we perform our tests, and all packets return with no errors. We have now eliminated Router A, the V.35 cable, and 90 percent of CSU/DSU (A).

The next step is to clear the local loopback on CSU/DSU (A) and set a remote line loopback on CSU/DSU (B). Because this is a remote loopback, we'll set the loop from CSU/DSU (A). (Alternatively, we could have called someone at the (B) location to manually set a line loopback on CSU/DSU (B).) Again, we run our tests, and all results are clean. We have now eliminated Router A, Router A's V.35 cable, CSU/DSU (A), and the T1 line itself (including all telco responsibility).

Now, we clear the remote line loopback on CSU/DSU (B), and set a remote payload loopback on CSU/DSU (B) (again, administered from CSU/DSU (A)). This time when we run our test, CSU/DSU (A) reports many BPV, OOF, and CRC6 errors. We have found the source of the trouble—CSU/DSU (B) is not functioning properly. By systematically moving our loopback point further and further away from one side of the link, we were able determine the point at which the trouble started to appear. Replacing CSU/DSU (B) solves our problem, and we're back in business.

Integrated CSU/DSUs

T1 WAN interface cards (WICs) with integrated CSU/DSUs are about the coolest thing to happen to routers in the past 10 years. Call me a nerd, but the idea of

removing another physical piece of equipment for each T1 installed (as well as those horrible V.35 cables from equipment racks) is the closest thing to geek paradise I've ever seen.

The integrated CSU/DSU WICs serve the same purpose as standalone units, and they have the added benefit of being controlled via IOS from the router. Figure 20-13 shows the loopback points for a T1 CSU/DSU WIC as they are described in IOS.

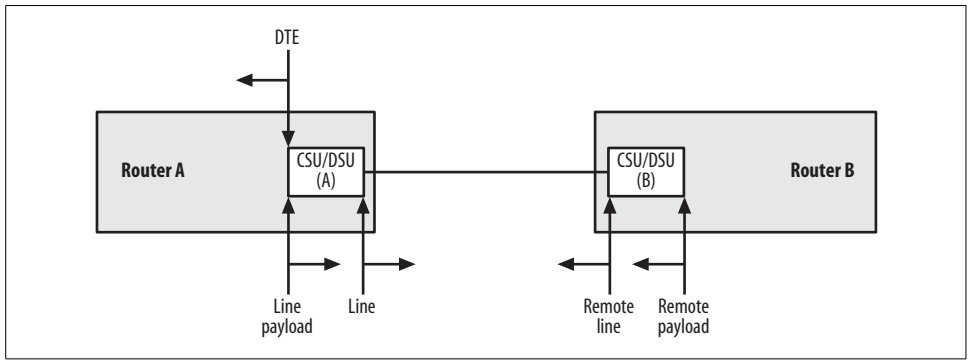


Figure 20-13. Integrated CSU/DSU loopback points

Some CSU/DSUs even include a feature that lets them run BERT tests. While this can be useful, if you're running BERT tests, you should probably call telco. Most problems discovered by BERT tests cannot be fixed with router configuration.

Configuring T1s

There are two steps involved in configuring T1s for use on a router. The first step is the configuration of the CSU/DSUs. The second step is the configuration of the router interface. When using integrated CSU/DSUs, the lines might seem blurred, but the concepts remain the same. Configuring the router interface is just like configuring any serial interface.

CSU/DSU Configuration

To get a T1 up and operational, you must:

Configure both sides with the same encoding that matches the circuit's provisioned encoding

Encoding options are AMI and B8ZS. Data T1s should always use B8ZS encoding. To configure encoding on a CSU/DSU WIC, use the `service-module t1 linecode` interface command:

```
Router(config)# int s0/1
Router(config-if)# service-module t1 linecode b8zs
```

Configure both sides with the same framing that matches the circuit's provisioned framing

Framing options are D4/SF and ESF. Data T1s should always use ESF framing. To configure framing on a CSU/DSU WIC, use the `service-module t1 framing` interface command:

```
Router(config)# int s0/1
Router(config-if)# service-module t1 framing esf
```

Configure how many channels will be used for the link, what channels will be used, and what speed they will be

If the T1 is being split or you have had fewer than 24 channels delivered to you, you must tell the CSU/DSU how many channels are in use. This is done for the CSU/DSU WIC with the `service-module t1 timeslots` interface command. Here, I've specified that channels 7–12 will be used at a speed of 64 Kbps:

```
Router(config)# int s0/1
Router(config-if)# service-module t1 timeslots 7-12 speed 64
```

By default, all channels are used with a speed of 64 Kbps per channel. In the event that you need to return a configured CSU/DSU WIC back to using all channels, you can do so with the `all` keyword:

```
Router(config)# int s0/1
Router(config-if)# service-module t1 timeslots all speed 64
```

Configure one side as the clock master, and the other side as the slave

T1s are asynchronous, so only one side has an active clock. The other side will determine the clocking from the data stream itself using a technology called *phase-locked loop* (PLL). To configure clocking on a CSU/DSU WIC, use the `service-module t1 clock source` interface command. The options are `internal` and `line`:

```
Router(config)# int s0/1
Router(config-if)# service-module t1 clock source internal
```

`internal` means that the CSU/DSU will provide clocking (master), and `line` indicates that the clocking will be determined from the data stream on the line (slave). The default behavior is to use the line for clocking.

Some environments may require that clocking be set to `line` on both ends. Check with your provider if you are unsure of your clocking requirements.

CSU/DSU Troubleshooting

Having a CSU/DSU integrated into a router is an excellent improvement over using standalone CSU/DSUs. The ability to telnet to the CSU/DSU is marvelous during an outage. Standalone CSU/DSUs often have serial ports on them that can be hooked to console servers, but the average corporate environment rarely uses this feature. Additionally, many companies use many different brands and models of CSU/DSUs, each with their own menus, commands, features, and quirks.

The Cisco T1 CSU/DSU WIC allows for CSU/DSU statistics to be viewed by telnetting to the router and issuing commands. The main command for troubleshooting a T1 CSU/DSU WIC is the `show service-module` *`interface`* command. This command outputs a wealth of information regarding the status of the CSU/DSU and the T1 circuit in general.

Let's look at the output of this command with a T1 that is not connected on the far end:

```
Router# sho service-module s0/1
Module type is T1/fractional
    Hardware revision is 0.112, Software revision is 0.2,
    Image checksum is 0x73D70058, Protocol revision is 0.1
Transmitter is sending remote alarm.
Receiver has loss of frame,
Framing is ESF, Line Code is B8ZS, Current clock source is line,
Fraction has 24 timeslots (64 Kbits/sec each), Net bandwidth is 1536 Kbits/sec.
Last user loopback performed:
    dte loopback
    duration 08:40:48
Last module self-test (done at startup): Passed
Last clearing of alarm counters 08:45:16
    loss of signal        :    1, last occurred 08:45:07
    loss of frame         :    2, current duration 00:01:38
    AIS alarm             :    0,
    Remote alarm          :    0,
    Module access errors  :    0,
Total Data (last 34 15 minute intervals):
    2 Line Code Violations, 0 Path Code Violations
    1 Slip Secs, 200 Fr Loss Secs, 2 Line Err Secs, 0 Degraded Mins
    0 Errored Secs, 0 Bursty Err Secs, 0 Severely Err Secs, 200 Unavail Secs
Data in current interval (896 seconds elapsed):
    255 Line Code Violations, 255 Path Code Violations
    32 Slip Secs, 109 Fr Loss Secs, 34 Line Err Secs, 0 Degraded Mins
    0 Errored Secs, 0 Bursty Err Secs, 0 Severely Err Secs, 116 Unavail Secs
```

Here, we can see that the CSU/DSU is sending a *remote alarm* (yellow alarm) out the T1 because it's receiving *loss of frame* errors. More importantly, one *loss of signal* event has occurred approximately 8 hours and 45 minutes ago. This T1 has not been up since the router booted. The last time the alarm stats were cleared was also 8 hours and 45 minutes ago.

This output essentially tells us that there's nothing on the other side of our circuit. Sure enough, the router on the far end is powered off. It also has an integrated CSU/DSU. After we power up the far-end router, let's clear the counters and see how the service module looks:

```
Router# clear counters s0/1
Clear "show interface" counters on this interface [confirm]
09:00:04: %CLEAR-5-COUNTERS: Clear counter on interface Serial0/1 by console
Router#
Router# sho service-module s0/1
Module type is T1/fractional
```

```
    Hardware revision is 0.112, Software revision is 0.2,
    Image checksum is 0x73D70058, Protocol revision is 0.1
Receiver has no alarms.
Framing is ESF, Line Code is B8ZS, Current clock source is line,
Fraction has 24 timeslots (64 Kbits/sec each), Net bandwidth is 1536 Kbits/sec.
Last user loopback performed:
    dte loopback
    duration 08:40:48
Last module self-test (done at startup): Passed
Last clearing of alarm counters 00:03:01
    loss of signal        :    0,
    loss of frame         :    0,
    AIS alarm             :    0,
    Remote alarm          :    0,
    Module access errors  :    0,
Total Data (last 96 15 minute intervals):
    258 Line Code Violations, 257 Path Code Violations
    33 Slip Secs, 309 Fr Loss Secs, 37 Line Err Secs, 1 Degraded Mins
    1 Errored Secs, 1 Bursty Err Secs, 0 Severely Err Secs, 320 Unavail Secs
Data in current interval (153 seconds elapsed):
    0 Line Code Violations, 0 Path Code Violations
    0 Slip Secs, 0 Fr Loss Secs, 0 Line Err Secs, 0 Degraded Mins
    0 Errored Secs, 0 Bursty Err Secs, 0 Severely Err Secs, 0 Unavail Secs
```

This output looks a lot better. It includes the welcome phrase "receiver has no alarms," and it indicates that we've just cleared the alarms, and no errors or alarms have been received since.

The last two paragraphs of the output are especially important. CSU/DSUs usually keep track of all the events that have occurred during the previous 24 hours. These events are recorded in 15-minute intervals and reported as such. The first paragraph (`Total Data`), while alarming, is not as important as the next paragraph (`Data in current interval`), which shows us the events that have occurred during the current interval. This can be a bit confusing, so let's take a closer look.

By using the command `show service-module` *`interface`* `performance-statistics`, we can see which events occurred during each of the last 96 15-minute intervals:

```
Router# sho service-module s0/1 performance-statistics
Total Data (last 96 15 minute intervals):
    258 Line Code Violations, 257 Path Code Violations
    33 Slip Secs, 309 Fr Loss Secs, 37 Line Err Secs, 1 Degraded Mins
    1 Errored Secs, 1 Bursty Err Secs, 0 Severely Err Secs, 320 Unavail Secs
Data in current interval (380 seconds elapsed):
    0 Line Code Violations, 0 Path Code Violations
    0 Slip Secs, 0 Fr Loss Secs, 0 Line Err Secs, 0 Degraded Mins
    0 Errored Secs, 0 Bursty Err Secs, 0 Severely Err Secs, 0 Unavail Secs
Data in Interval 1:
    1 Line Code Violations, 2 Path Code Violations
    0 Slip Secs, 0 Fr Loss Secs, 1 Line Err Secs, 0 Degraded Mins
    1 Errored Secs, 1 Bursty Err Secs, 0 Severely Err Secs, 0 Unavail Secs
```

```
Data in Interval 2:
    255 Line Code Violations, 255 Path Code Violations
    32 Slip Secs, 109 Fr Loss Secs, 34 Line Err Secs, 0 Degraded Mins
    0 Errored Secs, 0 Bursty Err Secs, 0 Severely Err Secs, 116 Unavail Secs
Data in Interval 3:
    0 Line Code Violations, 0 Path Code Violations
    0 Slip Secs, 0 Fr Loss Secs, 0 Line Err Secs, 0 Degraded Mins
    0 Errored Secs, 0 Bursty Err Secs, 0 Severely Err Secs, 0 Unavail Secs
Data in Interval 4:
    0 Line Code Violations, 0 Path Code Violations
    0 Slip Secs, 0 Fr Loss Secs, 0 Line Err Secs, 0 Degraded Mins
    0 Errored Secs, 0 Bursty Err Secs, 0 Severely Err Secs, 0 Unavail Secs
```

The first paragraph shows the combined totals for all the intervals in memory. The maximum amount of time for which the CSU/DSU will record events is 24 hours (96 15-minute intervals), at which point the oldest interval's data is discarded. The output is arranged by interval, not by something more obvious like actual time. This is a throwback to the way standalone CSU/DSUs reported historical information. The first interval listed is the current interval. The next interval, numbered as *Interval 1*, is the most recent interval. The intervals increment until they reach 96.

When looking at this information, search for patterns. If you see a number of line code violations, or any other error that shows up in all or most of the intervals, you've got a problem. Note that you will see errors when a T1 bounces for any reason. Interval 2 in the preceding output shows some pretty severe errors, which are the result of me unplugging the T1 cable from the jack.

If you're seeing errors incrementing, start troubleshooting, and remember: physical layer first! Most problems are caused by a bad cable or piece of equipment.

CHAPTER 21
DS3

I'm going to treat DS3s a little differently than I did T1s. While I believe knowledge of T1 framing and signaling is useful for a network engineer, I don't feel that the specifics are all that important when dealing with DS3s. For example, contrary to what many people will tell you (often adamantly), a DS3 is not defined as 28 DS1s. A DS3 is actually the result of multiplexing seven DS2s. If you're saying to yourself, "There's no such thing as a DS2," you're not alone. A DS2 is a group of four DS1s, and is not seen outside of multiplexing.

While it may be interesting to know that a DS3 is composed of DS2s, that knowledge won't help you build or troubleshoot a network today. In this chapter, I'll explain what you do need to know about DS3s: simple theory, error conditions, how to configure them, and how to troubleshoot them.

A DS3 is not a T3. *DS3* (Digital Signal 3) is the logical carrier sent over a physical T3 circuit. In practice, the terms are pretty interchangeable; most people will understand what you mean if you use either. However, from this point on, I'll refer to the circuit simply as a DS3, as we're really interested in the circuit, and not the physical medium.

You'll encounter two flavors of DS3s: channelized and clear-channel. A *channelized DS3* is one in which there are 672 DS0s, each capable of supporting a single POTS-line phone call. When a DS3 is channelized, Cisco will often refer to it as a "channelized T3." A *clear-channel DS3* has no channels and is used for pure data.

Framing

When I stated earlier that a DS3 is actually a group of seven DS2s multiplexed together, I was referring to a channelized DS3. When DS3s were designed in the 1960s, there really wasn't a need for data circuits like those we have today. DS3s were designed to handle phone calls, which is why they are multiplexed the way they are.

DS3s require framing for the same reasons that DS1s do. The difference is that there can be multiple DS1s multiplexed within a DS3. Each of those DS1s has its own clocking, framing, and encoding that must be maintained within the DS3. The DS3 must also have its own clocking, framing, and encoding, which must not interfere with the multiplexed circuits within it. There are a couple of different framing methods that can be used. Your choice should be dictated by the DS3's intended use.

M13

M13 (pronounced M-one-three, not M-thirteen) is short for *Multiplexed DS1 to DS3*. When a multiplexer builds a DS3, it goes through two steps: M12 and M23. The combination of these steps is referred to as M13. Figure 21-1 shows the steps involved in converting 28 DS1s into a single DS3.

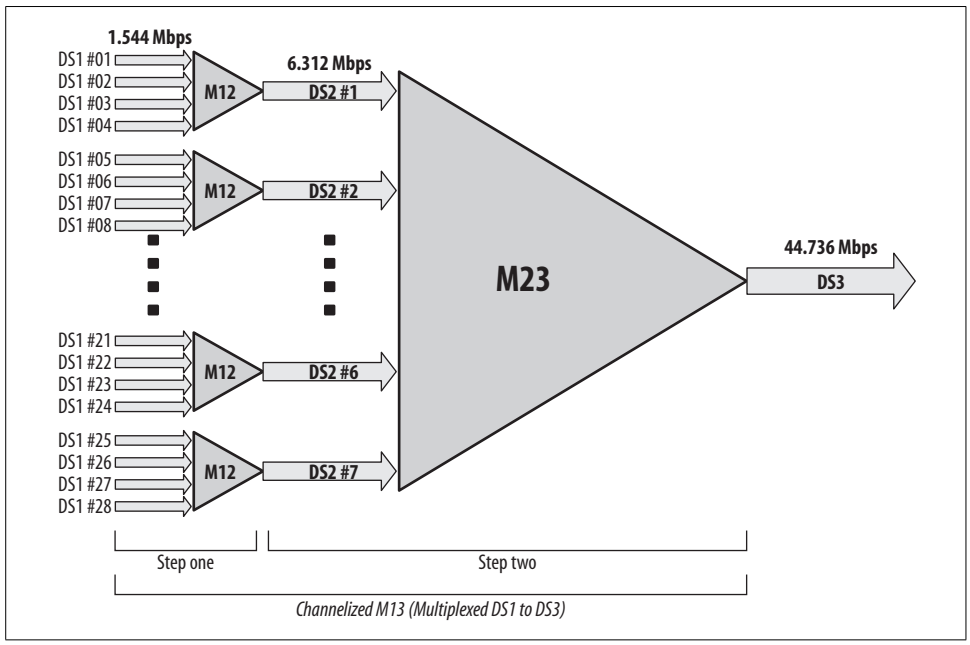


Figure 21-1. M13 multiplexing

Originally, DS3s were used for aggregating T1s. Imagine 28 T1s, all terminating at a CO, but originating at 28 different locations at varying distances from the CO. Because the T1s are not related, they may be out of sync with each other. The original designers knew that this was a probability, so they designed the first stage of the multiplexing process to deal with the problem.

The speed of a T1 is generally reported as 1.544 Mbps. If you multiply 1.544 Mbps * 4, you get 6.176 Mbps. Why, then, is a DS2, which is four T1s, shown as 6.312 Mbps? To compensate for T1s that are not delivering bits in a timely manner (+/– 77 Hz), the M12 multiplexer stuffs bits into the signal to get them up to speed with the other T1s in the group.

Each T1 is brought up to a line rate of 1,545,796 bits per second after *bit stuffing*. In all, 128,816 additional framing and overhead bits are added, which brings the total to 6.312 Mbps (1,545,796 * 4 + 128,816). The receiving multiplexer removes the extra bits.

Overclocking the T1s ensures that the DS3 should never cause a timing problem with the individual T1s. Remember that each T1 will have its own clock master and slave. The DS3 needs to support its own clocking without interfering with that of the individual T1s. (Modern networks that use SONET in the core do not really have this problem, but this technology was designed many years ago, before SONET existed.)

C-Bits

M13 framing is a bit outdated because it assumes that DS2 links may be terminated from remote locations, just as DS1s are. In practice, DS2s were not deployed, and as a result exist only within the multiplexer.

This means that the timing issues that require bit stuffing occur only at the M12 stage, and never at the M23 stage. Still, the M13 framing process provides positions for bit stuffing at the M23 stage.

Another framing technique was developed to take advantage of the unused bits. The original purpose of these bits (called *C-bits*) was to signal the presence of bits stuffed at the M23 stage of the multiplexing process. *C-bit framing* uses the C-bits in the DS3 frame differently than originally planned.

One of the benefits of C-bit framing is the inclusion of *far-end block errors* (FEBEs) reporting. FEBEs (pronounced *FEE-bees*) are DS3-specific alarms that indicate the far end of the link has received a C-parity or framing error. Figure 21-2 shows how FEBEs are sent on a DS3 with SONET in the middle of the link.

C-bit framing also allows for the introduction of *far-end out-of-frame* (FEOOF) signals. When a break is detected on the receiving interface of the remote end of the link, it sends a FEOOF signal back to the source. Figure 21-3 shows an example of FEOOFs in action.

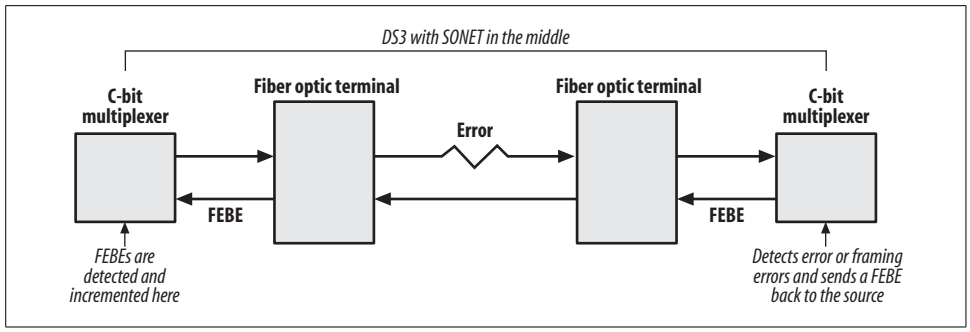


Figure 21-2. Far-end block errors

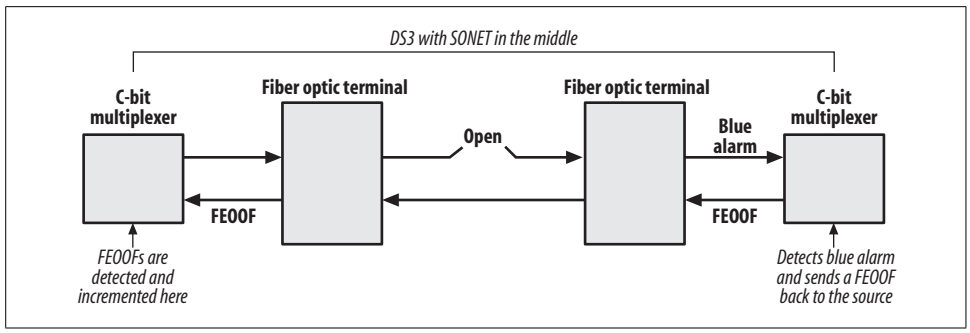


Figure 21-3. Far-end out-of-frame errors

Additional codes, called *far-end alarm and control* (FEAC) codes, are also available, and include:

- DS3 Equipment Failure—Service Affecting (SA)
- DS3 LOS/HBER
- DS3 Out-of-Frame
- DS3 AIS Received
- DS3 IDLE Received
- DS3 Equipment Failure—Non Service Affecting (NSA)
- Common Equipment Failure (NSA)
- Multiple DS1 LOS/HBER
- DS1 Equipment Failure (SA)
- Single DS1 LOS/HBER
- DS1 Equipment Failure (NSA)

FEAC codes are shown in the output of the `show controllers` command.

Clear-Channel DS3 Framing

Data links require the full capacity of the DS3 without any individual channels, as shown in Figure 21-4. Framing is still required, and either C-bit or M13 framing can be used to maintain clock synchronization between the two ends. The M12 and M23 steps are not needed, however, and nor are the overhead bits introduced by them. These bits are used for pure data.

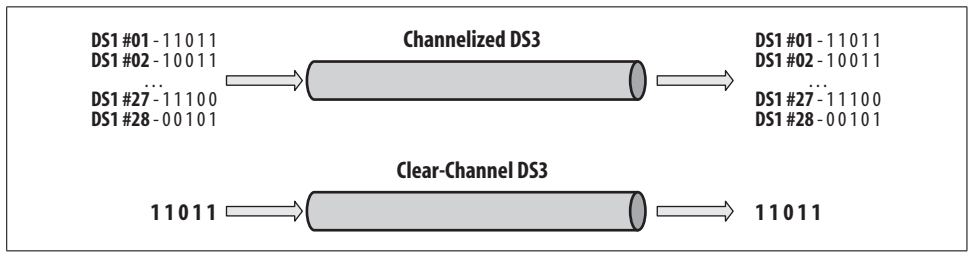


Figure 21-4. Channelized versus clear-channel DS3

Because there's no multiplexing overhead, the amount of bandwidth available over a clear-channel DS3 is 44.2 Mbps.

When using a DS3 for data links, C-bit framing should be used to gain the benefits of increased error reporting outlined previously.

Line Coding

DS-3 links support Alternate Mark Inversion (AMI), Bipolar Three Zero Substitution (B3ZS), and High-Density Bipolar Three (HDB3) line coding. AMI is the same as the AMI used on T1s, discussed in Chapter 20. B3ZS is similar to B8ZS, also discussed in Chapter 20, except that it replaces occurrences of three rather than eight consecutive zeros with a well-known bipolar violation. HDB3 is used primarily in Japan and Europe. The default line coding on Cisco DS3 interfaces is B3ZS. When using channelized DS3s, the line coding may be AMI, depending on how the circuit was ordered. Usually B3ZS is preferred.

Configuring DS3s

Recall that there are two flavors of DS3: clear-channel and channelized. Typically, clear-channel DS3s are used for data links, and channelized DS3s are used for voice links. The configurations in this section assume an integrated CSU/DSU in all cases. If you have an older router that requires an external CSU/DSU, you'll probably have a High-Speed Serial Interface (HSSI—pronounced *hissy*), which still looks like a serial interface within IOS. The difference will be that you cannot configure framing on a HSSI port. Additionally, because the CSU/DSU (which counts and reports these errors) is an external device, you won't be able to see DS3 errors on a HSSI port.

Clear-Channel DS3

Configuring a clear-channel DS3 is a pretty boring affair. Specify the framing with the `framing` interface command, then configure the interface like any other serial interface:

```
interface Serial3/1/0
 description DS3
 ip address 10.100.100.100 255.255.255.252
 framing c-bit
```

Showing the status of the interface is done the same as for any other interface. The errors and counters are generic, and not DS3-specific:

```
7304# sho int s3/1/0
Serial3/1/0 is up, line protocol is up
  Hardware is SPA-4XT3/E3
  Description: DS3
  Internet address is 10.100.100.100/30
  MTU 4470 bytes, BW 44210 Kbit, DLY 200 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation HDLC, crc 16, loopback not set
  Keepalive set (10 sec)
  Last input 00:00:02, output 00:00:01, output hang never
  Last clearing of "show interface" counters 6d03h
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: fifo
  Output queue: 0/40 (size/max)
  5 minute input rate 101000 bits/sec, 179 packets/sec
  5 minute output rate 98000 bits/sec, 170 packets/sec
     81589607 packets input, 2171970011 bytes, 0 no buffer
     Received 61914 broadcasts (229394 IP multicast)
     1072 runts, 0 giants, 0 throttles
              0 parity
     1136 input errors, 10 CRC, 0 frame, 0 overrun, 0 ignored, 54 abort
     80620669 packets output, 1165631313 bytes, 0 underruns
     0 output errors, 0 applique, 0 interface resets
     0 output buffer failures, 0 output buffers swapped out
     0 carrier transitions
```

The real meaty information is found using the `show controllers` command:

```
7304# show controllers s3/1/0
Interface Serial3/1/0 (DS3 port 0)
  Hardware is SPA-4XT3/E3
    Framing is c-bit, Clock Source is Line
    Bandwidth limit is 44210, DSU mode 0, Cable length is 10 feet
    rx FEBE since last clear counter 792, since reset 3693

    No alarms detected.

    No FEAC code is being received
    MDL transmission is disabled

    PXF interface number = 0x12
```

```
SPA carrier card counters:
  Input: packets = 81583462, bytes = 646644147O, drops = 7
  Output: packets = 80614617, bytes = 5460208896, drops = 0
  Egress flow control status: XON
  Per bay counters:
  General errors: input = 0, output = 0
  SPI4 errors: ingress dip4 = 0, egress dip2 = 0

SPA FPGA Packet Counters:
  Transmit : 80614638, Drops : 0
  Receive : 81583490, Drops : 0

SPA FPGA Invalid Channel Packets:
  Transmit : 0, Receive : 0

SPA FPGA IPC Counters:
  Transmit : 1057496, Drops : 0
  Receive : 1057496, Drops : 0

SPA FPGA Packet Error Counters:
  202 Receive error packets

Framer(PM5383) Counters:
  Transmit :  80614555 packets, 1165231422 bytes
  Errors : 0 aborts, 0 underruns

  Receive : 81583399 packets, 2171463422 bytes
  Errors : 10 crc, 1072 runts, 0 giants, 54 aborts
```

This output shows a healthy clear-channel DS3. The framing is C-bit.

Notice that there are FEBEs shown. FEBEs may increment in small numbers over time without serious impact. The preceding output indicates that the counters were cleared six days ago. Since then, 792 FEBEs have accumulated, which translates to about 5.5 per hour (assuming an even distribution). Another possibility is that something happened within the past six days that caused FEBEs to increment in a short amount of time. In this case, it would be a good idea to clear the counters again, and keep an eye on the interface. If FEBEs increment regularly, you might want to start troubleshooting further.

The `show controllers` output shows that there are no errors active, and no FEAC codes have been received. This information indicates a relatively healthy clear-channel DS3.

A known problem with the PA-A3-T3 and NM-1A-T3 modules results in the receiver being too sensitive (see Cisco bug ID CSCds15318). If you're seeing a large number of errors, and you have a short cable, this might be the problem. Short of replacing the interface with a different model, Cisco recommends either reducing the transmit level of the device on the other end of the DS3 cable connected to your router, or installing attenuators on the line. These cards are pretty common, so watch out for this.

Channelized DS3

A channelized DS3 can be configured for voice, data, or both. Because the DS3 is channelized, individual T1s can be broken out as either data links or voice links. This is done in the controller configuration for the interface.

In this example, I've configured the first 10 T1s in the DS3 to be serial data links. I did this by assigning the desired number of DS0s to a *channel group*. Because my links will all be full T1s, I've assigned all 24 DS0s (referenced as *timeslots*) to each channel group. However, because this is a channelized DS3, I could separate each T1 further by grouping DS0s together. Each group would get its own channel group number. Here's the controller configuration:

```
controller T3 2/0
 framing m23
 clock source line
 t1 1 channel-group 1 timeslots 1-24
 t1 2 channel-group 1 timeslots 1-24
 t1 3 channel-group 1 timeslots 1-24
 t1 4 channel-group 1 timeslots 1-24
 t1 5 channel-group 1 timeslots 1-24
 t1 6 channel-group 1 timeslots 1-24
 t1 7 channel-group 1 timeslots 1-24
 t1 8 channel-group 1 timeslots 1-24
 t1 9 channel-group 1 timeslots 1-24
 t1 10 channel-group 1 timeslots 1-24
 t1 1 clock source Line
 t1 2 clock source Line
 t1 4 clock source Line
 t1 6 clock source Line
 t1 7 clock source Line
 t1 8 clock source Line
 t1 9 clock source Line
 t1 10 clock source Line
 t1 11 clock source Line
 t1 12 clock source Line
 t1 13 clock source Line
 t1 14 clock source Line
 t1 15 clock source Line
 t1 16 clock source Line
 t1 17 clock source Line
 t1 18 clock source Line
 t1 19 clock source Line
 t1 20 clock source Line
 t1 21 clock source Line
 t1 22 clock source Line
 t1 23 clock source Line
 t1 24 clock source Line
 t1 25 clock source Line
 t1 26 clock source Line
 t1 27 clock source Line
 t1 28 clock source Line
```

Here, we have a DS3 connected to interface 2/0. This is a channelized DS3, so the framing is set to M23. The clock source defaults to `Line`. Unlike with T1s, this should normally be left alone, as clocking is usually provided from the SONET network upstream. Notice that there is a `clock` statement for the DS3, and additional `clock` statements for each T1 within the DS3.

Cisco only supports M23 and C-bit framing for channelized DS3s. When using M13 on telecom equipment, use M23 on your Cisco gear.

Once the controllers have been configured, the serial T1s can be configured as if they were regular T1s. The serial interfaces are a combination of the physical interfaces and the T1 numbers in the DS3, followed by a colon and the channel group assigned in the controller configuration:

```
interface Serial2/0/1:1
 description T1 #1
 ip address 10.220.110.1 255.255.255.252
!
interface Serial2/0/2:1
 description T1 #2
 ip address 10.220.120.1 255.255.255.252
!
interface Serial2/0/3:1
 description T1 #3
 ip address 10.220.130.1 255.255.255.252
!
interface Serial2/0/4:1
 description T1 #4
 ip address 10.220.140.1 255.255.255.252
```

You cannot create a serial interface larger than a T1 within a channelized DS3. If you need multimegabit speeds, you'll need to create multiple T1s, and either bundle them with Multilink-PPP, or load-balance them using CEF or a routing protocol.

This router (a 7304) has both channelized and clear-channel interface cards. When you do a `show version` on a router like this, the output can be confusing because the number of serial interfaces includes the clear-channel DS3s and any T1s you've configured from your channelized DS3s. This router contains four channelized DS3s and four clear-channel DS3s. If a channelized DS3 is not configured at the controller level, it does not appear in the output of the `show version` command. Because we have four clear-channel DS3s, which are serial interfaces by default, and we've

configured 10 T1s to be serial interfaces out of one of the channelized DS3s, the router reports a total of 14 serial interfaces:

```
7304# sho ver

[- Text Removed -]

1 FastEthernet interface
2 Gigabit Ethernet interfaces
14 Serial interfaces
4 Channelized T3 ports
```

Here, we can see the individual serial interfaces on the router. Ten of them are logical (s2/0/1:1 – s/2/0/10:1), and four of them are physical (s3/1/0 – s3/1/3):

```
7304# sho ip int brie
Interface              IP-Address      OK? Method Status                Protocol
FastEthernet0          unassigned      YES NVRAM  administratively down down
GigabitEthernet0/0     10.220.11.1      YES NVRAM  up                    up
GigabitEthernet0/1     10.220.12.1      YES NVRAM  up                    up
Serial2/0/1:1          10.220.110.1     YES NVRAM  up                    up
Serial2/0/2:1          10.220.120.1     YES NVRAM  up                    up
Serial2/0/3:1          10.220.130.1     YES NVRAM  up                    up
Serial2/0/4:1          10.220.140.1     YES NVRAM  up                    up
Serial2/0/5:1          unassigned      YES manual administratively down down
Serial2/0/6:1          unassigned      YES manual administratively down down
Serial2/0/7:1          unassigned      YES manual administratively down down
Serial2/0/8:1          unassigned      YES manual administratively down down
Serial2/0/9:1          unassigned      YES manual administratively down down
Serial2/0/10:1         unassigned      YES manual administratively down down
Serial3/1/0            10.100.100.100  YES manual up                    up
Serial3/1/1            unassigned      YES NVRAM  down                  down
Serial3/1/2            unassigned      YES NVRAM  down                  down
Serial3/1/3            unassigned      YES NVRAM  down                  down
```

The output of the `show controllers` command for a channelized DS3 is quite different from that for a clear-channel DS3. With a channelized DS3, you get a report of the line status for every 15-minute interval in the last 24 hours. The current alarm status, and the framing and line coding are shown here in bold:

```
7304# sho controllers t3 2/0
T3 2/0 is up. Hardware is 2CT3+ single wide port adapter
  CT3 H/W Version: 0.2.2, CT3 ROM Version: 1.0, CT3 F/W Version: 2.5.1
  FREEDM version: 1, reset 0 resurrect 0
  Applique type is Channelized T3
  Description:
  No alarms detected.
  Framing is M23, Line Code is B3ZS, Clock Source is Line
  Rx-error throttling on T1's ENABLED
  Rx throttle total 99, equipment customer loopback
  Data in current interval (29 seconds elapsed):
     0 Line Code Violations, 0 P-bit Coding Violation
     0 C-bit Coding Violation, 0 P-bit Err Secs
```

```
     0 P-bit Severely Err Secs, 0 Severely Err Framing Secs
     0 Unavailable Secs, 0 Line Errored Secs
     0 C-bit Errored Secs, 0 C-bit Severely Errored Secs
  Data in Interval 1:
     0 Line Code Violations, 0 P-bit Coding Violation
     0 C-bit Coding Violation, 0 P-bit Err Secs
     0 P-bit Severely Err Secs, 0 Severely Err Framing Secs
     0 Unavailable Secs, 0 Line Errored Secs
     0 C-bit Errored Secs, 0 C-bit Severely Errored Secs
  Data in Interval 2:
     0 Line Code Violations, 0 P-bit Coding Violation
     0 C-bit Coding Violation, 0 P-bit Err Secs
     0 P-bit Severely Err Secs, 0 Severely Err Framing Secs
     0 Unavailable Secs, 0 Line Errored Secs
     0 C-bit Errored Secs, 0 C-bit Severely Errored Secs

[- Text Removed -]
```

CHAPTER 22
Frame Relay

Frame relay is a method of transporting digital information over a network. The data is formatted into frames, which are sent over a network of devices usually under the control of a telecommunications company. Diagrams depicting frame-relay networks often display the network as a cloud, as the end user doesn't generally know (or care) how the network is actually designed. The end user only needs to know that *virtual circuits* through the cloud will allow the delivery of frames to other end points in the cloud.

Whatever goes into one end of the virtual circuit should come out the other end. The far end appears as though it is on the other end of a physical cable, though in reality, the remote end is typically many hops away.

The virtual circuits may be either *switched* or *permanent*. A frame-relay permanent virtual circuit (PVC) is always up, even if it's not in use. A switched virtual circuit (SVC) is up only when it's needed. Most data networking deployments use PVCs.

Figure 22-1 shows a typical simple frame-relay network using PVCs. Router A is connected to Router B with a PVC. Router A is also connected to Router C with a PVC. Router B and Router C are not connected to each other. The two PVCs terminate into a single interface on Router A.

Physically, each router is connected only to one of the provider's telecom switches. These switches are the entry and exit points into and out of the cloud.

Router A is connected with a DS3, while Routers B and C are connected to the cloud with T1s. Logically, frame relay creates the illusion that the routers on the far sides of the PVCs are directly connected.

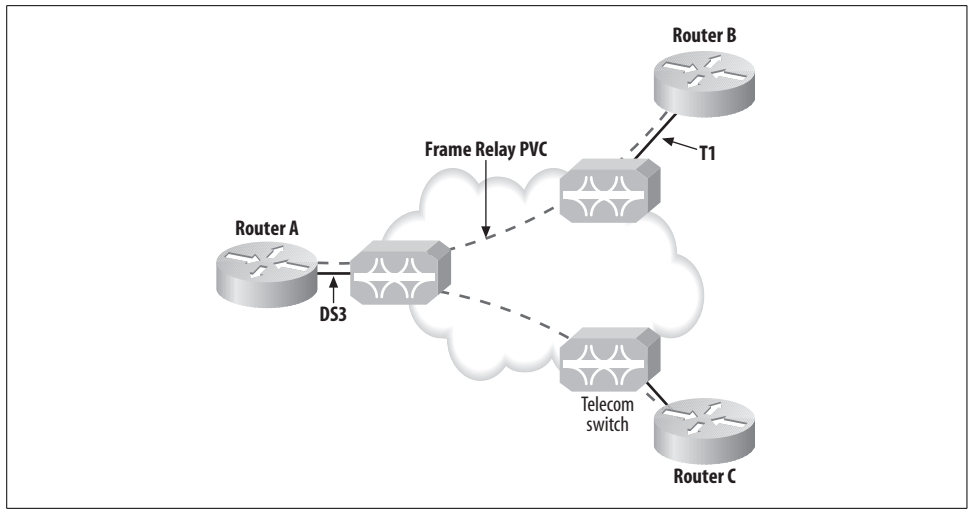


Figure 22-1. Simple frame-relay WAN

In reality, there may be many devices in the cloud, any of which may be forwarding frames to the destination. Figure 22-2 shows what the inside of the cloud might look like.

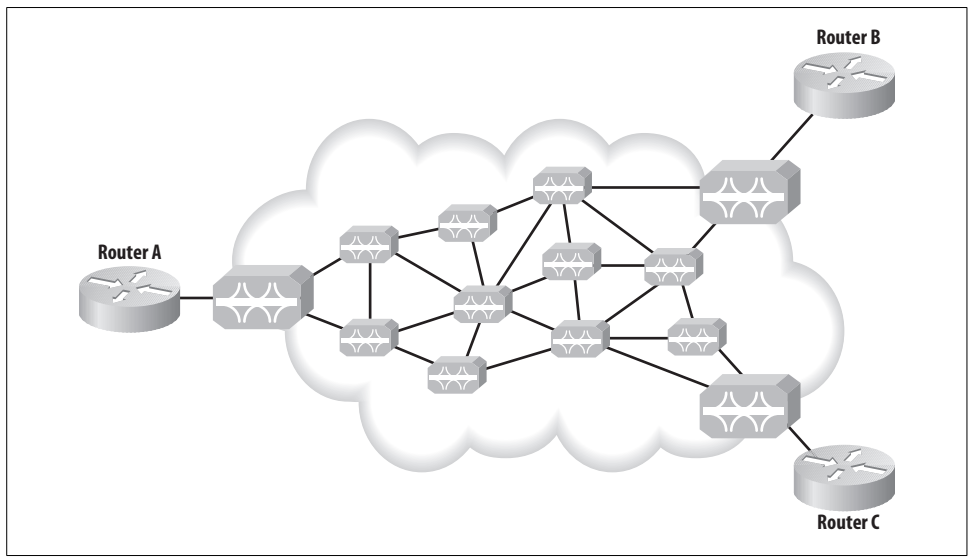


Figure 22-2. Frame-relay physical network

Frame relay functions in a manner similar to the Internet. When you send a packet to a remote web site, you don't really care how it gets there. You have no idea how many devices your packets may pass through. All you know is what your default gateway is; you let your Internet service provider worry about the rest. Frame relay is similar in that you have no idea what the intermediary devices are, or how they route your data. When using frame relay, there are a finite number of destinations, all specified by you and provisioned by your telecom provider.

For frame relay to create the illusion of a direct connection, each end of the virtual circuit is given a layer-2 address called a *data link control identifier* (DLCI, pronounced *dell-see*). These DLCIs (and your data) are visible only to the telecom switches that forward the frames and to your routers. Other customers connected to the frame-relay cloud cannot see your DLCIs or your data. Your telecom provider will determine the DLCI numbers.

Virtual circuits can be combined in such a way that multiple end points terminate to a single DLCI. An example of this type of design is shown in Figure 22-3. Each virtual circuit can also have its own DLCI on both sides. How you design the frame-relay PVCs will depend on your needs.

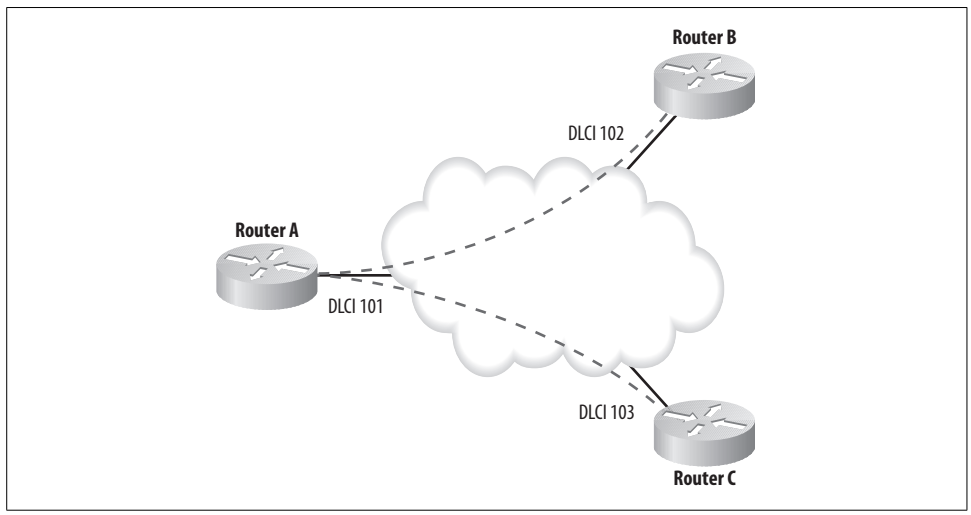
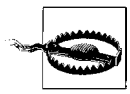


Figure 22-3. DLCIs in a frame-relay network

DLCIs are mapped to IP addresses in frame-relay networks in much the same way that Ethernet MAC addresses are mapped to IP addresses in Ethernet networks. Unlike with Ethernet, however, which learns MAC addresses dynamically, you'll usually have to map DLCIs to IP addresses yourself.

Ethernet networks use the *Address Resolution Protocol* (ARP) to determine how MAC addresses map to known IP addresses. Frame relay uses a protocol called *Inverse ARP* to try to map IP addresses to known DLCIs. Frame-relay switches report on the statuses of all configured DLCIs on a regular basis.



Be careful when configuring frame relay. There may be PVCs configured that you do not wish to enable, and Inverse ARP may enable those links without your knowledge.

The primary benefits of frame relay are cost and flexibility. A point-to-point T1 will cost more than a frame-relay T1 link between two sites, especially if the sites are not in the same geographic location, or LATA. Also, with a point-to-point T1, 1.5 Mbps of dedicated bandwidth must be allocated between each point, regardless of the utilization of that bandwidth. In contrast, a frame-relay link shares resources. On a frame-relay link, if bandwidth is not being used in the cloud, other customers can use it.

The notion of shared bandwidth raises some questions. What if someone else is using the bandwidth, and you suddenly need it? What if you're using the bandwidth, and someone else suddenly needs it? Frame relay introduces the idea of the *committed information rate* (CIR), which helps address these concerns.

Ordering Frame-Relay Service

When you order a frame-relay link, telco needs to know four pieces of information to provision the circuit. One of these is the CIR. Here are the requirements:

The addresses and phone numbers of the end points
: The street addresses—and, more importantly, phone numbers—in use at each end point are critical components of a circuit order. If the location does not have phone service, in most cases, it won't exist to telco.

The port speed
: The *port speed* is the size of the physical circuit you will deliver to each end. These physical links do not have to be the same size or type, but they must both be able to support the CIR requested for the frame-relay link. The port speed can be anything from a 56 Kbps DDS circuit up to and exceeding a DS3. These days, the most common frame-relay handoff is a full T1, though fractional T1s are still used as well. It all depends on the cost of the physical circuit.

The committed information rate (CIR)

The CIR is the rate of transfer that you want the carrier to provide. When requesting frame-relay service, you specify the amount of bandwidth you want to be available, and the provider guarantees that up to this level, all frames that are sent over this virtual circuit be forwarded through the frame-relay cloud to their intended destinations (additional frames may be dropped). The higher the CIR, the more the service will cost. A CIR is required for each virtual circuit.

The burst rate

The burst rate is the maximum speed of the frame-relay virtual circuit. Frames that exceed the CIR—but not the burst rate—are marked *discard-eligible* (DE). DE frames will be forwarded by the switches in the frame-relay cloud as long as there is sufficient bandwidth to do so, but may be dropped at the discretion of the frame-relay provider. Having a high burst rate is an inexpensive way to get more bandwidth for a lower cost. The burst rate is often a multiple of the CIR; you may hear the burst rate referred to as a "2X burst," which means the burst rate is twice the CIR. You can also order a virtual circuit with *zero burst*. A burst rate is required for each virtual circuit.

Figure 22-4 shows a bandwidth utilization graph for a frame-relay link. The link is running locally over a T1 with a port speed of 1.5 Mbps. The CIR is 512 Kbps, and the link has been provisioned with a 2X burst. As you can see in the graph, as traffic exceeds the 2X burst threshold, it is discarded. A graph such as this indicates that the link is saturated and needs to be upgraded. A good rule of thumb is to order more bandwidth when your link reaches 70 percent utilization. This circuit should have been upgraded long ago.

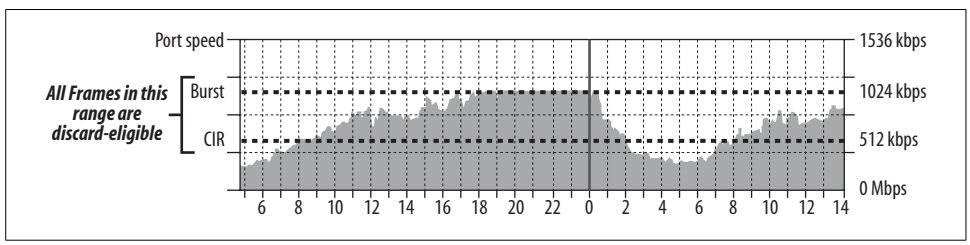


Figure 22-4. Frame-relay CIR and DE frames

Frame-Relay Network Design

Frame-relay links are more flexible than point-to-point links because multiple links can be terminated at a single interface in a router. This leads to design possibilities allowing connectivity to multiple sites at a significant cost savings over point-to-point circuits.

Figure 22-5 shows three sites networked together with frame relay. On the left, Router B and Router C are both connected to Router A, but are not connected to each other. This design is often referred to as a *partial mesh* or *hub and spoke* network. In this network, Router B can communicate to Router C only through Router A.

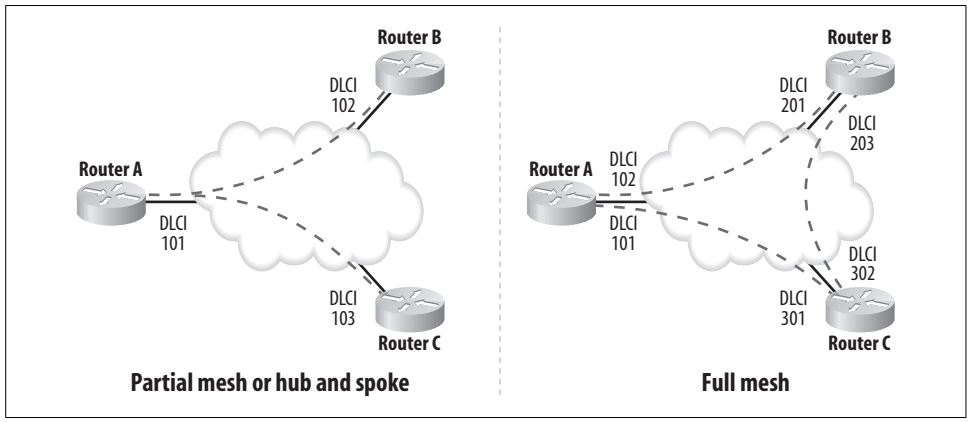


Figure 22-5. Meshed frame-relay networks

On the right side of Figure 22-5 is an example of a *fully meshed* network. The difference here is that all sites are connected to all other sites. Router B can communicate directly with Router C in the fully meshed network.

Meshed networks are not strictly the domain of frame relay. As you can see in Figure 22-6, a fully meshed network can easily be created with point-to-point T1s.

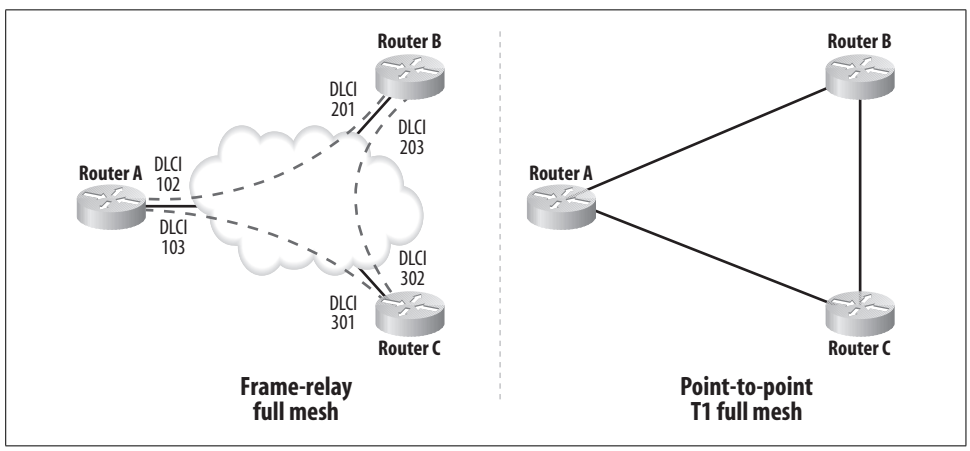


Figure 22-6. Frame-relay versus point-to-point T1 meshed networks

In a frame-relay network like the one shown on the left side of Figure 22-6, each location needs a router that can support a single T1. Each one of the PVCs can be configured as a separate virtual interface called a *subinterface*. Subinterfaces allow VCs to terminate into separate logical interfaces within a single physical interface.

With point-to-point links (T1s in this case), each router must be able to support two T1 interfaces. Routers that support two T1s are generally more expensive than single-T1 routers. Additionally, point-to-point T1s cost more than frame-relay T1 services, especially over long distances.

The example in Figure 22-6 is relatively simple, but what about larger networks? Figure 22-7 shows two networks, each with six nodes. On the left is a fully meshed network using frame relay, and on the right is a fully meshed network using point-to-point T1s.

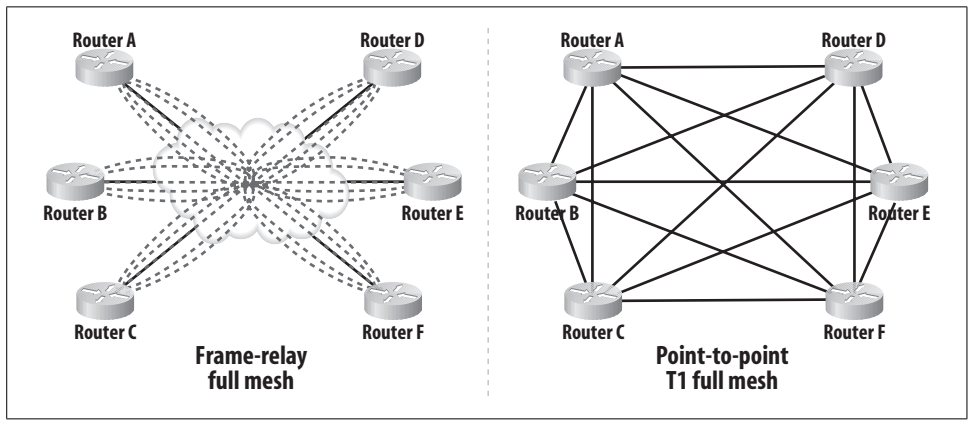


Figure 22-7. Six-node fully meshed networks

With six nodes in the network, there must be five links on each router. With frame relay, this can be accomplished with a single T1 interface at each router, provided the bandwidth for all the links will total that of a T1 or less. When using point-to-point links, however, routers that can support five T1s are required. In addition to the hardware costs, the telecom costs for a network like this would be very high, especially over longer distances.

To figure out how many links are required to build a fully meshed network, use the formula N (N – 1) / 2, where N equals the number of nodes in the mesh. In our example, there are six nodes, so 6 (6 – 1) / 2 = 15 links are required.

Oversubscription

When designing frame-relay networks, care must be taken to ensure that the total amount of bandwidth being provisioned in all CIRs within a physical link does not exceed the port speed. A CIR represents guaranteed bandwidth, and it is technically impossible to guarantee bandwidth beyond the port speed. A frame-relay link is considered *oversubscribed* when the total bandwidth of all the virtual circuits within the link exceeds the port speed of the link. For example, having four PVCs each with a 512 Kbps CIR is possible on a T1, even though the total of all the CIRs is 2,048 Kbps. Some providers will allow this, while others will not.

The burst rate has no bearing on the oversubscription of a link.

Careful planning should always be done to ensure that the CIRs of your PVCs total no more than the port speed of your physical link. I use spreadsheets to keep me honest, but any form of documentation will do. Often, simple charts like the one in Figure 22-8 are the most effective.

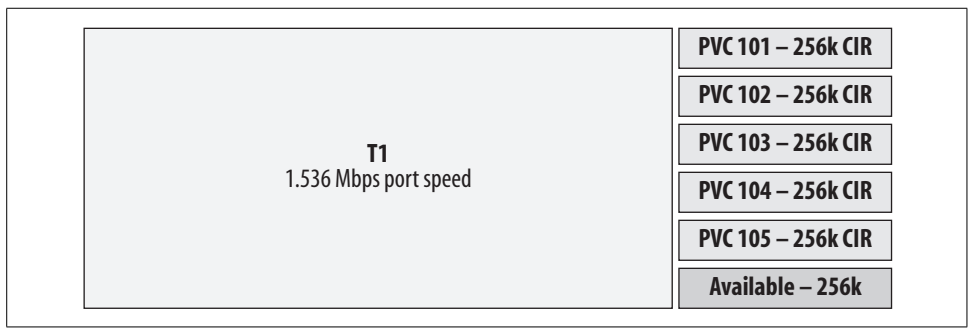


Figure 22-8. Subscription of a T1 using frame-relay PVCs

There are no technical limitations preventing oversubscription. During the Internet boom, small ISPs often debated the ethics of oversubscription. Many ISPs routinely oversubscribed their frame-relay links to save money and thereby increase profits. Oversubscription is never good for customers, though: eventually, usage will increase to the point where packets are dropped, even though you've committed to delivering them.

Be careful when ordering links from telecom providers. There are some providers who only provide 0 Kbps CIR frame-relay links. They do this to provide links at lower costs than their competitors. The drawback is that all data sent over these links will be discard-eligible. Be specific when ordering, and know what you are buying.

Local Management Interface (LMI)

In 1990, Cisco Systems, StrataCom, Northern Telecom, and Digital Equipment Corporation developed a set of enhancements to the frame-relay protocol called the *Local Management Interface*. LMI provides communication between the data terminal equipment, or DTE devices (routers, in our examples), and the data communication equipment, or DCE devices (telecom switches, in our examples). One of the most useful enhancements that LMI provides is the exchange of status messages regarding virtual circuits (VCs). These messages tell routers when a frame-relay PVC is available.

LMI messages are sent on a predefined PVC. The LMI type and PVC in use can be seen with the `show interface` command:

```
Router-A# sho int s0/0
Serial0/0 is up, line protocol is up
  Hardware is PowerQUICC Serial
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation FRAME-RELAY, loopback not set
  Keepalive set (10 sec)
  LMI enq sent  85, LMI stat recvd 86, LMI upd recvd 0, DTE LMI up
  LMI enq recvd 0, LMI stat sent  0, LMI upd sent  0
  LMI DLCI 1023  LMI type is CISCO  frame relay DTE
  FR SVC disabled, LAPF state down
  Broadcast queue 0/64, broadcasts sent/dropped 0/0, interface broadcasts 0
  Last input 00:00:03, output 00:00:03, output hang never
  Last clearing of "show interface" counters 00:30:11
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: weighted fair
  Output queue: 0/1000/64/0 (size/max total/threshold/drops)
     Conversations  0/1/256 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
     Available Bandwidth 1158 kilobits/sec
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     86 packets input, 1758 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     1 input errors, 0 CRC, 1 frame, 0 overrun, 0 ignored, 0 abort
     88 packets output, 1145 bytes, 0 underruns
     0 output errors, 0 collisions, 1 interface resets
     0 output buffer failures, 0 output buffers swapped out
     2 carrier transitions
     DCD=up  DSR=up  DTR=up  RTS=up  CTS=up
```

Three forms of LMI are configurable on Cisco routers: `cisco`, `ansi`, and `q933a` (Annex A). The DCE device (telecom switch) usually determines the type of LMI. The default LMI type on Cisco routers is `cisco`. The LMI type can be changed with the `frame-relay lmi-type` interface command:

```
Router-A(config-if)# frame-relay lmi-type ?
  cisco
  ansi
  q933a
```

Congestion Avoidance in Frame Relay

Frame relay includes provisions for congestion avoidance. Included in the frame-relay header are two bits titled *Forward-Explicit Congestion Notification* (FECN, pronounced *FECK-en*), and *Backward-Explicit Congestion Notification* (BECN, pronounced *BECK-en*). These flags are used to report congestion to the DTE devices (your routers).

The DCE devices (telecom switches) set the FECN bit when network congestion is found. When the receiving DTE device (your router) receives the FECN, it can then execute flow control, if so configured. The frame-relay cloud does not perform any flow control; this is left up to the DTE devices on each end.

The frame-relay switches set the BECN bit in frames when FECNs are found in frames traveling in the opposite direction. This allows the sending DTE device to know about congestion in the frames it is sending.

Figure 22-9 shows a frame-relay network where congestion is occurring. A PVC exists between Router A and Router B. The PVC traverses the topmost frame-relay switches in the drawing. Halfway through the cloud, a switch encounters congestion in Router B's direction. The switch marks packets moving forward (toward Router B) with FECNs, and packets moving in the opposite direction (toward Router A) with BECNs.

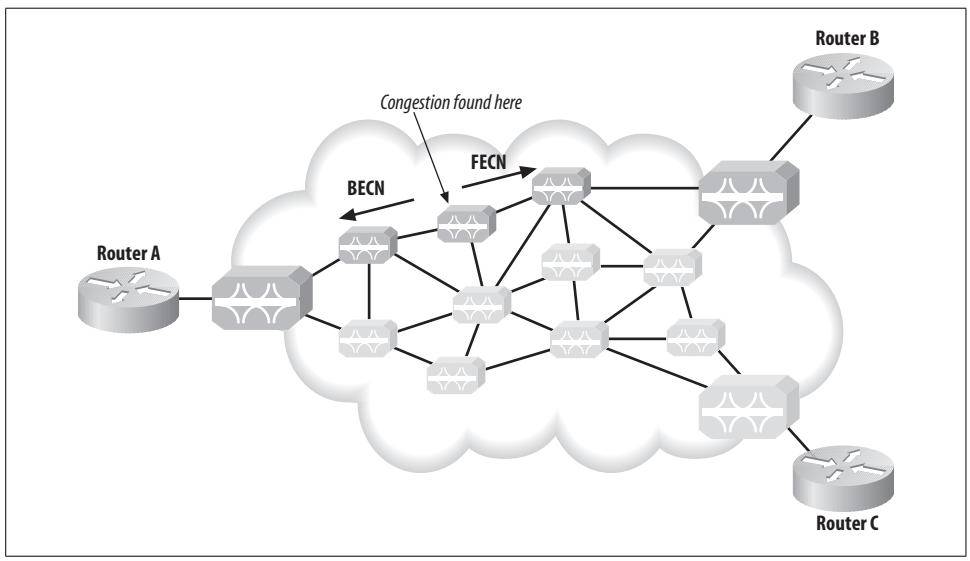


Figure 22-9. FECN and BECN example

Configuring Frame Relay

Once you understand how frame relay works, the mechanics of configuration are not very difficult. There are some interesting concepts, such as subinterfaces, that may be new to you; we'll cover those in detail here.

Basic Frame Relay with Two Nodes

Figure 22-10 shows a simple two-node frame-relay network. Router A is connected to Router B using frame relay over a T1. The port speed is 1.536 Mbps, the CIR is 512 Kbps, and the burst rate is 2X (1,024 Kbps).

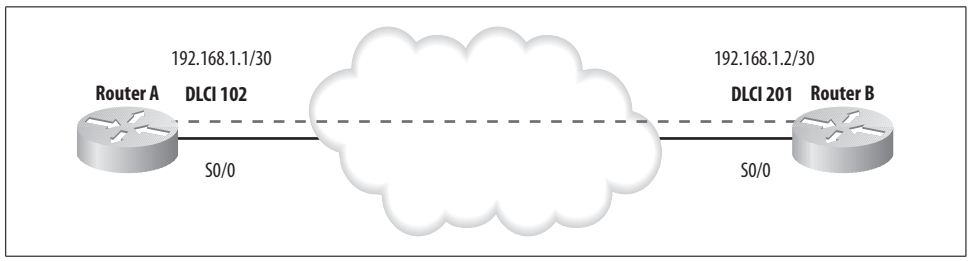


Figure 22-10. Two-node frame-relay network

The first step in configuring frame relay is to configure frame-relay encapsulation. There are two types of frame-relay encapsulation: `cisco` and `ietf`. The default type is `cisco`, which is configured with the `encapsulation frame-relay` command:

```
interface Serial0/0
encapsulation frame-relay
```

The `ietf` type is configured with the `encapsulation frame-relay ietf` command. `ietf` frame-relay encapsulation is usually used only when connecting Cisco routers to non-Cisco devices.

Once you've configured frame-relay encapsulation, and the interface is up, you should begin seeing LMI status messages. If the PVC has been provisioned, you can see it with the `show frame-relay PVC` command:

```
Router-A# sho frame pvc

PVC Statistics for interface Serial0/0 (Frame Relay DTE)

             Active     Inactive      Deleted       Static
  Local         0            0            0            0
  Switched      0            0            0            0
  Unused        0            1            0            0

DLCI = 102, DLCI USAGE = UNUSED, PVC STATUS = INACTIVE, INTERFACE = Serial0/0

  input pkts 0             output pkts 0            in bytes 0
  out bytes 0              dropped pkts 0           in pkts dropped 0
  out pkts dropped 0                out bytes dropped 0
```

```
  in FECN pkts 0           in BECN pkts 0           out FECN pkts 0
  out BECN pkts 0          in DE pkts 0             out DE pkts 0
  out bcast pkts 0         out bcast bytes 0
  switched pkts 0
  Detailed packet drop counters:
  no out intf 0            out intf down 0          no out PVC 0
  in PVC down 0            out PVC down 0           pkt too big 0
  shaping Q full 0         pkt above DE 0           policing drop 0
  pvc create time 00:19:12, last time pvc status changed 00:19:12
```

Notice that the information being sent relates status information about the local DLCI, not the remote side. LMI always reports on information critical to the local rather than the remote side of the link. The same output on Router B should show a similar report detailing the link from Router B's point of view:

```
Router-B# sho frame pvc

PVC Statistics for interface Serial0/0 (Frame Relay DTE)

              Active     Inactive      Deleted       Static
  Local          0            0            0            0
  Switched       0            0            0            0
  Unused         0            1            0            0

DLCI = 201, DLCI USAGE = UNUSED, PVC STATUS = ACTIVE, INTERFACE = Serial0/0

  input pkts 0             output pkts 0            in bytes 0
  out bytes 0              dropped pkts 0           in pkts dropped 0
  out pkts dropped 0                out bytes dropped 0
  in FECN pkts 0           in BECN pkts 0           out FECN pkts 0
  out BECN pkts 0          in DE pkts 0             out DE pkts 0
  out bcast pkts 0         out bcast bytes 0
  switched pkts 0
  Detailed packet drop counters:
  no out intf 0            out intf down 0          no out PVC 0
  in PVC down 0            out PVC down 0           pkt too big 0
  shaping Q full 0         pkt above DE 0           policing drop 0
  pvc create time 00:19:08, last time pvc status changed 00:19:08
```

Once you see that LMI is active, you can assign IP addresses to the interfaces like you would on any other type of interface. Here is the IP address configuration for Router A:

```
Router-A(config-if)# int s0/0
Router-A(config-if)# ip address 192.168.1.1 255.255.255.252
```

And here is the IP address configuration for Router B:

```
Router-B(config-if)# int s0/0
Router-B(config-if)# ip address 192.168.1.2 255.255.255.252
```

At this point, you can ping across the link:

```
Router-A# ping 192.168.1.2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.1.2, timeout is 2 seconds:
.!!!!
```

Ping works because the router has determined from the IP subnet mask that this is a point-to-point link.

To see the status of the PVC and which IP address has been mapped, use the `show frame-relay map` command:

```
Router-A# sho frame map
Serial0/0 (up): ip 192.168.1.2 dlci 102(0x66,0x1860), dynamic,
              broadcast,, status defined, active
```

Basic Frame Relay with More Than Two Nodes

Figure 22-11 shows a slightly more complex frame-relay network. There are three routers in this network. Router A has a PVC to Router B, and another PVC to Router C. Router B does not have a direct connection to Router C.

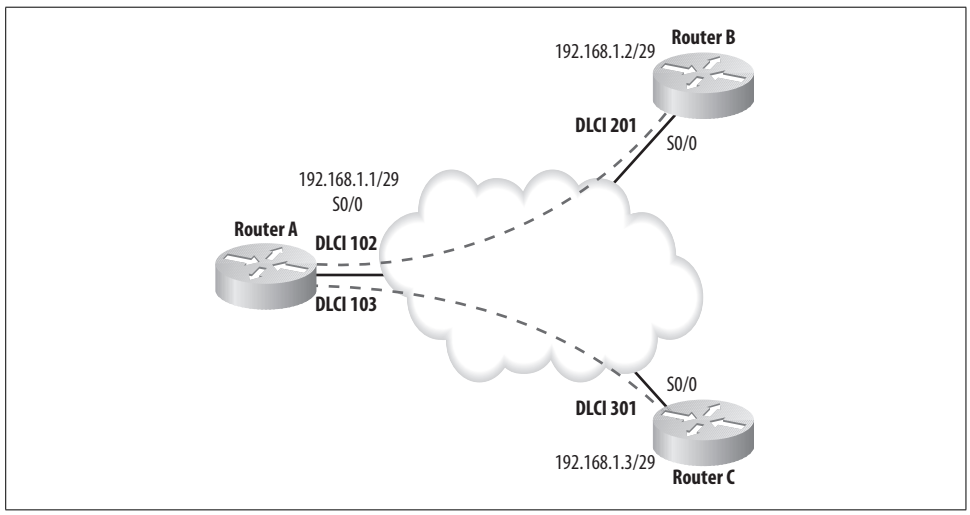


Figure 22-11. Three-node frame-relay network

To accomplish this design, as in the previous example, you'd begin by configuring frame-relay encapsulation. This step is the same on all routers:

```
interface Serial0/0
 encapsulation frame-relay
```

The IP address configuration is nearly the same as well; the only difference is the subnet masks. Here are the configurations for the three routers:

- Router A:

```
Router-A(config)# int s0/0
Router-A(config-if)# ip address 192.168.1.1 255.255.255.248
```

- Router B:

```
Router-B(config)# int s0/0
Router-B(config-if)# ip address 192.168.1.2 255.255.255.248
```

- Router C:

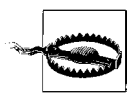
```
Router-C(config)# int s0/0
Router-C(config-if)# ip address 192.168.1.3 255.255.255.248
```

Performing these steps gives you a live frame-relay network, which you can test as follows:

```
Router-A# ping 192.168.1.2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.1.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 56/57/60 ms

Router-A# ping 192.168.1.3

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.1.3, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 56/57/60 ms
```

When there were only two nodes on the network, the routers were each able to determine the IP address of the far side by nature of the subnet mask. That's not the case here because there are more than two nodes.

When the subnet mask is other than 255.255.255.252, the routers will use Inverse ARP to determine what IP address belongs to what DLCI.



Beware of Inverse ARP on complex frame-relay networks. Inverse ARP will discover all PVCs and their IP addresses, if they exist. This can cause links that you may not expect or want to come online. Inverse ARP can be disabled with the `no frame-relay inverse-arp` command.

A better option for IP address–to–DLCI mappings is mapping them by hand. This is done with the `frame-relay map` interface command:

```
Router-A(config-if)# frame-relay map  ip 192.168.1.2 102 broadcast
Router-A(config-if)# frame-relay map  ip 192.168.1.3 103 broadcast
```

A mapping determined by Inverse ARP is considered a dynamic map, while a configured map is considered a static map.

Remember that each router only sees its own side of the PVC, so you are mapping the remote IP address to the local DLCI. Think of the local DLCI as pointing to the remote router. Take a look at the commands for Router B and Router C, and you'll see what I mean:

- Router B:

```
Router-B(config-if)# frame-relay map ip 192.168.1.1 201 broadcast
```

- Router C:

```
Router-C(config-if)# frame-relay map ip 192.168.1.1 301 broadcast
```

At the end of each `frame-relay map` command, you'll notice the keyword `broadcast`. This keyword should be included any time you execute this command. The `broadcast` keyword maps broadcasts and multicasts over the PVC as well as unicasts. Broadcasts and multicasts are an integral part of most routing protocols, so if you have a frame-relay WAN up, and you can't figure out why EIGRP or OSPF isn't establishing adjacencies, check to make sure you've included the `broadcast` keyword in your `map` statements.

The way IP-DLCI mapping works includes one odd little side effect. With a network configured as you'd expect, you cannot ping your own frame-relay interface:

```
Router-A# ping 192.168.1.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.1.1, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
```

This can burn you when troubleshooting because you may expect to be able to ping your own interface like you can with Ethernet. You cannot ping your own interface because there is no predefined layer-2 address. While Ethernet interfaces have permanent MAC addresses, frame-relay interfaces do not. With frame relay, all layer-2 addresses are configured manually.

To be able to ping yourself, you must map your own IP address to a remote router. As odd as this sounds, it will work—as soon as the packet arrives at the remote router, that router will send it back because it has a mapping for your IP address. Because there is no local layer-2 address, only a DLCI being advertised by the frame cloud, this is the only way to make it work. Beware that ping times to your local frame-relay interface will actually reflect the round-trip time for the PVC you specify in the mapping. Here's an example:

```
Router-A(config-if)# frame-relay map ip 192.168.1.1 102

Router-A# ping 192.168.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 112/112/112 ms
```

Notice the ping time compared with the previous example, where the remote side was pinged. Pinging yourself will take twice as long as pinging the remote side.

Local IP addresses mapped to remote DLCIs will not be locally available should the remote router fail, or the PVC become unavailable. In this example, should Router B fail, Router A will no longer be able to ping its own S0/0 IP address, though Router A will still be able to communicate with Router C.

As a matter of best practice, I like to always map my DLCIs to IP addresses, even if the router can do it reliably on its own. Placing the DLCI information in the configuration makes the config easier to read and troubleshoot.

Frame-Relay Subinterfaces

Sometimes, having two PVCs terminating on a single interface is not what you want, but, as we've seen, having a physical interface for each PVC is not beneficial for cost reasons. For example, with the network in Figure 22-11, each PVC terminates into a single interface. If you ran a routing protocol on these routers, Router B would advertise itself, but Router A would not advertise this route out to Router C because of the split-horizon rule. Splitting the PVCs into separate interfaces would allow the routing protocol to advertise the route, because the split-horizon rule would no longer apply.

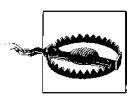
Cisco routers have a feature called *subinterfaces* that solves this problem. In a nutshell, you're able to configure virtual interfaces for each PVC. These virtual interfaces are named after the physical interfaces on which they are found. For example, a subinterface derived from S0/0 might be called S0/0.100. The subinterface number is user-definable, and can be within the range of 1 to 4,294,967,293. I like to name subinterfaces according to the DLCIs mapped to them.

There are two types of subinterfaces: *point-to-point* and *multipoint*. Point-to-point subinterfaces can have only one DLCI active on them, while multipoint subinterfaces can have many. Multipoint subinterfaces behave in much the same way that physical interfaces do: you can have a mix of point-to-point and multipoint subinterfaces on a physical interface. It is even possible to have some DLCIs assigned to subinterfaces, and others to the physical interface.

As mentioned earlier, one of the main benefits of frame-relay subinterfaces is the elimination of split-horizon issues with routing protocols. Creating multiple point-to-point subinterfaces, and assigning each of the PVCs to one of them, enables each PVC to be considered a different interface. Subinterfaces are created with the global `interface` command. Specify the name you'd like the subinterface to have, along with the keyword `point-to-point` or `multipoint`:

```
Router-A(config)# int s0/0.102 point-to-point
Router-A(config-subif)#
```

You're now in interface configuration mode for the newly created subinterface, and can configure this subinterface as you would a physical interface.



Be careful when you choose your subinterface type. If you choose the wrong type by mistake, the only way to change it is to negate the defining command in the configuration, save the config minus the subinterface, and reboot the router. As a result, the following error message will be displayed when you remove a frame-relay subinterface: `Not all config may be removed and may reappear after reactivating the sub-interface.`

You now need to assign specific virtual circuits to the subinterface. This can be done with the `frame-relay interface-dlci` subinterface command, or by mapping a DLCI to a layer-3 address with the `frame-relay map` subinterface command. If you're adding a subinterface after you've already configured the DLCI on the physical interface, you'll need to remove the maps on the physical interface before proceeding.

Mapping an IP address to a VC is a little different when using subinterfaces and the `interface-dlci` command:

```
interface Serial0/0.102 point-to-point
 frame-relay interface-dlci 102 protocol ip 192.168.1.2
```

I like this method because it shows that you've assigned the DLCI to the subinterface, and that you've mapped it to an IP address. If you just use the `map` statement, it doesn't seem as obvious. Still, either way is acceptable.

On point-to-point subinterfaces, you don't really need to map IP addresses to DLCIs, as the router will know that the far end is the only other IP address available (assuming a network mask of 255.255.255.252). Remember that if you make point-to-point links with subinterfaces, each PVC will now require its own IP network.

Figure 22-12 shows the same network as Figure 22-11, only this time with each of the PVCs assigned to specific frame-relay subinterfaces.

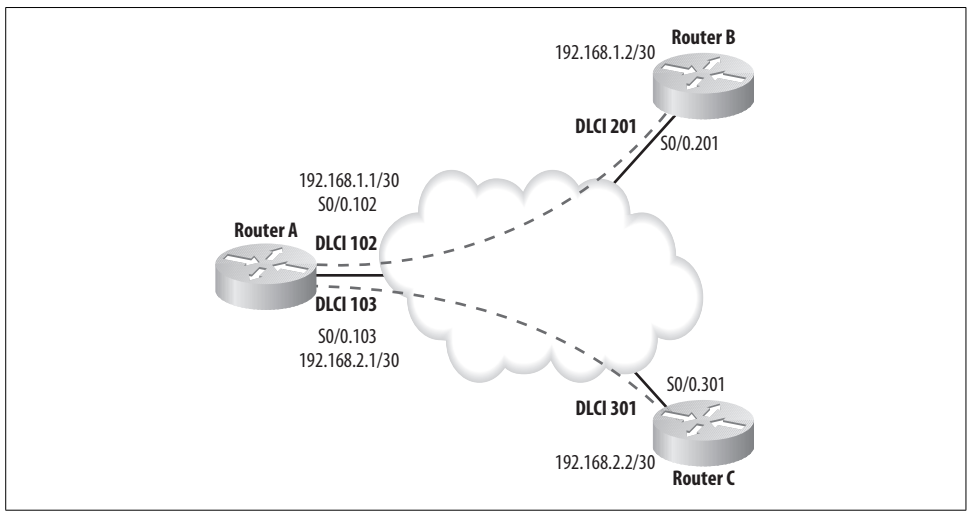


Figure 22-12. Three-node frame-relay network with subinterfaces

Routers B and C don't technically need subinterfaces in this scenario, but if you configured them on the physical interface, you'd need to change the configuration if you later added a PVC between Routers B and C. Configuring the subinterfaces now will potentially make life easier in the future.

Here are the configurations for the three routers:

- Router A:

```
interface Serial0/0
 no ip address
 encapsulation frame-relay
!
interface Serial0/0.102 point-to-point
 ip address 192.168.1.1 255.255.255.252
 frame-relay interface-dlci 102 protocol ip 192.168.1.2
!
interface Serial0/0.103 point-to-point
 ip address 192.168.2.1 255.255.255.252
 frame-relay interface-dlci 103 protocol ip 192.168.2.3
```

- Router B:

```
interface Serial0/0
 no ip address
 encapsulation frame-relay
!
interface Serial0/0.201 point-to-point
 ip address 192.168.1.2 255.255.255.252
 frame-relay interface-dlci 201 protocol ip 192.168.1.1
```

- Router C:

```
interface Serial0/0
 no ip address
 encapsulation frame-relay
!
interface Serial0/0.301 point-to-point
 ip address 192.168.2.2 255.255.255.252
 frame-relay interface-dlci 301 protocol ip 192.168.2.1
```

Troubleshooting Frame Relay

Troubleshooting frame relay is quite simple once you understand how it works. Remember that most of the information regarding PVCs is delivered from the DCE device, which is usually the telecom switch on the far end of your physical link.

The key to any troubleshooting process is *problem isolation*. You need to determine where the problem lies so you can determine a corrective course of action. Follow these steps, and you'll quickly determine where the trouble is:

Physical layer first!
: Is the cable plugged in? Is the cable a known good cable? Is the cable on the other end plugged in? This may sound silly, but you'll feel pretty foolish if you call Cisco for help only to find that the cause of your woes was an unplugged cable.

Is the serial link up?
: Make sure your serial link is up via a `show interface`. Leaving an interface in a shut-down state has a tendency to prevent traffic from being passed over it.

Are you receiving LMI?

Remember that LMI is sent from your locally connected telecom device. If you're not receiving LMI, you're not getting status messages regarding your VCs, so the router will not know that they exist. There are a couple of ways to see whether you're receiving LMI:

`show interface`

The output from a `show interface` command for a frame-relay-encapsulated interface will include LMI counters. LMI updates are received every 10 seconds, so executing the command and then waiting 10 seconds or more and executing the command again should show an increase in the LMI counters:

```
Router-A# sho int s0/0 | include LMI
  LMI enq sent  186, LMI stat recvd 186, LMI upd recvd 0, DTE LMI up
  LMI enq recvd 0, LMI stat sent  0, LMI upd sent  0
  LMI DLCI 1023  LMI type is CISCO  frame relay DTE
Router-A#
Router-A#
Router-A# sho int s0/0 | include LMI
  LMI enq sent  188, LMI stat recvd 188, LMI upd recvd 0, DTE LMI up
  LMI enq recvd 0, LMI stat sent  0, LMI upd sent  0
  LMI DLCI 1023  LMI type is CISCO  frame relay DTE
```

`debug frame-relay lmi`

The `debug frame-relay lmi` command will show every LMI update sent and received on the frame-relay interfaces. As always, be very careful when issuing debug commands on production devices. These commands can cause large or busy routers to stop functioning, due to the increased CPU load. When you run the `debug frame-relay lmi` command, every 10 seconds a small status message (which is not of much use) is sent, and every 30 seconds a summary of all the virtual circuits present on the link is sent. You'll recognize this message by the obvious inclusion of lines beginning with PVC or SVC:

```
Router-A# debug frame lmi
Frame Relay LMI debugging is on
Displaying all Frame Relay LMI data
Router-A#
00:33:05: Serial0/0(out): StEnq, myseq 197, yourseen 196, DTE up
00:33:05: datagramstart = 0x3CE9B74, datagramsize = 13
00:33:05: FR encap = 0xFCF10309
00:33:05: 00 75 01 01 01 03 02 C5 C4
00:33:05:
00:33:05: Serial0/0(in): Status, myseq 197, pak size 13
00:33:05: RT IE 1, length 1, type 1
00:33:05: KA IE 3, length 2, yourseq 197, myseq 197
00:33:15: Serial0/0(out): StEnq, myseq 198, yourseen 197, DTE up
00:33:15: datagramstart = 0x3CEA1B4, datagramsize = 13
00:33:15: FR encap = 0xFCF10309
00:33:15: 00 75 01 01 00 03 02 C6 C5
00:33:15:
00:33:15: Serial0/0(in): Status, myseq 198, pak size 53
00:33:15: RT IE 1, length 1, type 0
```

```
00:33:15: KA IE 3, length 2, yourseq 198, myseq 198
00:33:15: PVC IE 0x7 , length 0x6 , dlci 102, status 0x2 , bw 0
00:33:15: PVC IE 0x7 , length 0x6 , dlci 103, status 0x2 , bw 0
00:33:15: PVC IE 0x7 , length 0x6 , dlci 104, status 0x0 , bw 0
00:33:15: PVC IE 0x7 , length 0x6 , dlci 105, status 0x0 , bw 0
00:33:15: PVC IE 0x7 , length 0x6 , dlci 106, status 0x0 , bw 0
00:33:25: Serial0/0(out): StEnq, myseq 199, yourseen 198, DTE up
00:33:25: datagramstart = 0x3CEA574, datagramsize = 13
00:33:25: FR encap = 0xFCF10309
00:33:25: 00 75 01 01 01 03 02 C7 C6
```

In this example, the frame-relay switch is advertising five PVCs. The status of each is 0x0 or 0x2. 0x0 means that the VC is configured on the frame-relay switch, but is not active. This occurs most commonly because the far-end device is not configured (in other words, it's probably your fault, not telco's). A status of 0x2 indicates that the VC is configured and active. If the VC is not listed at all in the status message, either telco hasn't yet provisioned it, or it has been provisioned on the wrong switch or interface.

Are the VCs active on the router?

The command `show frame-relay pvc` will show the status of every known frame-relay PVC on the router:

```
Router-A# sho frame pvc

PVC Statistics for interface Serial0/0 (Frame Relay DTE)

             Active     Inactive      Deleted       Static
  Local         2            0            0            0
  Switched      0            0            0            0
  Unused        0            3            0            0

DLCI = 102, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial0/0.102

  input pkts 46            output pkts 55           in bytes 11696
  out bytes 14761          dropped pkts 0           in pkts dropped 0
  out pkts dropped 0                out bytes dropped 0
  in FECN pkts 0           in BECN pkts 0           out FECN pkts 0
  out BECN pkts 0          in DE pkts 0             out DE pkts 0
  out bcast pkts 45        out bcast bytes 13721
  pvc create time 00:44:07, last time pvc status changed 00:44:07

DLCI = 103, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial0/0.103

  input pkts 39            output pkts 47           in bytes 11298
  out bytes 13330          dropped pkts 0           in pkts dropped 0
  out pkts dropped 0                out bytes dropped 0
  in FECN pkts 0           in BECN pkts 0           out FECN pkts 0
  out BECN pkts 0          in DE pkts 0             out DE pkts 0
  out bcast pkts 42        out bcast bytes 12810
  pvc create time 00:39:13, last time pvc status changed 00:39:13

DLCI = 104, DLCI USAGE = UNUSED, PVC STATUS = INACTIVE, INTERFACE = Serial0/0

  input pkts 0             output pkts 0            in bytes 0
```

```
    out bytes 0             dropped pkts 0          in pkts dropped 0
    out pkts dropped 0              out bytes dropped 0
    in FECN pkts 0          in BECN pkts 0          out FECN pkts 0
    out BECN pkts 0         in DE pkts 0            out DE pkts 0
    out bcast pkts 0        out bcast bytes 0
    switched pkts 0
    Detailed packet drop counters:
    no out intf 0           out intf down 0         no out PVC 0
    in PVC down 0           out PVC down 0          pkt too big 0
    shaping Q full 0        pkt above DE 0          policing drop 0
    pvc create time 00:44:01, last time pvc status changed 00:44:01

[output truncated]
```

For every PVC, there is a paragraph that shows the status of the PVC and the interface on which it was discovered. Notice that PVCs that have been assigned to subinterfaces are shown to be active on those subinterfaces. All other PVCs are shown to be associated with the physical interfaces on which they were found. If a particular PVC is not shown here, you're probably not receiving LMI for that VC.

Each entry shows a status, which can be one of the following:

`active`
: This status indicates that the PVC is up end-to-end, and is functioning normally.

`inactive`
: This status indicates that a PVC is defined by the telecom switch, but you do not have an active mapping for it. If this is the PVC you're trying to use, you probably forgot to map it, or mapped it incorrectly.

`deleted`
: This status indicates that you have a mapping active, but the PVC you've mapped to doesn't exist. An incorrect mapping may cause this problem.

`static`
: This status indicates that `no keepalive` is configured on the frame-relay interface of the router.

Is the PVC mapped to an interface?

The `show frame-relay map` command shows a very concise report of the VCs that are mapped to interfaces:

```
Router-A# sho frame map
Serial0/0.103 (up): point-to-point dlci, dlci 103(0x67,0x1870), broadcast
          status defined, active
Serial0/0.102 (up): point-to-point dlci, dlci 102(0x66,0x1860), broadcast
          status defined, active
```

If a PVC is not listed here, but is listed in the output of the `show frame-relay pvc` command, you have a configuration problem.

PART V

Security and Firewalls

This section covers security topics, including ACLs and authentication, as well as general firewall theory and configuration. Cisco PIX firewalls are used for examples.

This section is composed of the following chapters:

Chapter 23, *Access Lists*
Chapter 24, *Authentication in Cisco Devices*
Chapter 25, *Firewall Theory*
Chapter 26, *PIX Firewall Configuration*

CHAPTER 23

Access Lists

The technical name for an access list is *access-control list*, or ACL. The individual entries in an access-control list are called *access-control entries*, or ACEs. The term access-control list isn't often used in practice; you'll typically hear these lists referred to simply as access lists or ACLs.

Access lists do more than just control access. They are the means whereby Cisco devices categorize and match packets in any number of interesting ways. Access lists are used as simple filters to allow traffic through interfaces. They are also used to define "interesting traffic" for ISDN dialer maps, and are used in some route maps for matching.

Designing Access Lists

This focus of this chapter will be less on the basics of access-list design, and more on making you conscious of the benefits and pitfalls of access-list design. The tips and tricks in this chapter should help you to write better, more efficient, and powerful access lists.

When creating access lists (or any configuration, for that matter), it's a good idea to create them first in a text editor, and then, once you've worked out all the details, try them in a lab environment. Any time you're working on filters, you risk causing an outage.

Wildcard Masks

Wildcard masks (also called *inverse masks*) can be confusing because they're the opposite, in binary, of normal subnet masks. In other words, the wildcard mask you would use to match a range that would be described with a subnet mask of 255.255.255.0 would be 0.0.0.255.

Here's a simple rule that will solve most of the subnet/wildcard mask problems you'll see:

> Replace all 0s with 255s, and all 255s with 0s.

Table 23-1 shows how class A, B, and C subnet masks are written as wildcard masks.

Table 23-1. Classful wildcard masks

Subnet mask	Matching wildcard mask
255.0.0.0	0.255.255.255
255.255.0.0	0.0.255.255
255.255.255.0	0.0.0.255

While this may seem obvious, in the real world, networks are not often designed on classful boundaries. To illustrate my point, consider a subnet mask of 255.255.255.224. The equivalent wildcard mask works out to be 0.0.0.31.

Luckily, there is a trick to figuring out all wildcard masks, and it's easier than you might think. Here it is:

> The wildcard mask will be a derivative of the number of host addresses provided by the subnet mask minus one.

In the preceding example (the subnet mask 255.255.255.224), there are eight networks with 32 hosts in each (see Chapter 34 for help figuring out how many hosts are in a subnetted network). 32 – 1 = 31. The wildcard mask is 0.0.0.31. Yes, it's really that simple.

All you really need to think about is the one octet that isn't a 0 or a 255. In the case of the wildcard mask being in a position other than the last octet, simply use the same formula, and consider the number of hosts to be what it would be if the dividing octet were the last octet. Here's an example, using the subnet mask 255.240.0.0:

1. 240 in the last octet of a subnet mask (255.255.255.240) would yield 16 hosts.
2. 16 – 1 = 15.
3. The wildcard mask is 0.15.255.255.

The more you practice subnetting in your head, the easier this becomes. Try a few for yourself, and you'll quickly see how easy it is.

Where to Apply Access Lists

One of the most common questions I hear from junior engineers is, "Do I apply the access list inbound or outbound?" The answer is almost always *inbound*.

Figure 23-1 shows a simple router with two interfaces, E0 and E1. I've labeled the points where an access list could be applied. The thing to remember is that these terms are from the viewpoint of the device.

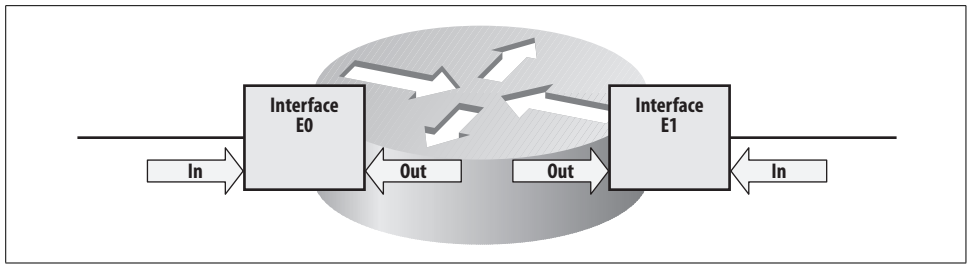


Figure 23-1. Access-list application points

Usually, when you're trying to filter traffic, you want to prevent it from getting into the network, or even getting to the device in the first place. Applying access lists to the inbound side of an interface keeps the packets from entering the device, thus saving processing time. When a packet is allowed into a device, and then switched to another interface, only to be dropped by an outbound filter, resources used to switch the packet have been wasted.

Reflexive access lists, covered later in this chapter, are applied in both directions.

Figure 23-2 shows a small network connected to the Internet by a router. The router is filtering traffic from the Internet to protect the devices inside. As traffic comes from the Internet, it travels inbound on E1, is switched in the router to E0, and is then forwarded to the inside network. If the ACL is applied inbound on E1, the packets will be denied before the router has to process them any further. If the ACL is applied outbound on E0, the router must expend resources switching the packets between interfaces, only to then drop them.

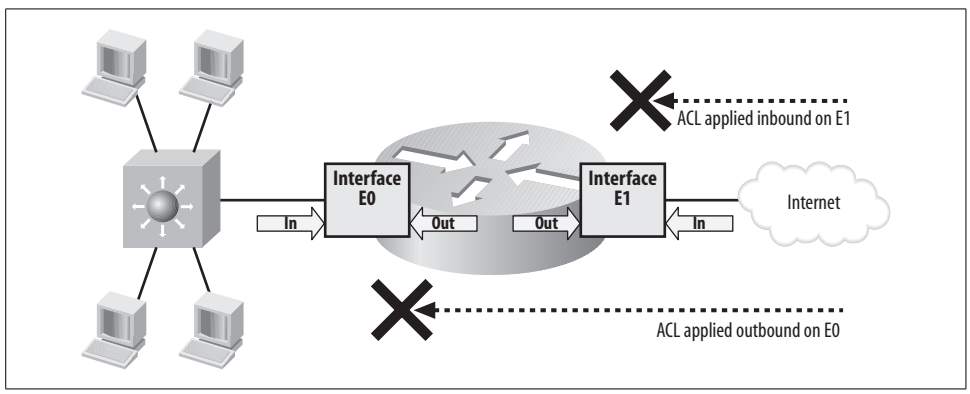


Figure 23-2. Access-list application in a network

Be careful when deleting access lists. If you delete an access list that is applied to an interface, the interface will deny all traffic. Always remove the relevant `access-group` commands before removing an access list.

Naming Access Lists

A quick word on naming access lists is in order. Naming access lists on Cisco routers with logical names rather than numbers is possible and encouraged, as it makes the configuration easier to read. The drawback of named access lists is that they cannot be used in many of the ways in which numbered access lists can be used. For example, route maps support named access lists, but dialer maps do not. PIX firewalls only support named access lists. That is, even if you create an access list named 10 on a PIX firewall, it will be considered a named access list rather than a standard (numbered) access list.

When you name access lists, it makes sense to name them well. I've seen many installations of PIX firewalls where the inbound access list is named something like "out." Imaging troubleshooting this command:

```
access-group out in interface outside
```

If you're not used to configuring PIX firewalls, that command might be difficult to interpret. If the access list were instead named Inbound, the command would be much more readable:

```
access-group Inbound in interface outside
```

The ability to quickly determine what a device is configured to do can save time during an outage, which can literally save your job. I like to begin my access list names with capital letters to aid in identifying them in code. This is a personal preference that may or may not suit your style—I've worked with people who complain when they have to use the Shift key.

Top-Down Processing

Access lists are processed from the top down, one line at a time. When a match is made, processing stops. This is an important rule to remember when building and troubleshooting access lists. A common mistake is to add a specific line to match something that's already been matched in a less-specific line above it:

```
access-list 101 permit tcp any 10.10.10.0 0.0.0.255 eq www
access-list 101 permit tcp any host 10.10.10.100 eq www
access-list 101 permit tcp any host 10.10.10.100 eq domain
```

In this example, the second line will never be matched because the IP address and protocol are matched in the first line. Even so, in the event that the first line doesn't match, the second line will still be evaluated, wasting time and processing power. This is a very commonly seen problem in enterprise networks. On larger firewalls, where more than one person is administering the device, the problem can be severe. It may also be hard to spot because it doesn't prevent protocols from working. This type of problem is usually uncovered during a network audit.

Most-Used on Top

Access lists should be built in such a way that the lines that are matched the most are at the beginning of the list. Recall that an ACL is processed until a match is made. Once a match is made, the remainder of the ACL is not processed. If you've only worked on routers with small ACLs, this may not seem like a big deal, but in real-world enterprise firewalls, ACLs can be extensive. (I've worked on PIX firewalls where the ACLs were up to 17 printed pages long!)

Here's an actual example from a PIX firewall. When my team built this small access list, we just added each line as we thought of it. This is a relatively common approach in the real world. We came up with a list of servers (web1, lab, web2), then listed each protocol to be allowed:

```
access-list Inbound permit tcp any host web1.gad.net eq www
access-list Inbound permit tcp any host web1.gad.net eq ssh
access-list Inbound permit udp any host web1.gad.net eq domain
access-list Inbound permit tcp any host web1.gad.net eq smtp
access-list Inbound permit tcp any host web1.gad.net eq imap4
access-list Inbound permit tcp any host lab.gad.net eq telnet
access-list Inbound permit tcp any host lab.gad.net eq 8080
access-list Inbound permit udp any host web2.gad.net eq domain
access-list Inbound permit tcp any host web2.gad.net eq smtp
access-list Inbound permit tcp any host web2.gad.net eq imap4
```

After letting the network run for a few days, we were able to see how our access list had fared by executing the `show access-list` command:

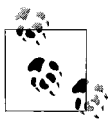
```
PIX# sho access-list
access-list cached ACL log flows: total 0, denied 0 (deny-flow-max 1024)
            alert-interval 300
access-list Inbound; 15 elements
access-list Inbound permit tcp any host web1.gad.net eq www (hitcnt=42942)
access-list Inbound permit tcp any host web1.gad.net eq ssh (hitcnt=162)
access-list Inbound permit udp any host web1.gad.net eq domain (hitcnt=22600)
access-list Inbound permit tcp any host web1.gad.net eq smtp (hitcnt=4308)
access-list Inbound permit tcp any host web1.gad.net eq imap4 (hitcnt=100)
access-list Inbound permit tcp any host lab.gad.net eq telnet (hitcnt=0)
```

```
access-list Inbound permit tcp any host lab.gad.net eq 8080 (hitcnt=1)
access-list Inbound permit udp any host web2.gad.net eq domain (hitcnt=10029)
access-list Inbound permit tcp any host web2.gad.net eq smtp (hitcnt=2)
access-list Inbound permit tcp any host web2.gad.net eq imap4 (hitcnt=0)
```

Look carefully at the `hitcnt` entries at the ends of the lines. They show how many times each of the lines in the ACL has been hit. The hit counts indicate that this ACL was not built optimally. To build it better, take the above output, and sort it by `hitcnt`, with the largest number first. The results look like this:

```
access-list Inbound permit tcp any host web1.gad.net eq www (hitcnt=42942)
access-list Inbound permit udp any host web1.gad.net eq domain (hitcnt=22600)
access-list Inbound permit udp any host web2.gad.net eq domain (hitcnt=10029)
access-list Inbound permit tcp any host web1.gad.net eq smtp (hitcnt=4308)
access-list Inbound permit tcp any host web1.gad.net eq ssh (hitcnt=162)
access-list Inbound permit tcp any host web1.gad.net eq imap4 (hitcnt=100)
access-list Inbound permit tcp any host web2.gad.net eq smtp (hitcnt=2)
access-list Inbound permit tcp any host lab.gad.net eq 8080 (hitcnt=1)
access-list Inbound permit tcp any host lab.gad.net eq telnet (hitcnt=0)
access-list Inbound permit tcp any host web2.gad.net eq imap4 (hitcnt=0)
```

This is an optimal design for this admittedly small access list. The entries with the most hits are now at the top of the list, and those with the fewest are at the bottom.



Beware of assumptions. You may think that SMTP should be high on your list because your firewall is protecting a mail server, but if you look at the preceding output, you'll see that DNS shows far more connections than SMTP. Check to see what's actually running on your network, and configure accordingly.

The problem with this approach can be a loss of readability. In this case, the original ACL is much easier to read and understand than the redesigned version. The second, more efficient ACL has an entry for web2 in the middle of all the entries for web1. This is easy to miss, and can make troubleshooting harder. Only you, as the administrator, can make the call as to the benefits or drawbacks of the current ACL design. In smaller ACLs, you may want to make some concessions to readability, but in the case of a 17-page access list, you'll find that putting the heavily hit lines at the top will have a significant impact on the operational speed of a heavily used firewall.

Using Groups in PIX ACLs

PIX firewalls now allow the use of *groups* in access lists. This is a huge benefit for access-list creation because it allows for very complex ACLs with very simple configurations. Using groups in ACLs also allows you to change multiple ACLs by changing a group—when a group that is in use is changed, the PIX will automatically change every instance where the group is applied. With complex access lists, using groups can help prevent mistakes because it's less likely that you'll forget an important entry: you don't have to make the addition in multiple places, you only have to remember to put it into the group.

Let's look at an example of groups in action. Here is the original ACL:

```
object-group service CCIE-Rack tcp
  description [< For Terminal Server Reverse Telnet >]
  port-object range 2033 2050

access-list Inbound permit tcp any host gto eq www
access-list Inbound permit tcp any host gto eq ssh
access-list Inbound permit tcp any host meg eq ssh
access-list Inbound permit tcp any host meg eq www
access-list Inbound permit tcp any host lab eq telnet
access-list Inbound permit tcp any host lab object-group CCIE-Rack
access-list Inbound permit udp any host PIX-Outside eq 5060
access-list Inbound permit tcp any host lab eq 8080
access-list Inbound permit udp any host meg eq domain
access-list Inbound permit udp any host gto eq domain
access-list Inbound permit tcp any host gto eq smtp
access-list Inbound permit tcp any host meg eq smtp
access-list Inbound permit tcp any host gto eq imap4
access-list Inbound permit tcp any host meg eq imap4
access-list Inbound permit esp any any
access-list Inbound permit icmp any any unreachable
access-list Inbound permit icmp any any time-exceeded
access-list Inbound permit icmp any any echo-reply
```

Notice that there is an object group already in use for CCIE-Rack. This may not seem necessary, as the same thing could be accomplished with the `range` keyword:

```
access-list Inbound line 3 permit tcp any host lab range 2033 2050
```

In fact, as you'll see shortly, the object group is converted to this line anyway. Some people argue that if an object group takes up more lines of configuration than the number of lines it is translated into, it shouldn't be used. I disagree. I like the fact that I can add a description to an object group. Additionally, I can easily add a service to the object group at a later time without having to change any access lists.

Here are the groups I've created based on the original access list. I've incorporated the services common to multiple servers into a group called *Webserver-svcs*. I've also created a group called *Webservers* that contains all of the web servers, another called *Webserver-svcs-udp* for UDP-based services like DNS, and one for ICMP packets called *ICMP-Types*. The ICMP-Types group is for return packets resulting from pings and traceroutes. The brackets in the description fields may look odd to you, but I like to add them to make the descriptions stand out:

```
object-group service CCIE-Rack tcp
  description [< For Terminal Server Reverse Telnet >]
  port-object range 2033 2050
object-group service Webserver-svcs tcp
  description [< Webserver TCP Services >]
  port-object eq www
  port-object eq ssh
  port-object eq domain
  port-object eq smtp
  port-object eq imap4
```

```
object-group service Webserver-svcs-udp udp
  description [< Webserver UDP Services >]
  port-object eq domain
object-group network Webservers
  description [< Webservers >]
  network-object host gto
  network-object host meg
object-group icmp-type ICMP-Types
  description [< Allowed ICMP Types >]
  icmp-object unreachable
  icmp-object time-exceeded
  icmp-object echo-reply
```

Now that I've organized all the services and servers into groups, it's time to rewrite the access list to use them:

```
access-list Inbound permit udp any object-group Webservers object-group Webserver-
svcs-udp
access-list Inbound permit tcp any object-group Webservers object-group Webserver-
svcs
access-list Inbound permit tcp any host lab eq telnet
access-list Inbound permit tcp any host lab object-group CCIE-Rack
access-list Inbound permit udp any host PIX-Outside eq 5060
access-list Inbound permit tcp any host lab eq 8080
access-list Inbound permit esp any any
access-list Inbound permit icmp any any object-group ICMP-Types
```

The access list has gone from 18 lines down to 8. This is only the visible configuration, remember. These lines will be expanded in the firewall's memory to the original 18 lines.

The lines may not be sorted optimally, which can be an issue with complex configurations. As with most things, there are tradeoffs. For complex installations, make sure you enable Turbo ACLs (discussed in the following section).

Note that groups do not necessarily mean less typing—in fact, the opposite is usually true. Even though this access list has shrunk from 18 to 8 lines, we had to type in more lines than we saved. The goal is to make the access list easier to read and maintain. It's up to you to determine whether the eventual benefits will justify the initial effort.

The actual result of the configuration can be seen using the `show access-list` command. The output includes both the `object-group` configuration lines, and the actual ACEs to which they translate. The `object-group` entries are shown in bold:

```
GAD-PIX# sho access-list

access-list cached ACL log flows: total 0, denied 0 (deny-flow-max 1024)
            alert-interval 300
access-list Inbound; 20 elements
access-list Inbound line 1 permit udp any object-group Webservers object-group
Webserver-svcs-udp
```

```
access-list Inbound line 1 permit udp any host gto eq domain (hitcnt=7265)
access-list Inbound line 1 permit udp any host meg eq domain (hitcnt=6943)
access-list Inbound line 2 permit tcp any object-group Webservers object-group
Webserver-svcs
access-list Inbound line 2 permit tcp any host gto eq www (hitcnt=21335)
access-list Inbound line 2 permit tcp any host gto eq ssh (hitcnt=4428)
access-list Inbound line 2 permit tcp any host gto eq domain (hitcnt=0)
access-list Inbound line 2 permit tcp any host gto eq smtp (hitcnt=1901)
access-list Inbound line 2 permit tcp any host gto eq imap4 (hitcnt=116)
access-list Inbound line 2 permit tcp any host meg eq www (hitcnt=23)
access-list Inbound line 2 permit tcp any host meg eq ssh (hitcnt=15)
access-list Inbound line 2 permit tcp any host meg eq domain (hitcnt=0)
access-list Inbound line 2 permit tcp any host meg eq smtp (hitcnt=1)
access-list Inbound line 2 permit tcp any host meg eq imap4 (hitcnt=0)
access-list Inbound line 3 permit tcp any host lab eq telnet (hitcnt=0)
access-list Inbound line 4 permit tcp any host lab object-group CCIE-Rack
access-list Inbound line 4 permit tcp any host lab range 2033 2050 (hitcnt=0)
access-list Inbound line 5 permit udp any host PIX-Outside eq 5060 (hitcnt=0)
access-list Inbound line 6 permit tcp any host lab eq 8080 (hitcnt=0)
access-list Inbound line 7 permit esp any any (hitcnt=26256)
access-list Inbound line 8 permit icmp any any object-group ICMP-Types
access-list Inbound line 8 permit icmp any any unreachable (hitcnt=359)
access-list Inbound line 8 permit icmp any any time-exceeded (hitcnt=14)
access-list Inbound line 8 permit icmp any any echo-reply (hitcnt=822)
```

Turbo ACLs

Normally, ACLs must be interpreted every time they are referenced. This can lead to significant processor usage, especially on devices with large ACLs.

One of the options for enhancing performance with large ACLs is to compile them. A compiled ACL is called a *Turbo ACL* (usually pronounced *turbo-ackle*). Compiling an ACL changes it to machine code, which no longer needs to be interpreted before processing. This can have a significant impact on performance.

PIX firewalls and Cisco routers support Turbo ACLs. On the PIX, the command `access-list compiled` tells the firewall to compile all access lists. Only Cisco routers in the 7100, 7200, 7500, and 12000 series (12.0(6)S and later) support Turbo ACLs. The IOS command to enable this feature is also `access-list compiled`.

When Turbo ACLs are enabled, the output of `show access-list` is altered to show the fact that the ACLs are compiled and how much memory each ACL is occupying:

```
PIX(config)# access-list comp
PIX(config)# show access-list

TurboACL statistics:
ACL                     State       Memory(KB)
----------------------- ----------- ----------
Inbound
                        Operational  2
```

```
Shared memory usage: 2056 KB

access-list compiled
access-list cached ACL log flows: total 0, denied 0 (deny-flow-max 1024)
            alert-interval 300
access-list Inbound turbo-configured; 20 elements
access-list Inbound line 1 permit udp any object-group Webservers object-group
Webserver-svcs-udp
access-list Inbound line 1 permit udp any host gto eq domain (hitcnt=7611)
access-list Inbound line 1 permit udp any host meg eq domain (hitcnt=7244)
access-list Inbound line 2 permit tcp any object-group Webservers object-group
Webserver-svcs
access-list Inbound line 2 permit tcp any host gto eq www (hitcnt=22578)
access-list Inbound line 2 permit tcp any host gto eq ssh (hitcnt=4430)
access-list Inbound line 2 permit tcp any host gto eq domain (hitcnt=0)
access-list Inbound line 2 permit tcp any host gto eq smtp (hitcnt=2035)
access-list Inbound line 2 permit tcp any host gto eq imap4 (hitcnt=157)
access-list Inbound line 2 permit tcp any host meg eq www (hitcnt=23)
access-list Inbound line 2 permit tcp any host meg eq ssh (hitcnt=16)
access-list Inbound line 2 permit tcp any host meg eq domain (hitcnt=0)
access-list Inbound line 2 permit tcp any host meg eq smtp (hitcnt=1)
access-list Inbound line 2 permit tcp any host meg eq imap4 (hitcnt=0)
access-list Inbound line 3 permit tcp any host lab eq telnet (hitcnt=0)
access-list Inbound line 4 permit tcp any host lab object-group CCIE-Rack
access-list Inbound line 4 permit tcp any host lab range 2033 2050 (hitcnt=0)
access-list Inbound line 5 permit udp any host PIX-Outside eq 5060 (hitcnt=0)
access-list Inbound line 6 permit tcp any host lab eq 8080 (hitcnt=0)
access-list Inbound line 7 permit esp any any (hitcnt=26423)
access-list Inbound line 8 permit icmp any any object-group ICMP-Types
access-list Inbound line 8 permit icmp any any unreachable (hitcnt=405)
access-list Inbound line 8 permit icmp any any time-exceeded (hitcnt=14)
access-list Inbound line 8 permit icmp any any echo-reply (hitcnt=822)
```

Allowing Outbound Traceroute and Ping

One of the more common frustrations with firewalls is the inability to ping and traceroute once the security rules are put in place. The idea that ICMP is dangerous is valid, but if you understand how ICMP behaves, you can allow only the types you need, and thus continue to enjoy the benefits of ping and traceroute.

Assuming that you're allowing all outbound traffic, you can apply packet filters that allow as inbound traffic only those reply packets that are the result of ping and traceroute commands. This will allow your tests to work when initiated from inside the network, while disallowing those same tests when they originate from outside the network. To allow these tools to work, you must allow the following ICMP packet types in from the outside:

ICMP unreachable
: There are many ICMP unreachable types, including network unreachable, and host unreachable. Generally, allowing them all is acceptable because they are response packets.

Time exceeded
: Time exceeded messages are sent back by traceroute at each hop of the path taken toward the intended destination.

Echo reply
: An echo reply is the response from a ping packet.

The packet filters are usually included at the end of whatever inbound access lists are already in place. They should generally be placed at the bottom of the ACL, unless there is a large amount of ICMP traffic originating inside your network. Here are some examples of deploying these filters for Cisco routers and PIX firewalls:

- Cisco routers:

```
access-list 101 remark [< Allows PING and Traceroute >]
access-list 101 permit icmp any any unreachable
access-list 101 permit icmp any any time-exceeded
access-list 101 permit icmp any any echo-reply
!
interface Ethernet1
ip access-group 101 in
```

- Firewalls:

```
object-group icmp-type ICMP-Types
  description [< Allowed ICMP Types >]
  icmp-object unreachable
  icmp-object time-exceeded
  icmp-object echo-reply
!
access-list Inbound permit icmp any any object-group ICMP-Types
!
access-group Inbound in interface outside
```

Allowing MTU Path Discovery Packets

MTU path discovery allows devices on remote networks to inform you of MTU limitations. To enable this, you must allow two more ICMP types: *source-quench* and *parameter-problem*. You can allow them on Cisco routers and PIX firewalls as follows:

- Cisco routers:

```
access-list 101 remark [< Allows PING and Traceroute >]
access-list 101 permit icmp any any unreachable
access-list 101 permit icmp any any time-exceeded
access-list 101 permit icmp any any echo-reply
access-list 101 permit icmp any any parameter-problem
access-list 101 permit icmp any any source-quench
!
interface Ethernet1
ip access-group 101 in
```

- Firewalls:

```
object-group icmp-type ICMP-Types
  description [< Allowed ICMP Types >]
```

```
  icmp-object unreachable
  icmp-object time-exceeded
  icmp-object echo-reply
  icmp-object source-quench
  icmp-object parameter-problem
!
access-list Inbound permit icmp any any object-group ICMP-Types
!
access-group Inbound in interface outside
```

ACLs in Multilayer Switches

Multilayer switches, by nature of their design, allow for some security features not available on layer-2 switches or routers.

The 3750 switch supports IP ACLs and Ethernet (MAC) ACLs. Access lists on a 3750 switch can be applied in the following ways:

Port ACLs
: Port ACLs are applied to layer-2 interfaces on the switch. They cannot be applied to EtherChannels, SVIs, or any other virtual interfaces. Port ACLs can be applied to trunk interfaces, in which case they will filter every VLAN in the trunk. Standard IP, extended IP, or MAC ACLs can be assigned as port ACLs. Port ACLs can be applied only in the *inbound* direction.

Router ACLs
: Router ACLs are applied to layer-3 interfaces on the switch. SVIs, layer-3 physical interfaces (configured with `no switchport`, for example), and layer-3 EtherChannels can have router ACLs applied to them. Standard IP and extended IP ACLs can be assigned as router ACLs, while MAC ACLs cannot. Router ACLs can be applied in both *inbound* and *outbound* directions.

VLAN maps
: VLAN maps are similar in design to route maps. VLAN maps are assigned to VLANs, and can be configured to pass or drop packets based on a number of tests. VLAN maps control all traffic routed into, out of, or within a VLAN. VLAN maps have no direction.

Configuring Port ACLs

Port ACLs are ACLs attached to a specific physical interface. Port ACLs can be used to deny a host within a VLAN access to any other host within the VLAN. They can also be used to limit access outside of the VLAN.

Imagine that VLAN 100 has many hosts in it, including host A. Host A should not be able to communicate directly with any of the other hosts within the same VLAN; it should only be able to communicate with the default gateway, to communicate with

the rest of the world. Assume that host A's IP address is 192.168.1.155/24, the default gateway's IP address is 192.168.1.1/24, and host A is connected to port G0/20 on the switch.

The first step in restricting host A's communications is to create the necessary ACL. You must allow access to the default gateway, then deny access to other hosts in the network, and, finally, permit access to the rest of the world:

```
access-list 101 permit ip any host 192.168.1.1
access-list 101 deny   ip any 192.168.1.0 0.0.0.255
access-list 101 deny   ip any any
```

Once you've created the ACL, you can apply it to the physical interface:

```
3750(config)# int g0/20
3750(config)# switchport
3750(config-if)# ip access-group 101 in
```

Notice that even though this is a layer-2 switch port, a layer-3 IP access list can be applied to it. The fact that the IP access list is applied to a switch port is what makes it a port ACL.

Port ACLs can also be MAC-based. Here's a small MAC access list that denies AppleTalk packets while permitting everything else:

```
mac access-list extended No-Appletalk
 deny   any any appletalk
 permit any any
```

Assigning this access list to an interface makes it a port ACL:

```
3750(config)# int g0/20
3750(config-if)# mac access-group No-Appletalk in
```

MAC ACLs can be mixed with IP ACLs in a single interface. Here, you can see that the MAC access list and the IP access list are active on the interface:

```
3750# show run int g0/20
interface GigabitEthernet0/20
 switchport mode dynamic desirable
 ip access-group 101 in
 mac access-group No-Appletalk in
end
```

Configuring Router ACLs

Router ACLs are probably what most people think of when they think of applying ACLs. Router ACLs are applied to layer-3 interfaces. Older routers only had layer-3 interfaces, so just about all ACLs were router ACLs.

If you were to take the previous example, and change the port from a layer-2 interface to a layer-3 interface, the ACL would become a router ACL:

```
3750(config)# int g0/20
3750(config)# no switchport
3750(config-if)# ip access-group 101 in
```

MAC access lists cannot be assigned as router ACLs.

When configuring router ACLs, you have the option to apply the ACLs outbound (though I'm not a big fan of outbound ACLs):

```
3750(config-if)# ip access-group 101 out
```

Remember that applying an ACL to any layer-3 interface will make the ACL a router ACL. Be careful when applying port ACLs and router ACLs together:

```
3750(config)# int vlan 100
3750(config-if)# ip address 192.168.100.1 255.255.255.0
3750(config-if)# ip access-group 101 in
2w3d: %FM-3-CONFLICT: Input router ACL 101 conflicts with port ACLs
```

This error message indicates that port ACLs and router ACLs are in place with overlapping ranges (in this case, the same IP addresses). This message is generated because both ACLs will be active, but the port ACL will take precedence.

Having a port ACL in place while a router ACL is also in place can cause a good deal of confusion if you don't realize the port ACL is in place.

Configuring VLAN Maps

VLAN maps allow you to combine access lists in interesting ways. VLAN maps filter all traffic *within* a VLAN.

A port ACL only filters inbound packets on a single interface, and a router ACL only filters packets as they travel into or out of a layer-3 interface. A VLAN map, on the other hand, filters every packet within a VLAN, regardless of the port type involved. For example, if you created a filter that prevented MAC address 1111.1111.1111 from talking to 2222.2222.2222, and applied it to an interface, moving the device to another interface would bypass the filter. But with a VLAN map, the filter would be applied no matter what interface was involved (assuming it was in the configured VLAN).

For this example, we'll create a filter that will disallow AppleTalk from VLAN 100. Here's the MAC access list:

```
mac access-list extended No-Appletalk
 permit any any appletalk
```

Notice that we're permitting AppleTalk, though our goal is to deny it. This is due to the nature of VLAN maps, as you're about to see.

To accomplish the goal of denying AppleTalk within the VLAN, we need to build a VLAN map. VLAN maps have clauses, similar to route maps. The clauses are numbered, although unlike in a route map, the action is defined within the clause, not in the title of the clause.

First, we need to define the VLAN map. This is done with the `vlan access-map` command. This VLAN map will have two clauses. The first (10) matches the MAC access list `No-Appletalk`, and drops any packets that match. This is why the access list needs to contain a `permit appletalk` instead of a `deny appletalk` line. The `permit` entry allows AppleTalk to be matched. The `action` statement in the VLAN map actually drops the packets:

```
vlan access-map Limit-V100 10
 action drop
 match mac address No-Appletalk
```

Next, we'll add another clause that forwards all remaining packets. Because there is no `match` statement in this clause, all packets are matched:

```
vlan access-map Limit-V100 20
 action forward
```

Here's the entire VLAN map:

```
vlan access-map Limit-V100 10
 action drop
 match mac address No-Appletalk
vlan access-map Limit-V100 20
 action forward
```

Now that we've built the VLAN map, we need to apply it to the VLAN. This is done with the `vlan filter` global command:

```
3750(config)# vlan filter Limit-V100 vlan-list 100
```

To apply a VLAN map to multiple VLANs, append each VLAN number to the end of the command.

You may be wondering, couldn't we just make a normal access list like the following one, and apply it to specific interfaces?

```
mac access-list extended No-Appletalk
deny   any any appletalk
permit any any
```

The answer is yes, but to do this, we'd have to figure out which interfaces might send AppleTalk packets, and hence, where to apply it. Alternatively, we could apply it to all interfaces within the VLAN, but then we'd need to remember to apply the access list to any ports that get added to the VLAN in the future. Assigning the access list to the VLAN itself ensures that any AppleTalk packet that arrives in the VLAN, regardless of its source or destination, will be dropped.

To see what VLAN maps are assigned, use the `show vlan filter` command:

```
SW2# sho vlan filter
VLAN Map Limit-V100 is filtering VLANs:
  100
```

Reflexive Access Lists

Reflexive access lists are dynamic filters that allow traffic based on the detection of traffic in the opposite direction. A simple example might be, "only allow telnet inbound if I initiate telnet outbound." When I first explain this to junior engineers, I often get a response similar to, "Doesn't it work that way anyway?" What confuses many people is the similarity of this feature to Port Address Translation (PAT). PAT only allows traffic inbound in response to outbound traffic originating on the network. This is due to the nature of PAT, in which a translation must be created for the traffic to pass. Reflexive access lists are much more powerful, and can be applied for different reasons.

Without PAT, a filter denies traffic without regard to other traffic. Consider the network in Figure 23-3. There are two hosts, A and B, connected through a router. The router has no access lists installed. Requests from host A to host B are answered, as are requests from host B to host A.

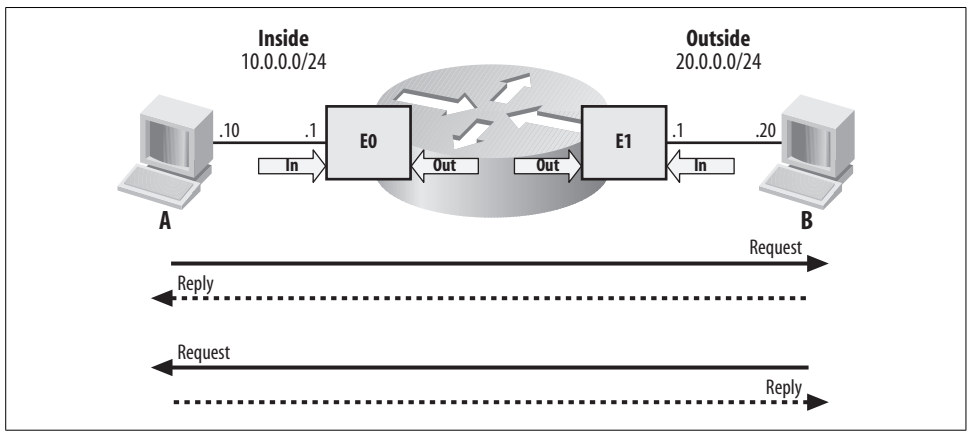


Figure 23-3. Simple network without ACLs

Say we want host A to be able to telnet to host B, but we don't want host B to be able to telnet to host A. If we apply a normal inbound access list to interface E1 on the router, we allow A to contact B, and prevent B from contacting A. Unfortunately, we also prevent B from replying to A. This limitation is shown in Figure 23-4.

This is too restrictive for our needs. While we've secured host A from host B's advances, we've also denied host A useful communications from host B. What we need is for the router to act more like a firewall: we need the router to deny requests from host B, but we want host B to be able to reply to host A's requests. Reflexive access lists solve this problem.

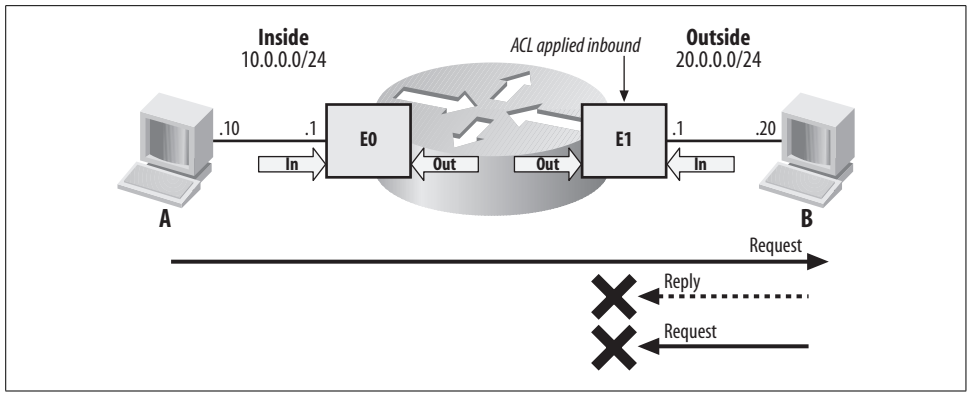


Figure 23-4. Simple access list applied inbound on E1

Reflexive access lists create ACLs on the fly to allow replies to requests. In this example, we'd like to permit traffic from B, but only if traffic from A is detected first. Should B initiate the traffic, we do not want to permit it. This concept is shown in Figure 23-5.

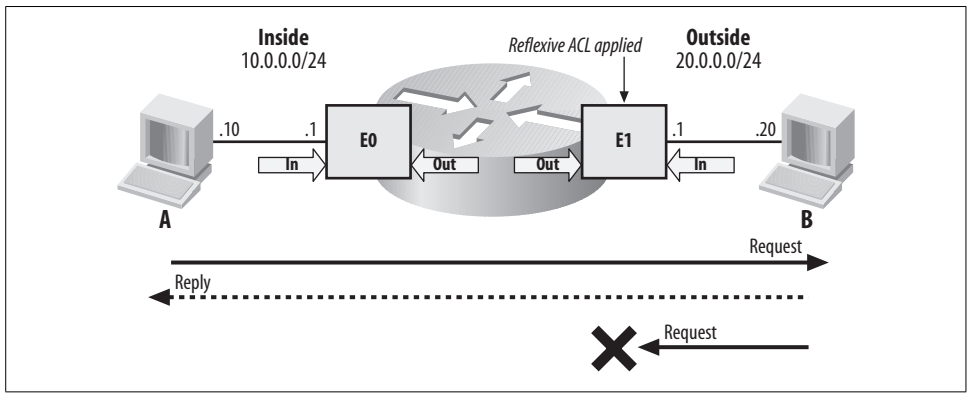


Figure 23-5. Reflexive access list applied to E1

Reflexive access lists create temporary `permit` statements that are reflections of the original statements. For example, if we permit telnet outbound, a temporary `permit` statement will be created for telnet inbound.

Reflexive access lists are very useful, but they do have some limitations:

- The temporary entry is always a `permit`, never a `deny`.
- The temporary entry is always the same protocol as the original (TCP, UDP, etc.).
- The temporary entry will have the opposite source and destination IP addresses from the originating traffic.

- The temporary entry will have the same port numbers as the originating traffic, though the source and destination will be reversed (ICMP, which does not use port numbers, will use type numbers).
- The temporary entry will be removed after the last packet is seen (usually a FIN or RST).
- The temporary entry will expire if no traffic is seen for a configurable amount of time (the default is five seconds).

You cannot create a reflexive access list that allows one protocol when another is detected. For example, you cannot allow HTTP inbound because a telnet was initiated outbound. If you want to reflexively allow HTTP inbound, you must test for HTTP outbound.

Because the port numbers in the temporary entries are always the reverse of the port numbers from the original traffic, they are not suitable for protocols such as RPC that change source port numbers. Reflexive ACLs are also not suitable for protocols that create new streams such as FTP.

FTP can still be used with reflexive access lists, provided *passive mode* is used.

Configuring Reflexive Access Lists

Reflexive access lists are a bit more complicated than regular access lists because you must nest one ACL within another. Consider the need to test for two types of traffic: the original request, and the resulting reply. An ACL must be created for each test. The ACL for the reply is created dynamically when the ACL for the original request is matched.

Cisco calls the way that reflexive access lists are configured *nesting*, though the configuration doesn't look like nested code to most programmers.

Continuing with the preceding example, let's create a reflexive access list for telnet. We want host A to be able to telnet to host B, but we'll deny everything else. This scenario is overly restrictive for most real-world applications, but it'll help illustrate the functionality of reflexive access lists.

To configure reflexive access lists, we must create one ACL for outbound traffic, and one for inbound traffic.

First, we'll create a named access list called TelnetOut:

```
ip access-list extended TelnetOut
 permit tcp host 10.0.0.10 host 20.0.0.20 eq telnet reflect GAD
 deny   ip any any
```

Reflexive access lists can only be created using named access lists.

This ACL is pretty straightforward, except for the addition of `reflect GAD` at the end of the `permit` line. This will be the name of the temporary access list created by the router when this `permit` entry is matched. The entry `deny ip any any` is not necessary, as all access lists include this by default, but I've included it here for clarity, and to show the counters incrementing as traffic is denied later.

Next, we'll create a named access list called TelnetIn:

```
ip access-list extended TelnetIn
 evaluate GAD
 deny   ip any any
```

This access list has no `permit` statements, but it has the statement `evaluate GAD`. This line references the `reflect` line in the TelnetOut access list. GAD will be the name of the new access list created by the router.

To make these access lists take effect, we need to apply them to the router. We'll apply TelnetOut to interface E1 *outbound*, and TelnetIn to interface E1 *inbound*. Figure 23-6 illustrates.

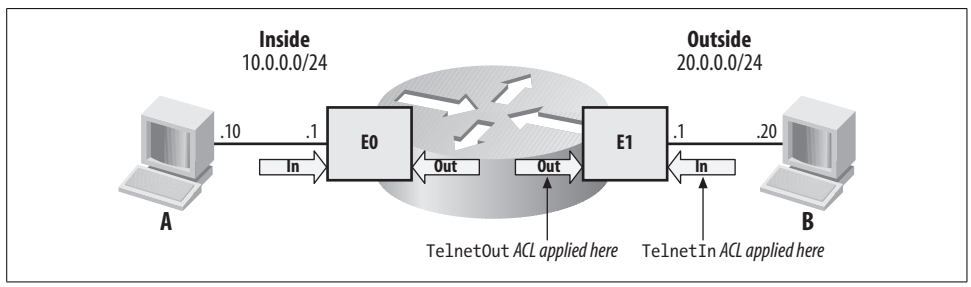


Figure 23-6. Application of reflexive access lists

Reflexive access lists are applied with the `access-group` interface command:

```
interface Ethernet1
 ip access-group TelnetIn in
 ip access-group TelnetOut out
```

The entire relevant configuration for the router is as follows:

```
interface Ethernet0
 ip address 10.0.0.1 255.255.255.0
!
interface Ethernet1
 ip address 20.0.0.1 255.255.255.0
 ip access-group TelnetIn in
 ip access-group TelnetOut out
!
```

```
ip access-list extended TelnetIn
 evaluate GAD
 deny   ip any any
ip access-list extended TelnetOut
 permit tcp host 10.0.0.10 host 20.0.0.20 eq telnet reflect GAD
 deny   ip any any
```

Looking at the access lists with the `show access-list` command, we see them both exactly as we've configured them:

```
Router# sho access-list
Reflexive IP access list GAD
Extended IP access list TelnetIn
    evaluate GAD
    deny ip any any
Extended IP access list TelnetOut
    permit tcp host 10.0.0.10 host 20.0.0.20 eq telnet reflect GAD
    deny ip any any (155 matches)
```

Here, we can see that all nontelnet traffic is being denied outbound. There really aren't any entries to permit anything inbound, but that will change when we trigger the reflexive access list.

After we initiate a telnet request from host A to host B, the output changes. There is now an additional access list named GAD:

```
Router# sho access-list
Reflexive IP access list GAD
    permit tcp host 20.0.0.20 eq telnet host 10.0.0.10 eq 11002 (12 matches)
Extended IP access list TelnetIn
    evaluate GAD
    deny ip any any
Extended IP access list TelnetOut
    permit tcp host 10.0.0.10 host 20.0.0.20 eq telnet reflect GAD
    deny ip any any (155 matches)
```

This temporary access list has been created in response to outbound traffic matching the `permit` entry containing the `reflect GAD` statement. The destination port number is 11002; this was the source port number for the outbound telnet request.

When the session has ended, or there is no activity matching the new access list, the reflexive access list is removed. The required inactivity period can be configured using the global `ip reflexive-list timeout` *seconds* command. This command affects all reflexive access lists on the router. The default timeout value is five seconds.

CHAPTER 24

Authentication in Cisco Devices

Authentication refers to the process of verifying a user's identity. When a router challenges you for a login username and password, this is an example of authentication.

Authentication in Cisco devices is divided into two major types: normal and AAA (Authentication, Authorization, and Auditing).

Basic (Non-AAA) Authentication

Non-AAA authentication is the basic authentication capability built into a router or other network device's operating system. Non-AAA authentication does not require access to an external server. It is very simple to set up and maintain, but lacks flexibility and scalability. Using locally defined usernames as an example, each username needs to be configured locally in the router. Imagine a scenario where a single user might connect to any of a number of devices, such as at an ISP. The user configurations would have to be maintained across all devices, and the ISP might have tens of thousands of users. With each user needing a line of configuration in the router, the configuration for the router would be hundreds of pages long.

Normal authentication is good for small-scale authentication needs, or as a backup to AAA.

Line Passwords

Lines are logical or physical interfaces on a router that are used for management of the router. The console and aux port on a router are lines, as are the logical VTY interfaces used for telnet and SSH. Configuring a password on a line is a simple matter of adding it with the `password` command:

```
R1(config-line)# password Secret
```

Passwords are case-sensitive, and may include spaces.

If a password is not set on the VTY lines, you will get an error when telneting to the device:

```
Password required, but none set
```

Passwords entered in clear text are shown in the configuration in clear text by default. To have IOS encrypt all passwords in the configuration, you need to enable the password-encryption service with the `service password-encryption` command. Here's an example of passwords in the running configuration displayed in clear text:

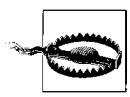
```
R1# sho run | include password
 password Secret1
```

Here's how to configure the password-encryption service to encrypt all the passwords:

```
R1# conf t
Enter configuration commands, one per line. End with CNTL/Z.
R1(config)# service password-encryption
R1(config)# exit
```

And here's what the passwords look like in the configuration with the password-encryption service running:

```
R1# sho run | include password
 password 7 073C244F5C0C0D54
```



Do not rely on encrypted passwords within IOS being totally secure. They can easily be cracked with tools freely available on the Internet.

If you would like to be able to telnet to your device without needing a password, you can disable the requirement with the `no login` command:

```
R1(config-line)# no login
```

This, of course, is a bad idea, and should only be done in a lab environment.

The `no login` command is available only when `aaa new-model` is not enabled. Once `aaa new-model` has been enabled, the `login` command takes on a different meaning and syntax. This command is discussed in the section "Enabling AAA."

Configuring Local Users

You can create users locally on your networking device. These usernames can then be used for authentication when users log into the device. This is useful when there are a small number of users, or when using an external authentication server (discussed in the section "AAA Authentication") is not practical. If you're using AAA authentication, this option is also useful as a backup: you can use the local users for authentication, should the normal external authentication server become unavailable.

Creating and managing users is done with the `username` command. Many options are available with this command; I'll focus on those that are useful for telnet or SSH access to a network device.

The first step in creating a local user is to define the username. Here, I'll use the username *GAD*:

```
R1(config)# username GAD ?
  access-class         Restrict access by access-class
  autocommand          Automatically issue a command after the user logs in
  callback-dialstring  Callback dialstring
  callback-line        Associate a specific line with this callback
  callback-rotary      Associate a rotary group with this callback
  dnis                 Do not require password when obtained via DNIS
  nocallback-verify    Do not require authentication after callback
  noescape             Prevent the user from using an escape character
  nohangup             Do not disconnect after an automatic command
  nopassword           No password is required for the user to log in
  password             Specify the password for the user
  privilege            Set user privilege level
  secret               Specify the secret for the user
  user-maxlinks        Limit the user's number of inbound links
  view                 Set view name
  <cr>
```

Simply specifying the command `username GAD` will create a user named GAD with no password. To add a password, include the `password` keyword followed by the password:

```
R1(config)# username GAD password Secret1
```

Passwords are case-sensitive, and can include spaces. Passwords are displayed in clear text in the configuration, unless the password-encryption service is running. To include an encrypted password without using the password-encryption service, use the `secret` keyword instead of the `password` keyword:

```
R1(config)# username GAD secret Secret1
```

This command results in the following configuration entry:

```
username GAD secret 5 $1$uyU6$6iZp6GLI1WGE1hxGDfQxc/
```

Every command and user has an associated *privilege level*. The levels range from 0 to 15. The standard user EXEC mode is level 1. When you enter the command `enable`, and authenticate with the `enable` or `enable secret` password, you change your level to *privileged EXEC mode*, which is level 15. If you'd like a user to be able to access privileged-level commands without entering the `enable` password, you can assign a higher privilege level to that user. In other words, configuring a user with a privilege level of 15 removes the need for that user to use the `enable` password to execute privileged commands:

```
R1(config)# username GAD privilege 15
```

When separate commands are entered for the same user (either in the same or separate command lines), as I've just done with the `secret` and `privilege` commands, the parser will combine the commands into a single configuration entry:

```
username GAD privilege 15 secret 5 $1$uyU6$6iZp6GLI1WGE1hxGDfQxc/
```

When this user logs into the router, he'll be greeted with the privileged # prompt instead of the normal exec prompt:

```
User Access Verification

Username: GAD
Password:
GAD-R1#
```

Another interesting feature of the `username` command is the ability to assign a command to run automatically when the user authenticates. Here, I've configured the `show ip interface brief` command to run upon authentication:

```
R1(config)# username GAD autocommand show ip interface brief
```

When this user logs in, he'll be shown the output from this command, and then promptly disconnected:

```
User Access Verification

Username: GAD
Password:
Interface                  IP-Address      OK? Method Status
Protocol
FastEthernet0/0            10.100.100.1    YES NVRAM  up                    up
FastEthernet0/1            unassigned      YES NVRAM  administratively down down
Serial0/0/0:0              unassigned      YES unset  down                  down
```

This may not seem like a useful feature until you consider the possibility that a first-line engineer may need to execute only one command. Why give him access to anything else?

Another possibility with the `autocommand` feature is to call a predefined menu of commands (configuring menus is outside the scope of this book):

```
R1(config)# username GAD autocommand menu root-menu
```

If you specify a command or menu that does not exist, the user will not be able to log in even with proper authentication.

You can disable the automatic disconnection after `autocommand` with the `nohangup` keyword:

```
R1(config)# username GAD nohangup autocommand menu root-menu
```

PPP Authentication

Authenticating a PPP connection is possible through one of two methods: Password Authentication Protocol (PAP), and Challenge Handshake Authentication Protocol (CHAP). PAP is the easier of the two to implement and understand, but it is of limited value because it transmits passwords in clear text. CHAP uses a more secure algorithm that does not include sending passwords. Both methods are outlined in RFC 1334. The RFC includes this warning about PAP:

> PAP is not a strong authentication method. Passwords are sent over the circuit "in the clear", and there is no protection from playback or repeated trial and error attacks. The peer is in control of the frequency and timing of the attempts.
>
> Any implementations which include a stronger authentication method (such as CHAP, described below) MUST offer to negotiate that method prior to PAP.

If a PPP authentication scheme is required, you must decide which scheme is right for you. Usually, CHAP is the right choice, due to its increased security, though PAP can be used when minimal security is desired, or the possibility of capturing packets is at a minimum. To use either method, you must have a local user configured at least on the receiving (called) router, although as you'll see, the requirements vary.

Configuring non-AAA authentication for PPP is covered here. To read about AAA PPP authentication using PAP and CHAP, see the section "Applying Method Lists" at the end of the chapter.

PAP

PAP can be configured for one-way or two-way authentication. One-way authentication indicates that only one side initiates a challenge. Two-way authentication indicates that both sides of the link authenticate each other.

One-way authentication. With one-way authentication, the calling router sends a username and password, which must match a username and password configured on the called router. The calling router must be configured as a `callin` router with the `ppp authentication pap callin` command. The calling router must also be configured to send a username and password with the `ppp pap sent-username` command. Here are some example configurations. Configuration entries not specific to PPP authentication are not shown, for clarity:

- Calling side:

```
interface BRI1/0
 encapsulation ppp
 ppp authentication pap callin
 ppp pap sent-username Bob password 0 ILikePie
```

- Called side:

```
username Bob password 0 ILikePie
!
interface BRI1/0
 encapsulation ppp
 ppp authentication pap
```

Two-way authentication. With two-way authentication, the `callin` keyword is not necessary. Because the authentication is done in both directions, both routers must be configured with usernames/passwords and the `ppp pap sent-username` command. The end result is that both sides are configured the same way:

- Calling side:

```
username Bob password 0 ILikePie
!
interface BRI1/0
 encapsulation ppp
 ppp authentication pap
 ppp pap sent-username Bob password 0 ILikePie
```

- Called side:

```
username Bob password 0 ILikePie
!
interface BRI1/0
 encapsulation ppp
 ppp authentication pap
 ppp pap sent-username Bob password 0 ILikePie
```

Debugging PPP authentication. Debugging PPP authentication is done with the `debug ppp authentication` command. The results are usually quite specific and easy to understand. Here, the password being sent from the calling router is incorrect. The debug was run on the called router:

```
8w4d: BR1/0:1 PPP: Using dialer call direction
8w4d: BR1/0:1 PPP: Treating connection as a callin
8w4d: BR1/0:1 PAP: I AUTH-REQ id 4 len 18 from "Bob"
8w4d: BR1/0:1 PAP: Authenticating peer Bob
8w4d: BR1/0:1 PAP: O AUTH-NAK id 4 len 27 msg is "Authentication failure" Bad
password defined for username Bob
```

In the next example, two-way authentication has succeeded. Notice that additional requests have been made. First, the called router receives the authorization request (AUTH-REQ) from the calling router. The called router then sends an authorization acknowledgment (AUTH-ACK) and an AUTH-REQ of its own back to the calling router. The final line in bold shows the AUTH-ACK returned by the calling router, completing the two-way authentication. The I and O before the AUTH-ACK and AUTH-REQ entries indicate the direction of the message (either *In* or *Out*):

```
00:00:41: %LINK-3-UPDOWN: Interface BRI1/0:1, changed state to up
00:00:41: BR1/0:1 PPP: Using dialer call direction
```

```
00:00:41: BR1/0:1 PPP: Treating connection as a callin
00:00:43: %ISDN-6-LAYER2UP: Layer 2 for Interface BR1/0, TEI 68 changed to up
00:00:45: BR1/0:1 AUTH: Started process 0 pid 62
00:00:45: BR1/0:1 PAP: I AUTH-REQ id 2 len 17 from "Bob"
00:00:45: BR1/0:1 PAP: Authenticating peer Bob
00:00:45: BR1/0:1 PAP: O AUTH-ACK id 2 len 5
00:00:45: BR1/0:1 PAP: O AUTH-REQ id 1 len 17 from "Bob"
00:00:45: BR1/0:1 PAP: I AUTH-ACK id 1 len 5
00:00:46: %LINEPROTO-5-UPDOWN: Line protocol on Interface BRI1/0:1, changed state
to up
00:00:47: %ISDN-6-CONNECT: Interface BRI1/0:1 is now connected to 7802000 Bob
```

CHAP

CHAP is more secure than PAP because it never sends passwords. Instead, it forms a hash value derived from the username and password, and sends that. The devices determine whether the hash values match in order to authenticate.

Figure 24-1 shows a simple two-router network. The Chicago router will call the New-York router. As with PAP, there are two ways to authenticate using CHAP: one-way and two-way.

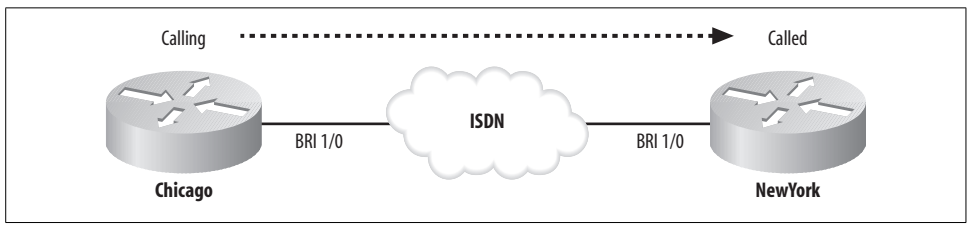


Figure 24-1. CHAP-authenticated ISDN call

CHAP can be a little harder to understand than PAP because of the way it operates. When Cisco routers authenticate using CHAP, by default, no username is needed on the calling router. Instead, the hostname of the router is used as the username. While people seem to grasp that concept easily enough, how passwords are handled is a little more complicated.

With PAP, one or more username/password pairs are configured on the called router. When the calling router attempts to authenticate, it must send a username and password that match a configured pair on the called router.

With CHAP, each router must have a username/password pair configured, but the username must be the hostname of the other router, and the passwords must be the same on *both* routers. Both the hostnames and the passwords are case-sensitive.

Be careful when configuring passwords. A common mistake is to enter a space or control character after a password during configuration. It's hard to catch such an error because everything will look normal. If you believe everything is configured correctly, but it's just not working, try removing the lines with the passwords and retyping them (using cut and paste usually doesn't solve the problem). While this can happen any time passwords are being configured, I find it especially maddening when I'm using CHAP because I'm constantly second-guessing my configuration.

One-way authentication. We'll begin with some examples of one-way authentication using CHAP. Notice that the username configured on each router matches the hostname of the other router. Notice also that the password is the same for both usernames. The password must be the same on both routers:

- Calling side (Chicago):

```
hostname Chicago
!
username NewYork password 0 Secret2
!
interface BRI1/0
 encapsulation ppp
 ppp authentication chap callin
```

- Called side (NewYork):

```
hostname NewYork
!
username Chicago password 0 Secret2
!
interface BRI1/0
 encapsulation ppp
 ppp authentication chap
```

Now, let's look at the debug output for a successful call using these configurations. The call was initiated from the Chicago router. If you look carefully, you'll see that the NewYork router is receiving a challenge from NewYork. The challenge entries in the debug output refer to the username, not the hostname. This can add to the confusion, as the username must match the hostname of the other router. Here's the debug output for both sides:

- Calling side:

```
20:08:11: %LINK-3-UPDOWN: Interface BRI1/0:1, changed state to up
20:08:11: BR1/0:1 PPP: Using dialer call direction
20:08:11: BR1/0:1 PPP: Treating connection as a callout
20:08:11: BR1/0:1 CHAP: I CHALLENGE id 3 len 28 from "NewYork"
20:08:11: BR1/0:1 CHAP: O RESPONSE id 3 len 28 from "Chicago"
20:08:11: BR1/0:1 CHAP: I SUCCESS id 3 len 4
20:08:12: %LINEPROTO-5-UPDOWN: Line protocol on Interface BRI1/0:1, changed state
to up
20:08:17: %ISDN-6-CONNECT: Interface BRI1/0:1 is now connected to 7802000
```

- Called side:

```
20:15:01: %LINK-3-UPDOWN: Interface BRI1/0:1, changed state to up
20:15:01: BR1/0:1 PPP: Using dialer call direction
20:15:01: BR1/0:1 PPP: Treating connection as a callin
20:15:02: BR1/0:1 CHAP: O CHALLENGE id 3 len 28 from "NewYork"
20:15:02: BR1/0:1 CHAP: I RESPONSE id 3 len 28 from "Chicago"
20:15:02: BR1/0:1 CHAP: O SUCCESS id 3 len 4
20:15:03: %LINEPROTO-5-UPDOWN: Line protocol on Interface BRI1/0:1, changed state
to up
20:15:07: %ISDN-6-CONNECT: Interface BRI1/0:1 is now connected to 7801000 Chicago
NewYork#
```

Two-way authentication. As with PAP, when configuring CHAP for two-way authentication, the difference is the removal of the `callin` keyword from the `ppp authentication chap` command on the calling router:

- Calling side:

```
hostname Chicago
!
username NewYork password 0 Secret2
!
interface BRI1/0
 encapsulation ppp
 ppp authentication chap
```

- Called side:

```
hostname NewYork
!
username Chicago password 0 Secret2
!
interface BRI1/0
 encapsulation ppp
 ppp authentication chap
```

Now, the output from `debug ppp authentication` is a bit more verbose, as authentication happens in both directions. I've included only the output for the called side here, for the sake of brevity:

```
20:01:59: %LINK-3-UPDOWN: Interface BRI1/0:1, changed state to up
20:01:59: BR1/0:1 PPP: Using dialer call direction
20:01:59: BR1/0:1 PPP: Treating connection as a callin
20:02:00: %ISDN-6-LAYER2UP: Layer 2 for Interface BR1/0, TEI 66 changed to up
20:02:00: BR1/0:1 CHAP: O CHALLENGE id 2 len 28 from "NewYork"
20:02:00: BR1/0:1 CHAP: I CHALLENGE id 2 len 28 from "Chicago"
20:02:00: BR1/0:1 CHAP: Waiting for peer to authenticate first
20:02:00: BR1/0:1 CHAP: I RESPONSE id 2 len 28 from "Chicago"
20:02:00: BR1/0:1 CHAP: O SUCCESS id 2 len 4
20:02:00: BR1/0:1 CHAP: Processing saved Challenge, id 2
20:02:00: BR1/0:1 CHAP: O RESPONSE id 2 len 28 from "NewYork"
20:02:00: BR1/0:1 CHAP: I SUCCESS id 2 len 4
20:02:01: %LINEPROTO-5-UPDOWN: Line protocol on Interface BRI1/0:1, changed state
to up
20:02:05: %ISDN-6-CONNECT: Interface BRI1/0:1 is now connected to 7801000 Chicago
```

Changing the sent hostname. Sometimes, the hostname of the calling router cannot be used for CHAP authentication. A common example is when you connect your router to an ISP. Figure 24-2 shows another simple two-router network: in this case, BobsRouter is connecting to a router named ISP. The service provider controlling the ISP router issues usernames and passwords to its clients. These usernames do not match the hostnames of the client's routers.

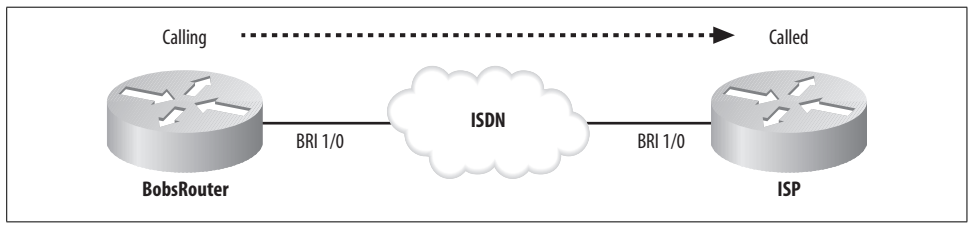


Figure 24-2. CHAP authentication with configured username

Bob, the client who is using BobsRouter, has been given the username Bob-01 and the password SuperSecret1. On the calling side, I've configured the additional command `ppp chap hostname`. This has the effect of using the name Bob-01 instead of the router's hostname for authentication. Notice that the username Bob-01 appears on the called side, and that there is no reference on the called side to the hostname of the calling side:

- Calling side:

```
hostname BobsRouter
!
username ISP password 0 SuperSecret1
!
interface BRI1/0
 encapsulation ppp
 ppp authentication chap callin
 ppp chap hostname Bob-01
```

- Called side:

```
hostname ISP
!
username Bob-01 password 0 SuperSecret1
!
interface BRI1/0
 encapsulation ppp
 ppp authentication chap
```

While this configuration works, chances are the only place you'd see it is on a certification exam. A far more logical approach is to configure the sent username and password in CHAP without having to configure a username that matches the hostname of the remote router:

- Calling side:

```
hostname BobsRouter
!
interface BRI1/0
```

```
 encapsulation ppp
 ppp authentication chap callin
 ppp chap hostname Bob-01
 ppp chap password 0 SuperSecret1
```

- Called side:

```
hostname ISP
!
username Bob-01 password 0 SuperSecret1
!
interface BRI1/0
 encapsulation ppp
 ppp authentication chap
```

Using this method, there is no confusion resulting from trying to match the hostnames and passwords in odd ways. Instead, you configure the username and password using the `ppp chap` interface commands on the calling side. The called side simply has the username and password configured to match. The hostnames, while included for completion, are not necessary in this example (though they are in all the previous CHAP examples).

Of all the non-AAA authentication methods available for PPP, this is the most secure, and the easiest to understand.

AAA Authentication

AAA stands for Authentication, Authorization, and Accounting. *Authentication* is the process of verifying a user's identity to determine whether the user should be allowed access to a device. *Authorization* is the act of limiting or permitting access to certain features within the device once a user has been authenticated. *Accounting* is the recording of actions taken by the user once she has been authenticated and authorized. In this section, I will cover only authentication, as it is the most commonly used feature offered by AAA.

To use AAA authentication on a switch or router, you must perform the following steps:

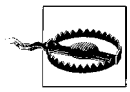
- Enable AAA by entering the `aaa new-model` command.
- Configure security server information, if using a security server. Configuring TACACS+ and RADIUS information is included in this step.
- Create method lists by using the `aaa authentication` command.
- Apply the method lists to interfaces or lines as needed.

Enabling AAA

To use the AAA features discussed here, you'll need to issue the command `aaa new-model`:

```
Router(config)# aaa new-model
```

If you don't execute this command, the AAA commands discussed in this section will not be available.



Be careful when configuring AAA for the first time. You can easily lock yourself out of the router by enabling AAA authentication without configuring any users.

Configuring Security Server Information

One of the benefits of using AAA is the ability to use an external server for authentication, authorization, and accounting. When an external server is used, all user information is stored externally to the networking device. Administration of user security is therefore centralized. This allows indivual users to access many devices, while also allowing the administrator to limit the users' access.

RADIUS and TACACS+ are two protocols used for authentication and authorization applications. Each is used to authenticate users, though they can also be used for various other features (logging command usage, call detail records, and so on). Both are widely used, and at some point, you'll need to decide which one to choose. Here's a quick rundown:

RADIUS
: Livingston Enterprises (now Lucent Technologies) originally developed the Remote Authentication Dial-In User Service (RADIUS) for its PortMaster series of network access servers. These devices were widely used by ISPs in the days when 33.6-Kbps modems were the norm. RADIUS was later described in RFCs 2058 and 2059. It is now available in open source server applications. RADIUS includes authentication and authorization in the user profile. RADIUS usually uses UDP ports 1812 or 1645 for authentication, and ports 1813 or 1646 for accounting.

TACACS+
: The Terminal Access Controller Access-Control System (TACACS) was originally designed for remote authentication of Unix servers. TACACS+, a new Cisco-proprietary protocol that is incompatible with the original version, has since replaced TACACS. This updated version is widely used for authentication and authorization in networking devices. TACACS+ separates authentication and authorization into separate operations. It is defined in a Cisco RFC draft (*http://tools.ietf.org/html/draft-grant-tacacs-02*) and utilizes TCP port 49 by default.

Cisco generally recommends TACACS+ over RADIUS, though both are usually available when configuring authentication. One important consideration is that RADIUS does not allow users to limit the commands a user can execute. If you need this feature, choose TACACS+. For more information on the differences between TACACS+ and RADIUS, see Cisco's document ID 13838 (*http://www.cisco.com/warp/public/480/10.html*).

To use a security server, you must configure *server groups*. Server groups are logical groups of servers that can be referenced with a single name. You can use default server groups, or create your own custom groups.

Default RADIUS and TACACS+ server groups

In a Cisco environment using ACS (Cisco's security authentication and authorization management system), TACACS+ is used. RADIUS is also supported if ACS or TACACS+ is not available.

TACACS+ servers are defined globally in a router using the `tacacs-server` command. Defining where to find a TACACS+ server is done with the `host` keyword:

```
tacacs-server host 10.100.100.100
```

A hostname can be used, provided that you have configured DNS on the router. You can also list multiple servers, in which case they will be referenced in the order in which they appear:

```
tacacs-server host 10.100.100.100
tacacs-server host 10.100.100.101
```

The router will query the second server in the list only if the first server returns an error, or is unavailable. A login failure is not considered an error.

Many installations require a secure key to be sent with the query. This key, which functions like a password for the server itself (as opposed to the user being authenticated), is configured through the `tacacs-server` command using the `key` keyword:

```
tacacs-server key Secret
```

The password will be stored in the configuration as plain text unless you have the password-encryption service enabled. With the password encrypted, the password line ends up looking like this:

```
tacacs-server key 7 01200307490E12
```

RADIUS servers are configured similarly. Most of the useful features are supplied in a single command. To accomplish the same sort of simple server configuration with a key, you could enter the following command:

```
radius-server host 10.100.200.200 key Secret
```

This will result in a configuration line that looks similar to this:

```
radius-server host 10.100.200.200 auth-port 1645 acct-port 1646 key Secret
```

Port 1645 is the default port for RADIUS, and was added automatically by the router.

As with TACACS+, you can add multiple servers:

```
radius-server host 10.100.200.200 auth-port 1645 acct-port 1646 key Secret
radius-server host 10.100.200.201 auth-port 1645 acct-port 1646 key Secret2
```

Again, the second server will be accessed only if the first returns an error. Notice, however, that with RADIUS you can have a different key for each server. TACACS+ only allows you to specify a global key for all servers.

Custom groups

Say you have two different sets of TACACS+ servers that you need to reference separately: two servers that you use for login authentication, and two servers that you use for PPP authentication.

IOS lets you specify custom groups for either RADIUS or TACACS+ servers. The `aaa group server` command is used to create these groups. Add the keyword `tacacs+` or `radius`, followed by the name of the group you'd like to create:

```
aaa group server tacacs+ Login-Servers
 server 10.100.1.100
 server 10.100.1.101

aaa group server radius PPP-Radius
 server 10.100.200.200 auth-port 1645 acct-port 1646
 server 10.100.200.201 auth-port 1645 acct-port 1646
```

Again, with RADIUS, the router adds the port numbers. The commands entered were simply:

```
R1(config)# aaa group server radius PPP-Radius
R1(config-sg-radius)# server 10.100.200.200
R1(config-sg-radius)# server 10.100.200.201
```

If you have a TACACS+ server that requires key authentication, you can add the key to an individual server within a group by using the `server-private` command instead of the `server` command:

```
aaa group server tacacs+ Login-Servers
 server-private 10.100.1.72 key Secret
```

Creating Method Lists

A *method list* is a list of authentication methods to be used in order of preference. For example, you may want to first try TACACS+ and then, if that fails, use local user authentication. Once you've created a method list, you can then configure an interface to call the method list for authentication.

A router can authenticate a user in a few different ways. They are:

Login
: Login authentication is the means whereby a user is challenged to access the router's CLI.

PPP
: PPP authentication provides authentication for Point-to-Point Protocol connectivity either on serial links, or through something like modem connectivity.

ARAP
: AppleTalk Remote Access Protocol is a remote access protocol for AppleTalk users.

NASI
: NetWare Asynchronous Services Interface is a remote access protocol for Novell Netware users.

In practice, you're only likely to encounter login and PPP authentication. With the advent of broadband Internet access in most homes, and the adoption of VPN at most companies, modem connectivity is becoming a thing of the past in most metropolitan areas. I haven't seen ARAP or NASI in the field in years, so I'll only cover login and PPP authentication here.

Login authentication

When logging into a network device, you can be challenged in a variety of ways. The default method is to be challenged to provide a password that's been entered in the configuration of the interface or line itself. For example, the following commands would secure the console with a simple password:

```
line con 0
 password Secret1
```

This behavior is called *line authentication* when using AAA, and is one of many methods available for authentication. The possible methods for login authentication are:

`enable`
: Use the configured `enable` password as the authentication password.

`krb5`
: Query a Kerberos 5 authentication server for authentication information.

`krb5-telnet`
: Use the Kerberos 5 telnet authentication protocol when using telnet to connect to the network device. This method must be listed first if used.

`line`
: Use the configured `line` password as the authentication password.

`local`
: Use locally configured usernames and passwords (entered in the configuration of the device itself).

`local-case`
: Same as local, but the usernames are case-sensitive.

`none`
: This method essentially removes authentication. If `none` is the only method, access is granted without challenge.

`group radius`
: Query the list of RADIUS servers for authentication information.

`group tacacs+`
: Query the list of TACACS+ servers for authentication information.

Custom (`group` *group-name)*
: Query a custom group as defined in the local configuration.

A method list can contain multiple methods, or just one. Method lists must be named. Here, I've specified a `login` method list called GAD-Method. The method being used is `local` users:

```
aaa authentication login GAD-Method local
```

If multiple methods are listed, they will be referenced in the order in which they appear. The second method will be referenced only if the first method fails; failure is not authentication failure, but rather, a failure to establish connectivity with that method.

Here, I've configured the GAD-Method method list to use TACACS+ first, followed by local users:

```
aaa authentication login GAD-Method group tacacs+ local
```

When using the server groups `tacacs+` and `radius`, you are referencing the globally configured TACACS+ and RADIUS servers. If you have defined a custom group of either type, you can reference the group name you created with the `aaa servers` command. For example, earlier we created a `Login-Servers` group. To reference this group in a method list, we would include the group name after the keyword `group`:

```
aaa authentication login default group Login-Servers
```

This example includes the `default` method, which, when implemented, is automatically applied to all interfaces.

If you're relying on external servers, and problems are encountered, you can sometimes lock everyone out of the router. The `none` method allows anyone to access the router without authenticating. When the `none` method is included as the last method in a list, anyone will be able to access the router in the event that all other authentication methods fail:

```
aaa authentication login default group tacacs+ local none
```

Again, failure is defined here as failed communication with the server listed, not an incorrect password entry. Of course, including `none` can be dangerous, as it means that a malicious party can launch a denial-of-service attack on the authentication servers, and thereby gain access to your devices (which is counterindicated by most network administrators). Instead, I like to use `local-case` as the last method in my method lists:

```
aaa authentication login default group tacacs+ local-case
```

I like to configure on all routers a standard username and password that will only be needed in the event of a server or network failure. I feel slightly better about using `local-case` than `local`, as it means I can include both upper- and lowercase characters in the usernames and passwords. Be careful, though, as this practice is frowned upon in environments where credit card transactions occur. Payment Card Industry (PCI) compliance dictates many details about how user data is stored and accessed.

You can use the same method list name for both PPP and login authentication, but be aware that creating a `login` method list doesn't automatically result in the creation of a `ppp` method list with the same name. If you want to have login and PPP method lists with the same names, you'll need to create them both:

```
aaa authentication login GAD-Login group GAD-Servers
none
aaa authentication ppp GAD-Login group GAD-Servers
none
```

PPP authentication

PPP authentication is used when Point-to-Point Protocol connections are initiated into the router. These can include modem connections into a serial interface, or connections of high-speed serial links such as T1s.

The possible methods for PPP authentication using AAA are:

`if-needed`
: If a user has already been authenticated on a TTY line, do not authenticate.

`krb5`
: Query a Kerberos 5 authentication server for authentication information (only for PAP).

`local`
: Use locally configured usernames and passwords (entered in the configuration of the device itself).

`local-case`
: Same as local, but the usernames are case-sensitive.

`none`
: This method essentially removes authentication. If `none` is the only method, there is no challenge.

`group radius`
: Query the list of RADIUS servers for authentication information.

`group tacacs+`
: Query the list of TACACS+ servers for authentication information

Custom (`group` *group-name)*
: Query a custom group as defined in the local configuration.

These methods are referenced in ppp method lists the same way that methods are referenced in login method lists. An interesting addition to the list of methods is if-needed. This method instructs the router to authenticate the incoming connection only if the user presented has not already been authenticated on a VTY, console, or aux line. Here is a sample ppp method list:

```
aaa authentication ppp default group tacacs+ local-case
```

Applying Method Lists

Once you have created a method list, it needs to be applied to the interface or line where you would like it to take effect. With login authentication, the command login is used to apply an authentication method list. Here, I'm applying the GAD-Login method list created earlier to VTY lines 0–4. This will have the effect of challenging telnet sessions to the router with whatever authentication methods exist in the GAD-Login method list:

```
line vty 0 4
 login authentication GAD-Login
```

To apply a PPP authentication method list to an interface, the interface must be configured with PPP encapsulation. The ppp authentication command is used to enable authentication. The authentication protocol must be specified along with the method list. Here, I have specified CHAP along with the method list GAD-Login:

```
interface Serial0/0/0:0
 no ip address
 encapsulation ppp
 ppp authentication chap GAD-Login
```

If you have not created a PPP method list, you will get an error, though the command will still be accepted:

```
R1(config-if)# ppp authentication pap GAD-Login
AAA: Warning, authentication list "GAD-Login" is not defined for PPP.
```

Using AAA with PAP or CHAP is much more scalable than using locally configured usernames and passwords.

CHAPTER 25

Firewall Theory

A firewall is the wall in a car that protects you from harm when the engine catches fire. At least, that's the definition that confused my mother when I told her I was writing this chapter. In networking, a firewall is a device that prevents certain types of traffic from entering or leaving your network. Usually, the danger comes from attackers attempting to gain access to your network from the Internet, but not always. Firewalls are often deployed when connecting networks to other entities that are not trusted, such as partner companies.

A firewall can be a standalone appliance, software running on a server or router, or a module integrated into a larger device, like a Cisco 6500 switch. These days, the functionality of a firewall is often included in other devices, such as the ubiquitous cable-modem/router/firewall/wireless-access-point devices in many homes.

Modern firewalls can serve multiple functions, even when they're not part of combination devices. VPN services are often supported on firewalls. A firewall running as an application on a server may share the server with other functions such as DNS or mail, though generally, a firewall should restrict its activities to security-related tasks.

Best Practices

One of the things I tell my clients over and over is:

> Security is a balance between convenience and paranoia.

We all want security. If I told you that I could guarantee the security of your family, wouldn't you jump at the chance? But what if I told you that to achieve this goal, I needed to put steel plates over all the windows in your home, replace the garage door with a brick wall, and change the front door for one made of cast iron? You might reconsider—it wouldn't be very convenient, would it? Companies also often want a high level of security, but like you, they may not be willing to give up too many conveniences to achieve it.

A while ago, I was working as a consultant in Manhattan for a large firm that was having security problems. We gave them some options that we knew had worked for other organizations, and these were the responses we received:

One-time password key fobs
: "We don't want that—the key fobs are a pain, and it takes too long to log in."

VPN
: "We like the idea, but can you make it so we don't have to enter any passwords?"

Putting the email server inside *the firewall*
: "Will we have to enter more than one password? Because if we do, forget it."

Password rotation
: "No way—we don't want to ever have to change our passwords!"

Needless to say, the meeting was a bit of a challenge. The clients wanted security, and they got very excited when we said we could offer them the same level of network security that the big banks used. But the minute they realized what was involved in implementing that level of security, they balked—they balked at the idea of any inconvenience.

More often than not, companies do come to an understanding that they need a certain level of security, even if some conveniences must be sacrificed for its sake. Sadly, for many companies, this happens after their existing security has been compromised. Others may be forced into compliance by regulations like Sarbanes-Oxley.

If you find yourself designing a security solution, you should follow these best practices:

Simple is good
: This rule applies to all of networking, but it is especially relevant for security rules. When you are designing security rules and configuring firewalls, keep it simple. Make your rules easy to read and understand. Where applicable, use names instead of numbers.

Monitor the logs
: You must log your firewall status messages to a server, and you must look at these messages on a regular basis. If you have a firewall in place, and you're not examining the logs, you are living with a false sense of security. Someone could be attacking your network right now, and you'd have no idea. I've worked on sites that kept buying more Internet bandwidth, amazed at how much they needed. When I examined their firewall logs, I discovered that the main bandwidth consumers were warez sites that hackers had installed on their internal FTP servers. Because no one looked at the logs, no one knew there was a problem.

Deny everything; permit what you need
: This is a very simple rule, but it's amazing how often it's ignored. As a best practice, this has got to be the one with the biggest benefit.

  In practical terms, blocking all traffic in both directions is often viewed as too troublesome. This rule should always be followed to the letter on inbound

firewalls—nothing should ever be allowed inbound unless there is a valid, documented business need for it. Restricting all outbound traffic except that which is needed is also the right thing to do, but it can be an administrative hassle. Here is a prime example of convenience outweighing security. On the plus side, if you implement this rule, you'll know that peer-to-peer file sharing services probably won't work, and you'll have a better handle on what's going on when users complain that their newly installed instant-messenger clients don't work. The downside is that unless you have a documented security statement, you'll spend a lot of time arguing with people about what's allowed and what's not.

The default behavior of many firewalls, including the Cisco PIX, is to allow all traffic outbound. Restricting outbound traffic may be a good idea based on your environment and corporate culture, though I've found that most small and medium-sized companies don't want the hassle. Additionally, many smaller companies don't have strict Internet usage policies, which can make enforcing outbound restrictions a challenge.

Everything that's not yours belongs outside the firewall

This is another simple rule that junior engineers often miss. Anything from another party that touches your network should be controlled by a firewall. Network links to other companies, including credit card verification services, should never be allowed without a firewall.

The corollary to this rule is that everything of yours should be inside the firewall (or in the DMZ). The only devices that are regularly placed in such a way that the firewall cannot monitor them are VPN concentrators. VPN concentrators are often placed in parallel with firewalls. Everything else should be segregated with the firewall. Segregation can be accomplished with one or more DMZs.

Firewalls get blamed for everything. It seems to be a law of corporate culture to blame the firewall the minute anything doesn't work. I believe there are two reasons for this. First, we naturally blame what we don't understand. Second, a firewall is designed to prevent traffic from flowing. When traffic isn't flowing, it makes sense to blame the firewall.

The DMZ

Firewalls often have what is commonly called a *DMZ*. DMZ stands for DeMilitarized Zone, which of course has nothing to do with computing. This is a military/political term referring to a zone created between opposing forces in which no military activity is allowed. For example, a demilitarized zone was created between North and South Korea.

In the realm of security, a DMZ is a network that is neither inside nor outside the firewall. The idea is that this third network can be accessed from inside, and probably outside the firewall, but security rules will prohibit devices in the DMZ

from connecting to devices on the inside. A DMZ is less secure than the inside network, but more secure than the outside network.

A common DMZ scenario is shown in Figure 25-1. The Internet is located on the outside interface. The users are on the inside interface. Any servers that need to be accessible from the Internet are located in the DMZ network.

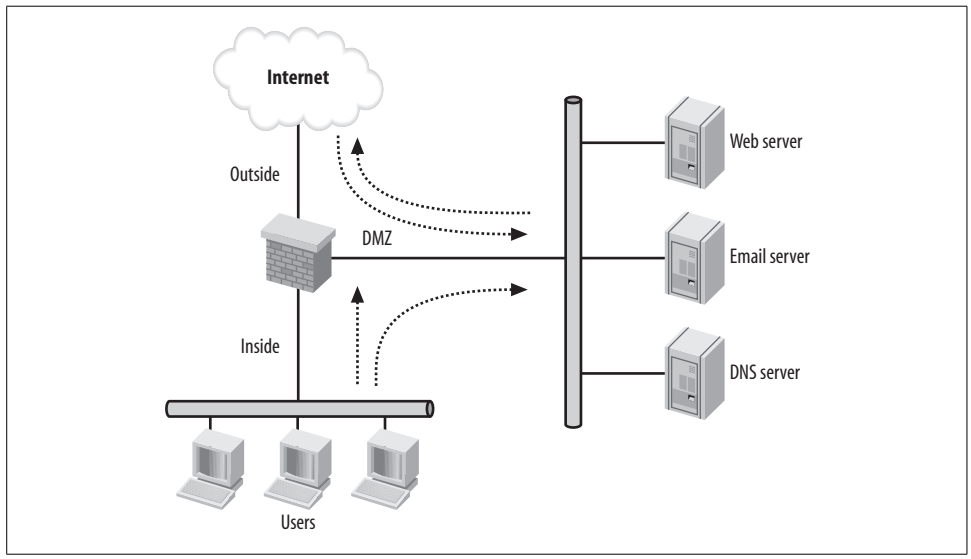


Figure 25-1. Simple DMZ network

The firewall would be configured as follows:

Inside network
: The inside network can initiate connections to any other network, but no other network can initiate connections to it.

Outside network
: The outside network cannot initiate connections to the inside network. The outside network can initiate connections to the DMZ.

DMZ
: The DMZ can initiate connections to the outside network, but not to the inside network. Any other network can initiate connections into the DMZ.

One of the main benefits of this type of design is isolation. Should the email server come under attack and become compromised, the attacker will not have access to the users on the inside network. However, in this design, the attacker *will* have access to the other servers in the DMZ because they're on the same physical network. (The servers can be further isolated with Cisco Ethernet switch features such as private VLANs, port ACLs, and VLAN maps. See Chapter 23 for more information.)

Servers in a DMZ should be locked down with security measures as if they were on the Internet. Rules on the firewall should be configured to allow services only as needed to the DMZ. For example:

Email server
: POP, IMAP, and SMTP (TCP ports 110, 143, and 25) should be allowed. All other ports should not be permitted from the Internet.

Web server
: HTTP and HTTPS (TCP ports 80 and 443) should be allowed. All other ports should be denied from the Internet.

DNS server
: Only DNS (UDP port 53, and, possibly, TCP port 53) should be allowed from the Internet. All other ports should be denied.

Ideally, only the protocols needed to manage and maintain the servers should be allowed from the managing hosts inside to the DMZ.

Another DMZ Example

Another common DMZ implementation involves connectivity to a third party, such as a vendor or supplier. Figure 25-2 shows a simple network where a vendor is connected by a T1 to a router in the DMZ. Examples of vendors might include a credit card processing service, or a supplier that allows your users to access its database. Some companies even outsource their email to a third party, in which case the vendor's email server may be accessed through such a design.

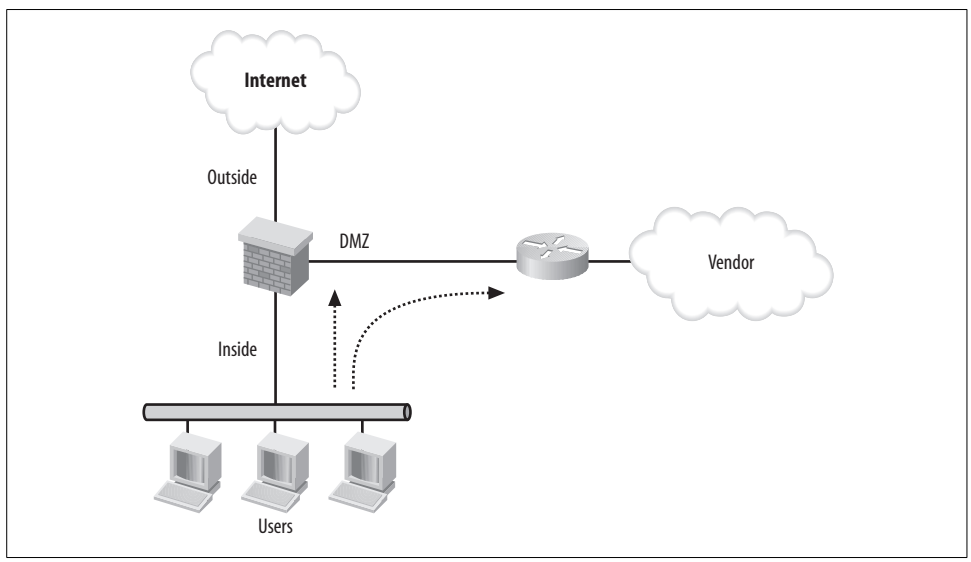


Figure 25-2. DMZ connecting to a vendor

In a network like this, the firewall would be configured as follows:

Inside network
: The inside network can initiate connections to any other network, but no other network can initiate connections to it.

Outside network
: The outside network cannot initiate connections to the inside network or to the DMZ. The inside network can initiate connections to the outside network, but the DMZ cannot.

DMZ
: The DMZ cannot initiate connections to any network. Only the inside network can initiate connections to the DMZ.

Multiple DMZ Example

The real world is not always as neat and orderly as my drawings would have you believe. The examples I've shown are valid, but larger companies have more complicated networks. Sometimes, a single DMZ is not enough.

Figure 25-3 shows a network with multiple DMZs. The design is a combination of the first two examples. Outside is the Internet, and inside are the users. DMZ-1 is a connection to a vendor. DMZ-2 is where the Internet servers reside. The security rules are essentially the same as those outlined in the preceding section, but we must now also consider whether DMZ-1 should be allowed to initiate connections to DMZ-2, and vice versa. In this case, the answer is no.

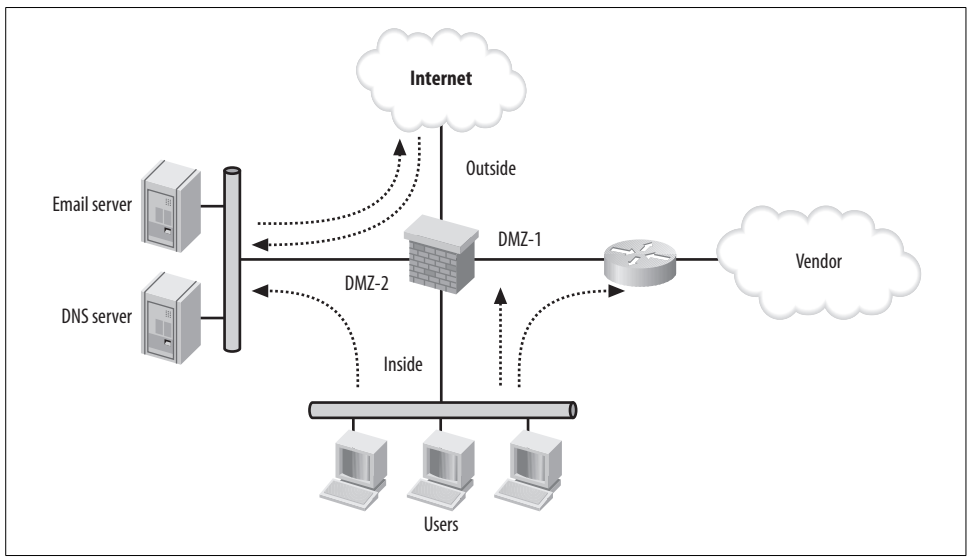


Figure 25-3. Multiple DMZs

The firewall should be configured as follows:

Inside network
: The inside network can initiate connections to any other network, but no other network can initiate connections to it.

Outside network
: The outside network cannot initiate connections to the inside network, or to DMZ-1. The outside network can initiate connections to DMZ-2.

DMZ-1
: DMZ-1 cannot initiate connections to any other network. Only the inside network can initiate connections into DMZ-1.

DMZ-2
: DMZ-2 can initiate connections only to the outside network. The outside network and the inside network can initiate connections to DMZ-2.

Alternate Designs

The Internet is not always the outside interface of a firewall. Many companies have links to other companies (parent companies, sister companies, partner companies, etc.). In each case, even if the companies are related, separating the main company from the others with a firewall is an excellent best practice to adopt.

Figure 25-4 shows a simplified layout where Your Company's Network is connected to three other external entities. Firewall A is protecting Your Company from the Internet, Firewall B is protecting Your Company from the parent company, and Firewall C is protecting Your Company from the sister company.

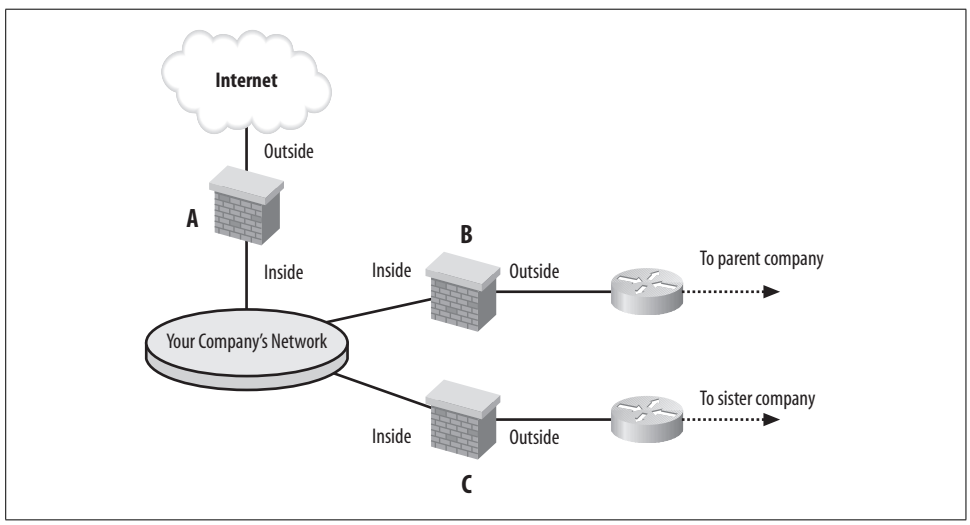


Figure 25-4. Multiple firewall example

Each of the firewalls has an inside and an outside interface. While each of the firewalls' inside interfaces are connected to the same network, the outside interfaces are all connected to different networks.

Firewalls are also often used in multitiered architectures like those found in e-commerce web sites. A common practice is to have firewalls not only at the point where the web site connects to the Internet, but between the layers as well. Figure 25-5 shows such a network.

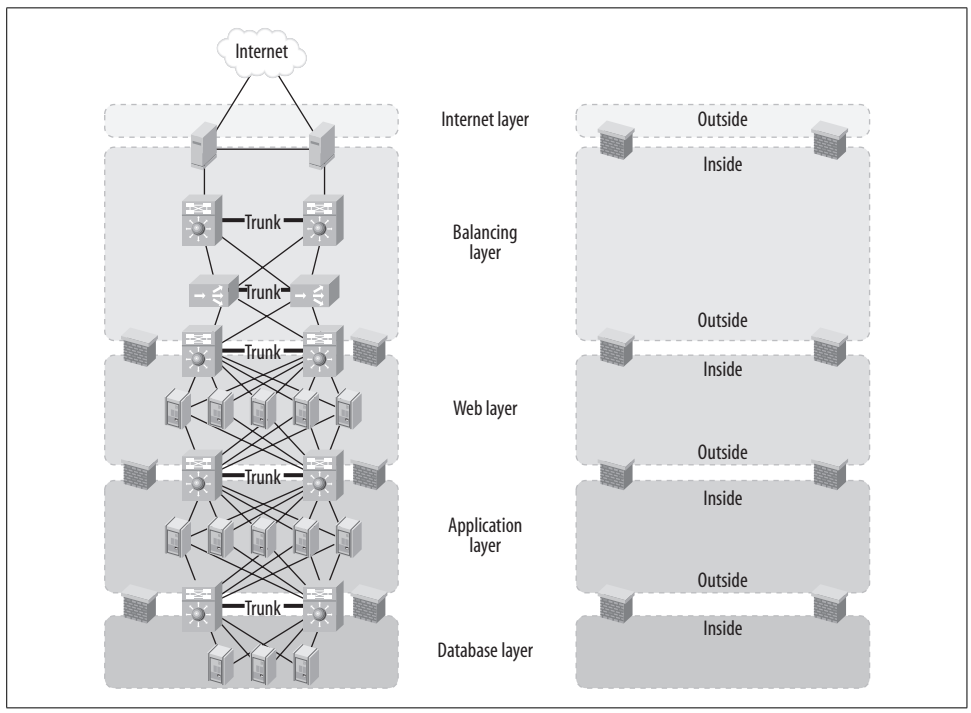


Figure 25-5. E-commerce web site

In a layered design like this, one firewall's inside network is the next firewall's outside network. There are four firewalls connected to the balancing layer. The top two, a failover pair, connect the balancing layer to the Internet layer. To these firewalls, the balancing layer is the inside network. The bottom two firewalls (another failover pair) connect the balancing layer to the web layer. To these firewalls, the balancing layer is the outside network.

Firewalls are another building block in your arsenal of networking devices. While there are some common design rules that should be followed, such as the ones I've outlined here, the needs of your business will ultimately determine how you deploy your firewalls.

CHAPTER 26

PIX Firewall Configuration

In this chapter, I will explain how to configure the most common features of a PIX firewall. Examples will be based on the PIX 515, which uses the same commands as the entire PIX line, from the PIX 501 to the 535, and the Firewall Services Module (FWSM).

Slight differences do appear between models. For example, the PIX 501 and 506e cannot be installed in failover pairs, and the PIX 506e has only two interfaces, and cannot be expanded. The FWSM also operates differently in that it is a module and has no configurable physical interfaces.

PIX firewalls can be a bit confusing for people whose experience is with IOS-based devices. While there are similarities in the way the command-line interpreter works, there are some pretty interesting differences, too. One of my favorite features of the PIX OS is the fact that you can execute the `show running-config` command from within configuration mode. Recent versions of IOS allow similar functionality using the `do` command (`do show run` from within configuration mode), but using the command in the PIX is, in my opinion, more natural.

Interfaces and Priorities

Each interface in a PIX firewall must have a physical name, a logical name, a priority, and an IP address. Interfaces may also be configured for features such as speed and duplex mode.

On the PIX 515, the standard physical interfaces are E0 and E1, even though the interfaces support 100 Mbps Ethernet. An expansion card can be installed to add interfaces, which are numbered incrementally, starting at E2. Each interface must be assigned a logical name. The default names are *inside* for the E1 interface, and *outside* for the E0 interface.

Each interface must also have a priority assigned. The priority establishes the level of security for the interface. By default, interfaces with higher priorities can send packets to interfaces with lower priorities, but interfaces with lower priorities cannot send packets to interfaces with higher priorities. An interface's priority is represented by an integer within the range of 0–100.

The default priority for the inside interface is 100. The default priority for the outside interface is 0. If you add a third interface, its priority will typically be somewhere between these values (PIX OS v7.0 introduced the ability to configure multiple interfaces with the same priority). Figure 26-1 shows a typical PIX firewall with three interfaces: outside and inside interfaces, and a DMZ.

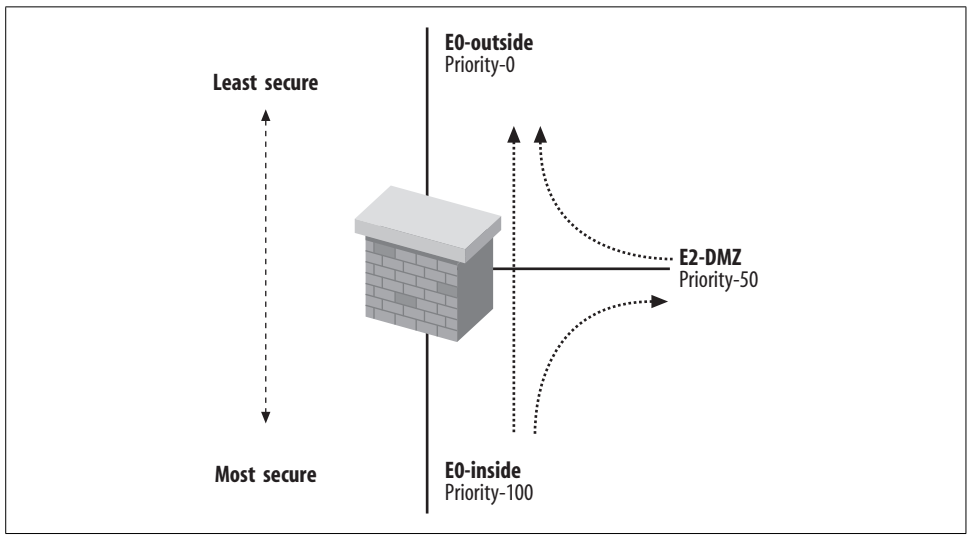


Figure 26-1. PIX firewall priorities

Given this design, by default, traffic may flow in the following ways:

- From inside to outside.
- From inside to the DMZ.
- From the DMZ to the outside, but not to the inside.
- Traffic from outside may not flow to any other interface.

The command to configure the name and priority of an interface is `nameif`. For example:

```
PIX(config)# nameif ethernet0 outside security0
PIX(config)# nameif ethernet1 inside security100
```

These commands are the default configuration for a PIX firewall with Ethernet interfaces. To configure the third interface in Figure 26-1, you would use the following command:

```
PIX(config)# nameif ethernet2 DMZ security50
```

To configure the speeds and duplex modes of the interfaces, use the `interface` command:

```
PIX(config)# interface ethernet0 auto
PIX(config)# interface ethernet1 100full
```

Notice that even though the inside interface's name is *ethernet1*, it's configured for 100 Mbps. In an IOS router or switch, an interface capable of 100 Mbps would be named *FastEthernet1*. PIX firewalls do not run IOS, though, so try not to assume anything based on your knowledge of IOS.

To show the statuses of your interfaces, use the `show interface` command. The output of this command is similar to that produced in IOS:

```
PIX# sho int
interface ethernet0 "outside" is up, line protocol is up
  Hardware is i82559 ethernet, address is 0055.55ff.1111
  IP address 10.10.10.1, subnet mask 255.255.255.248
  MTU 1500 bytes, BW 100000 Kbit full duplex
        95524227 packets input, 2181154879 bytes, 0 no buffer
        Received 82412 broadcasts, 0 runts, 0 giants
        0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
        82480261 packets output, 3285501304 bytes, 0 underruns
        0 output errors, 0 collisions, 0 interface resets
        0 babbles, 0 late collisions, 0 deferred
        6 lost carrier, 0 no carrier
        input queue (curr/max blocks): hardware (128/128) software (0/8)
        output queue (curr/max blocks): hardware (0/21) software (0/1)
interface ethernet1 "inside" is up, line protocol is up
  Hardware is i82559 ethernet, address is 0055.55ff.1112
  IP address 192.168.1.1, subnet mask 255.255.255.0
  MTU 1500 bytes, BW 100000 Kbit full duplex
        83033846 packets input, 3326592360 bytes, 0 no buffer
        Received 632782 broadcasts, 0 runts, 0 giants
        0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
        101485765 packets output, 3300678940 bytes, 0 underruns
        0 output errors, 0 collisions, 0 interface resets
        0 babbles, 0 late collisions, 0 deferred
        0 lost carrier, 0 no carrier
        input queue (curr/max blocks): hardware (128/128) software (0/21)
        output queue (curr/max blocks): hardware (1/53) software (0/1)
```

Names

One of the more useful features of the PIX OS is the ability to display IP addresses as names. To enable this feature, enter the `names` command in configuration mode:

```
PIX(config)# names
```

With the names feature enabled, you can configure any IP address to be associated with a name. This is similar in principle to a basic form of DNS, but the names are local to the PIX being configured. Say that 10.10.10.10 is the IP address of a server called *FileServer*. Using the `name` command, you can assign the name *FileServer* to the IP address within the PIX:

```
PIX(config)# name 10.10.10.10 FileServer
```

You can then configure an access list like the following:

```
PIX(config)# access-list 110 permit tcp any host 10.10.10.10 eq www
```

Access lists, including features specific to the PIX, are covered in detail in Chapter 23.

In the configuration, the IP address will be translated to the configured name:

```
PIX# sho run | include 110
access-list 110 permit tcp any host FileServer eq www
```

If you prefer to see the IP addresses, you can disable the names feature by negating the `names` command. The configuration will once again show the IP addresses:

```
PIX(config)# no names
PIX(config)# sho run | include 110
access-list 110 permit tcp any host 10.10.10.10 eq www
```

Even with names enabled, the output of the `show interface` command will always show the IP addresses.

If you need to see all the names configured on your PIX firewall, use the `show names` command:

```
PIX# sho names
name 10.10.10.1 PIX-Outside
name 10.10.10.10 FileServer
name 192.168.1.1 PIX-Inside
```

The names feature is extremely helpful in that it makes PIX firewall configurations easier to read. With very large configurations, the number of IP addresses can be staggering, and trying to remember them all is a practical impossibility.

Object Groups

Object groups allow a group of networks, IP addresses, protocols, or services to be referenced with a single name. This is extremely helpful when configuring complex access lists. Take the situation shown in Figure 26-2. There are three web servers, each of which offers the same three protocols: SMTP (TCP port 25), HTTP (TCP port 80), and HTTPS (TCP port 443).

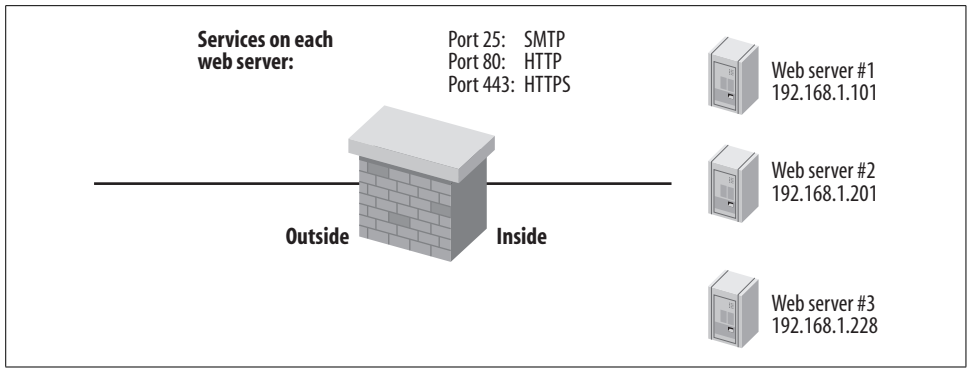


Figure 26-2. Complex access-list scenario

This example shows a collocated web site. On a normal enterprise network, web servers should not reside on the inside network, but rather in a DMZ.

Because the IP addresses of the three servers are not in a range that can be addressed with a single subnet mask, each of the servers must have its own access-list entry. Additionally, there must be an entry for each protocol for each server.

As a result, nine access-list entries must be configured to allow each of the three protocols to these three servers:

```
access-list In permit tcp any host 192.168.1.101 eq smtp
access-list In permit tcp any host 192.168.1.101 eq www
access-list In permit tcp any host 192.168.1.101 eq https
access-list In permit tcp any host 192.168.1.201 eq smtp
access-list In permit tcp any host 192.168.1.201 eq www
access-list In permit tcp any host 192.168.1.201 eq https
access-list In permit tcp any host 192.168.1.228 eq smtp
access-list In permit tcp any host 192.168.1.228 eq www
access-list In permit tcp any host 192.168.1.228 eq https
```

While this may not seem like a big deal, imagine if the firewall had six interfaces, and supported 40 servers. I've seen PIX firewalls that had access lists 17 printed pages long. Figuring out all the permutations of protocols and servers can be maddening. The potential complexity of the access lists has led many businesses to ignore the PIX when considering firewalls.

Version 6 of the PIX OS introduced the idea of object groups to solve this problem. With the `object-group` command, you can create a group of protocols, networks, ICMP types, or services that you can reference by a name:

```
PIX(config)# object-group ?
Usage:  [no] object-group protocol | network | icmp-type <obj_grp_id>
        [no] object-group service <obj_grp_id> tcp|udp|tcp-udp
        show object-group [protocol | service | icmp-type | network]
        show object-group id <obj_grp_id>
```

```
clear object-group [protocol | service | icmp-type | network]
clear object-group counters
```

In the preceding example, each of the web servers is using the same TCP ports. By assigning these common ports to a group, we can make the access list much smaller. Let's create an object group called *Webserver-svcs*. This will be a group of TCP services, which we'll define using `port-object` object-group commands:

```
object-group service Webserver-svcs tcp
  description For Webservers
  port-object eq smtp
  port-object eq www
  port-object eq https
```

Now, instead of listing each service for each web server, we can simply reference the group for each web server. We do this by using the `object-group` keyword, followed by the group name:

```
access-list In permit tcp any host 192.168.1.101 object-group Webserver-svcs
access-list In permit tcp any host 192.168.1.201 object-group Webserver-svcs
access-list In permit tcp any host 192.168.1.228 object-group Webserver-svcs
```

This reduces the number of access-list entries from nine to three, but we can do better. All of the IP addresses listed serve the same purpose—they are all web servers. Let's create another object group called *Webservers*. This time, the object-group type will be `network`, and we'll use the `network-object` command to add objects to the group:

```
object-group network Webservers
  description Webservers
  network-object host 192.168.1.101
  network-object host 192.168.1.201
  network-object host 192.168.1.228
```

We can now simplify the access list even more:

```
access-list In permit tcp any object-group Webservers object-group Webserver-svcs
```

What started as a nine-line access list has been compressed to one line. When we execute the `show access-list` command, the object groups will be expanded, and the resulting access list will be visible:

```
PIX# sho access-list

TurboACL statistics:
ACL                     State       Memory(KB)
----------------------- ----------- ----------
In                      Operational 2

Shared memory usage: 2056 KB
access-list compiled
access-list cached ACL log flows: total 0, denied 0 (deny-flow-max 1024)
            alert-interval 300
```

```
access-list In line 1 permit tcp any object-group Webservers object-group
Webserver-svcs
access-list In line 1 permit tcp any host 192.168.1.101 eq smtp (hitcnt=0)
access-list In line 1 permit tcp any host 192.168.1.101 eq www (hitcnt=0)
access-list In line 1 permit tcp any host 192.168.1.101 eq https (hitcnt=0)
access-list In line 1 permit tcp any host 192.168.1.201 eq smtp (hitcnt=0)
access-list In line 1 permit tcp any host 192.168.1.201 eq www (hitcnt=0)
access-list In line 1 permit tcp any host 192.168.1.201 eq https (hitcnt=0)
access-list In line 1 permit tcp any host 192.168.1.228 eq smtp (hitcnt=0)
access-list In line 1 permit tcp any host 192.168.1.228 eq www (hitcnt=0)
access-list In line 1 permit tcp any host 192.168.1.228 eq https (hitcnt=0)
```

Notice that the line number for each entry is the same (line 1). This indicates that these entries are a result of the expansion of line 1, which in this example is the only line in the access list.

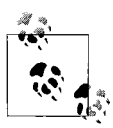
Fixups

Fixups are features that inspect application protocols. They are used to enable complex protocols such as FTP that have multiple streams. They are also used to make protocols more secure. For example, the SMTP fixup limits the commands that can be run through the PIX within the SMTP protocol.

To illustrate one of the common fixup applications, I've connected through a PIX firewall to a mail server using telnet. The PIX firewall is not running the SMTP fixup. When I issue the SMTP command `EHLO someserver`, I get a list of information regarding the capabilities of the mail server:

```
[GAD@someserver GAD]$ telnet mail.myserver.net 25
Trying 10.10.10.10...
Connected to mail.myserver.net.
Escape character is '^]'.
220 mail.myserver.net ESMTP Postfix
EHLO someserver
250-mail.myserver.net
250-PIPELINING
250-SIZE 10240000
250-ETRN
250 8BITMIME
```

This information is not necessary for the successful transfer of email, and could be useful to a hacker. For example, a hacker could try to pull email off of the server using the `ETRN deque` command. The SMTP fixup intercepts and disables the `ETRN` command.



ETRN is a very useful feature of SMTP that allows ISPs to queue mail for you should your email server become unavailable. If you need to use ETRN, you will have to disable the SMTP fixup on your PIX firewall.

I'll enable the fixup on the firewall now, using the `fixup` command. I must specify the protocol, and the port on which the protocol listens (in this case, port 25):

```
PIX(config)# fixup protocol smtp 25
```

Now the PIX will intercept and manage every SMTP request:

```
[GAD@someserver GAD]$ telnet mail.myserver.net 25
Trying 10.10.10.10...
Connected to mail.myserver.net.
Escape character is '^]'.
220 ************************
EHLO someserver
502 Error: command not implemented
```

Look at the items in bold, and compare the output to the previous example. Without the SMTP fixup enabled, the server responded to the telnet request with the name of the mail server, the version of SMTP supported, and the mail transfer agent (MTA) software in use:

```
220 mail.myserver.net ESMTP Postfix.
```

With the SMTP fixup enabled, the firewall intercepts this reply, and alters it to something useless:

```
220 ************************
```

This gives hackers much less to work with. Likewise, the fixup prevents the execution of the `EHLO someserver` command.

Different fixups are enabled by default on different versions of the PIX OS. On Version 6.2, the default fixups are:

```
fixup protocol ftp 21
fixup protocol http 80
fixup protocol h323 1720
fixup protocol rsh 514
fixup protocol smtp 25
fixup protocol sqlnet 1521
fixup protocol sip 5060
```

Some of these fixups may not be needed and can be disabled, though they usually don't hurt anything when left active. To see which ones are active on your PIX, use the `show fixup` command. Here, you can see that I've disabled the H323, RSH, Skinny, and SQLNET fixups:

```
PIX# sho fixup
fixup protocol dns maximum-length 512
fixup protocol ftp 21
no fixup protocol h323 h225 1720
fixup protocol h323 ras 1718-1719
fixup protocol http 80
no fixup protocol rsh 514
```

```
fixup protocol rtsp 554
fixup protocol sip 5060
fixup protocol sip udp 5060
no fixup protocol skinny 2000
fixup protocol smtp 25
no fixup protocol sqlnet 1521
fixup protocol tftp 69
```

Each fixup addresses the needs of a specific protocol. See the Cisco documentation for details.

Failover

PIX firewalls can be configured in high-availability pairs. In this configuration, should the primary PIX fail, the secondary will take over. All PIX firewalls are capable of being configured for failover, except the 501 and 506e models. To use this feature, the PIX must be licensed for it. To determine whether your PIX is capable of being configured for failover, use the `show version` command:

```
PIX# sho version | include Failover
Failover:                   Enabled
```

To be installed as a failover pair, each PIX firewall must have the same PIX software release installed. Each PIX in a failover pair must also have the exact same configuration. As a result, the hostname will be the same on both firewalls in the pair. If you attempt to configure the standby firewall, you will receive an error telling you that any changes you make will not be synchronized:

```
PIX# conf t
**** WARNING ***
     Configuration Replication is NOT performed from Standby unit to Active unit.
     Configurations are no longer synchronized.
```

You won't actually be prevented from making the changes, though. I have stared stupidly at this message more times than I can count while making changes after working for 18 hours straight.

Failover Terminology

When in a failover pair, PIX firewalls are referenced by specific names, depending on their roles:

Primary
: The primary PIX is the firewall on the `primary` end of the failover cable. This is a physical designation. On FWSMs, or models that do not use the failover cable, the primary PIX is configured manually using the `failover lan unit primary` command. The primary PIX is usually *active* when the pair is initialized. Once designated, it does not change.

Secondary
: The secondary PIX is the firewall on the `secondary` end of the failover cable, and is not usually configured directly, unless the primary PIX fails. This is a physical designation. The secondary PIX is usually the *standby* when the pair is initialized. Once designated, it does not change.

Active
: The active PIX is the firewall that is inspecting packets. It controls the pair. The active PIX uses the system IP address configured for each interface. This is a logical designation; either the primary or the secondary PIX can be the active PIX.

  Some PIX firewall models now support active/active failover, where both physical firewalls can pass traffic simultaneously. See the Cisco documentation for more information.

Standby
: The standby PIX is the firewall that is not inspecting packets. It uses the failover IP address configured for each interface. Should the active PIX fail, the standby PIX will take over and become active. This is a logical designation; either the primary or the secondary PIX can be the standby PIX.

Stateful failover
: When the active PIX fails, and the standby PIX takes over, by default, all conversations that were active through the active PIX at the time of the failure are lost. To prevent these connections from being lost, a dedicated Ethernet link between the active and standby PIX firewalls can be used to exchange the state of each conversation. With stateful failover configured, the standby PIX is constantly updated so that no connections are lost when a failover occurs.

Understanding Failover

The primary and secondary PIX communicate over the failover cable or the configured failover interface. The failover cable is a modified RS232 cable that connects the two PIX firewalls together.

In PIX software release 6.2, failover can be configured without a failover cable. Ethernet can be used instead to overcome the distance limitations of the failover cable.

Each PIX monitors the failover, power, and interface status of the other PIX. At regular intervals, each PIX sends a *hello* across the failover cable and each active interface. If a hello is not received on an interface on either PIX for two consecutive intervals, the PIX puts that interface into *testing* mode. If the standby PIX does not receive a hello from the active PIX on the failover cable for two consecutive intervals, the standby PIX initiates a failover.

The PIX platform is very flexible, so I won't cover all possible failover scenarios here. The underlying principles are the same for them all: if one PIX determines that the other is unavailable, it assumes that PIX has failed.

For failover to work, each PIX must be able to reach the other on each interface configured for failover. Usually, a pair of switches connects the firewalls. One switch connects to each PIX, and the switches are connected to each other, usually with trunks. An example of such a design is shown in Figure 26-3. The link that connects the two PIX firewalls directly is the stateful failover link, which should not be switched if possible. (In the case of the FWSM, this is not possible, as all interfaces are virtual.)

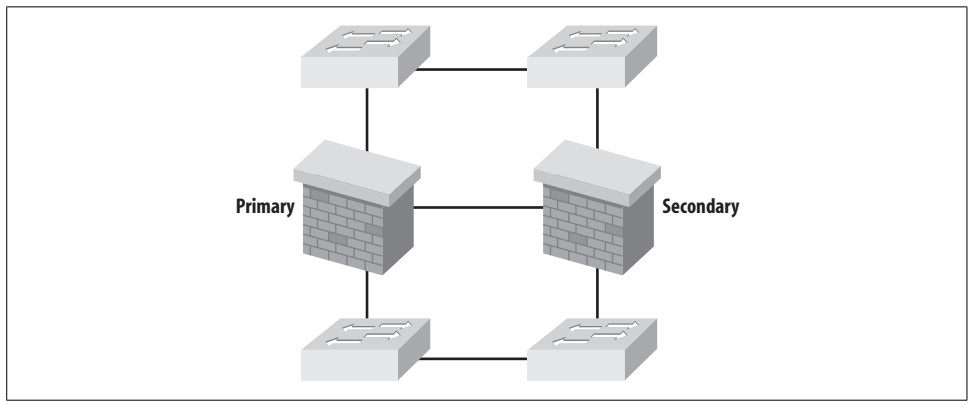


Figure 26-3. Common PIX failover design

Cisco recommends that the switch ports used to connect the PIX firewalls be set to `spanning-tree portfast`. With normal spanning-tree timers, hellos will not be received during the initial spanning-tree states. This might cause a PIX firewall to decide that the remote PIX is not responding, and to initiate a failover.

PIX OS v7.0 introduces the idea of *transparent mode* (the normal PIX behavior is called *routed mode*). In transparent mode, the PIX firewall acts as a bridge. When it's used in this manner, the spanning-tree requirements are different. Consult the Cisco documentation for more information on transparent mode.

When a failover occurs, the standby PIX assumes the IP and MAC addresses of all interfaces configured for failover on the active PIX. This is transparent to the network:

```
PIX(config)# sho int e1 | include Hardware
  Hardware is i82559 ethernet, address is 0050.54ff.1312
PIX(config)# failover active
PIX(config)# sho int e1 | include Hardware
  Hardware is i82559 ethernet, address is 0050.54ff.33c5
```

PIX failover works so well that PIX device failures can go unnoticed. When using this feature, it is imperative that the firewalls be managed in some way so that failures can be resolved. I have seen occasions where a primary PIX failed, and the secondary ran for months, until it, too, failed. When the secondary failed without another PIX to back it up, a complete outage occurred. If you're not monitoring your firewalls, your network is not as secure as you might think. Using SNMP with network management software such as CiscoWorks or OpenView will keep you apprised of PIX firewall failover events.

Configuring Failover

Both PIX firewalls in a failover pair must have the same operating system version, or they will not synchronize their configurations. They should be the same models with the same hardware as well.

The first step in configuring failover is to enable the feature with the `failover` command:

```
PIX(config)# failover
```

Each interface you wish to include (usually all of them) needs to have a failover IP address assigned to it. It's a good idea to assign pairs of IP addresses for firewalls when designing IP networks, even when you're only installing a single PIX. For example, if the inside IP address on your firewall would normally be 192.168.1.1, reserve 192.168.1.2 as well. This way, if you expand your firewall to include another PIX for failover, the IP address for failover will already be there.

To illustrate failover configuration, in this section, I'll build a pair of PIX firewalls to support the network shown in Figure 26-4.

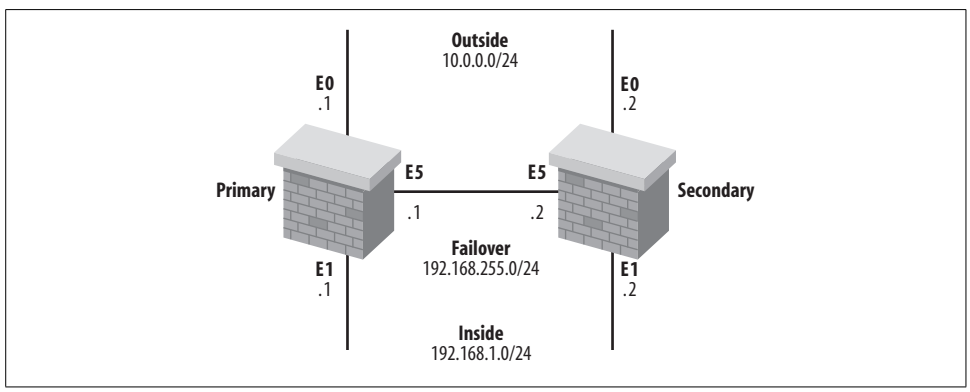


Figure 26-4. Sample PIX failover design

The interface configuration for the primary PIX is as follows:

```
nameif ethernet0 outside security0
nameif ethernet1 inside security100
nameif ethernet2 intf2 security4
nameif ethernet3 intf3 security6
nameif ethernet4 intf4 security8
nameif ethernet5 Failover security0
!
ip address outside 10.0.0.1 255.255.255.0
ip address inside 192.168.1.1 255.255.255.0
no ip address intf2
no ip address intf3
no ip address intf4
ip address Failover 192.168.255.1 255.255.255.0
```

The firewalls I'm using for these examples are PIX 515s with six Ethernet interfaces each. There are two onboard interfaces (E0 and E1), and a four-port interface card (E2–E5). I'm using E0 and E1 in their default *outside* and *inside* roles, and I've assigned interface E5 as the stateful failover interface.

Each interface must be configured with a failover IP address to be used on the secondary PIX. These commands are entered on the primary PIX. The entries in bold are the default configuration entries, and do not need to be entered manually:

```
failover ip address outside 10.0.0.2
failover ip address inside 192.168.1.2
no failover ip address intf2
no failover ip address intf3
no failover ip address intf4
failover ip address Failover 192.168.255.2
```

To configure the failover interface to be used for stateful failover, use the `failover link` command:

```
PIX(config)# failover link Failover
```

Interfaces that are not in use should always be placed in the `shutdown` state when employing PIX failover. Interfaces that are active, but not cabled, can trigger a failover, as the PIX will try unsuccessfully to contact the failover address if one was previously configured.

Monitoring Failover

The primary means of showing failover status is the `show failover` command:

```
PIX# sho failover
Failover On
Cable status: Normal
Reconnect timeout 0:00:00
```

```
Poll frequency 15 seconds
Last Failover at: 22:06:24 UTC Sat Dec 16 2006
        This host: Primary - Active
                Active time: 18645 (sec)
                Interface outside (10.0.0.1): Normal
                Interface inside (192.168.1.1): Normal
                Interface intf2 (0.0.0.0): Link Down (Shutdown)
                Interface intf3 (0.0.0.0): Link Down (Shutdown)
                Interface intf4 (0.0.0.0): Link Down (Shutdown)
                Interface Failover (192.168.255.1): Normal
        Other host: Secondary - Standby
                Active time: 165 (sec)
                Interface outside (10.0.0.2): Normal
                Interface inside (192.168.1.2): Normal
                Interface intf2 (0.0.0.0): Link Down (Shutdown)
                Interface intf3 (0.0.0.0): Link Down (Shutdown)
                Interface intf4 (0.0.0.0): Link Down (Shutdown)
                Interface Failover (192.168.255.2): Normal

Stateful Failover Logical Update Statistics
        Link : Failover
        Stateful Obj    xmit       xerr       rcv        rerr
        General         6651       0          2505       0
        sys cmd         2468       0          2475       0
        up time         4          0          0          0
        xlate           22         0          0          0
        tcp conn        4157       0          30         0
        udp conn        0          0          0          0
        ARP tbl         0          0          0          0
        RIP Tbl         0          0          0          0

        Logical Update Queue Information
                        Cur     Max     Total
        Recv Q:         0       1       2497
        Xmit Q:         0       1       6236
```

This command shows you the state of both of the firewalls in the pair, as well as statistics for the stateful failover link. If this link is incrementing errors, you may lose connections during a failover.

Remember that with stateful failover active, you may experience a failover without knowing it. This command will show you if your primary PIX has failed. If the primary PIX is the standby PIX, the original primary has failed at some point:

```
PIX(config)# sho failover | include host
        This host: Primary - Standby
        Other host: Secondary - Active
```

If the active PIX fails, the standby PIX takes over. If the failed PIX comes back online, it does not automatically resume its role as the active PIX. Cisco's documentation states that there is "no reason to switch active and standby roles" in this circumstance. While I would have preferred a *preempt* ability similar to that used in HSRP, unfortunately, Cisco didn't invite me to write the failover code.

To force a standby PIX to become active, issue the `no failover active` command on the active PIX, or the `failover active` command on the standby PIX:

```
PIX(config)# failover active
104001: (Primary) Switching to ACTIVE - set by the CI config cmd.
```

Assuming a successful failover, the primary PIX should now be the active PIX once again. When you force a failover, don't be impatient about checking the status. The CLI will pause for a few seconds (exactly how long depends on the model of PIX in use), and if you check the status too soon, you may see some odd results:

```
PIX(config)# sho failover | include host
        This host: Primary - Active
        Other host: Secondary – Active
```

If you see this after initiating a failover, take a deep breath, wait a second or two more, and try again:

```
PIX(config)# sho failover | include host
        This host: Primary - Active
        Other host: Secondary – Standby
```

Version 7.0 and later releases of the PIX OS can be configured in an active/active state. This is an advanced topic not covered in this book.

NAT

Network Address Translation (NAT) is technically what Cisco refers to as translating one IP address to another. The majority of installations, including most home networks, translate many IP addresses to a single address. This is actually called *Port Address Translation* (PAT). PAT has also been called *NAT Overload* in IOS.

To complicate matters, in the PIX OS, NAT is used in a number of ways that may not seem obvious. For example, you may have to use a `nat` statement to allow packets from one interface to another, even though they both have public IP addresses, and would normally require no translation.

NAT Commands

A few commands are used to configure the majority of NAT scenarios. Some, such as the `nat` command, have many options that I will not list here. The subject of NAT on a PIX firewall could fill a book itself. My goal is to keep it simple. If you need more information than what I've provided here, the Cisco command references are a good place to start. The commands you're most likely to need are:

`nat`
: The `nat` command is used when translating addresses from a more secure interface to a less secure interface. For example, if you needed to translate an address on the inside of your PIX to an address on the outside, you would use the `nat` command. Private IP addresses on the inside of a PIX are translated to one or more public IP addresses using the `nat` command. (Technically, the addresses do not need to be private and public addresses as described by RFC1918. The PIX documentation uses the terms "global" and "local" to describe addresses seen outside the PIX as opposed to those seen inside.)

`static`
: The `static` command is used when translating addresses from a less secure interface to a more secure interface. For example, if you had a server inside your PIX that needed to be accessed from outside, you would assign a public IP address to the private IP address of the server using the `static` command.

`global`
: The `global` command is used for PAT configurations where many addresses are translated to one address. It is also used to provide a pool of NAT addresses. This command is used in conjunction with the `nat` command.

NAT Examples

There are many possible NAT scenarios, some of which can become quite complicated. I will cover some of the more common scenarios here.

For these examples, I will be using 10.0.0.0 to represent a publicly routable IP network, and 192.168.1.0 as a private, unroutable network.

Simple PAT using the outside interface

One of the most common scenarios for a firewall is providing an office with protection from the Internet. Assuming that all nodes inside the firewall require access to the Internet, and no connections will be initiated inbound, a simple PAT configuration can be used.

Here, I've configured the outside interface to be used as the IP address for the global PAT. In other words, all packets that originate from the inside will be translated to the same IP address as the one used on the outside interface of the PIX:

```
global (outside) 1 interface
nat (inside) 1 0.0.0.0 0.0.0.0 0 0
```

All internal IP addresses will be translated because the `nat` statement references 0.0.0.0, which means all addresses.

Simple PAT using a dedicated IP address

Older releases of the PIX OS (before 6.0) did not allow PAT to be configured using an interface. This was a problem for installations with limited public IP addresses.

To accomplish PAT without using the interface's IP address, use the same configuration as the previous one, but specify the IP address used for the global PAT in the `global` command instead of the keyword `interface`:

```
global (outside) 1 10.0.0.5
nat (inside) 1 0.0.0.0 0.0.0.0 0 0
```

Simple PAT with public servers on the inside

Small installations may have a server inside (not on a DMZ) that must be accessible from the public Internet. While this is usually not a good idea, it is nevertheless a distinct possibility that you'll need to configure such a solution. Smaller companies—and even home networks—often require such configurations because a DMZ is either impractical or impossible.

Here, I've designed a global PAT using the outside interface, with all addresses from the inside being translated. Additionally, I've created two static entries. The first forwards packets sent to the public IP address 10.0.0.10 to the private IP address 192.168.1.10. The second translates 10.0.0.11 to the private IP address 192.168.1.11:

```
global (outside) 1 interface
nat (inside) 1 0.0.0.0 0.0.0.0 0 0
static (inside,outside) 10.0.0.10 192.168.1.10 netmask 255.255.255.255 0 0
static (inside,outside) 10.0.0.11 192.168.1.11 netmask 255.255.255.255 0 0
```

`static` statements override the more generic `nat` statement, so these commands can be used together in this way without issue. Be wary when configuring `static` statements, however. The order of interfaces and networks can be confusing. If you look carefully at the preceding example, you'll see that the references are essentially:

```
(inside-int,outside-int) outside-net inside-net
```

Remember that these `static` statements are allowing connections from outside to come into these two IP addresses, which reside inside the secure network. This may sound dangerous, but because the outside interface has a security level of 0, and the inside interface has a security level of 100, traffic cannot flow from outside to inside unless it's permitted with an access list.

In other words, an access list must now be created to allow the desired traffic to pass.

Port redirection

Port redirection is different from Port Address Translation. While PAT translates a pool of addresses to a single address by translating the ports within the packets being sent, port redirection does something else entirely.

Port redirection allows you to configure a static NAT where, though there is one IP address on the public side, there can be many IP addresses on the private side, each of which responds to a different port. PAT does not permit inbound connections. Port translation does.

Imagine you have only eight IP addresses on your public network, which are all in use:

.0 – Network address
.1 – ISP Router VIP (HSRP)
.2 – ISP Router 1
.3 – ISP Router 2
.4 – Primary PIX
.5 – Secondary PIX
.6 – Web server public IP
.7 – Broadcast address

While it might not seem realistic to have so much resilient equipment on such a small network, you might be surprised what happens in the field. Many small business networks are limited to eight addresses. In reality, many don't need any more than that.

In this example, we only need to have one static NAT configured—the web server. Here is the configuration relating to NAT:

```
global (outside) 1 interface
nat (inside) 1 0.0.0.0 0.0.0.0 0 0
static (inside,outside) 10.0.0.6 192.168.1.6 netmask 255.255.255.255 0 0
```

This configuration works fine, but what if the need arises for another web server to be available on the Internet? Say a secure server has been built using HTTPS, which listens on TCP port 443. The problem is a lack of public IP addresses. Assuming that the original web server only listens on TCP port 80, we can solve the problem using port redirection.

Using capabilities introduced in the 6.x release of the PIX OS, we can specify that incoming traffic destined for the 10.0.0.6 IP address on TCP port 80 be translated to one IP address internally, while packets destined for the same IP address on TCP port 443 be sent to another IP address:

```
static (inside,outside) tcp 10.0.0.6 80 192.168.1.6 80 netmask 255.255.255.255
static (inside,outside) tcp 10.0.0.6 443 192.168.1.7 443 netmask 255.255.255.255
```

Normally, the `static` command includes only the outside and inside IP addresses, and all packets are sent between them. Including the port numbers makes the `static` statement more specific.

The result is that packets destined for 10.0.0.6 will be translated to different IP addresses internally, depending on their destination ports: a packet sent to 10.0.0.6:80 will be translated to 192.168.1.6:80, while a packet destined for 10.0.0.6:443 will be translated to 192.168.1.7:443.

DMZ

Here is a very common scenario. A company has put a PIX in place for Internet security. Certain servers need to be accessed from the Internet. These servers will be in a DMZ. The outside interface connects to the Internet, the inside interface connects to the company LANs, and the DMZ contains the Internet-accessible servers. This network is shown in Figure 26-5.

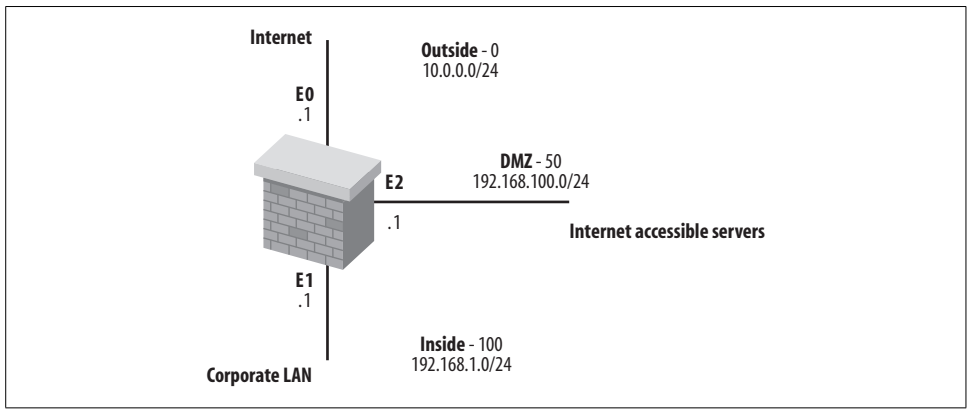


Figure 26-5. Firewall with DMZ

From a NAT point of view, we must remember that the security levels are important. The outside interface has a security level of 0, the inside interface has a level of 100, and the DMZ has a level of 50.

In this case, we want the servers in the DMZ to be accessible from outside. We also want hosts on the inside network to be able to access the DMZ servers, although the DMZ servers should not be able to access the inside network.

First, we need the `nat` and `global` statements for the inside network using the Internet:

```
global (outside) 1 interface
nat (inside) 1 192.168.1.0 255.255.255.0 0 0
```

Specifying a specific network, rather than using 0.0.0.0 as the address in the `nat` statement, ensures that only that network will be able to access the Internet. Should other networks that need Internet access be added internally, they will need to be added to the PIX with additional `nat (inside) 1` statements.

Now, we need to add the `static` statements so the servers on the DMZ can be accessed from the Internet:

```
static (DMZ,outside) 10.0.0.11 192.168.100.11 netmask 255.255.255.255
static (DMZ,outside) 10.0.0.12 192.168.100.12 netmask 255.255.255.255
static (DMZ,outside) 10.0.0.13 192.168.100.13 netmask 255.255.255.255
```

By default, the DMZ will not be able to access the inside network because the DMZ has a lower security level than the inside network. In this case, we must use a `static` statement to allow the connections. Where it gets a little strange is that we don't need to translate the source network; we just need to allow the connection. As odd as it sounds, to accomplish this, we must statically NAT the inside network to itself:

```
static (inside,DMZ) 192.168.1.0 192.168.1.0 netmask 255.255.255.0
```

A PIX firewall must translate a higher-security interface for the network to be seen by a lower-security interface. This can be confusing because doing this creates a "translation," even though nothing is being translated. The PIX must have a translation in place for the hosts on the inside network to be able to connect to the hosts on the DMZ. The IP addresses do not need to be changed, but the path needs to be built.

Once NAT is in place, all that's left to do is configure access lists to allow the required traffic from the DMZ to the inside network.

Miscellaneous

The following items are things that trip me up again and again in the field.

Remote Access

To be able to telnet or SSH to your PIX firewall, you must specify the networks from which you will do so. This is done with the `telnet` and `ssh` commands:

```
PIX(config)# telnet 192.168.1.0 255.255.255.0 inside
PIX(config)# ssh 192.168.1.0 255.255.255.0 inside
```

Saving Configuration Changes

If you are in the habit of shortening the `write memory` command in IOS to `wri`, you will be frustrated to find that the abbreviation does not work on a PIX:

```
PIX# wri
Not enough arguments.
Usage:  write erase|floppy|mem|terminal|standby
```

```
        write net [<tftp_ip>]:<filename>
PIX# wri mem
Building configuration...
Cryptochecksum: f4f6sf4b 045a1327 1b4eaac1 670e1e41
```

The `copy running startup` command also does not work.

When you're configuring the active PIX in a failover pair, each command should be sent to the standby PIX automatically after it's submitted, and when you save your changes on the active PIX, the `write memory` command should also write the configuration to the standby PIX. To force a save to the standby PIX, use the `write standby` command:

```
PIX# write standby
Building configuration...
[OK]
PIX# Sync Started
.
Sync Completed
```

Note that the `Sync Started` entry above is not a command, but rather the output of normal PIX logging when logging is enabled.

Logging

If you have a firewall in place, you should save and periodically review the logs it generates. When configured for logging, PIX firewalls create a great deal of information. Even on small networks, the logs can be substantial.

While logging to the PIX buffer may seem like a good idea, the logs can scroll by so fast that the buffer becomes all but unusable. If you log too much detail to the console, you can impact the firewall's performance. If you log to the monitor (your telnet session), the logs will update so frequently that you'll end up turning them off so you can work.

I like to send all my PIX firewall logs to a syslog server. I generally use some flavor of Unix to do this, though of course, you are free to use whatever you like. Two steps are required to enable logging: you must enable logging with the `logging on` command, and you must specify one or more logging destinations. When configuring logging destinations, you must also specify the level of logging for each destination. The levels are:

```
0 - System Unusable
1 - Take Immediate Action
2 - Critical Condition
3 - Error Message
4 - Warning Message
5 - Normal but significant condition
6 - Informational
7 - Debug Message
```

The useful logging information regarding traffic traversing the firewall is found in level 6. Here's a sample of level-6 logging:

```
302015: Built outbound UDP connection 3898824 for outside:11.1.1.1/123
(11.1.1.1/123) to inside:192.168.1.5/123 (10.1.1.5/334)
302013: Built inbound TCP connection 3898825 for outside:12.2.2.2/2737
(12.2.2.2/2737) to inside:192.168.1.21/80 (10.1.1.21/80)
302013: Built inbound TCP connection 3898826 for outside:13.3.3.3/49050
(13.3.3.3/49050) to inside:192.168.1.21/80 (10.1.1.21/80)
304001: 15.5.5.5 Accessed URL 10.1.1.21:/lab/index.html/
```

On a live network, this information will probably scroll by so fast you won't be able to read it. Unfortunately, debug messages are a higher log level, so if you need to run debugs on the PIX, the output will be buried within these other log entries.

Here, I've enabled logging, and set the console to receive level-5 logs, while all other logging destinations will receive level-7 logs:

```
logging on
logging console notifications
logging monitor debugging
logging buffered debugging
```

These commands apply only to the PIX itself. To send logs to a syslog host, you must configure a trap destination level, the syslog facility, and the host to receive the logs:

```
logging trap debugging
logging facility 22
logging host inside 192.168.1.200
```

On the syslog server, you then need to configure syslog to receive these alerts. Detailed syslog configuration is outside the scope of this book, so I'll just include the */etc/syslog.conf* entries from my Solaris server:

```
# Configuration for PIX logging
local6.debug                              /var/log/PIX
```

This will capture the PIX syslog entries, and place them into the */var/log/PIX* file. Notice that the PIX is configured for `facility 22`, but the server is configured for `local6.debug`. The facilities are mapped as 16(LOCAL0)–23(LOCAL7). The default is 20(LOCAL4).

Once you've begun collecting syslog entries into a file, you can use the server to view and parse the log file without affecting your CLI window. On Unix systems, you can use commands like `tail -f /var/log/PIX` to view the log in real time. You can also add filters. For example, if you only wanted to see log entries containing the URL */lab/index.html/*, you could use the command `tail -f /var/log/PIX | grep '/lab/index.html/'`.

For more information on logging, type `help logging` in PIX configuration mode. On a Unix system, you can learn more about syslog with the `man syslog` and `man syslogd` commands.

Troubleshooting

If you change an access list, change NAT, or do anything else that can alter what packets are allowed to flow through the firewall, you may not see the results until you execute the `clear xlate` command.

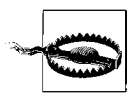
Xlate is short for translation. A translation is created for every conversation that is active on the PIX. To see what xlates are active on your PIX, use the `show xlate` command:

```
PIX# sho xlate
10 in use, 114 most used
PAT Global 10.0.0.5(9364) Local 192.168.1.110(1141)
PAT Global 10.0.0.5(1211) Local 192.168.1.100(3090)
PAT Global 10.0.0.5(1210) Local 192.168.1.100(3089)
PAT Global 10.0.0.5(1209) Local 192.168.1.100(3088)
PAT Global 10.0.0.5(1215) Local 192.168.1.100(3094)
PAT Global 10.0.0.5(1213) Local 192.168.1.100(3092)
PAT Global 10.0.0.5(1212) Local 192.168.1.100(3091)
PAT Global 10.0.0.5(9324) Local 192.168.1.110(1127)
PAT Global 10.0.0.5(1047) Local 192.168.1.100(2958)
Global 10.0.0.11 Local 192.168.1.11
```

The `PAT Global` entries are live connections from my PC to the Web. I had a download running through a web browser, plus a few web pages open. The last entry is a static translation resulting from the static configuration entered earlier.

To clear xlates, use the `clear xlate` command:

```
PIX# clear xlate
```



When you clear xlates, every session on the firewall will be broken, and will need to be rebuilt. If your PIX is protecting an e-commerce web site, transactions will be broken, and customers may become unhappy. Clearing xlates should not be done unless there is a valid reason.

While the `clear xlate` command runs with no fanfare on the PIX, every connection has been cleared. Now the output of the `show xlate` command shows only the single static entry:

```
PIX# sho xlate
1 in use, 114 most used
Global 10.0.0.11 Local 192.168.1.11
```

My IM client reset, and the download I had running aborted as a result of the xlates being cleared.

Another useful command for troubleshooting is `show conn`, which shows all of the active connections on the PIX:

```
PIX# sho conn
8 in use, 199 most used
```

```
TCP out 10.233.161.147:80 in LAB-PC2:1151 idle 0:00:18 Bytes 6090 flags UIO
TCP out 10.46.109.49:1863 in LAB-SVR1:1736 idle 0:03:28 Bytes 7794 flags UIO
TCP out 10.188.8.176:5190 in LAB-PC2:4451 idle 0:00:52 Bytes 32827 flags UIO
TCP out 10.120.37.15:80 in LAB-PC:1789 idle 0:00:03 Bytes 19222477 flags UIO
TCP out 10.120.37.15:80 in LAB-PC:1802 idle 0:00:02 Bytes 20277173 flags UIO
TCP out 10.172.118.250:19093 in LAB-SVR2:80 idle 0:00:09 Bytes 11494 flags UIOB
TCP out 10.172.118.250:19075 in LAB-SVR2:80 idle 0:00:09 Bytes 219866 flags UIOB
UDP out 10.67.79.202:123 in RTR1:123 idle 0:00:32 flags -
```

This command shows the protocol, direction, source, and destination of each connection, as well as how long each connection has been idle, and how many bytes have been sent. The flags are very useful, if you can remember them. The entire list of flags can be viewed along with a different format of the same data by appending the `detail` keyword to the command. This example was taken a few minutes later than the previous one:

```
PIX# sho conn detail
17 in use, 199 most used
Flags: A - awaiting inside ACK to SYN, a - awaiting outside ACK to SYN,
       B - initial SYN from outside, C - CTIQBE media, D - DNS, d - dump,
       E - outside back connection, F - outside FIN, f - inside FIN,
       G - group, g - MGCP, H - H.323, h - H.225.0, I - inbound data, i - incomplete,
       k - Skinny media, M - SMTP data, m - SIP media, O - outbound data,
       P - inside back connection, q - SQL*Net data, R - outside acknowledged FIN,
       R - UDP RPC, r - inside acknowledged FIN, S - awaiting inside SYN,
       s - awaiting outside SYN, T - SIP, t - SIP transient, U - up
TCP outside:10.46.109.49/1863 inside:LAB-PC2/1736 flags UIO
TCP outside:10.188.8.176/5190 inside:LAB-PC/4451 flags UIO
TCP outside:10.241.244.1/48849 inside:LAB-SVR1/80 flags UIOB
UDP outside:10.30.70.56/161 inside:RTR1/1031 flags -
```

PART VI

Server Load Balancing

This section is designed to give you a quick view into the world of server load balancing. It reviews server load-balancing technology, and shows examples regarding real-world implementations.

This section is composed of the following chapters:

Chapter 27, *Server Load-Balancing Technology*
Chapter 28, *Content Switch Modules in Action*

CHAPTER 27

Server Load-Balancing Technology

Server load balancing (SLB) is what enables multiple servers to respond as if they were a single device. This idea becomes very exciting when you think in terms of web sites, or anywhere a large amount of data is being served. Large web sites can sometimes serve gigabits of data per second, while most servers as of this writing can only provide a gigabit at a time at the basic level. EtherChannels (*trunks* in Sun Solaris parlance) can increase that rate, but the real issue becomes one of power and scalability. Having many smaller, less expensive web servers is often more viable financially than having one large, extremely powerful server. Additionally, the idea of high availability comes into play, where having multiple smaller servers makes a lot of sense.

Figure 27-1 shows a simple load-balanced network.

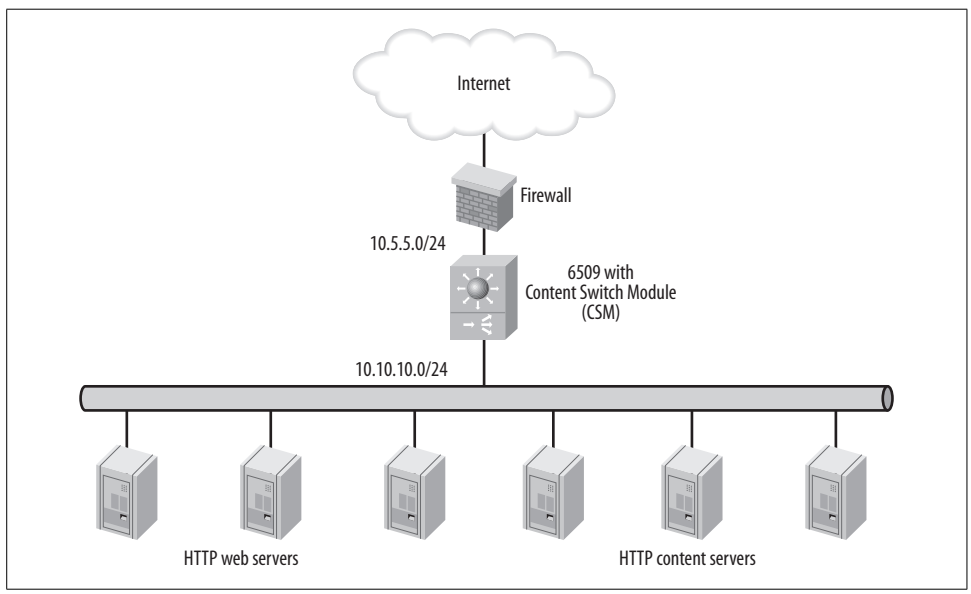


Figure 27-1. Simple load-balanced network

Within Figure 27-1, there is one feed to the Internet, with six servers behind a firewall. The device connecting all the servers to the firewall is a Cisco Content Switch. For our examples, we'll use a Cisco Content Switching Module (CSM) blade in a Cisco 6509 Catalyst switch. One of the real benefits of using an integrated service module like this is that it can be completely removed from the network using only command-line instructions.

Types of Load Balancing

Server load balancing is only one of many ways to accomplish the goal of multiple servers responding as one. Let's look at some different possibilities, and the pros and cons of each:

DNS load balancing

As the name implies, balancing is done with DNS. A single name resolves to multiple other names or IP addresses. These real names or IP addresses are then hit in a round-robin manner.

Pro:

- Very simple to configure and understand.

Cons:

- No intelligence other than round-robin.
- No way to guarantee connection to the same server twice if needed (sticky connections).
- DNS cannot tell if a server has become unavailable.
- Load may not be evenly distributed, as DNS cannot tell how much load is present on the servers.
- Each server requires a public IP address, in the case of publicly available web servers.

Bridged load balancing

Load balancing at layer two, or *bridged* load balancing, is a very simple model. A virtual IP address is created in the same IP network as the real servers. Packets destined for the virtual IP address are forwarded to the real servers.

Pros:

- Can be inserted into an existing network, with no additional IP networks required.
- Possibly easier to understand for simple networks.
- Usually less expensive than a routed model.

Cons:

- Layer-2 issues including loops and spanning-tree problems can arise if the solution is not designed carefully.
- Can be harder to understand for people used to layer-3 environments.
- Usually limited to a single local network.

Routed load balancing

Load balancing at layer three, or *routed* load balancing, is slightly more complex than bridged load balancing. In this model, the virtual IP address exists on one network, while the real servers exist on one or more others.

Pros:

- Expandability. Routed models allow for the real servers to be geographically diverse. The possibilities here are almost limitless.
- Easier to understand for people used to layer-3 environments.
- No spanning-tree issues.

Cons:

- Layer-3 load balancing can be costly. The CSM-S module for the 6500 switch is one of the most expensive modules Cisco produces.
- Requires network design and additional IP address space to implement. Because the real servers must be on a different broadcast domain from the virtual server, a routed load balancer cannot be dropped into an existing flat network without redesigning the network.

Load balancing in the Cisco world can be accomplished in one of the following ways:

IOS IP server load balancing

Load balancing in IOS is a very easy way to get started. The commands are very easy to understand, and they are very similar to those used with CSMs. Should you ever decide to upgrade, the transition will be smoother than with other methods such as Local Directors. The downside of IOS load balancing is that it can be CPU-intensive. If you're balancing more than a few servers, or if you find yourself building multiple virtual servers, it's time to upgrade to dedicated hardware. IOS SLB requires an SLB-capable version of IOS.

Local Directors

I hesitate to include Local Directors because they are end-of-sale as of this writing (they are supported until 2008). It's a shame that Cisco has discontinued them, because they were an excellent product. The Cisco Local Director is very easy to configure and manage. Its only shortcoming is that it must be deployed as a bridge. This not only makes it far less useful in complex environments (no doubt the cause of its demise), but also makes it a touch harder to understand for those engineers who live in a layer-3-only world.

Content Switches
: Either as standalone units or in the excellent CSMs for the 6500-series switch, the Content Switch is the top dog in Cisco's catalog when it comes to load balancing. Content Switches can be configured in bridged or routed mode (routed is preferred); they can be used for global load balancing; and, in the case of the CSMs, their configurations are included in the IOS configurations of the switch itself (4 Gbps of throughput is supported).

How Server Load Balancing Works

Each of the previously listed Cisco load balancers is configured in a different way. Server load balancing (IP SLB) and CSMs are both configured similarly. I will concentrate on these technologies. For information regarding the Cisco CSS content switches or Local Directors, consult the Cisco documentation.

With IP SLB and CSM modules, SLB is implemented by creating a *virtual server* that is mapped to a logical *server farm* that contains physical *real servers*:

Virtual server
: The virtual server is the IP address that will be used for accessing the services running on the real servers. The point of the virtual server is to make a single IP address appear to be a single server. The load-balancing system will then translate this single address, and forward the packets on to one of the server farms that have been *bound* to this virtual server.

Server farm
: Server farms are logical groups of real servers.

Real server
: A real server is a means of referencing the IP address of a physical server on the network. Real servers are grouped into one or more server farms. The server farms are then bound to one or more virtual servers.

This may seem needlessly complex, but consider the idea that you can have a real server in multiple server farms, and server farms assigned to multiple virtual servers. Let's say you have 100 real servers that are bound to a single virtual server. If you create 10 server farms, each containing 10 real servers, you'll be able to take 10 real servers offline at a time by shutting down a single server farm. This can be very useful in large environments where the entire site cannot be brought down for maintenance.

Balancing Algorithms

Every time the load balancer receives a request, it must choose which real server will be used to serve the request. Options for making this decision include algorithms such as round-robin and least-used. The *round-robin* algorithm connects to each real server

in turn, regardless of the number of connections active on each server. The *least-used* algorithm keeps track of the number of connections active on each real server, and sends each new request to the server with the least number of active connections.

In some cases, a user who connects more than once may need to connect to the same real server (this is called a *sticky* connection). A common example of this situation is a web site where you need to log in to view your account. If you were to log into one real server and then connect a minute later without logging out, the default behavior would most likely forward you to a different real server, which would not have the information from your original session. Server load balancers can be configured so that a single user, once connected to a real server, will reconnect to the same real server. After a timeout period has elapsed, the user will again be sent to any of the available servers, according to the balancing algorithm in use.

Configuring Server Load Balancing

In this section, I will explain how to configure real servers, server farms, and virtual servers in IOS SLB and CSMs.

IOS SLB

Configuration relating to SLB is done with the `ip slb` command, or in SLB configuration mode. Real servers and virtual servers must be on different VLANs when using SLB.

Real servers

When using SLB, real servers are not configured independently, but rather, within server farms.

Server farms

Server farms are created in SLB with the `ip slb serverfarm` *`farm-name`* command. Here, I've created a server farm named GAD-FARM. The `nat server` command is a default and will be inserted by IOS:

```
ip slb serverfarm GAD-FARM
 nat server
```

Don't bother trying to come up with names for your servers in lowercase or a combination of upper- and lowercase. No matter how you enter names when using SLB or CSMs, the parser will convert them to uppercase.

Once in `ip slb` configuration mode, add the real servers to be included in the server farm with the `real` *`ip-address`* command. Here, I've configured two real servers. The first one is in service, and the second is out of service:

```
real 10.10.10.100
 inservice
!
real 10.10.10.101
 no inservice
```

The final configuration for the server farm is as follows:

```
ip slb serverfarm GAD-FARM
 nat server
 real 10.10.10.100
  inservice
 !
 real 10.10.10.101
  no inservice
```

Virtual servers

Virtual servers are configured in SLB with the `ip slb vserver` *`server-name`* command. You must configure the IP address for the virtual server, the port or ports on which it will listen, and the server farms to be used:

```
ip slb vserver VIRT-GAD
 virtual 10.1.1.1 tcp 0
 serverfarm GAD-FARM
 inservice
```

This configuration creates a virtual server named VIRT-GAD that listens on the IP address 10.1.1.1 on all TCP ports (TCP port 0 indicates all ports). Any request that comes into this IP address using TCP will be load balanced to the real servers configured within the server farm GAD-FARM. To create a virtual server that listens to a specific port, enter the port number on the `virtual` command line. For example, if I needed my VIRT-GAD virtual server to respond only to SSH, I would include port 22 instead of port 0:

```
virtual 10.1.1.1 tcp 22
```

Port translation using SLB

You can configure individual ports for virtual servers and real servers. Configuring a real server on one port—and assigning the server farm to a virtual server that listens on a different port—causes the router to perform port translation when load balancing. Here, I've created real servers listening on port 8080 in the server farm WEB-FARM,

and a virtual server named VIRTUAL-WEB that listens on port 80. When requests come into the virtual server on port 80, they will be forwarded to the real servers on port 8080:

```
ip slb serverfarm WEB-FARM
 nat server
 real 10.10.10.101 8080
  inservice
 !
 real 10.10.10.102 8080
  inservice
 !
 real 10.10.10.103 8080
  inservice
!
ip slb vserver VIRTUAL-WEB
 virtual 10.1.1.1 tcp 80
 serverfarm WEB-FARM
 inservice
```

Content Switch Modules

All configuration for the CSM is done in IOS, using the `module ContentSwitchingModule` *`module#`* command. This command can be abbreviated `module CSM` *`module#`*. Here, I'm configuring a CSM residing in slot 8 of a 6509:

```
module CSM 8
```

CSMs can be installed in pairs, with one module in each of two physical switches. A dedicated VLAN needs to be created for the stateful failover traffic. This VLAN should have its own physical links, if possible.

Failover is called *fault tolerance* in the CSM, and is configured with the `ft` command. You must specify a group (you will probably need only one) and the VLAN being used for stateful failover traffic. Fault tolerance is similar in behavior to HSRP in that each side must be configured with a priority. You can configure preemption to have the CSM failback to the primary when it comes back online after a failure:

```
ft group 1 vlan 88
 priority 15
 preempt
```

Because the CSM is a physical device and not just a software feature like SLB, you must configure the VLANs that the CSM will be using. The CSM works with client VLANs and server VLANs: the *client VLAN* is where the virtual servers will reside, and the *server VLAN* is where the real servers will reside.

Content Switch Modules can operate in bridged mode, where the virtual servers and real servers reside in the same VLAN. This configuration is not recommended and not covered here. See the Cisco documentation for more details.

For my examples, I will use VLAN 3 for the client side, and VLAN 4 for the server side. The IP network for VLAN 3 will be 10.5.5.0/24. The IP network for VLAN 4 will be 10.10.10.0/24.

When configuring the client VLAN, you must configure the IP address of the CSM on the VLAN, and an alternate IP address for the failover CSM. This IP address is configured only on the primary CSM. The secondary will learn its IP address through the fault tolerance configuration. A gateway must also be configured for the client VLAN. The gateway is often an SVI on the 6500 switch, but it can be any IP address you like on the VLAN:

```
vlan 3 client
 ip address 10.5.5.3 255.255.255.0 alt 10.5.5.4 255.255.255.0
 gateway 10.5.5.1
```

The server VLAN must be configured with primary and secondary IP addresses (assuming a failover pair of CSMs), but it does not require a gateway. An alias IP address must also be configured when using failover. This alias IP address is the IP address that the real servers will use as their default gateway. If you do not include an alias, and use only the IP address of the CSM VLAN, the CSMs will still failover, but the servers will not be able to route because the primary address will cease to exist in a failure:

```
vlan 4 server
 ip address 10.10.10.2 255.255.0.0 alt 10.10.10.3 255.255.0.0
 alias 10.10.10.1
```

The Cisco documentation regarding the `alias` command is not clear. I recommend testing failover thoroughly in a lab environment before you implement a CSM. The need for the `alias` command is not very clear in the documentation, and leaving it out has burned me in the field.

Real servers

Real servers are configured with the CSM command `real`. The simplest form of a real server configuration lists the IP address and its status:

```
real CONTENT-01
  address 10.10.10.17
  inservice
```

```
real CONTENT-02
  address 10.10.10.18
  inservice
real CONTENT-03
  address 10.10.10.18
  inservice
```

Server farms

Server farms are configured with the CSM command `serverfarm`. The `nat server` and `no nat client` commands are defaults, and are usually what you'll need. Here, I've created a server farm named CONTENT_FARM. The server farm is composed of the three real servers configured previously:

```
serverfarm CONTENT_FARM
  nat server
  no nat client
  real name CONTENT-01
   inservice
  real name CONTENT-02
   inservice
  real name CONTENT-03
   inservice
```

Notice that you can put a real server into or out of service from within the server farm. This is useful because a real server can be in more than one server farm. If you were to make the real server inactive in the real server configuration, it would become inactive in every server farm in which it was configured. This can be useful, too, but it may not be what you want. It's handy to be able to make a real server inactive in one server farm while keeping it active in others.

Virtual servers

Virtual servers, or *vservers*, are configured using the `vserver` *`server-name`* CSM command. The IP address for the vserver is specified with the `virtual` command, which is followed by the port to be balanced (or the keyword `any` for all ports). The server farm to include is referenced with the `serverfarm` command.

For a sticky vserver, include the `sticky` command along with the timeout value in minutes. The `replicate` and `persistent` commands are defaults inserted by IOS:

```
vserver V-CONTENT
  virtual 10.5.5.6 any
  serverfarm CONTENT_FARM
  sticky 10
  replicate csrp sticky
  replicate csrp connection
  persistent rebalance
  inservice
```

Port redirection

Port redirection is accomplished by specifying a port number for the real servers in the server farm, and a different port number in the `virtual` statement of the vserver. In this example, I've created a virtual server that balances requests on port 443 (HTTPS), and forwards those requests to the real servers on port 1443:

```
serverfarm HTTP_FARM
  nat server
  no nat client
  real name CONTENT-01 1443
   inservice
  real name CONTENT-02 1443
   inservice
  real name CONTENT-03 1443
   inservice

vserver V-HTTP
  virtual 10.5.5.7 tcp https
  serverfarm HTTP_FARM
  replicate csrp sticky
  replicate csrp connection
  persistent rebalance
  inservice
```

CHAPTER 28

Content Switch Modules in Action

Figure 28-1 shows a simple load-balanced network that I've built to illustrate some common configuration tasks. The load balancer in use is a pair of Cisco Content Switch Modules in a pair of Cisco 6509 switches. The virtual servers reside on the 10.5.5.0/24 network, while the real servers reside on the 10.10.10.0/24 network.

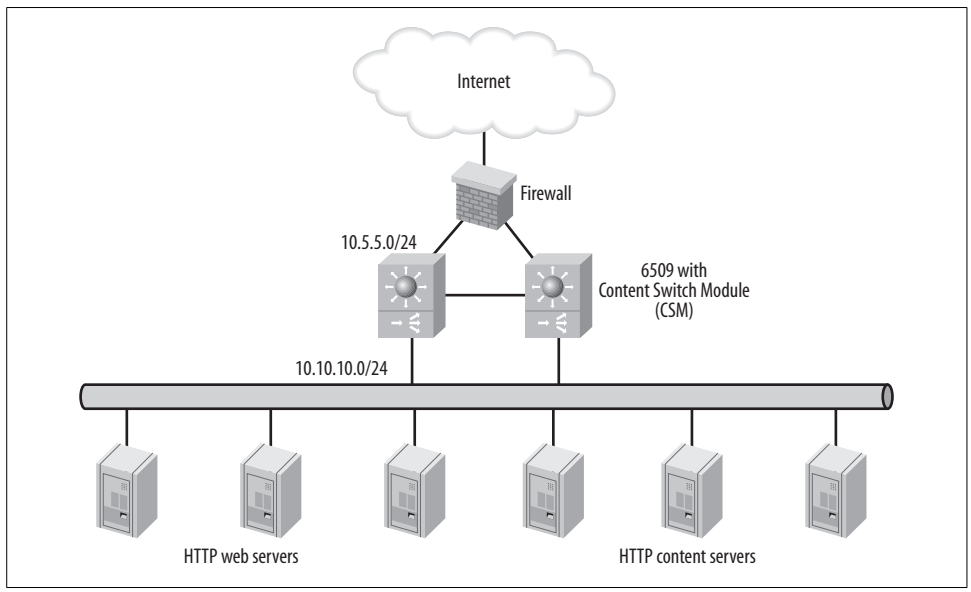


Figure 28-1. Simple load-balanced network

Here is the configuration for the CSM in the first switch. The second switch will inherit the configuration from the first through the `ft group` command, and the `alt` configurations in the VLAN IP addresses. This configuration was created in the previous chapter, so it should all be familiar:

```
module ContentSwitchingModule 8
 ft group 1 vlan 88
```

```
  priority 15
  preempt
!
 vlan 3 client
  ip address 10.5.5.3 255.255.255.0 alt 10.5.5.4 255.255.255.0
  gateway 10.5.5.1
!
 vlan 4 server
  ip address 10.10.10.1 255.255.0.0 alt 10.10.10.2 255.255.0.0
  alias 10.10.10.1
!
 real CONTENT-00
  address 10.10.10.16
  inservice
 real CONTENT-01
  address 10.10.10.17
  inservice
 real CONTENT-02
  address 10.10.10.18
  inservice
 real CONTENT-03
  address 10.10.10.19
  inservice
 real CONTENT-04
  address 10.10.10.20
  inservice
!
 real HTTP-00
  address 10.10.10.32
  no inservice
 real HTTP-01
  address 10.10.10.33
  no inservice
 real HTTP-02
  address 10.10.10.34
  inservice
 real HTTP-03
  address 10.10.10.35
  inservice
 real HTTP-04
  address 10.10.10.36
  inservice
!
 serverfarm CONTENT_FARM
  nat server
  no nat client
  real name CONTENT-00
   inservice
  real name CONTENT-01
   inservice
  real name CONTENT-02
   inservice
  real name CONTENT-03
   inservice
```

```
 real name CONTENT-04
  inservice
!
serverfarm HTTP_FARM
 nat server
 no nat client
 real name HTTP-00
  inservice
 real name HTTP-01
  inservice
 real name HTTP-02
  inservice
 real name HTTP-03
  inservice
 real name HTTP-04
  inservice
!
vserver V-CONTENT
 virtual 10.5.5.6 any
 serverfarm CONTENT_FARM
 replicate csrp sticky
 replicate csrp connection
 persistent rebalance
 inservice
!
vserver V-HTTP
 virtual 10.5.5.7 any
 serverfarm HTTP_FARM
 replicate csrp sticky
 replicate csrp connection
 persistent rebalance
 inservice
```

Common Tasks

All configuration of the CSMs is done on the Catalyst switch using IOS. Every status command for the CSM module is prefixed with `show module csm` *`module#`*. The *`module#`* following the keyword `csm` references the slot in which the module resides. For example, to see which CSM is active in a fault-tolerance pair, use the `show module csm` *`module#`* `ft` command:

```
Switch# sho mod csm 8 ft
FT group 1, vlan 88
 This box is active
 priority 20, heartbeat 1, failover 3, preemption is on
```

Configuration of CSM modules must be done in CSM configuration mode. To enter CSM configuration mode, use the `module csm` *`module#`* command:

```
Switch(config)# mod csm 8
Switch(config-module-csm)#
```

In a fault-tolerant design with two CSMs connected by a stateful failover link, changes to the primary CSM are not automatically replicated to the secondary CSM. You must replicate the changes manually with the command `hw-module csm` *`module#`* `standby config-sync` (this command can be executed only on the primary CSM):

```
Switch# hw-module csm 8 standby config-sync
Switch#
Sep 19 13:02:37 EDT: %CSM_SLB-6-REDUNDANCY_INFO: Module 8 FT info: Active: Bulk
sync started
Sep 19 13:02:39 EDT: %CSM_SLB-4-REDUNDANCY_WARN: Module 8 FT warning: FT
configuration might be out of sync.
Sep 19 13:02:48 EDT: %CSM_SLB-4-REDUNDANCY_WARN: Module 8 FT warning: FT
configuration back in sync
Sep 19 13:02:49 EDT: %CSM_SLB-6-REDUNDANCY_INFO: Module 8 FT info: Active: Manual
bulk sync completed
```

The `hw-module csm` *`module#`* `standby config-sync` command was introduced on the CSM in revision 4.2(1), and on the CSM-S (CSM with SSL module) in revision 2.1(1).

A real server might need to be removed from service for maintenance, or because it is misbehaving. To do this, negate the `inservice` command in the real server configuration:

```
Switch(config)# mod csm 8
Switch(config-module-csm)# real CONTENT-02
Switch(config-slb-module-real)# no inservice
```

The method is the same for real servers, server farms, and vservers. To put the server back into service, remove the `no` from the `inservice` command:

```
Switch(config)# mod csm 8
Switch(config-module-csm)# real CONTENT-02
Switch(config-slb-module-real)# inservice
```

To show the statuses of the real servers, use the `show module csm` *`module#`* `real` command:

```
Switch# sho mod csm 8 real

real                    server farm          weight  state          conns/hits
----------------------------------------------------------------------------
CONTENT-00              CONTENT-FARM         8       OPERATIONAL    26
CONTENT-01              CONTENT-FARM         8       OPERATIONAL    21
CONTENT-02              CONTENT-FARM         8       OPERATIONAL    16
CONTENT-03              CONTENT-FARM         8       OPERATIONAL    24
CONTENT-04              CONTENT-FARM         8       OPERATIONAL    21
HTTP-00                 HTTP-FARM            8       OPERATIONAL    221
HTTP-01                 HTTP-FARM            8       OPERATIONAL    231
HTTP-02                 HTTP-FARM            8       OPERATIONAL    216
HTTP-03                 HTTP-FARM            8       OPERATIONAL    224
HTTP-04                 HTTP-FARM            8       OPERATIONAL    215
```

If one of the servers is having issues, you might see a state of *failed*. Another indication of a server problem might be a zero in the conns/hits column, or a very high number in this column in relation to the other servers.

To show detailed status information for the real servers, use the `show module csm` *`module#`* `real detail` command. This shows the number of current connections, as well as the total number of connections ever made. Also included in the output is the number of connection failures for each server farm:

```
Switch# sho mod csm 8 real detail
CONTENT-00, CONTENT-FARM, state = OPERATIONAL
  address = 10.10.10.16, location = <NA>
  conns = 23, maxconns = 4294967295, minconns = 0
  weight = 8, weight(admin) = 8, metric = 4, remainder = 0
  total conns established = 5991, total conn failures = 12
CONTENT-01, CONTENT-FARM, state = OPERATIONAL
  address = 10.10.10.17, location = <NA>
  conns = 25, maxconns = 4294967295, minconns = 0
  weight = 8, weight(admin) = 8, metric = 3, remainder = 1
  total conns established = 5991, total conn failures = 8
CONTENT-02, CONTENT-FARM, state = OPERATIONAL
  address = 10.10.10.18, location = <NA>
  conns = 23, maxconns = 4294967295, minconns = 0
  weight = 8, weight(admin) = 8, metric = 2, remainder = 7
  total conns established = 5991, total conn failures = 8
CONTENT-03, CONTENT-FARM, state = OPERATIONAL
  address = 10.10.10.19, location = <NA>
  conns = 20, maxconns = 4294967295, minconns = 0
  weight = 8, weight(admin) = 8, metric = 2, remainder = 4
  total conns established = 5991, total conn failures = 18
[- output snipped -]
```

To show the statuses of the virtual servers, use the `show module csm` *`module#`* `vserver` command:

```
Switch# sho mod csm 8 vserver

vserver    type  prot virtual             vlan state        conns
------------------------------------------------------------------
V-CONTENT  SLB   any  10.5.5.6/32:0        ALL  OPERATIONAL  615
V-HTTP     SLB   any  10.5.5.7/32:0        ALL  OPERATIONAL  382
```

Unless you've taken a vserver out of service, the state should be *operational*. If the state is anything other than operational, check the server farms and real servers for failures.

To show detailed status information for vservers, use the `show module` *`module#`* `vserver detail` command:

```
Switch# sho mod csm 8 vserver detail
V-CONTENT, type = SLB, state = OPERATIONAL, v_index = 10
  virtual = 10.5.5.6/32:0 bidir, any, service = NONE, advertise = FALSE
  idle = 3600, replicate csrp = sticky/connection, vlan = ALL, pending = 30, layer 4
  max parse len = 2000, persist rebalance = TRUE
```

```
  ssl sticky offset = 0, length = 32
  conns = 645, total conns = 22707980
  Default policy:
    server farm = CONTENT_FARM, backup = <not assigned>
    sticky: timer = 0, subnet = 0.0.0.0, group id = 0
  Policy          Tot matches  Client pkts  Server pkts
  ------------------------------------------------------
  (default)       22707980     1844674407   31902598137

V-HTTP, type = SLB, state = OPERATIONAL, v_index = 17
  virtual = 10.5.5.7/32:0 bidir, any, service = NONE, advertise = FALSE
  idle = 3600, replicate csrp = sticky/connection, vlan = ALL, pending = 30, layer 4
  max parse len = 2000, persist rebalance = TRUE
  ssl sticky offset = 0, length = 32
  conns = 637, total conns = 2920304957
  Default policy:
    server farm = HTTP_FARM, backup = <not assigned>
    sticky: timer = 0, subnet = 0.0.0.0, group id = 0
  Policy          Tot matches  Client pkts  Server pkts
  ------------------------------------------------------
  (default)       2920305029   3043400882   2154679452
```

To show the statuses of the server farms, use the `show module csm` *`module#`* `serverfarms` command:

```
Switch# sho mod csm 8 serverfarms

server farm      type     predictor    nat   reals   redirect  bind id
---------------------------------------------------------------------
CONTENT-FARM     SLB      RoundRobin   S     5       0         0
HTTP-FARM        SLB      RoundRobin   S     5       0         0
```

This command gives a quick status summary, and shows the number of real servers in each server farm.

To show detailed status information for the server farms, use the `show module csm` *`module#`* `serverfarms detail` command. This command shows the statuses of the server farms, and of every real server within them:

```
Switch# sho mod csm 8 serverfarms detail
CONTENT_FARM, type = SLB, predictor = RoundRobin
  nat = SERVER
  virtuals inservice = 1, reals = 5, bind id = 0, fail action = none
  inband health config: <none>
  retcode map = <none>
  Real servers:
    CONTENT_00, weight = 8, OPERATIONAL, conns = 25
    CONTENT_01, weight = 8, OPERATIONAL, conns = 15
    CONTENT_02, weight = 8, OPERATIONAL, conns = 22
    CONTENT_03, weight = 8, OPERATIONAL, conns = 21
    CONTENT_04, weight = 8, OPERATIONAL, conns = 15
  Total connections = 98

[- text removed -]
```

I use this command a lot because it gives me a nice snapshot of the server farms, with connection details for each real server contained within them.

Upgrading the CSM

I'm including a section on how to upgrade the CSM because I've found the Cisco documentation to be lacking. This process may change with future releases. Here, I'll show you how to upgrade a CSM-S (the S indicates an included SSL daughter card) from Version 2.1 to 2.1(1).

First, you must TFTP the new image to the supervisor's bootflash. The Cisco documentation indicates that the image should be placed on the device `sup-bootflash:`, but, in my experience, this process will not work unless the image is located on the `bootflash:` device. If you're having trouble getting this to work, try another bootflash device.

Then, using the `tftp-server` command, configure the TFTP server to serve the file you just loaded to the bootflash:

```
Switch(config)# tftp-server bootflash:c6slb-csms-k9y9.2-1-1.bin
```

To proceed, you'll need to control the CSM directly. To do this, issue the `session` command, followed by the *`module#`*, the keyword `processor`, and the *`processor#`* (which should always be zero):

```
Switch# session slot 8 proc 0
The default escape character is Ctrl-^, then x.
You can also type 'exit' at the remote prompt to end the session
Trying 127.0.0.80 ... Open


 wwwwwwwwwwwwwwwwwwwwwwww
 www.C o n t e n t     w
 www.S w i t c h i n g  w
 www.M o d u l e       w
 wwwwwwwwwwwwwwwwwwwwwwww
```

You will now find yourself in the CSM module's CLI. Issue the `upgrade` command followed by `slot0:`, a space, and the name of the image you downloaded earlier:

```
CSM> upgrade slot0: c6slb-csms-k9y9.2-1-1.bin

Upgrading System Image 1
CSM ExImage Mon May 09 18:38:02 2005

R|W|  Reading:shakira.rom......    Writing:lam_ppc.bin..DONE
Pinging SSL daughtercard ... success.
Transferring 3958636 byte image...
total bytes sent 3958636 0x3c676c
```

```
Shakira Image transfer successful.  Waiting for flash burn to complete ..
................................................................................
................................................................................
................................................................................
.............................................................success

Read 14 files in download image. (16,0,64)
Saving image state for image 1...done.
```

Using `upgrade slot0:` on the CSM may seem misleading, as there is often a `slot0:` CompactFlash device available in IOS. The CSM has no concept of a CompactFlash drive, though, and instead references the MSFC by the name `slot0:`. In other words, this basically means to upgrade using the MSFC as your TFTP server. This may change in future releases.

This process can take a long time. Be patient. When you're done, exit the CSM, and reset the card with the `hw-module module` *`module#`* `reset` command:

```
CSM> exit
Good Bye.

[Connection to 127.0.0.80 closed by foreign host]
Switch# hw-module module 8 reset
Proceed with reload of module?[confirm]
% reset issued for module 8
Switch#

01:45:03: %C6KPWR-SP-4-DISABLED: power to module in slot 8 set off (Reset)
01:46:21: %PM_SCP-SP-4-UNK_OPCODE: Received unknown unsolicited message from module
8, opcode 0x330
01:46:55: %DIAG-SP-6-RUN_MINIMUM: Module 8: Running Minimum Diagnostics...
01:46:56: %MLS_RATE-4-DISABLING: The Layer2 Rate Limiters have been disabled.
01:46:56: %SVCLC-5-FWTRUNK: Firewalled VLANs configured on trunks
01:46:56: %DIAG-SP-6-DIAG_OK: Module 8: Passed Online Diagnostics
01:46:56: %OIR-SP-6-INSCARD: Card inserted in slot 8, interfaces are now online
```

When the CSM is rebooted, use the `show module` command to make sure the new software revision is loaded:

```
Switch# sho mod
Mod Ports Card Type                              Model              Serial No.
--- ----- ---------- ----- ---
  1   48  CEF720 48 port 10/100/1000mb Ethernet  WS-X6748-GE-TX     SAL09858F2K
  4   48  CEF720 48 port 10/100/1000mb Ethernet  WS-X6748-GE-TX     SAL09322F2K
  5    2  Supervisor Engine 720 (Active)         WS-SUP720-3B       SAL0935896A
  6    0  Supervisor-Other                       Unknown            Unknown
  7    6  Firewall Module                        WS-SVC-FWM-1       SAD092803DF
  8    0  CSM with SSL                           WS-X6066-SLB-S-K9  SAD094107YN
  9    8  CEF720 48 port 10/100/1000mb Ethernet  WS-X6748-GE-TX     SAL09881F2K
```

```
Mod MAC addresses                       Hw    Fw           Sw           Status
- --------- -- --- --- --
  1  0015.2bca.1df4 to 0015.2bca.1e23   2.3   12.2(14r)S5  12.2(17d)SXB Ok
  4  0014.a90c.6ce0 to 0014.a90c.6ce7   3.0   7.2(1)       3.4(1a)      Ok
  5  0014.a97d.b5d4 to 0014.a97d.b5d7   4.4   8.1(3)       12.2(17d)SXB Ok
  6  0000.0000.0000 to 0000.0000.0000   0.0   Unknown      Unknown      Unknown
  7  0014.a90c.3a58 to 0014.a90c.3a5f   3.0   7.2(1)       2.3(2)       Ok
  8  0015.636e.9ea2 to 0015.636e.9ea9   1.1                2.1(1)       Ok
  9  0014.a9bc.f8b8 to 0014.a9bc.f8bf   5.0   7.2(1)       4.1(4)S91    Ok
```

PART VII
Quality of Service

Quality of Service (QoS), with an emphasis on low-latency queuing (LLQ), is the focus of this section. The first chapter explains QoS, and the second walks you though the steps necessary to deploy LLQ on a WAN link. Finally, I'll show you how congested and converged networks behave, and how LLQ can be tuned.

This section is composed of the following chapters:

Chapter 29, *Introduction to QoS*
Chapter 30, *Designing a QoS Scheme*
Chapter 31, *The Congested Network*
Chapter 32, *The Converged Network*

CHAPTER 29

Introduction to QoS

Quality of Service (QoS) is deployed to prevent data from saturating a link to the point that other data cannot gain access to it. Remember, WAN links are serial links, which means that bits go in one end, and come out the other end, in the same order: regardless of whether the link is a 1.5 Mbps T1 or a 45 Mbps DS3, the bits go in one at a time, and they come out one at a time.

QoS allows certain types of traffic to be given a higher priority than other traffic. Once traffic is classified, traffic with the highest priority can be sent first, while lower-priority traffic is queued. The fundamental purpose of QoS is to determine what traffic should be given priority access to the link.

Figure 29-1 shows two buildings connected by a single T1. Building B has a T1 connection to the Internet. There are servers, and roughly 100 users in each building. The servers replicate their contents to each other throughout the day. The users in each building have IP phones, and inter-building communication is common. Users in both buildings are allowed to use the Internet.

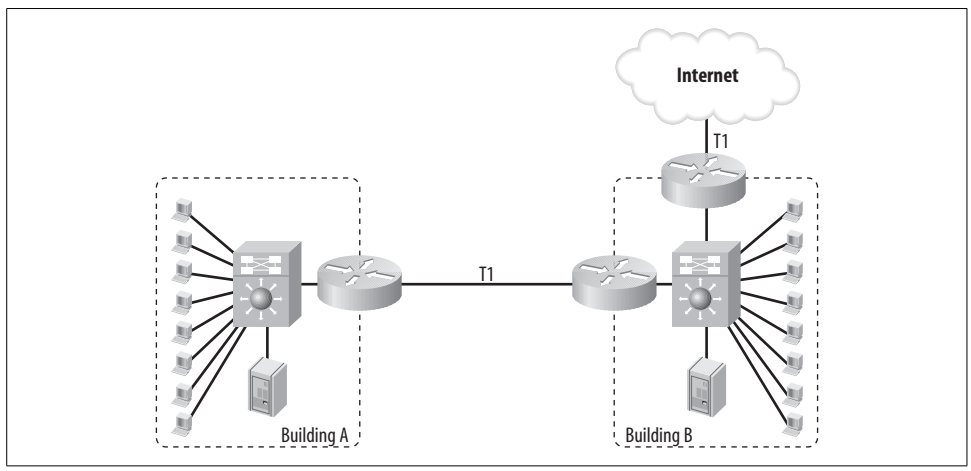


Figure 29-1. Simple two-building network

The only path out of the network in Building A is the T1 to Building B. What happens when each of the users in that building decides to use that single link at once? The link is only 1.5 Mbps, and each user may have a 100 Mbps (or even 1 Gbps) Ethernet connection to the network.

A good designer should never have built a network with these limitations, and the admin (if one exists) should not have let the problem get so severe. Still, the real world is filled with networks like this just waiting for someone to fix them. Many smaller companies don't have dedicated network administrators on staff, so problems like this can and do occur.

Let's say, for example, that 30 of the 100 users start to download the free demo of GAD's Rockin' Guitar Solos, Volume VI. Let's also say that 20 of the users decide to surf the O'Reilly web site. Another 20 need to download the latest service packs from their favorite operating system vendors, and the remaining 30 want to use their VoIP phones.

The problem is that the T1 isn't "big" enough for the amount of traffic that's about to be created. Imagine you had a hose with a funnel on top, as shown in Figure 29-2. The hose is capable of allowing one liter per minute to flow through it. As long as you only pour one liter of water every minute into the funnel, you'll be fine.

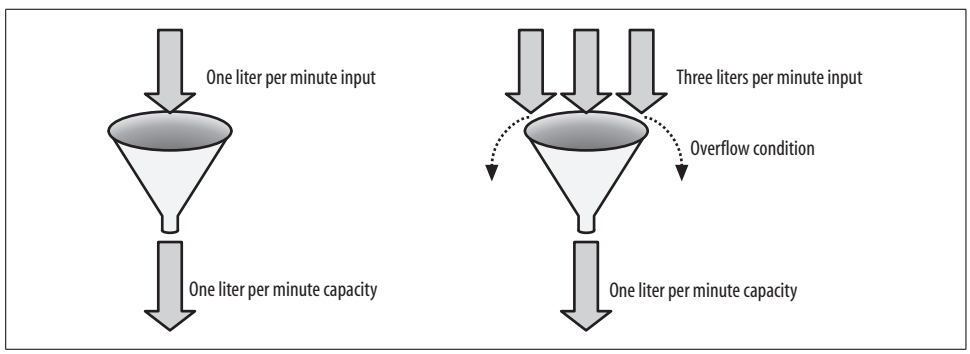


Figure 29-2. Overflow condition

Now, imagine you decide to get a little crazy and pour three liters into the funnel in one minute's time. The hose will only allow one liter per minute to flow out of the funnel. So what happens to the other two liters? They pour over the top of the funnel all over the floor, literally creating an *overflow* condition.

Our T1 is similar, but instead of water, we've got packets, which have the property of being discrete logical objects as opposed to being a fluid. A T1 can handle 1,500,000 packets per second. When the rate of packets being sent to the T1 exceeds this rate, the T1 interface on the router buffers as many of the packets as possible

(like the funnel does with the water), and then, when the buffer gets full, it overflows. Instead of spilling water all over the floor, the interface drops the packets. At this point, the packets go into the proverbial "bit bucket." Sadly, that's just a fun term. There's no bucket—when packets are dropped, they are gone forever.

Water is just water, but some types of packets may be more important than other types of packets. Packets are categorized in different ways: UDP, TCP, FTP, HTTP, and VoIP, to name a few. Not only can different packets have different priorities, but in many instances, they can be time-sensitive as well. Some protocols require that packets arrive in order. Other protocols may be sensitive to packet loss. Let's take a look at some of these protocols and see how they differ:

TCP
: TCP includes algorithms that alert the sending station of lost or damaged packets so they can be resent. Because of this, TCP-based applications are generally not sensitive to lost packets. TCP-based applications tend to be less time-sensitive than UDP-based applications.

UDP
: UDP does not do any error checking, and does not report on lost packets. Because of this, UDP-based applications may be sensitive to packet loss.

HTTP
: HTTP is TCP-based. Generally, HTTP applications are not time-sensitive. When viewing a web page, having to wait longer for an image to load due to a dropped packet is not usually a problem.

FTP
: FTP is TCP-based. FTP is not a real-time protocol, nor is it time-sensitive. If packets are dropped while downloading a file, it's usually not a problem to wait the extra time for the packets to be resent.

Telnet and SSH
: Telnet and SSH are both TCP-based. While they may appear to be real-time, they're not. Lost packets being resent manifest as slow responses while typing. This may be annoying, but no damage is done when packets are dropped and resent.

VoIP
: Voice over IP is UDP-based for the Real-Time Protocol (RTP) voice stream, and TCP-based for the call-control stream. VoIP requires extreme reliability and speed, and cannot tolerate packets being delivered out of order. The use of UDP may seem odd, since UDP is not generally used for reliable packet delivery. VoIP uses UDP to avoid the processing and bandwidth overheads involved in TCP. The speed gained using UDP is significant. Reliability issues can be resolved with QoS; in fact, VoIP is one of the main reasons that companies deploy QoS.

Assuming a FIFO (First In First Out) interface for this example, packets may be sent out of a serial interface in a noncontiguous fashion. In Figure 29-3, there are three types of packets being delivered: HTTP, FTP, and voice. Voice information, in particular, is very sensitive to packets being delivered out of order. It is also very sensitive to packets being lost. Remember, voice traffic is UDP-based, and there is no reliable transport mechanism, so if the buffer overflows, and a voice packet is dropped, it's gone forever. This will result in choppy voice calls and irritated users.

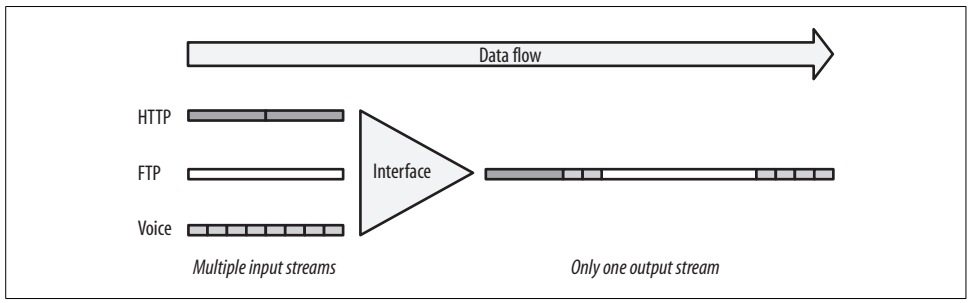


Figure 29-3. Packets through a serial interface

QoS can alleviate (if not, perhaps, solve) these problems. QoS can be daunting, as it has a lot of facets. If you decide at some point to pursue the CCIE, you'll need to know them all in detail, and you'll need to know how to abuse them in all sorts of odd ways. Most of those test-specific games don't have any basis in the real world, though. For our purposes, we will stick to the most common scenarios.

In the real world, QoS is most often used for a couple of reasons. Either there's some sort of streaming media (like voice or video) that requires low latency and timely delivery, or you've got an oversubscribed link, and the powers that be want to make it work better so they don't have to buy more bandwidth. Perhaps the new circuit has been ordered, but due to facility problems, it won't be delivered for 120 days. QoS is the answer to help make the existing link a little more useful until the bigger link is delivered.

QoS can also be used for some more interesting applications. For example, you could configure your network so that telnet and SSH have priority over all other traffic. When a virus hits, and you need to telnet to your routers, the telnet and/or SSH traffic will always get through (assuming you've rate-limited the CPUs on said routers). Or, if you're eager to please the boss, you could prioritize his traffic above everyone else's (except, of course, your own) so he'll have a better online experience. While these examples may seem far-fetched, I've been asked to do just these sorts of things for customers. In many cases, the operations department needs better network access than the rest of the company. And once an executive learns about QoS, he may demand that he get "better" treatment on the network. I wouldn't recommend this, but I've seen it happen.

In a nutshell, any traffic that can be identified can be prioritized. Deciding whether the traffic should be prioritized (and, if so, in what order) is your job, and it can be an interesting one, to say the least.

Types of QoS

The term *QoS* is used to describe any of the myriad functions used to limit how much bandwidth may be used, guarantee how much bandwidth may be used, or prioritize certain traffic over other traffic. Technically, QoS is only one of a broader range of protocols and ideas. The same term can refer to CoS (Class of Service), QoS, queuing, marking, policing, or even traffic shaping.

For example, on Cisco routers, any serial link under 2 Mbps has weighted fair queuing (WFQ) enabled. To quote the Cisco documentation:

> WFQ is one of Cisco's premier queuing techniques. It is a flow-based queuing algorithm that does two things simultaneously: it schedules interactive traffic to the front of the queue to reduce response time, and it fairly shares the remaining bandwidth between high bandwidth flows.

Sounds like QoS to me! It is, and it's probably already in use on your network. Turning off WFQ on a saturated link can have a dramatic impact.

Some forms of QoS are very simple to implement, and affect only the interfaces on which they are configured. Other types can be installed on a single interface, or deployed from end to end on an enterprise network. Voice over IP is probably the biggest reason a large-scale QoS deployment might be considered.

Consider Figure 29-4, which shows a simple two-building network with VoIP phones. There are PCs and IP phones in both buildings. With a simple install, all resources share the network equally. But, this is a problem in a VoIP world because voice packets must be delivered in a timely fashion, in order, and should not be dropped. Because the VoIP voice stream is UDP-based, reliable transport for these packets is not guaranteed like it is in TCP. VoIP packets must compete for bandwidth with other traffic when other protocols are in use.

Let's assume that a VoIP call is active between buildings A and B, and everything is working fine. Now, let's say a PC in Building A pulls a large file from the server in building B using FTP. Suddenly, the voice call gets choppy. If you've got luck like mine, the call will be from the CEO to the payroll company about your raise—which will now be canceled because the voice quality was so bad.

At any rate, let's look at the problem, and see if we can come up with some ways to solve it by discussing the way QoS works, and how different types of QoS might be deployed.

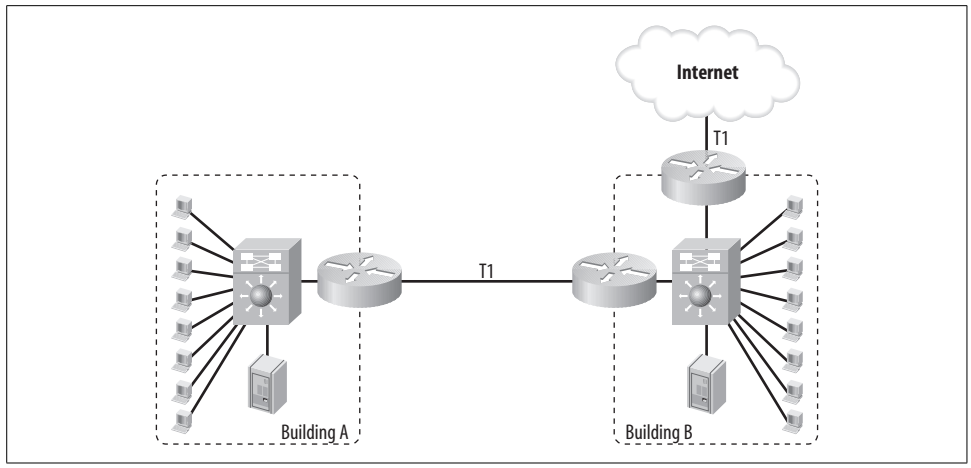


Figure 29-4. Simple two-building VoIP network

QoS Mechanics

QoS is usually deployed by first marking packets, then policing the packets, and, finally, scheduling the packets. *Marking* refers to deciding what priority a packet should be and labeling it accordingly. For example, a voice RTP stream would have the highest priority. *Policing* refers to the actions the router takes based on how the packets are marked. For example, we might specify that any packet marked with the highest priority should be guaranteed 10 percent of the overall link's available bandwidth. *Scheduling* refers to the interface actually serving the packets, in the order determined by how the marked packets are policed. In other words, in the case of high-priority voice RTP packets, we will deliver them first, and then deliver all other packets according to their priorities (if applicable).

All of these steps can be carried out on the same device, or on separate devices. For example, Cisco IP phones can automatically set their voice RTP packets to a high priority. This means you don't have to test for that packet type on the router. Marking on a router adds CPU load, so if it can be offloaded to a device that can do it natively, that's generally a good idea.

Priorities

Every IP packet has a field in it called the Type of Service (TOS) field. This eight-bit field is split up into a couple of sections that can be a little confusing. The beauty of the design is buried in its logic, so hang in there.

Two primary types of IP prioritization are used at layer three: *IP precedence* and *differential services* (*diffserv*).

IP precedence goes way back to the early days of the Internet. Diffserv is much newer. The values for diffserv are called *differential service code point* (DSCP) values.

The TOS field is illustrated in Figure 29-5.

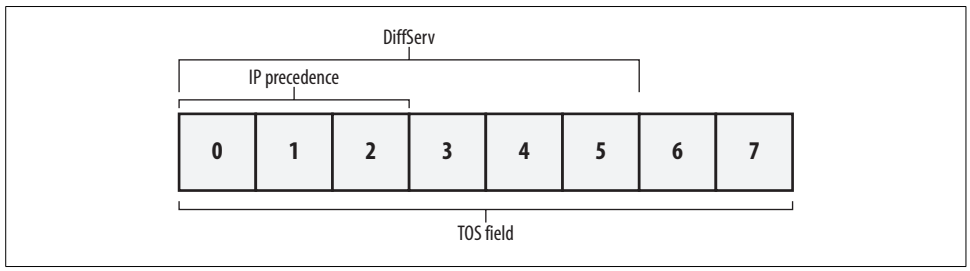


Figure 29-5. The IP TOS field

The field is eight bits long, with the first three bits used for IP precedence. The IP precedence values are called *service mappings*, and are defined in RFC 795 as follows:

```
Network Working Group                                          J. Postel
Request for Comments:  795                                          ISI
                                                          September 1981
                            SERVICE MAPPINGS
                            ----------------
This memo describes the relationship between the Internet Protocol (IP) [1]
Type of Service and the service parameters of specific networks.

The IP Type of Service has the following fields:

   Bits 0-2:  Precedence.
   Bit    3:  0 = Normal Delay,      1 = Low Delay.
   Bits   4:  0 = Normal Throughput, 1 = High Throughput.
   Bits   5:  0 = Normal Relibility, 1 = High Relibility.
   Bit  6-7:  Reserved for Future Use.

      0     1     2     3     4     5     6     7
   +-----+-----+-----+-----+-----+-----+-----+-----+
   |                 |     |     |     |     |     |
   |   PRECEDENCE    |  D  |  T  |  R  |  0  |  0  |
   |                 |     |     |     |     |     |
   +-----+-----+-----+-----+-----+-----+-----+-----+

   111 - Network Control
   110 - Internetwork Control
   101 - CRITIC/ECP
   100 - Flash Override
   011 - Flash
   010 - Immediate
   001 - Priority
   000 - Routine
```

The early Internet (this RFC was written in 1981) was pretty simple by today's standards, and three bits of IP precedence were fine. But eventually, the issue of scalability became apparent as users began to want more than eight levels of distinction in their priorities. So, along came RFC 2474, which redefined the TOS field:

```
3.  Differentiated Services Field Definition

   A replacement header field, called the DS field, is defined, which is
   intended to supersede the existing definitions of the IPv4 TOS octet
   [RFC791] and the IPv6 Traffic Class octet [IPv6].

   Six bits of the DS field are used as a codepoint (DSCP) to select the
   PHB a packet experiences at each node. A two-bit currently unused
   (CU) field is reserved and its definition and interpretation are
   outside the scope of this document. The value of the CU bits are
   ignored by differentiated services-compliant nodes when determining
   the per-hop behavior to apply to a received packet.

   The DS field structure is presented below:


     0   1   2   3   4   5   6   7
   +---+---+---+---+---+---+---+---+
   |         DSCP          |  CU   |
   +---+---+---+---+---+---+---+---+

     DSCP: differentiated services codepoint
     CU:   currently unused
```

Here's where the beauty of the design comes into play. Let's assume our TOS field contains the bits 10100000. The TOS field is always eight bits (an octet). If we look at it using the rules of IP precedence, only the first three bits are significant, giving us a value of 5. If we look at it using the rules of diffserv, however, the first six bits are significant, giving us a value of 40. And if we look at the entire field, with all eight bits being significant, we get a value of 160. The relationship between IP precedence, diffserv, and the TOS field is shown in Figure 29-6.

Knowing that a value of 160 in the TOS field equals an IP precedence of 5 can be valuable when looking at packet captures. Because the field is known only as TOS to IP, the packet-capture tool will usually report the TOS value.

CoS is a layer-2 form of QoS. CoS works under the same principles as IP precedence, in that there are only eight values determined within a three-bit field. The difference is that these bits are in the 802.1P frame header, not the IP header. CoS values are Ethernet-specific, so they will be lost if frames are forwarded out of Ethernet networks. The good news is that because they map so perfectly to IP precedence values, rules can be created to read CoS values and translate them to IP precedence values.

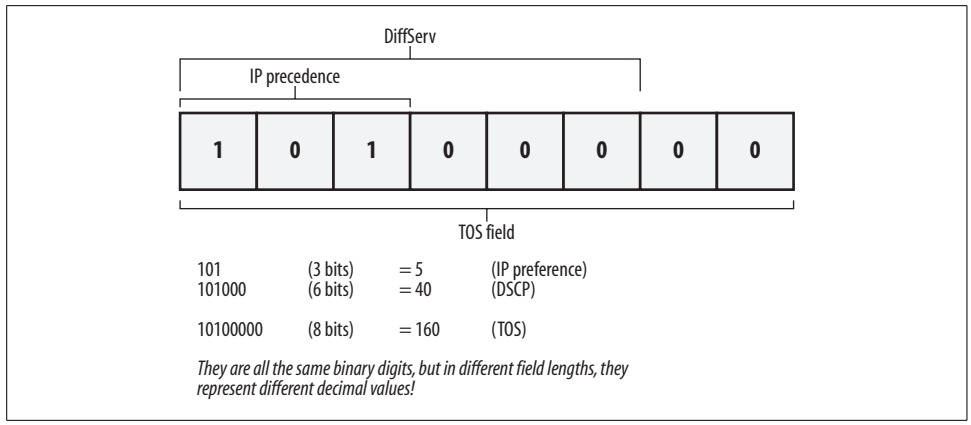


Figure 29-6. Different decimal values depending on the number of significant bits in the TOS field

In Cisco catalyst switches, there can be multiple queues for each Ethernet interface. CoS values can be mapped to these queues, allowing for prioritization. Ethernet priorities can be different from WAN-link priorities, though in practice, they're usually mapped one to one.

Table 29-1 shows how CoS, IP precedence, and DSCP values are related. If you take a look at IP precedence 5, you'll see that it is the same as a DSCP value of between 40 and 47. The decimal value 40 is 101000 in binary. The decimal value 47 is 101111 in binary. However, 48 is 110000 in binary—the first three bits change from 101 to 110, thus changing the IP precedence significant bits in the TOS field from 5 to 6.

Table 29-1. Priority levels of different QoS types

CoS	IP precedence	DSCP	Name
0	0	0–7	Routing (Best Effort)
1	1	8–15	Priority
2	2	16–23	Immediate
3	3	24–31	Flash
4	4	32–39	Flash-Override
5	5	40–47	Critical
6	6	48–55	Internet
7	7	56–63	Network

Because the IP precedence values and the DSCP values are so closely related, they can usually be interchanged. For example, if you're running a diffserv environment that uses DSCP values, but have IP phones that will only mark packets using IP precedence values, you're still OK because a diffserv value of 40 is the same bitwise as an IP precedence value of 5.

Flavors of QoS

These are the most commonly seen forms of QoS. If you're working with a VoIP network, you'll probably see CBWFQ on your routers, and CoS on your switches:

Weighted fair queuing (WFQ)
: Weighted fair queuing is the default queuing mechanism on 2 Mbps and slower serial links. For most implementations, the default configuration works fine. WFQ can be configured very specifically, but it usually isn't configured at all. When VoIP is involved, low-latency queuing is typically used in place of WFQ.

Class-based weighted fair queuing (CBWFQ)
: Class-based weighted fair queuing allows you to configure classes of traffic and assign them to priorities and queues. CBWFQ is the basis for low-latency queuing.

Priority queuing
: Priority queuing works just how it sounds: queues are created, and each class of packet is assigned to an appropriate queue based on priorities you design.

Custom queuing
: Custom queuing is one of those features that you probably won't see much. You'll see it on Cisco exams, and you may see it on networks that had specific problems to be solved where voice was not a concern at the time of the resolution. That's not to say that custom queuing is not suitable for voice, but again, low-latency queuing is the QoS method of choice for voice-enabled networks.

Low-latency queuing (LLQ)
: Low-latency queuing is class-based weighted fair queuing with a *strict priority queue*. A strict priority queue is one in which hardware is used to send packets that need to be sent with the lowest latency possible. This is especially useful for voice and video, where any latency or delay causes problems. LLQ is the preferred method of QoS for voice networks, and is the QoS method I focus on in this book.

Traffic shaping
: Traffic shaping is slightly different from queuing. Traffic shaping monitors traffic, and, when it reaches a threshold, instead of discarding packets, keeps them until a point where bandwidth has lowered and they can be sent. The benefits are a smoother use of bandwidth (Figure 29-7) and the fact that packets are buffered and not just dropped. The downside is that traffic shaping requires memory to function because it buffers packets. Also, if the configured buffer overflows, packets will be discarded.

  Traffic shaping is not really suitable for voice, as voice packets must be delivered in order and with low latency. Queuing of packets for transmission at a later time is not a viable solution for VoIP protocols.

Traffic shaping can be used on data networks where TCP can assemble packets received out of order. However, in today's converged networks, traffic shaping is not often used.

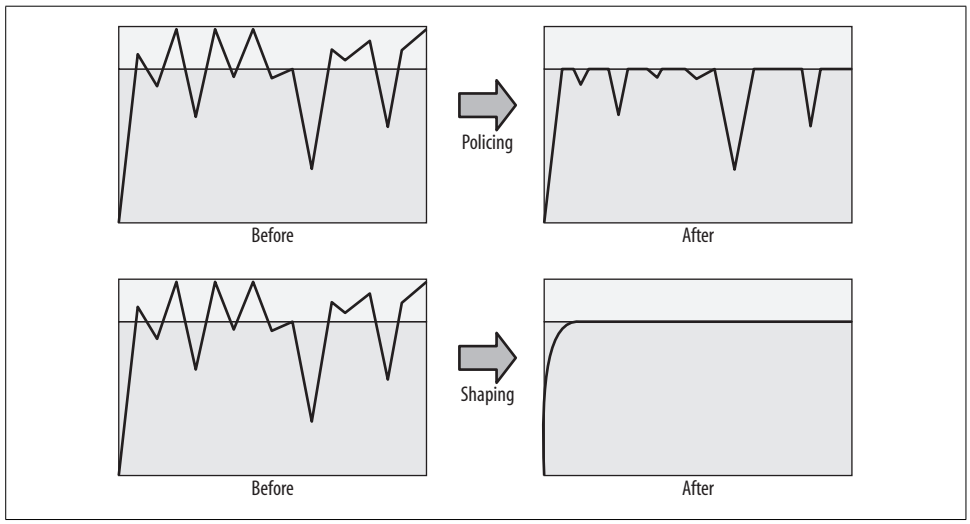


Figure 29-7. Traffic policing versus traffic shaping

Common QoS Misconceptions

There are some pretty wide-ranging misconceptions about what QoS can do. QoS is pretty complex, and a lot of people just skim the documentation, and make assumptions based on what they've read. Still others read only executive summaries, or worse, extrapolate from information they've heard from other misinformed people.

QoS is a very useful technology, but it is not a miracle. It cannot make a T1 perform like a DS3, and it cannot completely prevent packets from being dropped. What QoS can do for you depends on your network and your requirements. If you have a DS3's worth of bandwidth requirements, no amount of queuing or reservation will make a T1 good enough. Let's examine a few common misconceptions:

QoS "carves up" a link into smaller logical links
: I can't tell you how many times I've heard this in meetings and technical discussions regarding QoS. While the analogy is useful in certain circumstances, it is a dangerous one because the perception is that each "chunk" of the link cannot be used for other traffic. This is simply not true. The important thing to remember here is that while scheduling of packets always takes place, the limits set are really only enforced during congestion.

Using low-latency queuing as an example, let's think about the process. Packets come into the interface, and are scheduled according to your configuration. Say you've used the following allocations for the link:

- Voice—10 percent
- Voice Control—10 percent
- FTP—10 percent
- Email—10 percent
- Telnet/SSH—10 percent
- Remainder (default queue)—remaining 50 percent

It is very tempting (and appealing to many) to assume that the link is now divided into six pieces, with half evenly divided amongst voice, voice control, FTP, email, and telnet/SSH, and the other half devoted to everything else. I think the reason this is such a common misconception is because it's a very simple view, and, as such, is very easy to understand.

In reality, if email is the only traffic riding over the link, it will use up 100 percent of the link. The same is true for any type of traffic. When there is no congestion, any protocol can use any amount of available bandwidth it needs. Say a single user decides to download a massive file using HTTP. If there is no other data traversing the link, that download may saturate the link.

Now, say someone else starts a similar download using FTP. Because FTP is guaranteed 10 percent of the link, the scheduler will forward the FTP packets before it will forward the HTTP packets. Because the FTP queue will be serviced before the default queue, the FTP download will impact the HTTP download. Without getting too deep into how the scheduler works, it's worth pondering the following issue: in this scenario, with no other traffic, will FTP get better service, or will HTTP get better service? FTP is only guaranteed 10 percent of the link. That means that HTTP can take up the remainder, right? Wouldn't that mean that HTTP would get better service than FTP?

The truth lies in the way the scheduler works. Because FTP is configured as a priority over HTTP (which falls into the default queue), the scheduler will service FTP packets before HTTP packets. The difference will probably not be obvious to the users, though, *unless there is congestion*. This is an important distinction. The scheduler will service FTP first, which means there will be a higher probability that HTTP packets will be dropped. Packets are only dropped when there is congestion, however, so both protocols will seem to operate normally until the link is saturated.

To summarize, if there is congestion, QoS helps determine which packets should be serviced, and which packets can be dropped. In the case of the FTP vs. HTTP example, as FTP is configured as a priority, and HTTP is not, if there is congestion, HTTP packets will probably be dropped. Without congestion, FTP packets will be sent first, but the HTTP packets will at worst be queued for a short time before being sent.

QoS limits bandwidth

Let's get semantic. QoS does not *limit* bandwidth; it *guarantees* it, which is not the same thing. That is to say, if you have a 1.5 Mbps T1, and you want to run 1 Mbps of FTP across it, a configuration that guarantees 10 percent of the link for FTP will not limit FTP to only 10 percent.

Again, with no congestion, any protocol can use up as much bandwidth as it likes, provided no other configured protocol needs its share. If I configure 20 protocols on a DS3, and each one needs 1 Mbps, any one of those protocols can use all the bandwidth it wants, provided that (are you sick me saying this yet?) there is no congestion on the link.

QoS resolves a need for more bandwidth

As I've said earlier, if you need a DS3, QoS will not make your T1 "just as good." If you have an oversubscribed link, the best QoS can do is to prioritize which packets should be sent first, which also means it's determining which packets can be dropped! That's right—not only does QoS help some protocols, but it makes others worse. Remember the distinction, though: only during congestion.

If you have a T1, and you're trying to shove 20 Mbps through it, I don't care how you prioritize and queue your packets, you're going to drop a lot of them.

Of course, QoS not resolving a need for bandwidth is the corollary to "Throwing more bandwidth at the problem will not always fix it." Knowing the difference is what makes you, the engineer, valuable to management. If you have a T1 link, and you're dropping 50 percent of your packets, assuming there are no physical-layer problems, you probably need more bandwidth.

If you're running VoIP over a T1, and the calls are choppy, adding more bandwidth may not solve your problem because the problem probably isn't simply a matter of bandwidth. The key here again is congestion. If you've got choppy calls, and your links aren't congested, you need QoS to prioritize those voice packets so they get delivered first.

QoS prevents packets from being dropped

If you've got a problem with excessive packet drops, QoS will not solve it. What QoS will do for you is help you get the important packets through so that only the less important packets get dropped.

Traffic shaping *can* prevent packets from being dropped (assuming buffers are not similarly saturated), though this is not really suitable for voice or video. Additionally, traffic shaping only buffers packets to a point, so if you're oversubscribing your link all day long, traffic shaping won't help you.

QoS will make you more attractive to the opposite sex

Sad to say, I'm afraid this isn't true either, though in theory, it is possible that your dream date could be reading this very book right now!

CHAPTER 30

Designing a QoS Scheme

Designing a QoS scheme can be a relatively simple task, or a very large project. The scale of the solution depends on the complexity of your network, and the needs of your business. As a general rule, I like to plan everything I do with an end state design in mind. Even if I'm implementing a QoS scheme for a company that currently doesn't have VoIP, I'll design a scheme that assumes VoIP is coming in the future.

In this chapter, I'll walk you through designing a QoS scheme for a network that will require VoIP. Designing QoS is a two-part process: first, you must determine the requirements for the system, and then you have to configure the routers. We'll use low-latency queuing (LLQ), which is the recommended solution according to Cisco.

Determining Requirements

The first part of the QoS design process will require some investigative work on your part. You should interview all the business groups that use the network, and determine what protocols are important to them. Assume from the onset that every one of them will tell you that their protocols are the most important, and should get the lion's share of the bandwidth. You will need to assimilate all of this data, and make some decisions about how to allocate the bandwidth. With the help of good management, you should be able to come up with a list of requirements from which you can work.

Protocols

Compile a list of the protocols in use on your network. Tools to help you with this task include packet-capture applications such as Ethereal, netflow switching on Cisco routers, and Network Analysis Modules in 6500 switches. Take careful notes. Also, bear in mind that companies like to change the default ports for popular protocols. You might have web traffic on ports 80, 443, and 8080, for example.

Record who is using each protocol, and how the protocols behave. Some protocols can be tricky. For example, RTP is used for the voice stream of Cisco VoIP. If you assume that all RTP packets should get top priority to ensure voice quality, there may be unintended consequences. Other applications, such as streaming video, may also use RTP. If you design a QoS scheme that prioritizes RTP at the highest level without any further granularity, video conferencing could cause quality problems in voice calls. If you know that video conferencing is only done from the executive boardroom, however, you can prioritize that traffic accordingly to avoid problems.

Priorities

Determining priorities can be tough. Politics may well play a part in this step, so be prepared to play the game.

There are certain best practices for QoS. For example:

- You shouldn't use IP precedence 6 and 7, as they are reserved for *internetwork control* and *network control*.
- Voice RTP packets should always be marked IP precedence 5 (DSCP: EF).
- All voice-control packets should be marked IP precedence 3 (DSCP: AF31).
- All data traffic should be marked with lower priorities than voice RTP and control packets.

You may have noticed that if you follow these best practices, you'll now be running pretty short of IP precedence levels. Of the original eight, only levels 0, 1, 2, and 4 remain. IP precedence 4 should be used for critical applications that are not quite as critical as voice, such as video conferencing. IP precedence 0 (essentially, no priority) is used for "best effort" traffic, which leaves only levels 1 and 2. (As you can see, IP precedence doesn't scale well. It's generally best to use diffserv where possible.)

Don't forget about yourself. When the links get congested, and you need to troubleshoot a router problem on the other side of your network, you don't want to have your telnet or SSH traffic being dropped in favor of an FTP download. Because SSH and telnet don't require much bandwidth, the impact of giving them a high priority is small, but the return during a crisis is enormous.

By now, you should be compiling an ordered list of how you want to prioritize your protocols. The best effort line should always be at the bottom. This will end up being your default queue. It's also a good idea to keep an entry labeled *to be determined*, which serves as a placeholder for normal traffic. Assuming you keep the highest priorities on top, the list might look something like this:

- Voice RTP
- Voice control

- Telnet/SSH (multiple protocols can share the same priority and queue)
- ---to be determined---
- Everything else (default—best effort)

In general, you should resist the urge to put HTTP traffic in any queue other than the default. HTTP traffic is not typically mission-critical. However, in some environments—for instance, call-center agents using a web-based application on a remote server—it is important to prioritize HTTP. We'll use this as an example, and assume that all traffic to or from the server (10.10.100.100) on port 80 should be prioritized. The list should now look like this:

- Voice RTP
- Voice control
- Telnet/SSH
- HTTP to/from 10.10.100.100
- ---to be determined---
- Everything else (default—best effort)

Notice I've kept telnet/SSH above the call-center traffic. Remember, we're keeping that as our ace in the hole for when there's an outage.

Our next consideration is email, which can eat up a lot of bandwidth on a network. In a Microsoft Exchange environment, email is stored in files called postoffice (*.pst*) files. You may have users on one end of a link with their *.pst* files stored on a file server in a remote office. Often, the execs will have 1 GB *.pst* files that get opened across the WAN. What's more, all the users who are configured this way probably come in around the same time, and request their huge *.pst* files at the same time every morning, potentially bringing the network to a crawl. Still, the executives insist that email should be prioritized, so we'll put it just above the default queue:

- Voice RTP
- Voice control
- Telnet/SSH
- HTTP to/from 10.10.100.100
- ---to be determined---
- Email (SMTP, POP, IMAP, Exchange)
- Everything else (default—best effort)

In my experience, email problems can usually be attributed to poorly enforced (or even nonexistent) acceptable use policies. For example, one user sending an email with a 20 MB attachment to everyone in the company can cause a significant impact on the network.

Removing the placeholder, we now have the following final list of priorities, in order:

- Voice RTP
- Voice control
- Telnet/SSH
- HTTP to/from 10.10.100.100
- Email (SMTP, POP, IMAP, Exchange)
- Everything else (default—best effort)

This list is a good representation of some real-world environments I've seen, and implementing it will illustrate how to mark different types of traffic in different ways.

Determine Bandwidth Requirements

To illustrate how to determine the bandwidth requirements for the various protocols, we'll use the sample network from the previous chapter, shown in Figure 30-1.

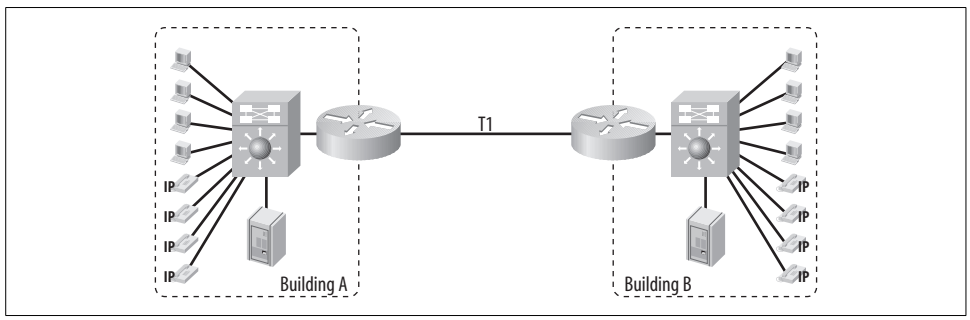


Figure 30-1. Two-building network with a single T1

Low-latency queuing is the preferred QoS method for VoIP. When using this method, you create a single high-priority queue for data streams that cannot suffer dropped packets (voice, video, etc.), and the remaining data is queued according to a hierarchy of priorities.

The priority queue may be assigned a percentage of the available bandwidth (15 percent, for example), or a finite amount of bandwidth (300 Kbps, for example). The priority queue should not consume more than three-quarters of the available bandwidth. In fact, the Cisco IOS prevents you from allocating more than 75 percent to the priority queue, though this value can be altered with the `max-reserved-bandwidth` interface command:

```
Config-if# max-reserved-bandwidth 80
```

WAN links should never be built with more than 75 percent planned utilization. 25 percent should always be left for overhead, routing protocols, administration, etc. Because of the way the scheduler works, if you assign an unusually large value to the priority queue, none of the other queues will ever get serviced. This will result in apparent outages because only the priority queue's traffic will get through.

We're going to assign voice RTP to the strict priority queue. For our purposes, let's say we'll need to allocate space for four G.729a codec VoIP calls (50 pps) at any given time. Some quick research shows that each call should take no more than 37 Kbps. Four calls at 37 Kbps equals 148 Kbps, which is the amount we'll assign to the priority queue:

- Voice RTP—priority queue—148 Kbps

Call control requires very minimal bandwidth. Some Cisco documentation states that 150 bps (plus overhead) per phone should be guaranteed for call control. For four calls, that's only 600 bps. Other documentation states that 2–5 percent of overall link bandwidth should be allocated for call control. For a T1, that's about 30–77 Kbps. Let's take the middle ground, and say we need 30 Kbps for call control:

- Voice control—30 Kbps

We're only prioritizing telnet and SSH for emergencies, so these don't need a lot of bandwidth. 64 Kbps should be more than enough, and we can allocate this amount without impacting other protocols:

- Telnet/SSH—64 Kbps

HTTP to and from 10.10.100.100 is a critical app on our network, so we'll give it a sizable chunk of bandwidth. Let's say during our discovery process http traffic hovered at around 300 Kbps. We'll plan for growth, and allocate 384 Kbps:

- HTTP to/from 10.10.100.100—384 Kbps

Email is prioritized only to appease the executives, so we'll allocate it a small amount of bandwidth. 128 Kbps should give it enough bandwidth to function while still leaving plenty for other applications:

- Email (SMTP, POP, IMAP, Exchange)—128 Kbps

The default queue will be allocated the remaining amount of bandwidth after the other queues are considered. Assuming a T1 at 1544 Kbps, and the queues we've decided upon, we get the following results: 1544 – (148 + 30 + 64 + 384 + 128) = 754 Kbps. So, the final element will be:

- Everything else (default—best effort)—remainder (754k)

Our final list looks thus like this:

- Voice RTP—priority queue—148 Kbps
- Voice control—30 Kbps
- Telnet/SSH—64 Kbps
- HTTP to/from 10.10.100.100—384 Kbps
- Email (SMTP, POP, IMAP, Exchange)—128 Kbps
- Everything else (default—best effort)—remainder (754 Kbps)

Configuring the Routers

We've done all of our discovery and planning. Now, it's time to get to the meat of the exercise: configuring the routers. For this implementation, we're going to configure QoS on the edge routers that connect the T1 to the individual buildings. We will do marking, classification, and policing on the same device, but remember that these steps can be separated onto discrete devices as needed. VoIP phones will usually mark their packets, so we'll assume that they're doing so on our network.

To configure LLQ, we need to configure three things: *class maps*, *policy maps*, and *service policies*.

Class Maps

We need to identify what packets should be marked. To do this, we will create *class maps*, which will match specific types of traffic. There are a wide variety of matching possibilities for class maps—they can match specific IP precedence values, DSCP values, or even ACLs. The only function class maps accomplish is matching.

The following class map, called Voice-Calls, matches either DSCP EF (express forwarding), or IP precedence 5:

```
class-map match-any Voice-Calls
  description [---[ Actual Calls (Set on IP Phones and Dial-Peers) ]---]
  match ip dscp ef
  match ip precedence 5
```

This is technically unnecessary, however, because (as was discussed in the last chapter) the two are synonymous. Let's change it so it's easier to read:

```
class-map match-any Voice-Calls
  description [---[ Actual Calls (Set on IP Phones and Dial-Peers) ]---]
  match ip precedence 5
```

We could just as easily have used DSCP EF, but this is a simple configuration, so we'll stick with IP precedence.

We now need to create class maps for the other items in our list. Next is call control:

```
class-map match-any Voice-Control
  description [---[ Call Control (Set on CallManager and IP Phones) ]---]
  match ip precedence 3
```

Call Manager, the voice gateways, and the IP phones are all capable of setting the precedence values for us, so we'll take advantage of that, and assume that all voice call-control traffic has been marked with an IP precedence of 3.

The next item in our list is telnet and SSH. In this example, we'll match the protocols by name in the class map:

```
class-map match-any Telnet-Traffic
  description [---[ Telnet & SSH ]---]
  match protocol telnet
  match protocol ssh
```

For HTTP, however, we need to include more than just the protocol name. We can't simply say `match protocol http` because then we'd match HTTP traffic from any source to any destination. Our requirements indicate that we need a way to match HTTP (only port 80) traffic either sourced from or destined for 10.10.100.100. Access lists to the rescue:

```
class-map match-any HTTP-Call-Center
  description [---[ Call Center HTTP ]--]
  match access-group 121
```

Now, we need to create an ACL to match our `access-group` statement:

```
access-list 121 remark [---[ Call Center HTTP ]---]
access-list 121 permit tcp any 10.10.100.100 0.0.0.0 eq www
access-list 121 permit tcp any eq www 10.10.100.100 0.0.0.0
```

ACLs allow for a great deal of flexibility when used in class maps. For example, we could:

- Create a time-based ACL (different QoS for different times of the day!).
- Create an IP-based ACL (certain IP addresses get priority).
- Give a group minus one person priority (put a single `deny` on top, then a `permit` for the group).
- Create a dynamic ACL ("If I'm using SSH, don't prioritize Bob's SSH packets.").

Pretty much anything you can dream up can be used, but be careful—the more complex your rules are, the harder they are to manage and maintain.

For the email item on our list, we can again use protocol names. Because email involves many protocols, we'll list them all. The router is even smart enough to reference "exchange" as a traffic type:

```
class-map match-any Mail-Traffic
  description [---[ Any Mail Traffic (Set on WAN Router) ]---]
  match protocol smtp
```

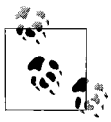
```
match protocol pop3
match protocol imap
match protocol exchange
```

Policy Maps

Now that we have our class maps defined, we need to create the policy maps. While class maps determine *which* packets get policed, the policy map determines *how* they get policed. This step is actually pretty easy if you've done all your planning up front like we have.

Because we're only worried about one T1, we only need to make one policy map. Let's call it WAN-Link:

```
policy-map WAN-Link
 description [---[ Apply Treatment for Classes ]---]
```



You've probably noticed that I use a lot of descriptions. I think `description` statements are the single best thing to ever happen to Cisco IOS. As I've said, simple is good, and well-documented code is always a plus. Any time you have the option of adding a comment to your configuration, I highly encourage you to do so.

A policy map is simply a list of all your class maps with commands for each. First, we'll create the policy map, and an entry for the `Voice-Calls` class:

```
policy-map WAN-Link
 description [---[ Apply Treatment for Classes ]---]
 class Voice-Calls
  priority 148
```

Using the `priority` command within the class map tells the router to assign this class to the strict priority queue. The number after the command is the amount of bandwidth to allocate to the priority queue. In this case, we're allocating 148 Kbps to the priority queue. There are additional subcommands that I encourage you to learn about, but for the present example, this is all you need.

Some newer versions of IOS code allow you to allocate based on a percentage of the entire link. To do this, use a command like `priority percent 10` (this would allocate 10 percent of the circuit's total bandwidth to the priority queue).

Now that we've configured the priority queue, let's configure the rest of the queues. We no longer need to use the `priority` command. From this point on, we'll allocate bandwidth using the `bandwidth` command in each class. This set of commands allocates 30 Kbps of the link's total bandwidth to the Voice-Control class:

```
class Voice-Control
 bandwidth 30
```

On newer versions of IOS, you can specify a percentage of the remaining bandwidth to allocate to each nonpriority queue. For example, if you'd applied 10 percent to the priority queue, and wanted to apply 5 percent of the remaining bandwidth (that is, the bandwidth left after the first 10 percent is used) to the Voice-Control class, you could use the following commands instead:

```
class Voice-Control
 bandwidth remaining percent 5
```

Each of the remaining classes will be configured the same way, including the appropriate bandwidth assignments:

```
class Telnet-Traffic
 bandwidth 64
class HTTP-Call-Center
 bandwidth 384
class Mail-Traffic
 bandwidth 128
```

Finally, we need to configure the default queue. To ensure that weighted fair queuing is applied to the remainder of the traffic, we must reference the *class-default* class in our policy map:

```
class class-default
   fair-queue
```

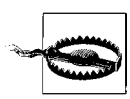
class-default is a reserved name; you cannot create this class, but you should configure it in your policy maps.

Service Policies

`service-policy` is simply the command that applies your class maps and policy maps to an interface. Because we're looking to improve performance on the WAN link, we'll apply our `service-policy` statement to the T1 interfaces on the WAN routers:

```
int S0/0
 service-policy output WAN-Edge
```

Service policies are applied on the output side of an interface. Remember that we're trying to affect how packets are sent onto a link. Received packets have already traversed the link, so there's nothing we can do about them on the receive side.



Be warned that when you change the service policy on an interface, you clear the interface so it can apply the new queuing mechanism. For IP data traffic this is not usually a problem, but it may disconnect, or, at the very least, cause audible gaps in voice calls.

Once you apply the service policy, all of your policies are in effect.

Our final configuration is as follows:

```
class-map match-any Mail-Traffic
  description [---[ Any Mail Traffic (Set on WAN Router) ]---]
  match protocol smtp
  match protocol pop3
  match protocol imap
  match protocol exchange
!
policy-map WAN-Link
  description [---[ Apply Treatment for Classes ]---]
  class Voice-Calls
   priority 148
  class Voice-Control
   bandwidth 30
  class Telnet-Traffic
   bandwidth 64
  class HTTP-Call-Center
   bandwidth 384
  class Mail-Traffic
   bandwidth 128
  class class-default
   fair-queue
!
int S0/0
 service-policy output WAN-Edge
```

CHAPTER 31

The Congested Network

A congested network is one where there's too much data, and not enough bandwidth to support it. QoS can help with a congested network, but it cannot cure the root problem. The only ways to cure congestion on a network are to add more bandwidth, or reduce the amount of data trying to flow over it. That being said, let's look at a congested network, and see how we might fix it.

Determining Whether the Network Is Congested

How do you know if your network is congested? Let's look at our favorite two-building company again (Figure 31-1).

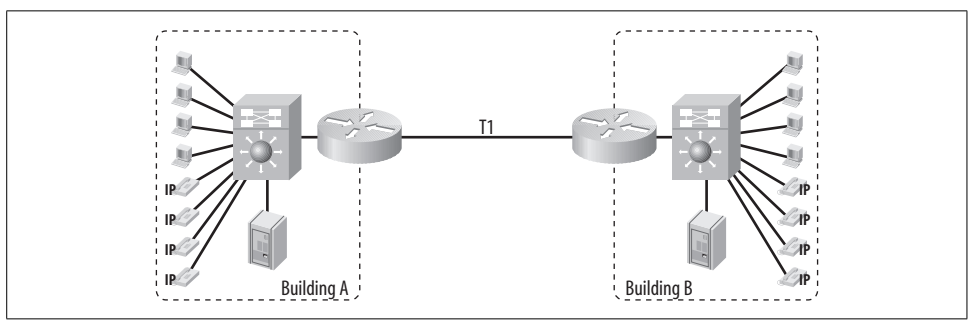


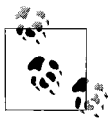
Figure 31-1. Typical two-building network

Users have been complaining that access to the other building is slow. So, let's take a look at the interfaces on one of the routers that connects the T1 between buildings. Here's the output from the `show interface` command for the serial interface on Building B's router:

```
1  Bldg-B-Rtr# sho int s0/0
2  Serial0/0 is up, line protocol is up
3    Hardware is PowerQUICC Serial
4    Description: <[ T1 WAN Link ]>
```

```
5   Internet address is 10.10.10.2/30
6   MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
7      reliability 255/255, txload 42/255, rxload 249/255
8   Encapsulation PPP, loopback not set
9   Keepalive set (10 sec)
10  LCP Open
11  Open: IPCP, CDPCP
12  Last input 00:00:00, output 00:00:00, output hang never
13  Last clearing of "show interface" counters 3w4d
14  Queueing strategy: fifo
15  Output queue 0/40, 548941 drops; input queue 0/75, 3717 drops
16  5 minute input rate 1509000 bits/sec, 258 packets/sec
17  5 minute output rate 259000 bits/sec, 241 packets/sec
18     287554125 packets input, 3659652949 bytes, 0 no buffer
19     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
20     819 input errors, 559 CRC, 227 frame, 0 overrun, 0 ignored, 33 abort
21     282883104 packets output, 3613739796 bytes, 0 underruns
22     0 output errors, 0 collisions, 1 interface resets
23     0 output buffer failures, 0 output buffers swapped out
24     0 carrier transitions
25     DCD=up  DSR=up  DTR=up  RTS=up  CTS=up
```

Looking at this information, you should notice some things right away. First, this is a T1, and the bandwidth is 1,544 Kbps (lines 4 and 6).



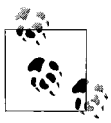
Remember that the bandwidth shown when using `show interface` is not necessarily representative of the true bandwidth of the link. While this value is correct by default, it can be changed with the `bandwidth` interface command. This field is used for calculations in EIGRP and other protocols, which is why it can be altered.

Looking further down (line 16), you can see that the five-minute input rate is 1,509,000 bps, 258 pps. That's 1,509 Kbps used on a 1,544 Kbps link. You might be tempted to say, "Hey, the link's got 33 Kbps left!" However, remember that this is a five-minute *average*.

Perhaps an easier way to get a sense of the utilization is on line 7, which shows:

```
reliability 255/255, txload 42/255, rxload 249/255
```

Looking at these numbers, you can see that the transmit load (txload) is only 42 out of 255, but the receive load (rxload) is 249 out of 255. If you use some simple math (249/255 * 100), you can see that the link is 97.65 percent utilized.



Ever wonder why the rxload and txload numbers are based on 255, and not 100? The decimal number 100 in hex is 64. If we need to burn two hex digits in the router's memory, why not use them to their fullest? The largest number we can represent with two hex digits is FF, which is 255. By using 255 instead of 100, we get more than twice the granularity in our measurements.

The link is saturated, but only in one direction: while txload is 249/255, rxload is only 42/255. To view this as bandwidth used, look at lines 16 and 17. They show we're receiving 1,509 Kbps, but transmitting only 259 Kbps. Remember that T1s are full duplex. What happens in one direction, for the most part, does not affect what happens in the other direction.

T1s, like most telecom links, are full duplex. They are rated at 1.54 Mbps, and can transmit this speed in both directions simultaneously.

An even easier way to see whether your links are saturated is to deploy a tool such as the Multi Router Traffic Grapher (MRTG), which gives you a quick graphical representation of the traffic on your links.

Figure 31-2 shows an MRTG graph for this link. Each number along the bottom of the graph represents an hour of the day. The small triangle on the righthand edge of the graph indicates the current time (i.e., the time the graph was created—in this case, 12:12 p.m.). The vertical line at 0 on the graph indicates midnight. Starting at around 9:30 or 10:00 a.m., and continuing until the current time, the link became completely saturated in one direction. This coincides nicely with what we're seeing on the router.

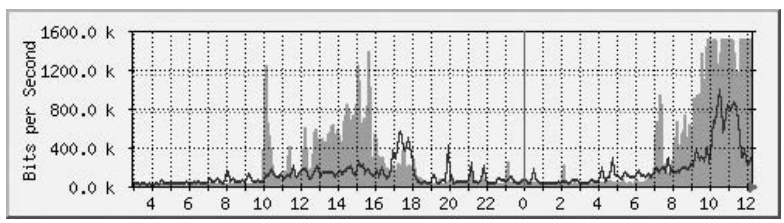


Figure 31-2. MRTG graph showing T1 utilization

OK, so the link is saturated. So what? We're getting our money's worth, right?

If that's the case, why are users complaining of slowdowns? Let's look deeper. The first thing you should always look for is problems at the physical layer. Look for errors in the `show interface` output. The bold lines toward the bottom of the output do indeed indicate some errors:

```
819 input errors, 559 CRC, 227 frame, 0 overrun, 0 ignored, 33 abort
0 output errors, 0 collisions, 1 interface resets
```

These two lines show input errors, CRC errors, framing errors, abort errors, and an interface reset. Hmm. Not so good. Let's not get excited just yet, though. The first thing you should do when you see errors—especially when they're relatively small in number, as in this case—is to wait a few minutes, and do a `show interface` again. Here are the relevant results after executing the same command 10 minutes later:

```
819 input errors, 559 CRC, 227 frame, 0 overrun, 0 ignored, 33 abort
0 output errors, 0 collisions, 1 interface resets
```

As you can see, they're identical; none of the counters are incrementing. This indicates to me that there was probably a problem in the past where someone pulled out the T1 cable and quickly reset it. Or, it could be that the telecom provider had a blip on the line. Either way, it's not worth worrying about right now. Once we've established that the counters are not incrementing, we should clear them so we don't waste time chasing ghosts the next time we're in the router. We don't want to do that yet, though—we still need the data.

The next thing we need to look at is the health of the input and output queues. The previous `show interface` output includes the following:

```
Queueing strategy: fifo
Output queue 0/40, 548941 drops; input queue 0/75, 3717 drops
```

The first thing you should notice is that the *queueing strategy* is FIFO (First In First Out). Links 2 Mbps and below should be using weighted fair queuing (WFQ), so there's something to investigate. Looking further, you'll see that there have been 548,941 drops on the output queue, and 3,717 drops on the input queue. This means that since the last time the counters were cleared (now you know why we didn't clear them), we've dropped 548,941 packets while transmitting.

Line 13 shows how long it's been since the counters were cleared:

```
Last clearing of "show interface" counters 3w4d
```

Line 22 shows how many packets we've sent:

```
282883104 packets output, 3613739796 bytes, 0 underruns
```

So, in three weeks and four days, we've transmitted 282,883,104 packets and dropped 548,941. Some simple math shows that this is only 1/10th of one percent of all the packets. So, why do we have a problem?

OK, I'll admit it, I led you down the wrong path, but I did it to give you some real-world experience. Let's think for a moment here. These numbers look good to me. Sure, we're saturated, but we're not dropping all that many packets, are we? Why, then, are we getting complaints?

The answer is a deceptively simple one: we're looking at the wrong side of the link. Take another look at lines 16 and 17:

```
5 minute input rate 1509000 bits/sec, 258 packets/sec
5 minute output rate 259000 bits/sec, 241 packets/sec
```

The input is almost maxed, not the output! When we drop packets due to congestion, we drop them on the outbound journey. Remember the funnel example from Chapter 29? We're looking at the bottom of the hose here. The only packets that can come out of it are the ones that made it through the funnel! We'll never see the ones that were dropped on the other side.

So, let's take a look at the other side of the link with the same `show interface` command:

```
Bldg-A-Rtr# sho int s0/0
Serial0/0 is up, line protocol is up
  Hardware is PowerQUICC Serial
  Description: [-- T1 WAN Link --]
  Internet address is 10.10.10.1/30
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
     reliability 255/255, txload 250/255, rxload 47/255
  Encapsulation PPP, loopback not set
  Keepalive set (10 sec)
  LCP Open
  Open: IPCP, CDPCP
  Last input 00:00:00, output 00:00:00, output hang never
  Last clearing of "show interface" counters 3w4d
Input queue: 0/75/0 (size/max/drops); Total output drops: 152195125
  Queueing strategy: weighted fair
  Output queue: 63/1000/64/152195113 (size/max total/threshold/drops)
     Conversations  7/223/256 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
  5 minute input rate 261000 bits/sec, 238 packets/sec
  5 minute output rate 1511000 bits/sec, 249 packets/sec
     282883104 packets input, 3613739796 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     872 input errors, 472 CRC, 357 frame, 0 overrun, 0 ignored, 12 abort
     1326931662 packets output, 2869922208 bytes, 0 underruns
     0 output errors, 0 collisions, 1 interface resets
     0 output buffer failures, 0 output buffers swapped out
     0 carrier transitions
     DCD=up  DSR=up  DTR=up  RTS=up  CTS=up
```

A quick look at the load stats shows us that we're looking at the right side now:

```
     reliability 255/255, txload 250/255, rxload 47/255
```

The errors look about the same, and are not of concern:

```
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     872 input errors, 472 CRC, 357 frame, 0 overrun, 0 ignored, 12 abort
```

Now, let's look at the queues:

```
Input queue: 0/75/0 (size/max/drops); Total output drops: 152195125
  Queueing strategy: weighted fair
  Output queue: 63/1000/64/152195113 (size/max total/threshold/drops)
     Conversations  7/223/256 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
```

The first thing that pops out at me is that this end of the link is running with WFQ enabled (as it should be). Remember that the other side was FIFO. They don't need to match, but there's probably no reason for the other side to be FIFO.

And wow! 152,195,113 packets dropped on the output queue! If we divide that into the total number of packets we get (152,195,113 / 1,326,931,662 * 100) = 11.47 percent of all packets dropped. I'd say we've found our problem. This link is so saturated that more than 1 out of every 10 packets is being discarded. One out of 10. It's no wonder users are complaining!

A total of 11.47 percent might not seem like a lot of dropped packets, but remember that VoIP cannot stand to have packets dropped. Even 1 percent would be a problem for VoIP. And for protocols that can recover lost packets, the result is a perceived slowdown. So, what should your dropped-packets counters look like? In a perfect world, they should be zero. More practically, they should not increment as you watch them, and they should be as close as possible to 0 percent of your total packets sent. If you're dropping even 1 percent of your packets over a serial link, you've got a congestion problem.

Now that we know what the problem is, we need to monitor the link over time. We want to ensure that all those packets weren't dropped in the last 10 minutes (though that's unlikely given the high numbers). It is probably safe to make some assumptions, though. If you've configured MRTG to monitor usage for you, the historical information it provides can help, and the pretty graphs make explaining things to the executives a whole lot easier.

Figure 31-3 shows that the link is saturated pretty much all day long. The problem starts roughly when the users come into work, and ends roughly when they all go home. See how the graph peaks and then runs flat up near 1,600 Kbps all day, every day? This link is oversubscribed.

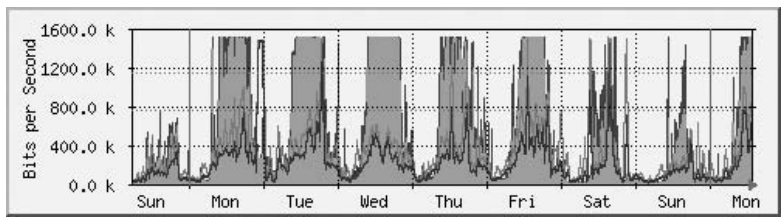


Figure 31-3. One week's historical MRTG graph for our link

Resolving the Problem

So, what do we do? Can QoS save the day? Well, yes and no. QoS is, in fact, already running on the link (weighted fair queuing), but it's making certain assumptions as to what traffic should be prioritized. WFQ is also only running in one direction (luckily, the direction in which we need it to be running, which is the direction that most of the data is flowing).

We've already talked about what QoS cannot do. One of the big things that QoS cannot do is resolve a need for more bandwidth. This link is clearly in need of a larger pipe because we're dropping more than 10 percent of all packets sent from Building A to Building B. In this case, the only ways to prevent these packets from being dropped is to install a larger link between the buildings, or stop sending so much data.

What we *can* use QoS to do is determine more specifically what packets should and should not be dropped. (See Chapters 29 and 30 for more information on designing a QoS scheme.)

Let's clear those counters now, so that the next time we look we'll have some fresh data:

```
Bldg-A-Rtr# clear counters s0/0
Clear "show interface" counters on this interface [confirm]
```

And on the other side as well:

```
Bldg-B-Rtr# clear counters s0/0
Clear "show interface" counters on this interface [confirm]
```

If you think you need a larger link, you will need to collect some real data to support your case. When you approach management with your concerns, having information similar to what I've shown you in this chapter will help justify the expense of more bandwidth. SNMP network monitoring is one of the most useful tools you can deploy on your network. You don't need fancy systems like CiscoWorks or OpenView, though if you can afford those tools, they are worth having. Free tools such as the MRTG are very powerful and useful, especially for justifying larger links.

CHAPTER 32

The Converged Network

In this chapter, you'll see a converged network in action. While the network will be very simple, the principles shown will scale to networks and links of almost any size.

Figure 32-1 shows the network we'll use for the examples in this chapter. R1 and R2 each have two Ethernet networks attached: one with an IP phone, and one with a personal computer. The routers are connected with a T1 that terminates into S0/1 on each.

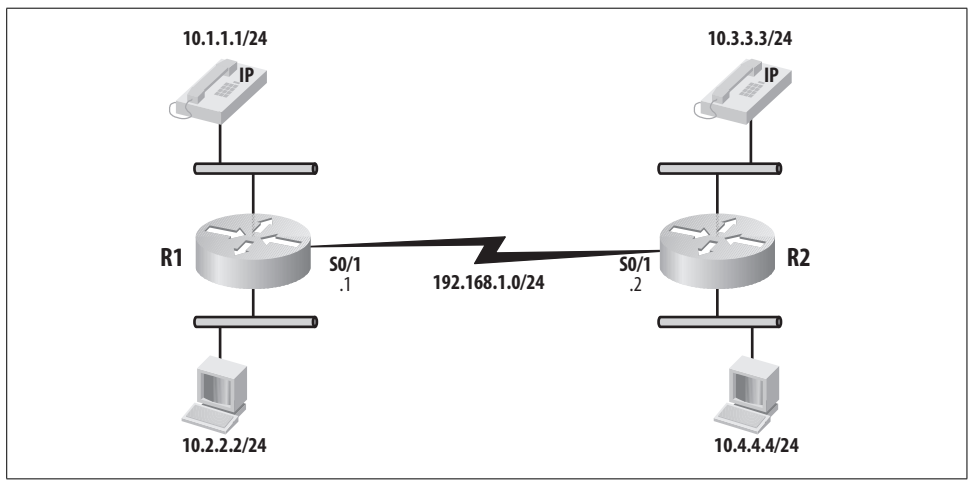


Figure 32-1. Simple converged network

Configuration

The interface we'll be concentrating on is S0/1 on R1. Here is the configuration:

```
interface Serial0/1
 ip address 192.168.1.1 255.255.255.0
 service-policy output WAN
```

The `service-policy` statement maps the policy map named *WAN* to the interface. Here is the configuration for the policy map:

```
policy-map WAN
  class Voice-RTP
    priority 128
  class Voice-Control
   bandwidth percent 5
  class HTTP
   bandwidth percent 10
  class class-default
   fair-queue
```

The policy map references four classes: Voice-RTP, Voice-Control, HTTP, and the special class *class-default*. Remember that *class-default* is a special class name used when building the default class. The default class is where packets not matching any other class are queued. The queues are designed as follows:

Voice-RTP
: Packets in this class will be put into the strict priority queue. The queue will be sized for 128 Kbps. This is roughly the size of two 64 Kbps G711-encoded VoIP calls. Packets with an IP precedence of 5 will be put into this queue. This is defined in the Voice-RTP class map.

Voice-Control
: The voice control for the VoIP class should not consume much bandwidth. I have guaranteed 5 percent of the link's total bandwidth to this class. Packets marked with an IP precedence of 3 will be put into this queue. This is defined in the Voice-Control class map.

HTTP
: I have guaranteed this class 10 percent of the link's total bandwidth. Any traffic with a destination port of 80 will match this class, as referenced by the HTTP class map, and access list 101.

class-default
: This queue is where packets that don't match any other queue will end up. The only definition for the queue is that these packets will be *fair-queued*. They will be treated as they would be on a regular T1 serial link, but only after the other queues have been processed.

The configuration for the class maps is as follows. *class-default* is a default class in IOS that requires no configuration:

```
class-map match-any HTTP
  match access-group 101
class-map match-any Voice-RTP
  match ip precedence 5
class-map match-all Voice-Control
  match ip precedence 3
```

The HTTP class references access list 101 for matching packets. The access list checks for the presence of HTTP packets:

```
access-list 101 permit tcp any any eq www
access-list 101 permit tcp any any eq 443
```

I could have used the statement `match protocol http` in the class map instead of calling an access list. However, this would have matched only port 80. Using an access list allows the flexibility of adding HTTPS (port 443), or HTTP running on nonstandard ports such as 8080.

Monitoring QoS

Monitoring queues is done primarily through a couple of simple commands. The output of these commands contains a lot of information, so I'll explain what you need to look for. First, run the `show interface` command:

```
R1# sho int s0/1
Serial0/1 is up, line protocol is up
  Hardware is PQUICC with Fractional T1 CSU/DSU
  Internet address is 192.168.1.1/24
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
     reliability 255/255, txload 21/255, rxload 22/255
  Encapsulation HDLC, loopback not set
  Keepalive set (10 sec)
  Last input 00:00:00, output 00:00:00, output hang never
  Last clearing of "show interface" counters 5d04h
  Input queue: 2/75/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: weighted fair
  Output queue: 0/1000/64/0 (size/max total/threshold/drops)
     Conversations  0/4/256 (active/max active/max total)
     Reserved Conversations 2/2 (allocated/max allocated)
     Available Bandwidth 1030 kilobits/sec
  5 minute input rate 134000 bits/sec, 53 packets/sec
  5 minute output rate 131000 bits/sec, 53 packets/sec
     22874566 packets input, 2838065420 bytes, 0 no buffer
     Received 52348 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     22964396 packets output, 2678320888 bytes, 0 underruns
     0 output errors, 0 collisions, 0 interface resets
     0 output buffer failures, 0 output buffers swapped out
     0 carrier transitions
     DCD=up  DSR=up  DTR=up  RTS=up  CTS=up
```

The output indicates that the queuing strategy is *weighted fair*. We're using low-latency queuing, so why does `show interface` tell us something different? Low-latency queuing is technically class-based weighted fair queuing (CBWFQ) with a strict priority queue. Because CBWFQ is a type of weighted fair queuing, the router simply reports the queuing strategy as *weighted fair*.

Note that `show interface` does not show the specific queues that we specified in the policy map. To show the statuses of the individual queues, we need to use the `show policy-map interface` *`interface#`* command. This command produces a lot of output, and the more queues you have configured, the longer the output will be:

```
R1# sho policy-map interface s0/1
 Serial0/1

  Service-policy output: WAN

    Class-map: Voice-RTP (match-any)
      19211963 packets, 3918472220 bytes
      5 minute offered rate 81000 bps, drop rate 0 bps
      Match: ip precedence 5
        19211963 packets, 3918472220 bytes
        5 minute rate 81000 bps
      Queueing
        Strict Priority
        Output Queue: Conversation 264
        Bandwidth 128 (kbps) Burst 3200 (Bytes)
        (pkts matched/bytes matched) 19211963/3918472220
        (total drops/bytes drops) 0/0

    Class-map: Voice-Control (match-all)
      793462 packets, 161097064 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: ip precedence 3
      Queueing
        Output Queue: Conversation 265
        Bandwidth 5 (%) Max Threshold 64 (packets)
        (pkts matched/bytes matched) 793462/161097064
        (depth/total drops/no-buffer drops) 0/0/0

    Class-map: HTTP (match-any)
      717928 packets, 42537234 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: access-group 101
        717928 packets, 42537234 bytes
        5 minute rate 0 bps
      Queueing
        Output Queue: Conversation 266
        Bandwidth 10 (%) Max Threshold 64 (packets)
        (pkts matched/bytes matched) 717934/42537620
        (depth/total drops/no-buffer drops) 0/0/0

    Class-map: class-default (match-any)
      2243454 packets, 2851898036 bytes
      5 minute offered rate 52000 bps, drop rate 0 bps
      Match: any
      Queueing
        Flow Based Fair Queueing
        Maximum Number of Hashed Queues 256
        (total queued/total drops/no-buffer drops) 0/0/0
```

Each queue is shown in detail with slightly different information reported. The strict priority queue is used in the class map Voice-RTP. Here is the output for that queue:

```
Class-map: Voice-RTP (match-any)
  19211963 packets, 3918472220 bytes
  5 minute offered rate 81000 bps, drop rate 0 bps
  Match: ip precedence 5
    19211963 packets, 3918472220 bytes
    5 minute rate 81000 bps
  Queueing
    Strict Priority
    Output Queue: Conversation 264
    Bandwidth 128 (kbps) Burst 3200 (Bytes)
    (pkts matched/bytes matched) 19211963/3918472220
    (total drops/bytes drops) 0/0
```

The interesting tidbits are in bold. The first bold line shows the *5 minute offered rate* followed by the *drop rate*. The drop rate should always be zero, especially in the priority queue. If you're dropping packets, your queue is too small. The five-minute offered rate lets us know how much of the queue is being used. This information is also shown later in the output as the *5 minute rate*.

Under the `Queuing` paragraph, we can see that this map is using the strict priority queue. Beneath that is a report of the queue size. Lastly, the number of packets and bytes dropped are reported. This should always be `0/0`. If your traffic spikes beyond your planned levels, however, you may start to drop packets. When dealing with voice calls in the strict priority queue, dropped packets result in distorted or choppy voice quality and unhappy users.

The information for the next queue is a little different. Because this is not the strict priority queue, the queue is described as we've designed it:

```
Class-map: Voice-Control (match-all)
  793462 packets, 161097064 bytes
  5 minute offered rate 0 bps, drop rate 0 bps
  Match: ip precedence 3
  Queueing
    Output Queue: Conversation 265
    Bandwidth 5 (%) Max Threshold 64 (packets)
    (pkts matched/bytes matched) 793462/161097064
    (depth/total drops/no-buffer drops) 0/0/0
```

In this example, the five-minute offered rate is zero. This is call-control, and packets should only hit this queue when calls are set up or torn down. If no calls have been started or completed in the last five minutes, this rate could be valid.

Further down, we can see that 5 percent of the bandwidth has been allocated for this queue. The last line again shows that no packets have been dropped for this queue. The *depth* is the number of packets currently in the queue. On busy links, this number may show fluctuating values. As long as there are no drops, you're in good shape.

Skipping down to the last queue, we get yet another view:

```
Class-map: class-default (match-any)
  2243454 packets, 2851898036 bytes
  5 minute offered rate 52000 bps, drop rate 0 bps
  Match: any
  Queueing
    Flow Based Fair Queueing
    Maximum Number of Hashed Queues 256
    (total queued/total drops/no-buffer drops) 0/0/0
```

The first line in bold should be closely monitored. If all your other queues are behaving properly, this line will show you how much bandwidth has been needed for the default queue. It also shows you the drop rate, which should be zero. If there is one queue where it's permissible to drop packets, it's this one. This is the queue where all your noncritical traffic goes. If you need to drop packets due to congestion, it should be the noncritical traffic that suffers, not the important stuff like voice calls.

Troubleshooting a Converged Network

Dropping packets within specific queues can be hard to diagnose. After you've worked with a converged network for a while, though, you will get a feel for how the queues should behave. While every network is different, here are some common symptoms to look out for.

Incorrect Queue Configuration

When all the queues are configured correctly, none of them should drop packets when the link is congestion-free. When congestion occurs, packets should be dropped from the least important queue first. The least important queue is the class-default queue, where all nonprioritized packets are queued.

Here, I have routed excessive traffic across our sample network's T1 link. While most of the queues are busy, only the class-default queue is showing dropped packets:

```
R1# sho policy-map interface s0/1
 Serial0/1

  Service-policy output: WAN

    Class-map: Voice-RTP (match-any)
      11091627 packets, 2262021588 bytes
      5 minute offered rate 324000 bps, drop rate 0 bps
      Match: ip precedence 5
        11091627 packets, 2262021588 bytes
        5 minute rate 324000 bps
```

```
    Queueing
      Strict Priority
      Output Queue: Conversation 264
      Bandwidth 500 (kbps) Burst 12500 (Bytes)
      (pkts matched/bytes matched) 11091627/2262021588
      (total drops/bytes drops) 0/0

  Class-map: Voice-Control (match-all)
    7312459 packets, 1477128588 bytes
    5 minute offered rate 255000 bps, drop rate 0 bps
    Match: ip precedence 3
    Queueing
      Output Queue: Conversation 265
      Bandwidth 5 (%) Max Threshold 64 (packets)
      (pkts matched/bytes matched) 7312784/1477194238
      (depth/total drops/no-buffer drops) 0/0/0

  Class-map: HTTP (match-any)
    79255 packets, 4690398 bytes
    5 minute offered rate 1000 bps, drop rate 0 bps
    Match: access-group 101
      79255 packets, 4690398 bytes
      5 minute rate 1000 bps
    Queueing
      Output Queue: Conversation 266
      Bandwidth 10 (%) Max Threshold 64 (packets)
      (pkts matched/bytes matched) 78903/4669112
      (depth/total drops/no-buffer drops) 0/0/0

  Class-map: class-default (match-any)
    23750755 packets, 2030244008 bytes
    5 minute offered rate 887000 bps, drop rate 4000 bps
    Match: any
    Queueing
      Flow Based Fair Queueing
      Maximum Number of Hashed Queues 256
      (total queued/total drops/no-buffer drops) 12/16501/0
```

If users are complaining at this point, there are a few things to consider:

- The link may be saturated, in which case it should be replaced with a larger link.
- Some traffic in the class-default queue may need to be prioritized above this queue.
- One or more of the queues may not be configured appropriately. Look at the HTTP queue. While it's performing well, the five-minute offered rate is only 1,000 bps. The queue has been allocated 10 percent of the link. If we make this queue's allocation smaller, the default queue might stop dropping packets. (Actually, given the amount of traffic on the link, in this case this is unlikely. The five-minute offered rate in the class-default queue is 887,000 bps. That's almost two-thirds of a T1 in and of itself!)

Priority Queue Too Small

The rate for the Voice-RTP queue seems stable at about 80 Kbps. For fun, I'll lower the size of the priority queue to 64 Kbps:

```
R1# conf t
R1(config)# policy-map WAN
R1(config-pmap)# class Voice-RTP
R1(config-pmap-c)# priority 64
```

Now, let's take another look at the queues using the `show policy-map interface` *`interface#`* command:

```
R1# sho policy-map interface s0/1
 Serial0/1

  Service-policy output: WAN

    Class-map: Voice-RTP (match-any)
      19682025 packets, 4014346028 bytes
      5 minute offered rate 79000 bps, drop rate 17000 bps
      Match: ip precedence 5
        19682025 packets, 4014346028 bytes
        5 minute rate 79000 bps
      Queueing
        Strict Priority
        Output Queue: Conversation 264
        Bandwidth 64 (kbps) Burst 1600 (Bytes)
        (pkts matched/bytes matched) 19682025/4014346028
        (total drops/bytes drops) 1538/313752
```

Look at the drop statistics. The first line in bold shows a drop rate of 17,000 bps. This is a severe problem indicating that the queue is too small. This line is telling us that the queue offered up 79,000 bps, but had to drop 17,000 bps. To prevent packets from being dropped, the queue should be set to a minimum of 79,000 + 17,000 = 90,000 bps. If you see anything other than zero in the drop rate section of this output, you've got a problem.

The last line of output shows the total drops and bytes dropped. This is a counter, so it will display an incrementing total until the interface is reset, or the counters are cleared. To illustrate this point, I've let the router run with the priority queue set too small for an hour, then set the bandwidth value back to 128 Kbps. I waited five minutes to let the counters settle, and then executed the command again:

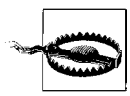
```
R1# sho policy-map interface  s0/1
 Serial0/1

  Service-policy output: WAN

    Class-map: Voice-RTP (match-any)
      20049682 packets, 4089333416 bytes
      5 minute offered rate 79000 bps, drop rate 0 bps
      Match: ip precedence 5
```

```
    20049682 packets, 4089333416 bytes
    5 minute rate 79000 bps
  Queueing
    Strict Priority
    Output Queue: Conversation 264
    Bandwidth 128 (kbps) Burst 3200 (Bytes)
    (pkts matched/bytes matched) 20049682/4089333416
    (total drops/bytes drops) 78648/16044192
```

The drop rate in the first bold line is now zero, as it should be. However, the last line still shows some very high numbers. If I were to see something like this on a normally healthy link, I would suspect that one of two things had happened:

- Someone altered the configuration like I just did, and did not clear the counters after resolving the issue.
- There was a surge of traffic in this queue that exceeded the queue's configured bandwidth.

In either case, you should try to determine what's happened. If there was a surge of traffic in the priority queue, it's possible that either there were more active phone calls than you anticipated, or the active calls were not using the proper codec. Even worse, packets might be getting marked as IP precedence 5 when they shouldn't be.



If the priority queue is saturated, even if there is additional bandwidth available, packets will still be dropped from the priority queue. The priority queue differs from the other queues in that it will not use the remaining available bandwidth on the link for overflow. When this queue is too small, packets will be dropped.

Priority Queue Too Large

When the priority queue is too large, it can affect network performance. However, this problem can be very difficult to diagnose. Here, the priority queue has been set to 256 Kbps, but the traffic hitting the queue is only about 80 Kbps:

```
Class-map: Voice-RTP (match-any)
  3150111 packets, 642496644 bytes
  5 minute offered rate 81000 bps, drop rate 0 bps
  Match: ip precedence 5
    3150111 packets, 642496644 bytes
    5 minute rate 81000 bps
  Queueing
    Strict Priority
    Output Queue: Conversation 264
    Bandwidth 256 (kbps) Burst 6400 (Bytes)
    (pkts matched/bytes matched) 3150111/642496644
    (total drops/bytes drops) 0/0
```

This queue looks healthy, and it is, for all practical purposes. The problem is far subtler than can be determined from the numbers.

The queues are serviced by something called the *scheduler*. The scheduler serves each queue according to its size, and its position in the configuration. The priority queue is always serviced first. Only when the priority queue is empty will other queues be serviced.

I have seen problems on large links such as DS3s when the priority queue is set to a very high level. For example, setting the priority queue near 75 percent of the total link speed causes the queues beneath the priority queue to be starved. In other words, the scheduler spends so much time servicing the priority queue that, even if there is little traffic, the other queues never get serviced. This results in packets being dropped in the lower queues, even if there is plenty of bandwidth to serve them all. Try to keep the priority queue near the level of the expected traffic peaks.

Nonpriority Queue Too Small

Our example is using a T1, and the Voice-Control queue is set to use 5 percent of the link. Five percent of 1.544 Mbps is about 77 Kbps. Here, I've increased the call-control traffic to exceed the queue size. I'm now pushing 90 Kbps through this queue:

```
Class-map: Voice-Control (match-all)
  213781 packets, 40614182 bytes
  5 minute offered rate 90000 bps, drop rate 0 bps
  Match: ip precedence 3
  Queueing
    Output Queue: Conversation 265
    Bandwidth 5 (%) Max Threshold 64 (packets)
    (pkts matched/bytes matched) 213900/40638220
    (depth/total drops/no-buffer drops) 3/0/0
```

Notice that the queue is not dropping packets, even though it is now too small. This is quite different behavior from what we saw in the priority queue. When saturated, the priority queue will drop packets, even if there is additional bandwidth available on the link. Other queues do not drop packets unless the entire link is saturated. The `3/0/0` in the last line shows the queue depth, total drops, and no-buffer drops values. The queue depth indicates that there are three packets queued, but that's a rarity even with the link now running at more than 50 percent utilization:

```
R1# sho int s0/1 | include minute
  5 minute input rate 883000 bits/sec, 830 packets/sec
  5 minute output rate 897000 bits/sec, 846 packets/sec
```

Even when I increase the traffic so this queue is running at over triple capacity, *as long as the link is not saturated*, no packets are dropped:

```
Class-map: Voice-Control (match-all)
  281655 packets, 54324322 bytes
  5 minute offered rate 237000 bps, drop rate 0 bps
  Match: ip precedence 3
  Queueing
    Output Queue: Conversation 265
```

```
    Bandwidth 5 (%) Max Threshold 64 (packets)
    (pkts matched/bytes matched) 281936/54381084
    (depth/total drops/no-buffer drops) 0/0/0
```

With a saturated link, the queue will drop packets when full.

Nonpriority Queue Too Large

As with the priority queue, when a nonpriority queue is too large, the problem will not be obvious. The scheduler may spend too much time servicing the queue unnecessarily, which can result in drops in lower-priority queues, but only after servicing the strict priority queue. Here, the HTTP queue is configured for 40 percent of the link's total speed. With the link being a T1, this translates to more than 600 Kbps. However, the queue is only showing 1 Kbps being offered:

```
Class-map: HTTP (match-any)
  79255 packets, 4690398 bytes
  5 minute offered rate 1000 bps, drop rate 0 bps
  Match: access-group 101
    79255 packets, 4690398 bytes
    5 minute rate 1000 bps
  Queueing
    Output Queue: Conversation 266
    Bandwidth 40 (%) Max Threshold 64 (packets)
    (pkts matched/bytes matched) 78903/4669112
    (depth/total drops/no-buffer drops) 0/0/0

Class-map: class-default (match-any)
  23750755 packets, 2030244008 bytes
  5 minute offered rate 887000 bps, drop rate 4000 bps
  Match: any
  Queueing
    Flow Based Fair Queueing
    Maximum Number of Hashed Queues 256
    (total queued/total drops/no-buffer drops) 12/16501/0
```

Notice that the default queue is dropping packets at a rate of 4,000 bps.

As soon as I change the HTTP queue to be allowed only 5 percent instead of 40 percent, the router almost completely stops dropping packets in the class-default queue:

```
Class-map: HTTP (match-any)
  86935 packets, 5150618 bytes
  5 minute offered rate 1000 bps, drop rate 0 bps
  Match: access-group 101
    86935 packets, 5150618 bytes
    5 minute rate 1000 bps
  Queueing
    Output Queue: Conversation 266
    Bandwidth 5 (%) Max Threshold 64 (packets)
    (pkts matched/bytes matched) 86204/5106232
    (depth/total drops/no-buffer drops) 0/0/0
```

```
Class-map: class-default (match-any)
  26519010 packets, 2404610690 bytes
  5 minute offered rate 849000 bps, drop rate 0 bps
  Match: any
  Queueing
    Flow Based Fair Queueing
    Maximum Number of Hashed Queues 256
    (total queued/total drops/no-buffer drops) 0/32252/0
```

The trick here is that the configuration of one queue directly affected the behavior of another queue—specifically, the one beneath it. In this case, there was nothing I could have done to the class-default queue to increase its performance; the changes had to be made to the queue above it.

Default Queue Too Small

The size of the class-default queue is not directly configurable. If the queue is too small, either the link is too small, or the other queues are too large. In either case, you'll end up dropping packets in the class-default queue.

Default Queue Too Large

I would argue that the default queue cannot be too large. If you have a large class-default queue, you have ample bandwidth and life should be good—for now, anyway!

PART VIII
Designing Networks

This section covers network design in a way that you won't see on certification exams. Instead of focusing on the technical details, I'll begin by walking you through the complete process of designing a network, including example spreadsheets, and formulas to help you get it right. The next chapters cover IP address allocation, IP subnetting, Network Time Protocol (NTP), and device failures. Finally, I'll cover some topics you won't find in other technical books, including how to sell your ideas to management, and when to upgrade.

This section is composed of the following chapters:

Chapter 33, *Designing Networks*
Chapter 34, *IP Design*
Chapter 35, *Network Time Protocol*
Chapter 36, *Failures*
Chapter 37, *GAD's Maxims*
Chapter 38, *Avoiding Frustration*

CHAPTER 33

Designing Networks

There are hundreds of books out there that will tell you how to build a three-tier corporate network. This is not one of them. Instead, I'm going to show you what you need to do *before* you build your network. Then I'll show you some real-world examples of network designs.

This is not the sort of technical information you get from getting certified. This is information that will help you do your job better. For the most part, this chapter is written with the assumption in mind that you'll be designing a network from scratch. While that's not often the case, the information contained herein is applicable to any network project.

Documentation

Documentation is the bane of many an engineer's existence. I'm not entirely sure why this is the case, but an engineer who likes to write documentation seems to be a rarity. Writing is hard work (try writing a book!), but the payoffs are enormous.

Some engineers seem to believe that if they hoard information, they become irreplaceable. Trust me on this one—you and I are replaceable. In fact, I've made a living by coming in to document networks when "irreplaceable" engineers were fired.

Well-written documentation saves time and money. If someone can fix your network by reading your documentation, you've done a good job. If no one but you can fix your network, you're not doing a good job.

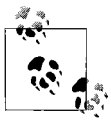
Whenever possible, even on small networks, you should document every detail of your network. Just because a network is small enough for you to memorize every IP address doesn't mean it should not be documented.

Requirements Documents

One of the things you should do at the start of any project, regardless of your perception of available time, is write a *requirements document*. This document should include all the requirements for the project as you understand them, as well as any and all assumptions being made. Even if the requirement is simply "design a new corporate network," write that down, and include all the assumptions that you'll be making to accomplish that goal.

When you've finished your requirements document, send it to everyone who's involved. This list should include your boss, at an absolute minimum. Depending on the corporate culture, you may also wish to include project managers, the VP who is responsible for funding the project (the sponsor), and anyone else involved in the process. Your boss may wish to forward this document to the other parties instead of having you send it to them directly. Either way, you need to publish something that dictates what you will be designing and why.

Writing a requirements document is one of the best things you can do to protect yourself later on when the project scope changes.



The scope will change. I can't think of a single project that I've worked on where it didn't. Scope change is a fact of life with projects.

For example, if you order a T1 for Internet connectivity, and the VP gives you grief because he doesn't think it's enough bandwidth, you should be able to point to the requirements document that you published at the beginning of the project (which hopefully backs up your decision). This takes the heat off of you because you sent out a document describing the needs being met by your design. If he didn't know about the T1, he didn't read the document.

A requirements document does not need to be large or complicated—in fact, the simpler the document is, the better chance it has of being understood. All points should be made as simply as possible (especially assumptions). Here's an example requirements document:

Requirement:

- The network must support 300 users.

Assumptions:

- Each user will have one workstation.
- Each workstation will have only one Ethernet interface.
- All interfaces will be cabled for 1 Gpbs Ethernet.
- The network does not need to support 1 Gbps for all users at the same time.
- Each user will have one phone supporting one phone line.

- All phones will be non-IP phones.
- VoIP will not be run on the network for the foreseeable future.
- Each user will equal one cube or office.
- Each cube or office will have two data jacks and one phone jack.
- All cabling runs will terminate into the computer room (home runs).

When a VP says, "We need a network to support 300 users," he doesn't always understand the implications of this simple statement. Your job is partially to make sure that the other people involved have all the information they need to understand what's happening, and partially to cover your own butt should someone fail to understand what you're doing. If you publish all your assumptions, the VP should be able to read your document and pose any questions he may have. Perhaps there's a plan in the works to move the phone system to VoIP in the next year, and you need to be advised to order equipment that will support that vision.

I've seen many projects fail because no requirements document was written. I've also witnessed (and been party to) many heated arguments about what was said months ago, but not put into writing.

Just because people say they understand something, don't assume they do—especially if they are not "technical" people. I once designed an elaborate Internet failover mechanism for a company. When the DS3s were being delivered (after a 60-day lead time), the VP who had signed the order for them started screaming at me because he didn't understand why we needed them. I'd never realized that he didn't understand the design. I had assumed we were all on the same page, but because I had not written a requirements document, there was literally no page for us to be on.

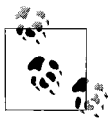
Port Layout Spreadsheets

The first step I take when designing any network is to compile a list of all the devices that will be accessing the network. In the case of LANs, here are some things to think about:

- How many users will the network need to support?
- How many servers will the network need to support?
- How many printers will be attached to the network, and what will their locations be?
- What are the applications running on the network? How will users interact with these applications (HTTP, client software, terminals, Citrix)?
- What type of security do you need?
- Is high availability desired and/or affordable?
- What percentage of growth should we assume?
- Do all interfaces need to be gigabit?

- Will the network need to support VoIP?
- Will you be supporting one physical location, or many (including multiple floors in a single building)?

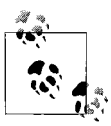
Your goal is to come up with a target number of each interface type for each location. Once you have these numbers in mind, you can decide what sort of equipment you need to order. Gigabit Ethernet switches come in multiples of 48 at the highest port densities currently available. To determine how many 48-port switches or modules you need, divide the total number of gigabit interfaces required by 48. For example, if you need 340 gigabit interfaces in one location, you'll need 340 / 48 = 7.08 (in other words, eight modules).



When figuring out how many ports you need, don't forget servers that offer high availability. Many servers today will allow multiple Ethernet interfaces to be connected to one of two switches in a failover pair. Talk with your systems people, and find out what they're doing, so you don't come up short.

You should also plan for a certain amount of growth. A good rule of thumb for capacity planning is to plan for a minimum of 15 percent growth: 340 * .15 = 51. Add this to the number of interfaces currently required, then divide this number by 48 to see how many modules you'll need if you want to allow for growth: 340 + 51 = 411, and 411 / 48 = 8.56. This means you'll need nine modules, which will provide a total of 432 ports. Nine is an odd number, which means one switch will have more modules than the other. This may or may not matter in your environment, but I like to have each side match exactly. It's always better to have too many ports, rather than too few, so long as the budget allows it. Rounding out the modules to 5 on each chassis brings you to 10 modules, totaling 480 ports.

Now you have 480 ports available, and a present requirement for 340. This allows for a growth factor of almost 30 percent. Chances are you've missed something (I always do), so having room for more than 15 percent growth is not a bad thing. In cases like this, I make sure to reserve extra ports for expansion purposes. They always come in handy later.



Another requirement that engineers often forget is the need for inter-switch trunks. Switches in failover pairs need to be connected to each other, usually with multigigabit links. If you're using Firewall Services Modules, they should have dedicated trunks. Content Service Modules should also have their own trunks, as should RSPAN, if you'll be using it. If you assume a 2 Gbps EtherChannel for each of these trunks, you've just allocated 16 ports (8 on each side).

I've worked on high-bandwidth networks where each of these trunks required 4 Gbps EtherChannels. A design like this would require 32 ports (16 on each side). That's almost an entire 48-port module just for inter-switch communication!

Once you've determined how many ports you'll require, map out the devices you'll need, and figure out what devices will serve what purposes. Here, I've decided that 6509 switches will be used in the core, and 2811 routers will be used for Internet and backend connectivity to the corporate web site, located in a collocation facility. Figure 33-1 shows my spreadsheet for this information.

					Interfaces			
Name	**Device**	**Function**	**Location**	**Slots**	**T1**	**DS3**	**1G**	**10G**
Core-1	6509-E	Core switch/router	Rack #2	9	0	0	240	0
Core-2	6509-E	Core switch/router	Rack #3	9	0	0	240	0
Internet	2811	Internet router	Rack #2	1	2	0	2	0
HQ-Colo	2811	Colo connection router	Rack #3	1	2	0	2	0

Figure 33-1. Sample equipment list

When you know what equipment you'll be using, you should break down each device so that you know exactly what hardware to order. This step will help you immensely throughout the process of designing and building the network. When you get the equipment, you can expand the spreadsheet to include the actual serial numbers of each part. Figure 33-2 shows a sample spreadsheet for planning one of the 6509s I've chosen. Notice that I've included not only the modules, but the extra memory as well.

				Interfaces		
Name	**Device**	**Function**	**Slots**	**10/100/1G**	**GBIC**	**SF-GBIC**
Core-1	**WS-C6509-E**	Core Switch/Router	**9**	**240**	**0**	**4**
Slot1	WS-X6748-GE-TX	48 Port 10/100/1000 FE Blade		48		
Slot2	WS-X6748-GE-TX	48 Port 10/100/1000 FE Blade		48		
Slot3	WS-X6748-GE-TX	48 Port 10/100/1000 FE Blade		48		
Slot4	WS-X6748-GE-TX	48 Port 10/100/1000 FE Blade		48		
Slot5	WS-SUP720-3B=	Supervisor - 720 Fabric Enabled				2
Slot6	WS-SUP720-3B=	Supervisor - 720 Fabric Enabled				2
Slot7	WS-SVC-FWM-1-K9=	Firewall Switch Module				
Slot8						
Slot9	WS-X6748-GE-TX	48 Port 10/100/1000 FE Blade		48		
MSFC Mem	MEM-MSFC2-512MB	MSFC - 512M DRAM				
Sup Mem	MEM-S2-512MB	Sup - 512M DRAM				
Flash Mem	MEM-C6K-CPTFL256M	256M Compact Flash Upgrade				
Fan Tray	WS-C6509-E-FAN	Fan Tray				
PS #1	WS-CAC-3000W-US	3000W Power Supply				
PS #2	WS-CAC-3000W-US/2	3000W Power Supply				

Figure 33-2. Core switch hardware detail

Once you have the hardware figured out, it's often up to management to get the equipment ordered. You'll probably need to help with the ordering process by providing lists of part numbers and quantities. Once the order gets submitted, you may have to wait several weeks for the equipment to arrive.

Now, you need to map every interface on every device. This may sound excessive to some, but planning where every device will connect will save you aggravation later. Having a plan like this can also let you hand off the work of connecting the devices to someone else. This step lets you work on a large chunk of the final documentation before the equipment arrives.

I create a spreadsheet for every device, and, sometimes, for every module in a device. It doesn't need to be anything elaborate. A simple list of the ports, the devices connected, and maybe the IP or VLAN to be configured (such as the list in Figure 33-3) suffices.

Physical	VLAN/Trunk/IP	Device	Remote Interface
1/1	VLAN 10	Internet Router	F0/0
1/2		Reserved	
1/3			
1/4			
1/5	VLAN 777	Core-2 FWSM Failover link-1	G1/5
1/6	VLAN 777	Core-2 FWSM Failover link-2	G1/6

Figure 33-3. Sample port allocation spreadsheet

If you have planned your network to this level, when the equipment arrives, all you have to do is install it in the racks, and cable together the devices according to the plan.

IP and VLAN Spreadsheets

Along with the physical planning of equipment and port allocation, you'll need to plan the IP network and VLAN layouts. I like to make pretty detailed spreadsheets of this information, just as I did for the physical devices, modules, and ports.

In Figure 33-4, I've allocated my IP networks. I'm allocating a /23 network for each VLAN, and reserving the /23 beyond each allocation for future expansion.

I create a spreadsheet for each IP network I'll be using, and populate it with specific information regarding every device on the network. This is an excellent exercise that will force you to once again think about every device you'll be connecting. Figure 33-5 shows a sample IP address layout spreadsheet.

Network	Mask	VLAN	Description
10.1.0.0			
10.1.1.0			
10.1.2.0			
10.1.3.0			
10.1.4.0			
10.1.5.0			
10.1.6.0			
10.1.7.0			
10.1.8.0	255.255.254.0	VLAN 10	Internet DMZ
10.1.9.0			
10.1.10.0			*reserved for expansion*
10.1.11.0			*reserved for expansion*
10.1.12.0	255.255.254.0	VLAN 100	VLAN 100
10.1.13.0			
10.1.14.0			*reserved for expansion*
10.1.15.0			*reserved for expansion*

Figure 33-4. IP network layout sheet

Navigation: Master IP spreadsheet

IP address	Subnet mask	VLAN	Description
10.1.32.0			HQ Networks
10.1.32.1	255.255.254.0	VLAN 130	Default Gateway (HSRP VIP)
10.1.32.2	255.255.254.0	VLAN 130	Core-1 VLAN 130 IP
10.1.32.3	255.255.254.0	VLAN 130	Core-2 VLAN 130 IP
10.1.32.4	255.255.254.0	VLAN 130	*Reserved*
10.1.32.5	255.255.254.0	VLAN 130	*Reserved*
10.1.32.6	255.255.254.0	VLAN 130	*Reserved*
10.1.32.7	255.255.254.0	VLAN 130	*Reserved*
10.1.32.8	255.255.254.0	VLAN 130	Color Printer in Charlie's office
10.1.32.9	255.255.254.0	VLAN 130	Color Printer in Lucy's office
10.1.32.10	255.255.254.0	VLAN 130	Color Copier #1
10.1.32.11	255.255.254.0	VLAN 130	Color Copier #2

Figure 33-5. IP address layout sheet

Once you have all this information documented, building the configurations should be a snap. As an added bonus, the spreadsheets are excellent documents for the network. They can be printed and put in binders, or simply stored somewhere for easy access.

Once the network is built, I suggest printing out a copy of all the documentation you've made. This way, when something changes, you have documented evidence of the condition of the network at the time of its implementation.

Bay Face Layouts

If you're unfamiliar with the term, *bay face layouts* are diagrams showing how each rack will be built, including every detail. For me, the biggest benefit of bay face layouts is that once I've created them, I can have someone else install the equipment, if need be: I can turn over all the boxes, and a copy of the bay face layouts and say, "Make it look like this." A bay face layout example is shown in Figure 33-6.

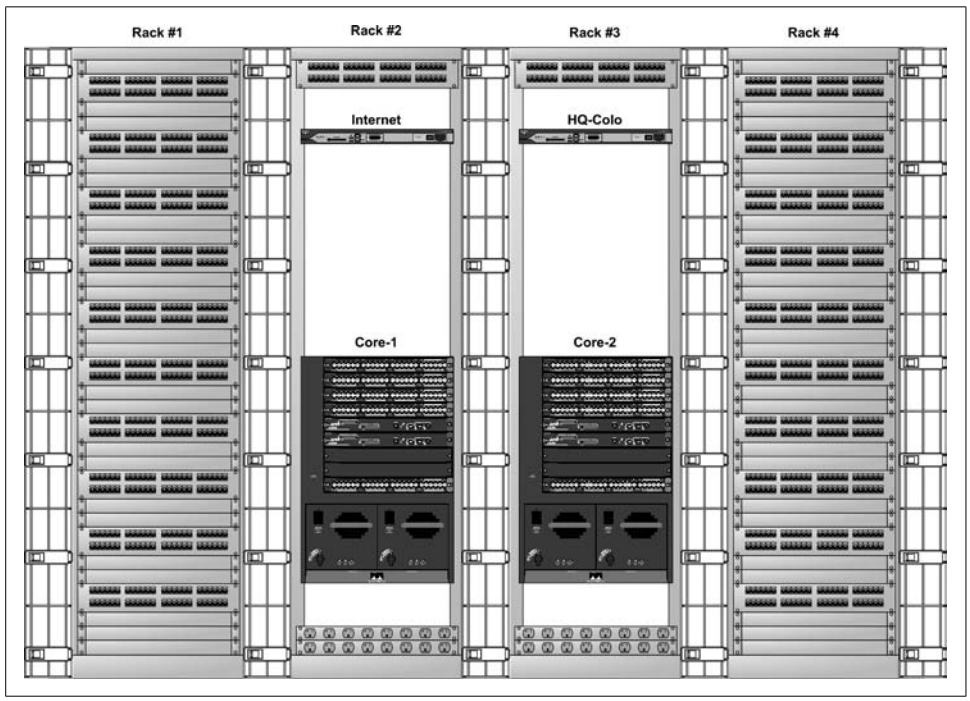


Figure 33-6. Bay face layout

There are three things that many engineers forget when designing or planning rack space: power, cabling, and patch panels. All of these items take up space in your racks.

Getting your rack space requirements wrong can be a costly mistake, especially if you're renting collocation space. Bay face layouts are an excellent sanity check to ensure that you have enough racks to support all the equipment you've bought.

Power and Cooling Requirements

This is a good time to work out your power requirements. Most vendors list the power consumption of their equipment on their web pages. Simply find out what kind of power the devices require (AC/DC, voltage, and amperage), and the connectors needed, and add up the requirements for each rack. Talk with whoever is responsible for the environment where your racks are located, and make sure that they can support the equipment you're putting into your racks.

Just because you can fit 40 1-RU servers in a rack doesn't mean you can power them! The default AC power installed in a rack in a collocation facility is often two 20-amp AC power strips. If your 1-RU servers each draw 2 amps, and have two power supplies, you can only install 10 in each rack. If you want to add more, you'll need to order extra power.

Don't forget cooling requirements, either. Along with power specifications, vendors will list the British Thermal Unit (BTU) values for their devices. The person responsible for the environment where the equipment will be installed will also need to know this information. Even though you can physically install two 6509s in a rack, the required air conditioning systems may not be in place to keep them cool.

To find power and heat values for 6500 switches, search Cisco's site for "6500 power and heat numbers." Because the power draw and heat output vary depending on the modules installed, you must figure out this information for each installation. The numbers for the 6509 I specified earlier are shown in Figure 33-7. A similar spreadsheet should be compiled for every device you will be installing.

Device	Slot	Module	Part number	Watts	BTU
Core-1 6509E	**Slot 1**	48 Port 10/100/1000	WS-X6748-GE-TX	367.50	1,255.01
	Slot 2	48 Port 10/100/1000	WS-X6748-GE-TX	367.50	1,255.01
	Slot 3	48 Port 10/100/1000	WS-X6748-GE-TX	367.50	1,255.01
	Slot 4	48 Port 10/100/1000	WS-X6748-GE-TX	367.50	1,255.01
	Slot 5	Sup-720	WS-SUP720-3B	350.80	1,204.81
	Slot 6	Sup-720	WS-SUP720-3B	350.80	1,204.81
	Slot 7	FWSM	WS-SVC-FWM-1-K9	214.73	733.29
	Slot 8				
	Slot 9	48 Port 10/100/1000	WS-X6748-GE-TX	367.50	1,255.01
	Fan Tray		WS-C6509-E-FAN	188.00	642.00
		Total		**2,941.83**	**10,059.96**

Power based on AC Power Supplies

Figure 33-7. Power and BTU values for a 6509E

Remember that there will be limitations regarding how many devices can be placed in a rack. To illustrate this point, I've built a rack full of 1-RU servers. The specs and rack layout are shown in Figure 33-8. Assume that in the collocation facility where this rack is located, we have been given a limitation of 20,000 BTU per rack. The AC power per rack is limited to two 20-amp PDUs, each containing eight outlets.

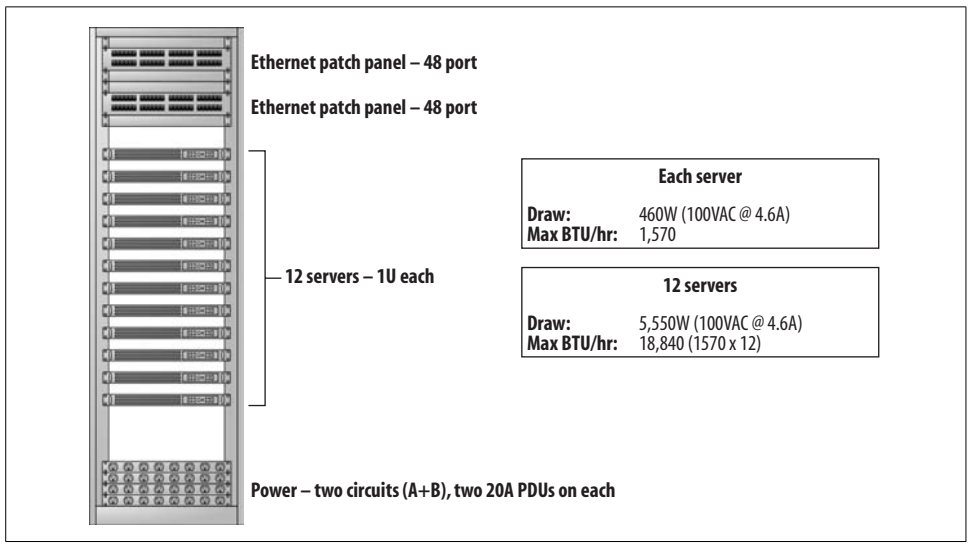


Figure 33-8. Figuring out how many servers can fit in a rack

When you first look into 1-RU servers, they may seem very appealing because in theory you can cram 30 or 40 of them into a rack. However, that's not usually the case in the real world. In my example, each server produces a maximum of 1,570 BTU per hour. (You should always plan based on the maximum numbers provided.) Some simple math shows that 12 servers will potentially produce 18,840 BTU/hr. Based on heat output alone, therefore, we cannot have more than 12 of these servers in a single rack.

Looking at power, more limitations become apparent. Because each server draws 460 watts (100 VAC @ 4.6 amps), we should really only plug four or five servers in each PDU (not six). The easy way to figure out power limits is to use the amperage numbers. If each PDU can only support 20 amps, and each server will use 4.6 amps, you can only fit 4.34 servers in each PDU. Look carefully, though—the specs indicate that the server draws 460 watts when using 100-volt AC power at 4.6 amps. In the U.S., our power is usually 120 VAC, not 100 VAC, so these numbers will change a bit. What happens if we change the voltage and keep the watts the same? If W = A * V, then A = W / V. So, 460 watts divided by 120 volts = 3.83 amps. Five servers using 3.83 amps each will require a total of 19.15 amps.

Of course, this assumes a full load on the electrical circuits. Many collocation facilities will not allow you to load your circuits over a set threshold (typically, 70

percent). If we can only load our circuits to 70 percent, we can only put three or maybe four servers on each PDU. Thus, even though we only have 12 servers, if each PDU is only capable of 20 amps, we'll still need four PDUs.

Because we can only fit 12 servers in a rack (assuming four PDUs), it makes sense to space them out. Working on 1-RU servers can be difficult with all the connections they may have. As you'll see, having six Ethernet interfaces in each server is not unusual. Don't forget that each server will also have console cables, power, and the wire management necessary to support it all. Additionally, if you leave a large empty space in the rack, executives will expect you to fill it before they allow you to order additional racks.

Tips for Network Diagrams

Engineers seem to like to make their documents so complicated that even they have trouble reading them. Here are some tips to help you produce documentation that will get you noticed:

Keep it simple
: Take a look at any of the drawings in this book. Each one of them is designed to convey a single idea. The more you try to include in a drawing, the harder it will be to understand.

Separate physical and logical ideas
: Physical connectivity is very important, but try to keep it separate from VLANs, routing, and other logical subjects. I like to make two drawings: one for the physical ports, and another with the VLANs and IP addresses. With the use of SVIs, this becomes even more important.

Don't cross lines
: Every time you cross a line in a drawing, the drawing gets harder to read. Sometimes, it's unavoidable, but try to keep crossed lines to a minimum.

Orient your straight lines
: If you put together a drawing in which the straight lines are slightly off the horizontal or vertical, the drawing will look like the etchings of a serial killer. When you take the time to orient all the lines, the difference is dramatic. Similarly, lines drawn at an angle should all be at the same angle where possible.

Delineate when you can
: If there are two locations in your drawing, separate them somehow. Put each one in a rectangle (rounded rectangles are preferred by many people). Using colors, or even shades of gray can help as well.

Line up your icons
: If you have a row of icons in your drawing, take the time to line them all up along a single axis.

Naming Conventions for Devices

Hostnames should be constructed so that anyone with a rudimentary knowledge of the network can determine the devices' functions. However, there seems to be a tendency among IT people to give their devices the most incomprehensible names possible. For example, here's a real hostname from my time in the field (I've changed the company name and details to protect the guilty): gadnslax1mai750901. What the heck does that mean? Is it relevant? Can I tell what the system is from the hostname? More importantly, can you?

In a training session for the network containing these wacky hostnames, one of the students asked what the hostnames meant. No one could answer without looking up the document describing the hostname layout. Hostnames should not require research! Here's the breakdown of the offending name:

gadnslax1mai750901

gad – The name of the company (GAD Technology)

ns – Network services

lax – Los Angeles

1 – It's the first, um, thing in Los Angeles

mai – Main Street

7509 – The device is a Cisco 7509

01 – It's the first 7509 in this location

The purpose of a hostname is to identify a device. Hostnames should be easy to remember. When a hostname is harder to remember than an IP address, using a hostname is counterproductive.

When a hostname is coupled with a domain name to make a *fully qualified domain name* (FQDN), the resulting string should be obvious and simple. *sw1.gad.net* is a simple and obvious FQDN that describes the first switch in the *gad* network provider domain.

I like hostnames that describe one thing—the function of the device. *WAN-Router* is an excellent hostname. For companies that have multiple locations, adding the location to the hostname (e.g., *LAX-WAN-Router*) is useful. Equipment is usually deployed in pairs, in which case it's also a good idea to number the device (e.g., *LAX-WAN-Router-1*). In my opinion, though, that name is too long. The fact that the device is a router is irrelevant to its function; *LAX-WAN-1* will suffice. If you want to use DNS hierarchies, *wan-1.lax.domain.com* works nicely as well.

Every company has unique needs. If you work at an ISP, you may have devices at the premises of many customers. In this case, including the customer name is beneficial (e.g., *GAD-WAN-1*). You should resist the urge to document your network with the

hostname. You don't need to include the serial number of the device in the hostname, either (yes, I've really seen this). Of course, everyone has their own opinions on hostnames, but my advice is to keep them as simple as possible. That being said, Lauren is not a good name for a router either. It's a pretty name, but it doesn't give any indication of the router's function!

Once I have my hostnames figured out, I prepend interface names to the hostnames for DNS. For example, Serial 0/0/0:1 on *LAX-WAN-1* would have the DNS hostname of *s0-0-0-1-lax-wan-1* (because DNS can only use hyphens, I replace each slash and colon with a hyphen). This makes traceroutes very readable because each node and IP interface within the node is clearly labeled:

```
[gad]$ traceroute switch9.mydomain.com
traceroute to switch9.mydomain.com (10.10.10.10), 30 hops max, 40 byte packets
1  s0-0.router1.mydomain.com 9.854 ms  10.978 ms  11.368 ms
2  f0-1.switch2.mydomain.com 2.340 ms  1.475 ms  1.138 ms
3  g0-0-12.switch9.mydomain.com 1.844 ms  1.430 ms  1.833 ms
```

Network Designs

I can't tell you how you should design your network. I can show you some of the more common corporate and e-commerce network designs out there, and explain why I've chosen to focus on these designs.

Corporate Networks

Most networks are designed along the lines of the classic *three-tier model*. The three-tier model has *core*, *distribution*, and *access* levels. These levels are clearly delineated and served by different devices. Traditionally, routing was the slowest, most expensive process. For these reasons, routing was done in the core. All the other levels were usually switched.

With the advent of inexpensive layer-3 switching, the three-tiered model is now often collapsed for corporate networks. We'll look at the traditional model as it might be used today, as well as a couple of collapsed-core models.

Three-tiered architecture

The three-tiered architecture that is most commonly seen in textbooks is still widely used in the industry. Physical separation of the three levels usually occurs when there is a physical need to do so. An excellent example would be a college or business campus: there might be core switches (possibly in a central location), distribution switches in each building, and access switches close to the users in each building.

The specifics would depend on the physical layout of the campus. Figure 33-9 shows a textbook three-tiered corporate network.

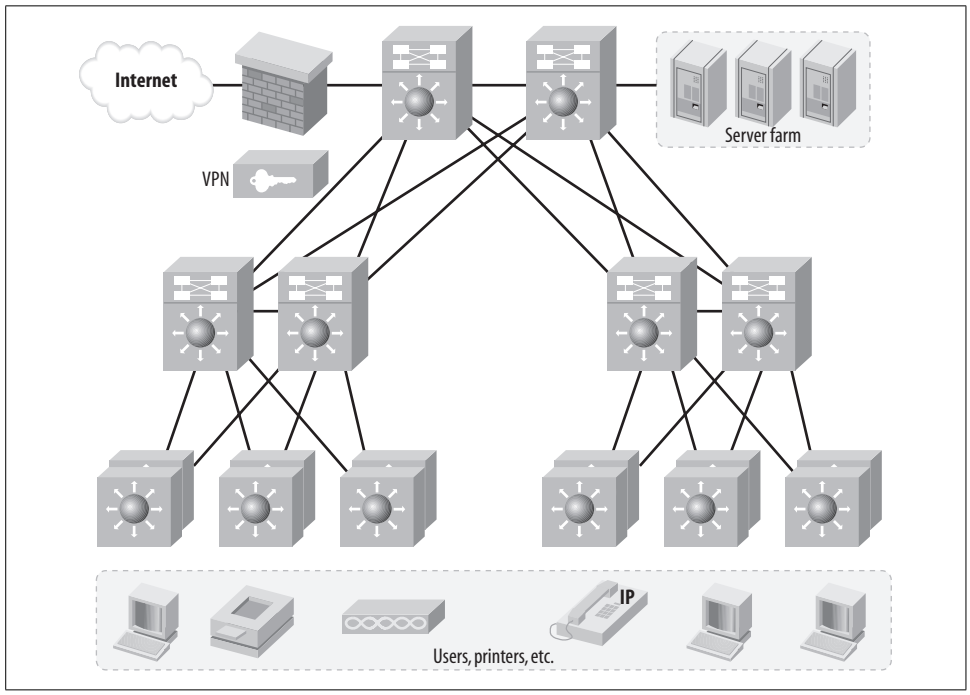


Figure 33-9. Typical three-tiered corporate network

At the bottom of the drawing, users, printers, IP phones, and wireless access points (APs) are connected to the access-layer switches. The access-layer switches are connected to the distribution-layer switches in the middle row. The distribution switches connect to the top two switches, which are the core of the network. The Internet and the server farm are both connected to the core in this example.

Collapsed core—no distribution

Collapsed-core networks are very common. I work a lot in Manhattan, where skyscrapers are the norm. Office space in skyscrapers is usually allocated on a floor-by-floor basis. Wiring is usually run from points on the floor to a central location on the same floor. Floors are usually interconnected with conduits that run between floors. When a company occupies multiple floors, the wiring of each floor lends itself to a collapsed-core design because there are limited wiring possibilities between floors.

Because the amount of space on each floor is limited, there is typically little need for more than two physical network layers. With the core switches on one floor, and access switches on the remaining floors, the access switches can act as distribution-layer switches as well. Port density is usually not an issue, as each floor does not

Server Farms—Where Do They Belong?

Do servers belong near the users that they serve, or in a farm near the core? Where servers belong in your network depends on how your network is designed, and often on the type of server in question. Some servers should be in the core. Centrally locating email servers, for instance, makes sense. Other servers should be closer to the users they serve. For example, in a campus network, it would not make sense to put accounting servers in the core if the entire accounting department were located in one building. Then again, there may not be space in each building for servers.

A lot depends on the layout of your network, too. Many companies these days are building completely flat networks, with the core/distribution/access model being completely collapsed into a single pair of large switches, such as Cisco 6509s. In this case, everything is connected to the same switches, making the argument moot.

occupy much physical space. From a logical standpoint, the distribution layer may be collapsed into the core as well. You may not even need a logical distribution layer at all. Again, each environment is different.

Another scenario I've seen that lends itself to this type of design is the segmented building. One project I worked on involved redesigning the network for a major stadium. The stadium was divided into four segments, each of which had its own infrastructure room (closet). The users' cables were run to these closets, and the closets were linked together with fiber.

Figure 33-10 shows an example of a collapsed-core network without a distribution layer.

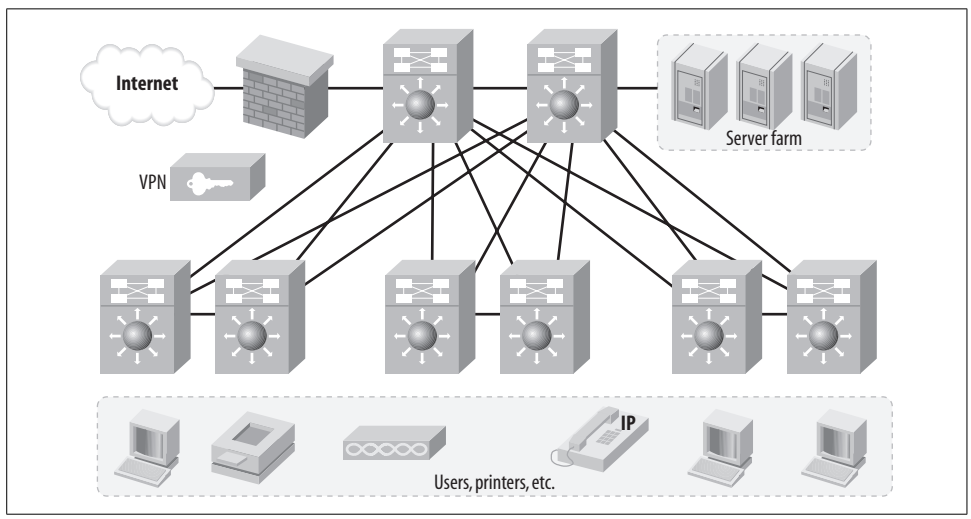


Figure 33-10. Collapsed-core network without a distribution layer

Collapsed core—no distribution or access

A very popular design in companies that are contained within a single building is the collapsed network, where there are only core switches. A company with hundreds of employees that are all in a single location can often manage to have its computer room central to the environment. As long as the Ethernet runs can remain within the distance limitations of the cabling in use, all runs can be home runs back to the core switches. A pair of high-density, high-availability switches like Cisco 6509s can support hundreds of users. Such a network design is shown in Figure 33-11.

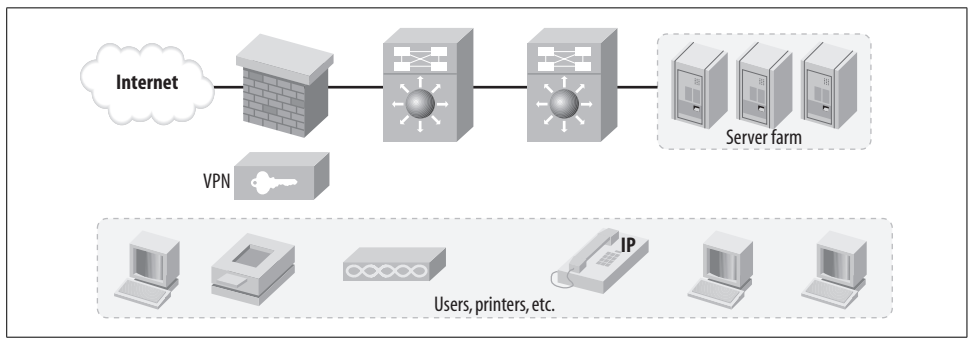


Figure 33-11. Collapsed-core network consisting of only one layer

Configuration concerns

I'm not going to go through the detailed steps of configuring a corporate network here. Configuration details for the various devices are provided elsewhere in this book.

Instead, I'll point out some elements you should consider when designing such a network, and try to point you in the right direction. I will be using a collapsed-core model with no distribution layer as an example. You can use Figure 33-10 as a reference.

Trunks. Trunks may be necessary anywhere switches are interconnected. The obvious points are between the two core switches and the links connecting the access switches to the core. Don't forget the links between each pair of access switches, either.

You may choose to design your network such that the links between switches are IP links and not trunks. This is perfectly valid, and can make the network easier to understand for many people. If you do choose a layer-3 network design, you will need fewer trunks.

EtherChannels. I like to link my switches together with a minimum of two connections bound together by an EtherChannel. This provides a bit of resiliency, and increases the bandwidth between the switches. Depending on your network, you may want to increase the size of the bundles to three, four, or even more.

You may have servers that require EtherChannels as well. Check with your server group to see if this feature is required, and, if so, how it should be configured.

If you will have in your core switches modules such as FWSMs that require dedicated links between failover pairs, you may need to design EtherChannels for these as well. For more information on EtherChannels, refer back to Chapter 7.

Spanning Tree. Determine which ports will be user or server ports, and configure them for `spanning-tree portfast`. Configure one of the core switches to be the spanning-tree root bridge, and the other core switch to be the secondary root bridge. See Chapter 8 for additional details.

VTP. I'm not a big fan of VTP, especially for small networks, but even if you don't use it, you'll need to configure a VTP domain name. Trunk negotiation requires the VTP domain to be set properly. See the Chapter 5 for more details.

VLANs. How many VLANs will you need? Make sure you plan them all out ahead of time. Here's a list I came up with just by looking at Figure 33-10:

- Internet
- Internet inside
- Server farm
- User VLANs

Planning ahead of time will save you from having to make last-minute decisions and wasting time.

E-Commerce Web Sites

An e-commerce web site is one on which goods are sold and money is exchanged. E-commerce web sites involve some concerns that don't exist with simple web sites. The biggest challenge is security. If you throw together a Linux server, and host a simple web site on it, you only have to worry about security for the web server. With an e-commerce web site, you will probably have multiple web servers. Each of those web servers will probably need to access a database, which should not be accessible from the Internet. In fact, the database shouldn't even be directly accessible from the web servers! Instead, there should be a layer of servers that process requests to the database on the web servers' behalf. This layer of servers is called the *application layer*. Now, you have to worry about security for web servers, application servers, and database servers. Some or all of these servers may need to be load-balanced, and all of them will need to be managed. Management can be an interesting challenge because as I've said, database servers should not be accessible from the Internet.

The database servers will contain customer information, and possibly, credit card data. This information must be protected. Keeping the servers away from the Internet is one of the best ways to accomplish the required protection.

The standard design for an e-commerce web site is composed of three tiers. The first tier contains the web servers, which are reachable from the Internet. This is called the Internet layer. The second tier contains the application servers, which cannot usually be reached from the Internet. These servers can talk to the web servers above them, and the database servers beneath them. The lowest layer is the database layer, which can be accessed only from the application layer.

There are many ways to design an e-commerce web site. The network design will be determined, in large part, by the software in use, and the developer's methods. I've worked on e-commerce web sites that had only two layers, and others with four. Sometimes, developers will insist on a single layer. If the application simply does not support multiple layers, forcing the issue will only waste time. Still, when dealing with servers that store credit card data, make sure your design adheres to Payment Card Industry (PCI) standards for security.

The Internet layer is the least secure layer because it is accessible from the Internet. Even though this layer is protected by a firewall, the general population has access to services residing in this layer. The most secure layer is the database layer. The only way to access the database layer is through the application servers. These servers will have special applications residing on them that can process and possibly cache database information for the web servers.

The three-tier e-commerce architecture is shown in Figure 33-12. The Internet layer contains the most servers. The number of servers generally decreases in the lower layers, though this is not a universal truth. For example, the database layer may be composed of an Oracle cluster, which could use many smaller servers.

The servers at each level connect with servers in the adjacent level or levels. For example, application servers have interfaces in the database layer and the Internet layer. All servers must have multiple interfaces. In the case of high-availability networks, the servers may need four interfaces: two for the level above (one in each switch), and two for the level below.

There are two generally accepted methods for accomplishing this layering. I'll call them *bridging* and *routing*. In the bridging method, the lower interfaces of the upper layer are connected to the same VLAN as the upper interfaces of the layer beneath them. Figure 33-13 shows an example of a bridged e-commerce design. The advantages of

this design include simplicity and speed. There are only three VLANs, and there are no routers or firewalls separating the layers. The disadvantage is decreased security, as there are no devices separating the servers on one layer from the servers on the next.

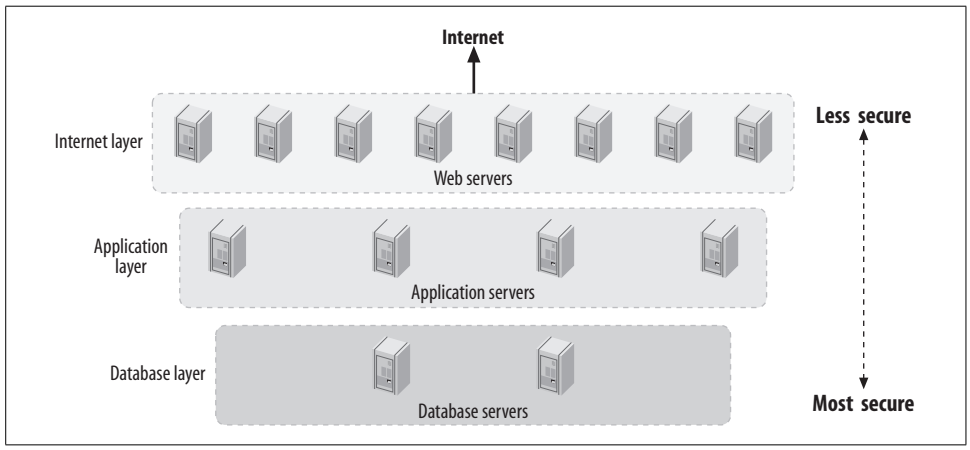


Figure 33-12. Typical three-tier e-commerce web site

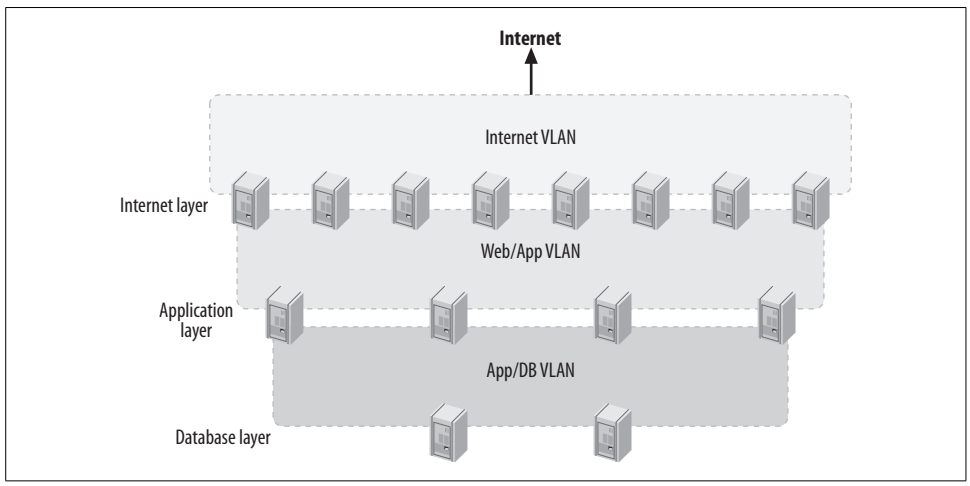


Figure 33-13. Bridged three-tier e-commerce design

The servers in a bridged design only need to talk to the networks directly connected to them. The web servers are the only servers that require a default gateway in this design.

The more secure alternative to the bridged design is the routed design. Routing between the layers allows firewalls to be placed between the layers. Figure 33-14 shows a typical routed three-tier e-commerce design.

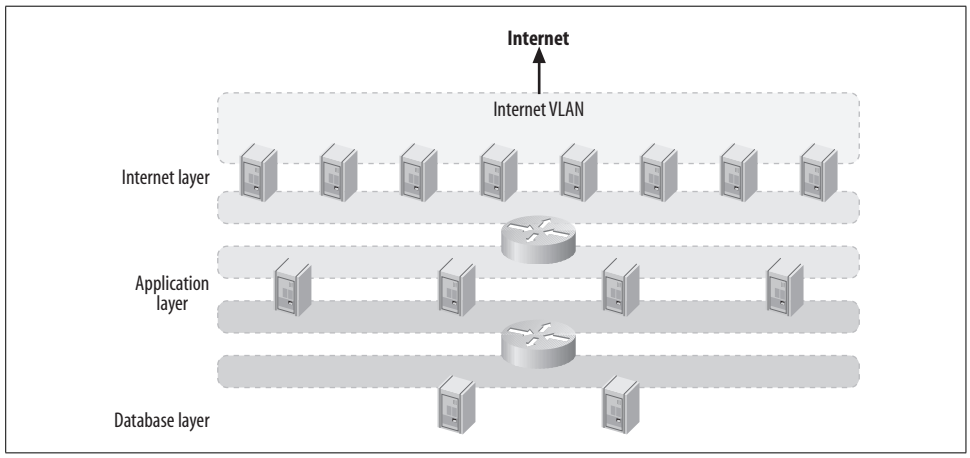


Figure 33-14. Routed three-tier e-commerce design

The routed design is more complicated because there are almost twice as many VLANs, each of which must have an SVI or router interface associated with it. Because there are physical or logical routers separating the VLANs, the servers on each level must have default gateways or route statements configured. The benefit of a routed design is security. Because all traffic must pass through a router, the router can be a firewall.

The problem with both of these designs is that there is no easy way to manage the servers remotely. If you have easy access to the servers, having a Keyboard Video Monitor (KVM) switch in each rack can solve this problem. However, e-commerce web sites are typically hosted in separate collocation facilities. In this case, remote management capabilities are vital.

Remote management is often accomplished with another VLAN that connects to every server and network device at the site. This management VLAN then connects to a router or firewall that allows connectivity to the main site. Figure 33-15 shows such a design.

You've gone through great pains to make sure the servers are secure. You've separated all of the servers into layers, and prohibited the web servers from communicating with the database servers. Now, you've added another VLAN that connects them all together. If you're thinking that this looks like a bad idea, you're right. The benefits of a management network do outweigh the risks, but only if you design the network properly.

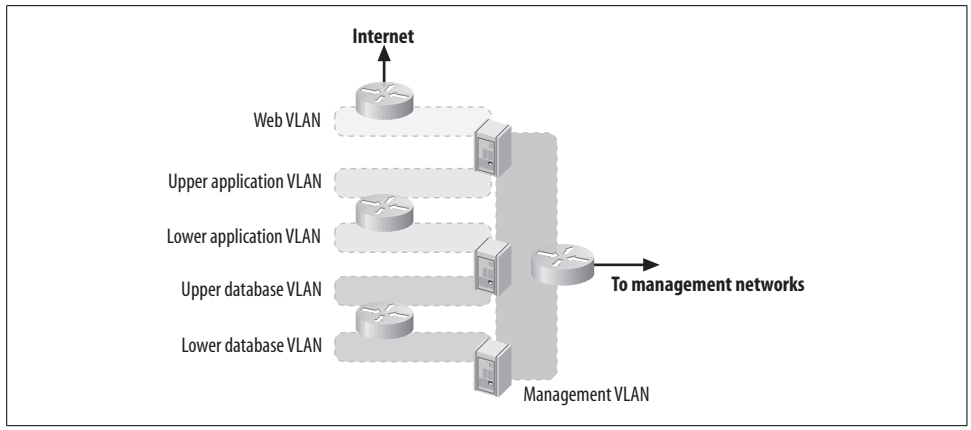


Figure 33-15. E-commerce management network

Management networks should be very secure. Features like port security and private VLANs can help keep the network secure. No server should be allowed to talk to any other server on this network, with the exception of backup and telemetry servers.

The problem with all these VLANs is that they are usually combined with a high-availability network design. For every VLAN a server must connect to, it must use two Ethernet interfaces: one for each of the switches in the high-availability pair. While this may not sound like a big deal, with only three VLANs for the average server, that's six Ethernet interfaces in use. Figure 33-16 shows a typical server connected to three VLANs on two switches.

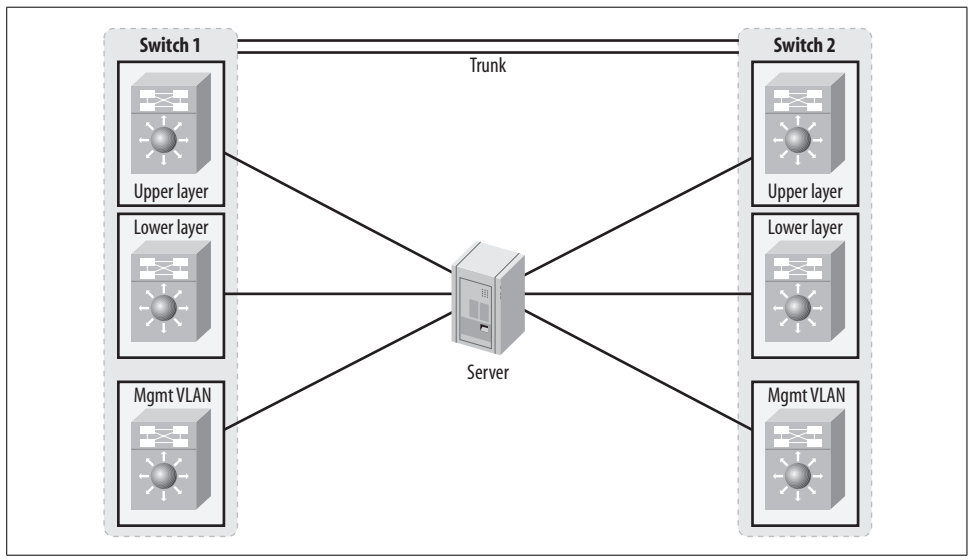


Figure 33-16. Ethernet interfaces in use on a server

When designing a network like this, make sure you work closely with your systems people. There are a lot of things to think about. Here's a short list:

- Every interface will need an IP address.
- In some server high-availability solutions, a third IP address will be needed for each VLAN. For example, Solaris IP Multipathing requires a virtual IP address on each VLAN in addition to one for each physical interface.
- Every IP address you assign may need a DNS entry (including virtual IP addresses).
- Which interface is primary?
- Does the server need a default gateway? If so, where does it go? Can the server support multiple defaults? How will this work? Web servers need a default gateway that points to the Internet. This will require your management VLAN to have specific routes on the servers.
- How many physical network cards do you need in a server to support six Ethernet interfaces? Make sure you have enough. Extra interfaces are even better.
- Will the servers have both interfaces active in each VLAN, or just one? Some server high-availability solutions require the switches to be configured a certain way, while others require different configurations. Work this out in a lab before you build your network.
- Will your servers support remote Ethernet consoles? Will you need a dedicated network for this traffic?

When figuring out your IP scheme, it's a good idea to make the last octet (or octets) the same for every interface on each server. In other words, if your upper network is 10.1.10.0/24, and your lower network is 10.1.20.0/24, make the last octet match for the server on each network. Thus, the upper IP address would be 10.1.10.10, and the lower IP address would be 10.1.20.10. Remember that you must assign an IP address for each interface, so make the last octet match for each switch. Figure 33-17 shows how such an IP scheme might work.

Small Networks

Many small companies don't have the need for elaborate three-tiered networks. Maybe they have only one office, or even three, but the offices are small, and the networks are simple. Even some larger companies do not have elaborate networks. No matter the size or complexity of the network, every aspect of it should be thoroughly documented.

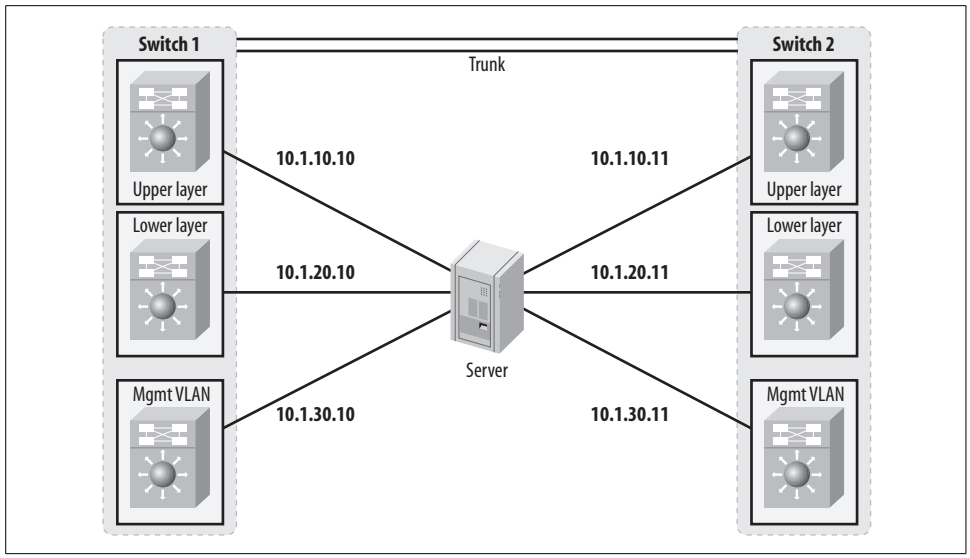


Figure 33-17. Matching last octet on multiple VLANs and switches

CHAPTER 34

IP Design

When a network or group of networks is designed well, the payoff can be substantial. The payoff, however, is in hours *not* spent, which can be very hard to quantify. Believe me, though—designing IP space the right way, the first time, can save literally thousands of man-hours over the lifetime of the network.

IP address allocation is rarely done properly, and many unlucky network administrators end up inheriting a mess of IP networks that's just been thrown together over time. In many cases, small networks are built with no vision of where the companies might end up, resulting in massive undertakings when the IP networks need to be changed. And even the best of IP address schemes can be rent asunder by a merger or acquisition.

Think about how long it takes to put an IP address, subnet mask, and default gateway on a server. Not long at all, when you're installing the server. Now think about what is involved when the IP address, subnet mask, or default gateway needs to be changed. The server must be taken offline, which usually involves a change control. In many environments, the change needs to happen during a change-control window, which may involve you coming into the office or data center at 2:00 a.m. Now imagine that your company has 100, 200, or 1,000 servers. Don't forget that DNS and/or WINS and Active Directory will need to be updated, too.

IP network design is not a glamorous project. It is not something that the CTO will slap you on the back for in a meeting. IP network design is not something that many people will appreciate at all, until it is explained to them. Still, careful planning and design are required to create an IP schema that will allow growth for years to come.

Public Versus Private IP Space

If you've taken any certification exams, you should be familiar with the idea of public versus private IP space. In a nutshell, there exist a select number of IP networks

that should not be routed on the public Internet. These networks are described in RFC 1918 ("Address Allocation for Private Internets") as follows:

```
3. Private Address Space

   The Internet Assigned Numbers Authority (IANA) has reserved the
   following three blocks of the IP address space for private internets:

     10.0.0.0        -   10.255.255.255  (10/8 prefix)
     172.16.0.0      -   172.31.255.255  (172.16/12 prefix)
     192.168.0.0     -   192.168.255.255 (192.168/16 prefix)
```

A quick note about these ranges is in order. First, the 172.16.0.0 network is not just the /16 range, and it is not the 172.0.0.0/8 range. The range allocated for 172.16.0.0 is a /12. The IP range included in this private range is 172.16.0.0–172.31.255.255. The ranges 172.0.0.0–172.31.255.255 and 172.32.0.0–172.255.255.255 are not composed of private addresses, and should not be used as such. This is a mistake commonly made in the real world.

Other ranges are also reserved, and should not be used. A now common example is the use of the 169.254.0.0/16 network when DHCP requests go unanswered. RFC 3330 (Special-Use IPv4 Addresses) describes these networks. The entry for 169.254.0.0 is as follows:

```
169.254.0.0/16 - This is the "link local" block. It is allocated for
communication between hosts on a single link. Hosts obtain these
addresses by auto-configuration, such as when a DHCP server may not
be found.
```

Over the years, I have encountered more than one company whose entire scheme was based on the network 128.0.0.0 because it was easy to decipher in binary (10000000.00000000.00000000.00000000). This is a dangerous practice. Again, RFC 3330 provides the details:

```
128.0.0.0/16 - This block, corresponding to the numerically lowest of
the former Class B addresses, was initially and is still reserved by
the IANA. Given the present classless nature of the IP address
space, the basis for the reservation no longer applies and addresses
in this block are subject to future allocation to a Regional Internet
Registry for assignment in the normal manner.
```

Similarly, the 127.0.0.0/8 network should never be used when designing IP networks. Though many people are familiar with the fact that 127.0.0.1 is a local loopback address, many are unaware that the entire network is reserved. RFC 3330 explains:

```
127.0.0.0/8 - This block is assigned for use as the Internet host
loopback address. A datagram sent by a higher level protocol to an
address anywhere within this block should loop back inside the host.
This is ordinarily implemented using only 127.0.0.1/32 for loopback,
but no addresses within this block should ever appear on any network
anywhere [RFC1700, page 5].
```

Many companies design their IP networks with a complete disregard for the rules, often using IP ranges that correspond to addresses, building numbers, or branch numbers. The risks of such designs have to do with connectivity and the rules of routing. If a host on the improperly numbered network 15.1.1.1/24 tries to get to a web page on a server with the same IP network and mask, the rules of routing will indicate that they are on the same network, so the packet will never be forwarded to the default gateway. Some would argue that a firewall with NAT is the solution for poor IP design, but because the host will never forward the packets to the default gateway, they will never be NATed.

Additionally, when companies form partnerships with other companies, they often link their networks together. A smart company will not link with a company that violates the IP rules, as they too will be affected since they will inherit routes to the improperly used IP spaces.

Remember, if there is a route in the routing table for a destination network, the default gateway is not involved. Should a valid public network be misused internally, your users will never be able to get to any services on that valid public network.

Figure 34-1 shows an example of an improperly used public IP network. In this example, the world's most interesting web page has a legitimate IP address of 25.25.25.40, with a subnet mask of 255.255.255.0. Bob, in Company A, has an improperly used IP address of 25.0.0.15, with a subnet mask of 255.0.0.0. What is important to understand in this example is that, even though the subnet masks are different, no one in either Company A or Company B can get to the world's most interesting web page.

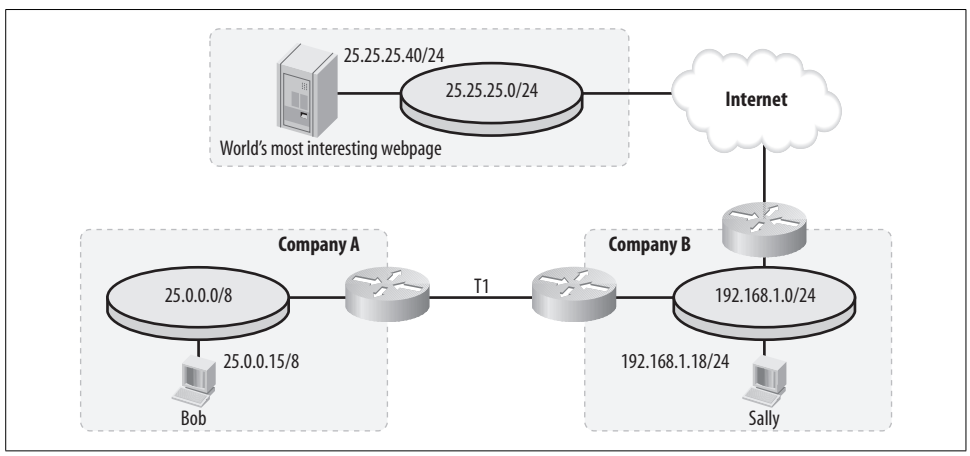


Figure 34-1. Invalid use of a public IP network

Bob cannot get to the world's most interesting web page because as far as the routers are concerned, he is on the same IP network as the destination. The IP stack on his machine will assume that because the destination is on the same network, it does not need to forward the packets to the default gateway. In this case, Bob's computer will either send the packets to a local device (should one exist with the same IP address as the world's most interesting web page), or fail to resolve the IP address to a local MAC address. Either way, Bob will never be able to see the world's most interesting web page.

Assuming Company B would like its employees or servers to be able to get to Company A's devices (why else would there be a link?), the router connecting Company B to the Internet must have a route to Company A's network; in this case, 25.0.0.0 255.0.0.0. As the default route will only be used if there are no matching routes in the routing table, and because, in this case, there is a matching route in the routing table, the packets destined for the world's most interesting web page will be forwarded to Company A. Therefore, Sally will not be able to access the world's most interesting web page either.

VLSM

Variable Length Subnet Masking (VLSM) is the best thing to happen to IP networking since subnet masks.

In a nutshell, when using what we'll call *classful subnetting rules*, if a network is divided into subnets, each subnet within the classful network must be the same size. Figure 34-2 shows the ways in which a Class C network can be divided. Using the traditional classful rules, these are the only divisions possible. The reason for this is the binary nature of the subnet mask.

When IP was developed, there were more IP addresses available than anyone could have possibly wanted—or so we all thought! But as the world progressed, we quickly discovered that we were running out of IP address space at an alarming rate. Some bright minds determined that if the rules of subnetting could be altered a bit, we could give ourselves some breathing room. The result was VLSM.

VLSM states that when subnetting a network, the individual subnets do *not* need be of equal size. The caveat is that the subnets must still follow the rules of binary; that is, they must exist where they would normally fall when subnetting using the traditional rules.

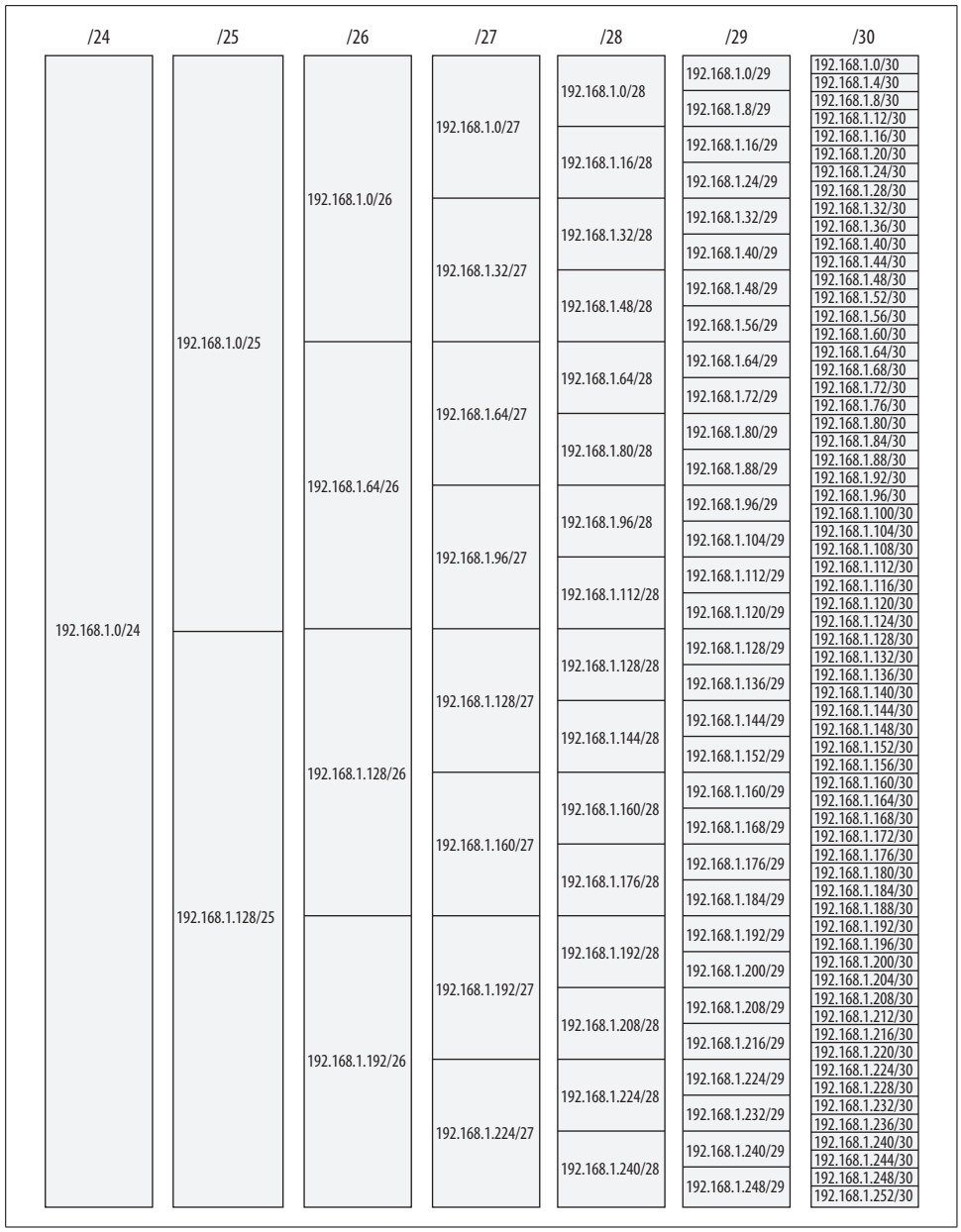


Figure 34-2. Classful subnets of a /24 network

Figure 34-3 shows examples of a normally subnetted network, a proper VLSM network, and an invalid VLSM network. In the invalid example, the network 192.168.1.200/28 is not permitted by the rules of subnetting. Remember, a subnet must exist where it would normally fit had you subnetted the entire network with that subnet mask. If you were to configure an IP address of 192.168.1.200/28, the network address would be 192.168.1.192, and the broadcast address would be 192.168.1.207.

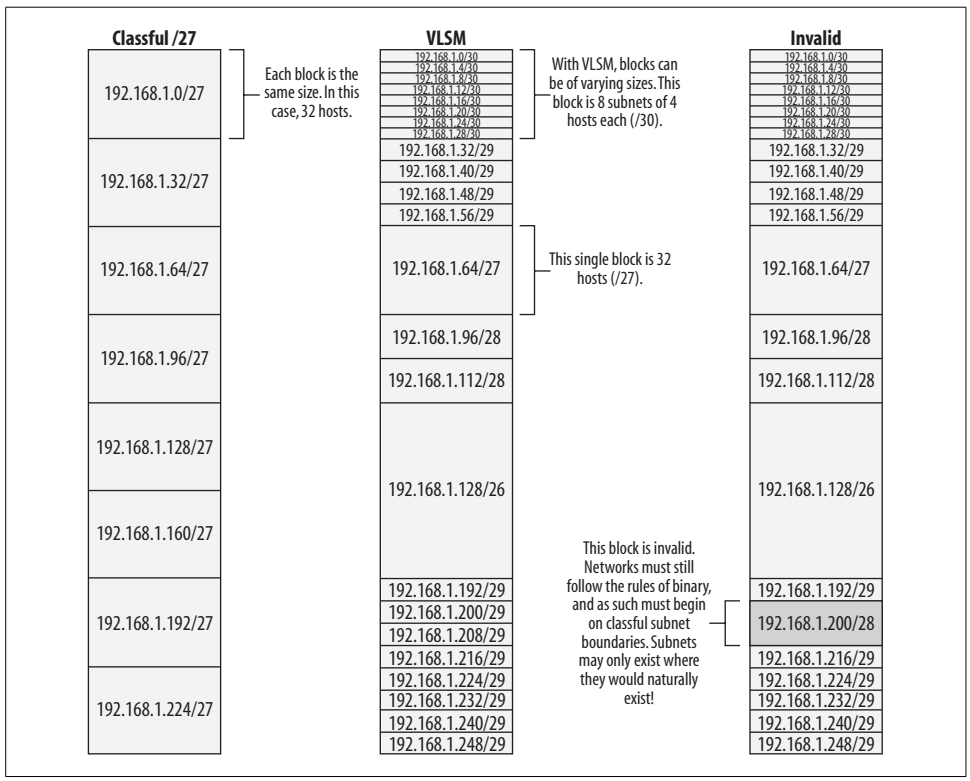


Figure 34-3. Benefits and limitations of VLSM

The benefits of VLSM should be apparent. In the old days, we used to have to put aside entire networks just for /30 subnets (once a /30 subnet was allocated within a classful network, the entire network had to be allocated with the same subnet mask). Often, 80 percent of the network was wasted because it had to be allocated as /30 subnets, when only 10 or so /30 subnets were needed. With VLSM, you can use 25 percent of a network for /30 subnets, another 25 percent for /27 subnets, and so on. Needless to say, VLSM has had a serious impact on the allocation of IP space, both on the Internet, and in private networks.

CIDR

If you deal with large numbers of IP networks, like ISPs do, *Classless Internet Domain Routing* (CIDR) is a most useful tool. While VLSM has had a dramatic impact on IP space allocation within corporate networks, CIDR has had an equally impressive impact on public Internet networks allocated to ISPs.

CIDR is sort of the inverse of VLSM: whereas VLSM prescribes rules for subdividing networks, CIDR prescribes rules for referencing groups of networks with a single route statement.

Aggregating routes may seem like a solution looking for a problem if you've only ever dealt with small or medium-sized corporate networks, but, rest assured, it provides a real benefit. Usually, small or medium-sized companies use one of the private IP networks described by RFC 1918. If a company used the entire 10.0.0.0/8 network, and subdivided it to maximize efficiency, each of these subdivisions would technically be a subnet. While VLSM deals with subnets, CIDR deals with groups of *major* or *classful* networks. Figure 34-4 shows how a single route statement can reference eight Class C networks. The route is called an *aggregate* route or a *summary* route.

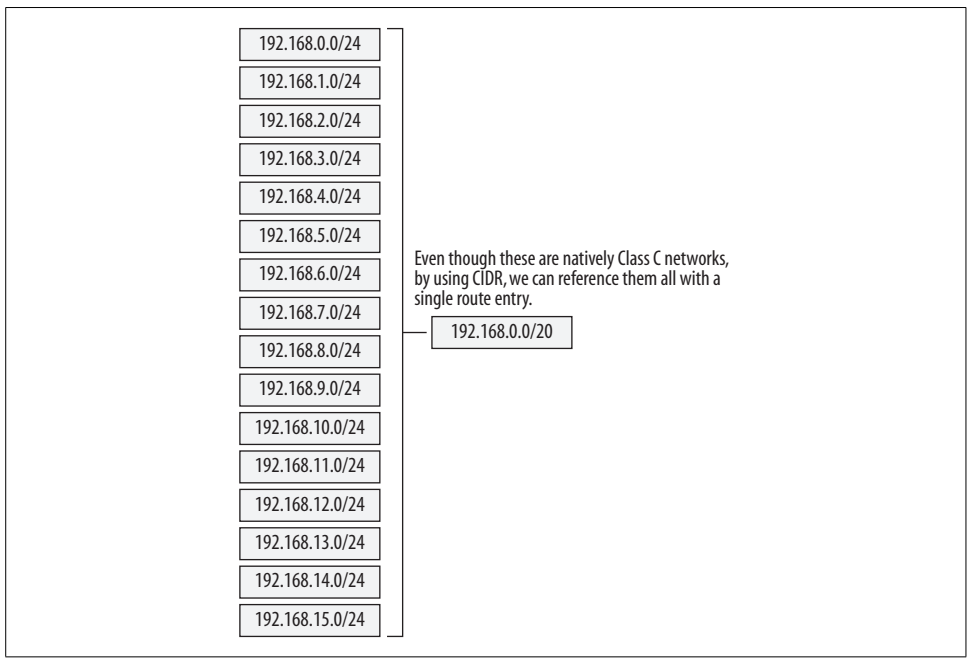


Figure 34-4. CIDR route aggregation

On Internet-attached routers with *full tables*, the routing tables may contain hundreds of thousands of routes. Anyone can see them at any time by connecting to one of many publicly available route servers. To illustrate this point, here's a sample of the number of routes the AT&T IP Services Route Monitor shows as of this writing:

```
################### route-server.ip.att.net ###################

route-server> sho ip bgp summary
BGP router identifier 10.1.2.5, local AS number 65000
BGP table version is 1094053, main routing table version 1094053
182347 network entries using 18417047 bytes of memory
3464386 path entries using 166290528 bytes of memory
42502 BGP path attribute entries using 2380392 bytes of memory
38401 BGP AS-PATH entries using 997216 bytes of memory
4 BGP community entries using 96 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 188085279 total bytes of memory
Dampening enabled. 850 history paths, 742 dampened paths
182190 received paths for inbound soft reconfiguration
BGP activity 192312/9965 prefixes, 4364639/900253 paths, scan interval 60 secs
```

There are 182,347 network entries on this route server. It is reasonable to assume that any router connected to the Internet that is receiving full tables will contain a similar number of network entries. Notice also the fourth line:

```
182347 network entries using 18417047 bytes of memory
```

Those entries are taking up 18.4 MB of memory in the router. More importantly, any time a change is made to the paths to any of these networks, the entire world of BGP-connected routers must be notified.

CIDR comes into play on the Internet because it helps to make these massive tables smaller. It also helps smaller networks (and, more importantly, their administrators), by making local routing tables smaller. Smaller routing tables are more logical, easier to understand, and much simpler to troubleshoot during outages.

Allocating IP Network Space

One of the problems I've seen repeatedly over the years is improper allocation of IP network space. Whether the trouble is the use of public IP addresses that are not rightfully owned, or just a lack of understanding of the benefits of proper IP allocation, a poorly designed IP network can cause daily headaches and increased troubleshooting time. And because a poorly designed IP scheme can take months to rectify, the problem usually gets worse over time because no one wants to take on the task of renumbering the entire network.

There are some simple rules that can help when designing a network using IP address space. Here is the first one:

> When allocating IP address space, endeavor to allocate a block that can be referenced with a single access-list entry.

Following this simple rule will make everything from server allocations to global network design much simpler.

Assume for a minute that you've been requested to create a network for a new server farm. The powers that be insist there will only ever be 30 servers in the farm. You, being the bright admin, ask what kind of servers they will be. The response is that they will all be Oracle servers.

The average admin might be quick to say that a /27 (255.255.255.224) network would be in order, as it provides 32 host addresses. Two of these are used for network and broadcast addresses, however, leaving no room for growth. This leads us to our next rule:

> Always allocate more IP address space than is originally requested.

If your boss requests 30 IP addresses, give her a block of 64; if she requests 60, give her 128. While this may seem wasteful on the surface, you are always better off with excess IP addresses within a subnet. You must learn to balance the need for growth with the overall availability of addresses. Allocating 256 addresses when only 30 are requested is usually foolish, given normal growth patterns. Allowing for 30 percent growth is a good rule of thumb. Rounding up when you get to binary boundaries (16, 32, 64, etc.) makes this rule pretty easy to implement.

To make things even more interesting, let's say there is already a server farm in your network. The IP network for the server farm is 10.100.100.0/24. There are already 10 servers in place, as well as the router port that connects the network to the rest of the company. Assume the IP addresses in use are 1–11 (people like to allocate numbers in order).

When asked for 30 more IP addresses, why not just provide the addresses 12–42? You could do this, and everyone would be fat, dumb, and happy. But here's where you can be smarter than the average admin. Instead of allocating a random list of IP addresses based on the last number used, allocate a block that agrees with our first rule: *When allocating IP address space, endeavor to allocate a block that can be referenced with a single access-list entry*. If you allocate the range of 32–63, not only have you allocated enough IP addresses, but you can reference the range like this:

```
Access-list 101 permit ip any 10.100.100.32 255.255.255.224 eq web
```

But what about our second rule? *Always allocate more IP address space than is originally requested.* Because 30 IP addresses were requested, you should think ahead, and

allocate 64. This changes things a bit, because according to the rules of subnetting, you can't allocate 64 contiguous IP addresses starting at 32. Looking back at Figure 34-3, you can see that you'll need to start at a multiple of 64. So, allocate 64–127:

```
Access-list 101 permit ip any 10.100.100.64 255.255.255.192 eq web
```

Still only one ACL entry!

Now, you can create security rules based on logical groups of devices where none previously existed. To compare, if you'd used 12–42, you would have needed the following lines to achieve the same thing as the single line above:

```
Access-list 101 permit ip and 10.100.100.12 255.255.255.252 eq web
Access-list 101 permit ip and 10.100.100.16 255.255.255.248 eq web
Access-list 101 permit ip and 10.100.100.32 255.255.255.250 eq web
Access-list 101 permit ip and 10.100.100.40 255.255.255.254 eq web
Access-list 101 permit ip and 10.100.100.42 255.255.255.255 eq web
```

You've allocated twice the requested number of IP addresses, and you can address them using one-fifth the number of ACL entries.

When you allocate groups of similar servers into what I like to call *subnettable ranges*, another benefit comes to light. When servers are grouped like this, you can remove them from the network and place them on their own physical networks without changing the IP scheme. This takes some planning, but when it pays off, it pays off in a big way.

Figure 34-5 shows how this idea might be applied. Even though the network in place is 10.100.100.0/24, if you apply your servers within logical groups of IP addresses corresponding to subnet boundaries, later on you can actually subnet the network without changing the IP addresses of any servers. You'll only need to change the subnet masks and default gateways.

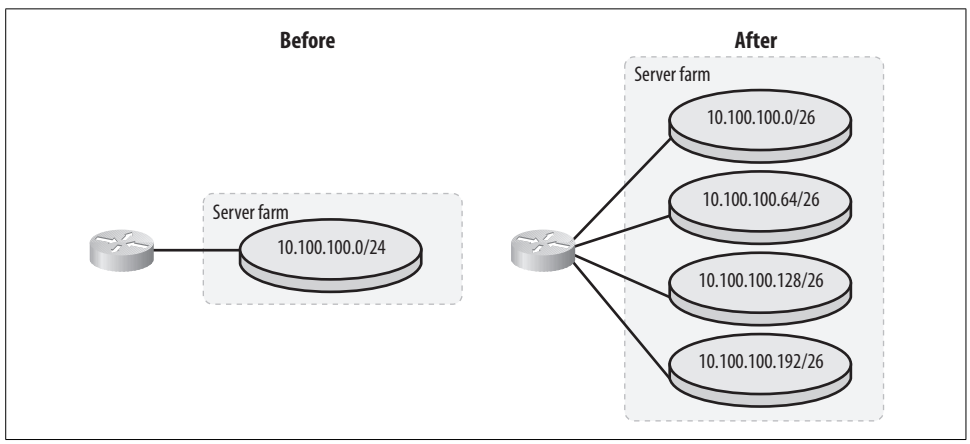


Figure 34-5. Subnetting an existing range

This kind of thing happens more often than you might think. Perhaps a decision will be made that all of the Oracle servers should be on their own physical network, for security reasons. No problem! Because you were smart enough to allocate their IP space along subnet boundaries, the entire range can easily be pulled out and moved to another interface. All the IP addresses will stay the same; only the subnet masks will change.

The catch (isn't there always a catch?) is that you must be vigilant! If even one IP address is allocated in violation of these rules, you will have to renumber something to move a section of the existing network.

Allocating IP Subnets

When allocating IP subnets within a network, care should be taken to allocate them in a logical fashion. When allocating subnets, you should strive for two goals:

- Allow for the largest possible remaining addressable space (i.e., the largest subnets possible in the remaining space).
- Allow as many subnets as possible to be expanded into the surrounding available space.

As you will see, achieving both of these goals is a balancing act.

I've encountered three methods for allocating IP subnets. I'll call these methods *sequential*, *divide by half*, and *reverse binary*.

Sequential

Most people's first inclination seems to be to allocate IP addresses and IP subnets in numerical order. That is, they allocate the first numerical subnet, then the next, and so on. If such a person were subnetting the 192.168.100.0/24 network into /30 subnets, they would likely get allocated in this order: 0, 4, 8, 12, etc. Sequential allocation of subnets in a network is what most admins seem naturally prone to do. It works, it's easy to understand, and it's pretty easy to tell where the next available network will be.

Of the three methods I'll discuss here, this is the least desirable, and the most often deployed. There are some serious problems with this method. First, there is no room for expansion in any of the subnets, except for possibly the last one used. If the subnet 192.168.100.16 exhibits growth, and now needs 18 addresses instead of 16, the entire subnet must be renumbered to a larger subnet space elsewhere. Second, this method does not allocate addresses in a manner that allows for the largest available block to be concurrent and addressable. Another problem is future availability of subnets. Remember, with VLSM, you are not limited to having every subnet be the same size. If you plan carefully, you can use your given space very efficiently.

Figure 34-6 shows how we might sequentially allocate a series of /28 subnets within the 192.168.1.0/24 network. Each time we need another /28 subnet, we allocate the next one available. I've broken out the subnets to show the largest possible network space remaining after each allocation. Because we cannot use the last subnet due to the rules of networking, a lot of space is wasted. This wasted space is mostly theoretical, as we could always use VLSM to split the last subnet into smaller networks. But, the fact remains that by using this method, as soon as a single subnet is allocated, we can no longer assign a /25 space, as the only remaining /25 space is the broadcast subnet.

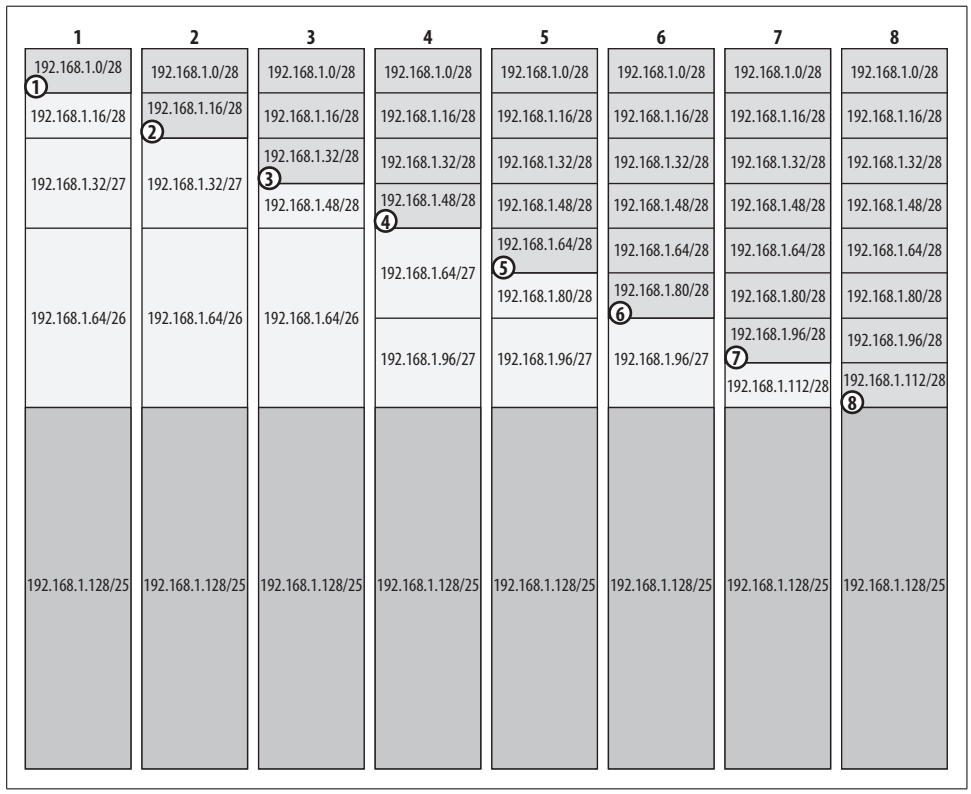


Figure 34-6. Sequential IP subnet allocation

The diagrams in these sections are keyed as follows: the lightest subnets have not been allocated, the middle-shaded subnets have been allocated, and the darkest subnets are not usable (i.e., are broadcast subnets).

Divide by Half

This method prescribes that every time a new network is allocated, the smallest available chunk of addresses is divided by half. The idea here is to maintain the largest possible block of addressable space. To that end, we allocate the middle subnet first, thus leaving the large /25 block available.

Figure 34-7 shows the divide-by-half method in action. In this example, I've taken the same 192.168.1.0/24 network, and allocated /28 subnets accordingly. Using this method allows for each subnet to be expanded, as there is space after each used subnet equal to the size of the subnet in use. This method also allows for the largest possible subnet to be available at any given time. I like this method because it reasonably balances the need for subnet expandability while keeping the largest free space available for as long as possible.

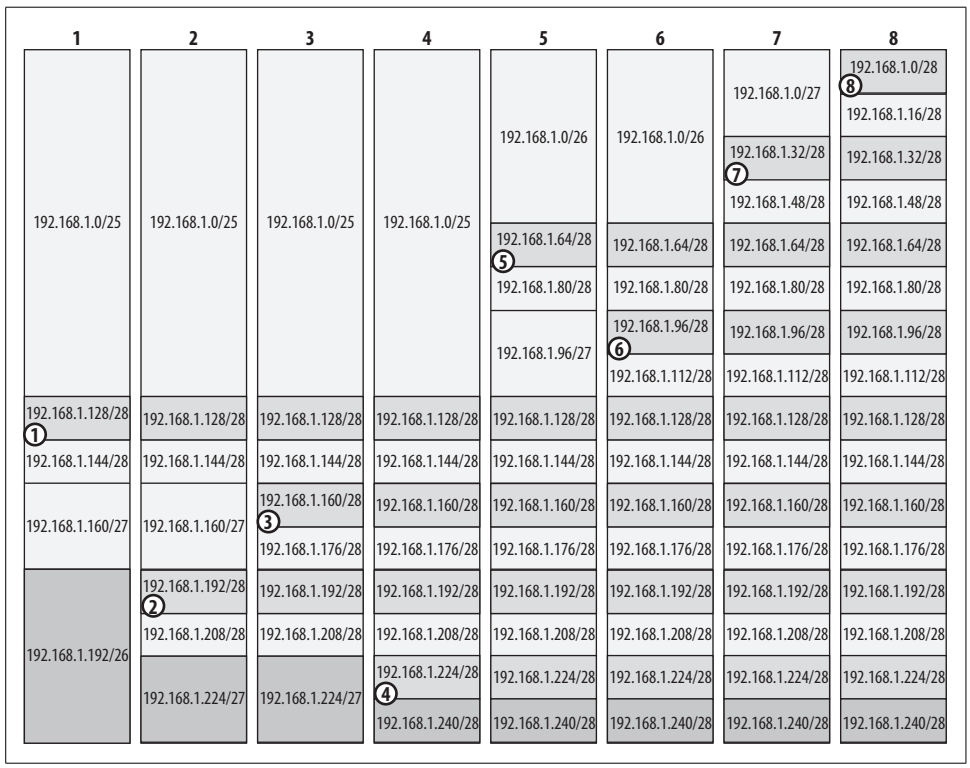


Figure 34-7. Divide-by-half IP subnet allocation

Reverse Binary

Reverse binary is Cisco's recommended method for IP subnet allocation. In this method, subnets are allocated by counting in binary, but with the most and least significant bits reversed. Figure 34-8 shows how reverse binary works. The normal binary representations of the digits 0–7 are shown on the left. By reversing the significant bit order, we create a mirror image of the numbers in binary (shown on the right). These numbers happen to correspond to the subnets that should be used, in order, regardless of size. The pattern may be continued for as long as is useful.

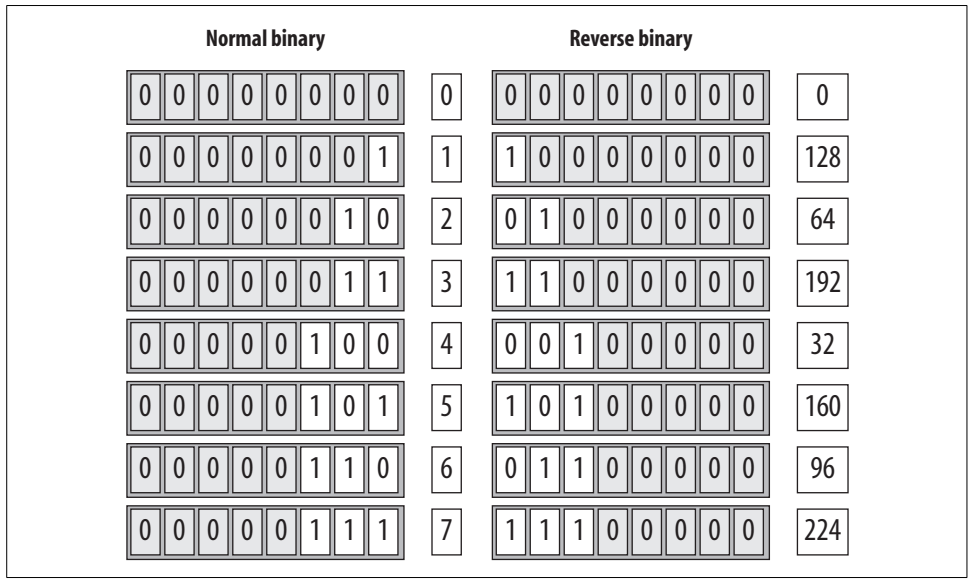


Figure 34-8. Reverse-binary subnet allocation

By allocating subnets in this order, we end up spacing the subnets so that no two subnets are adjacent unless the subnets are of varying sizes (a distinct possibility using VLSM). This method also offers a reasonable tradeoff of allowing the largest possible remaining addressable space while offering the expandability desired. The real difference between this method and the divide-by-half method is the size of the largest remaining block in the source IP network.

The exception to these rules for me is the allocation of /30 networks. Because these networks are usually used for point-to-point links, and as such, will never expand, I usually allocate an entire /24 network (or more, depending on the overall need) just for /30 subnets.

The results of using the reverse-binary method to divide our 192.168.1.0/24 network are shown in Figure 34-9. Each of the allocated networks can be expanded with no IP address changes. A reasonable balance is achieved, with large pools of available addresses remaining after each allocation. The differences between the reverse-binary and divide-by-half methods all but disappear by the eighth subnet allocation, where, in our example, each subnet has an equally sized subnet adjacent to it.

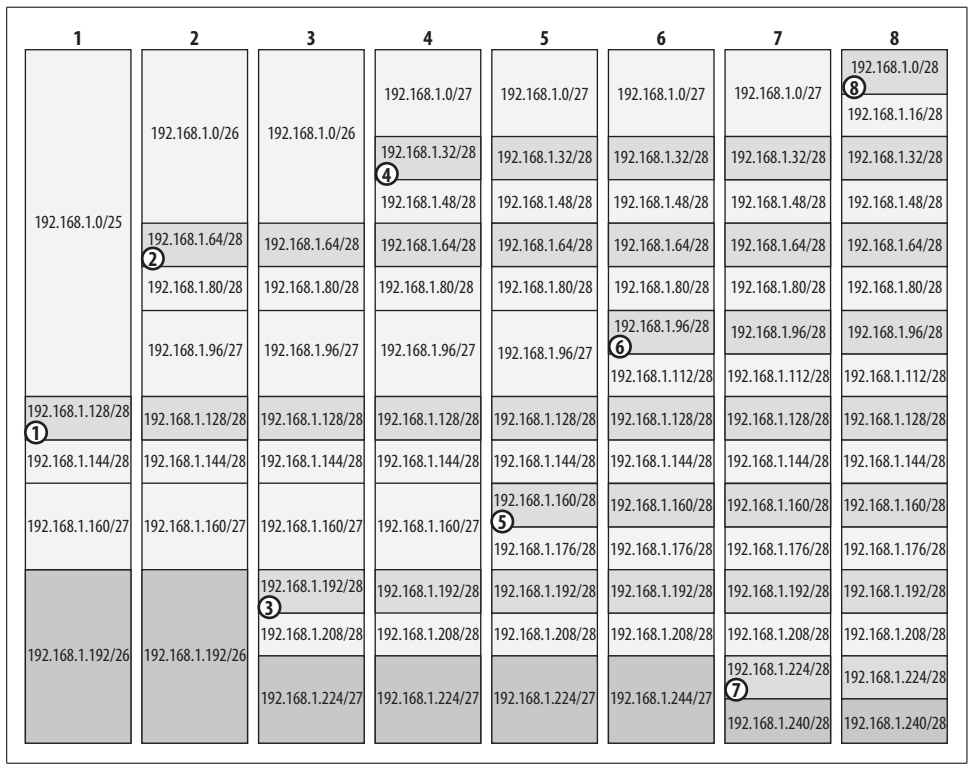


Figure 34-9. Reverse binary IP subnet allocation

The reverse-binary method of IP subnet allocation is the most logical choice mathematically, but it can be the hardest to understand. Proper documentation should be produced at all times to avoid confusion, regardless of the method used.

IP Subnetting Made Easy

IP subnetting seems to trip up quite a few people in the networking world. I've known experienced consultants who have worked in the industry for 15 years and still resort to subnet calculators.

IP subnetting can be a daunting subject for those who are not familiar with it. The principles of subnetting are based on binary (and some mathematical) principles such as eXclusive-OR (XOR), which, for many people, are foreign concepts or aspects of college courses long-since forgotten.

There are really only two times when your average networking people need to know how subnet math truly works: when they study for their first networking certification (the CCNA), and when they study for their last (the CCIE). In fact, I don't really think the math is needed for the CCNA coursework, although Cisco makes you learn it. The CCIE exam does make you do bizarre things with subnet masks that require a full understanding of the math behind the topic. However, these things are rarely, if ever, seen in the real world, and are not a topic for this book.

If you want to be able to do IP subnetting in your head, there are a couple of things you will need to understand. First, Cisco, along with every other major manufacturer out there, wants you to learn subnetting its way. Why? So its tests are harder to pass, and so everyone who's certified speaks the same language.

Cisco, in particular, seems to want people to think in a way that makes it harder to figure out what's really going on. As an example, in Cisco parlance, a native class C network with a subnet mask of 255.255.255.224 is said to consist of 6 networks with 30 hosts in each network.

Call me pedantic, but that's incorrect. The subnet mask actually results in 8 networks with 32 hosts each. Cisco's point is that, using classful networking rules, there are only 6 *available* networks, with 30 *available* hosts in each network. While this is a valid concept, I believe it causes confusion.

The reason Cisco's method causes confusion is simple: there is no easy way to prove the answer. Using the method I will outline here, however, there will always be an easy way to prove your answer. In this case, 8 * 32 = 256.

Everything having to do with subnet masks has something to do with the number 256. In fact, this will be our first rule:

> Every result will either produce 256, or be divisible by 256.

Looking at a subnet mask, the maximum value for an octet is 255. Remember, though, that the values start with 0, so 255 is really the 256th number possible. This is a very important concept because everything else is predicated on this idea.

The second rule astounds many people. In fact, it astounded me when I discovered it while writing my own subnet calculator program:

> Only nine values are possible for any octet in a subnet mask. They are: 0, 128, 192, 224, 240, 248, 252, 254, and 255.

Subnet masks are, by their nature, inclusive. That is to say, you can only add or subtract bits from the mask in bit order, and they must be added from left to right—you cannot skip a bit in a subnet mask. Figure 34-10 shows bits validly and invalidly used, with their resulting decimal octets.

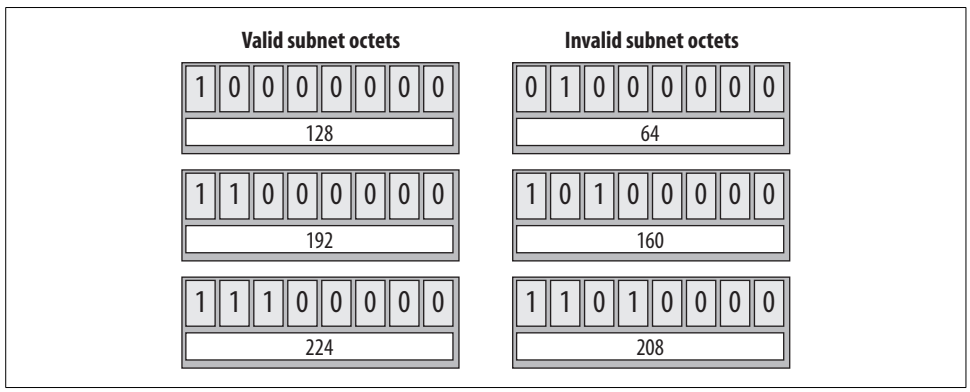


Figure 34-10. Valid and invalid subnet octets

The CCIE exam requires knowledge of manipulating subnet masks in unnatural ways, but you should never see this in the real world. While it is possible to use subnet masks in unobvious ways, you should never do it outside of a lab environment.

Because there are only eight bits in an octet, if you limit yourself to only allowing bits to be added or subtracted, there can only be a finite number of values represented in binary. Those values are shown in Figure 34-11.

Mask	Binary	Ratio
0	0000 0000	1:256
128	1000 0000	2:128
192	1100 0000	4:64
224	1110 0000	8:32
240	1111 0000	16:16
248	1111 1000	32:8
252	1111 1100	64:4
254	1111 1110	128:2
255	1111 1111	256:1

Figure 34-11. Possible subnet octet values

Notice the column titled Ratio. This is something you won't see in the manufacturers' texts, which is a shame because I believe it is the very essence of subnet masks in the real world.

In Figure 34-12, I've laid out the ratios as they apply to any octet in any position in the subnet mask. None of this needs to be memorized, as it will all become clear shortly. Notice the patterns at work here. The ratio you will learn to use for a class C network works for any class network. Simply by moving the ratio to the proper position in the subnet mask, you can figure out what the octet should be.

Mask	First octet		Second octet		Third octet		Fourth octet	
0	1	256 * 256 * 256 * 256	256 * 1	256 * 256 * 256	256 * 256 * 1	256 * 256	256 * 256 * 256 * 1	256
128	2	128 * 256 * 256 * 256	256 * 2	128 * 256 * 256	256 * 256 * 2	128 * 256	256 * 256 * 256 * 2	128
192	4	64 * 256 * 256 * 256	256 * 4	64 * 256 * 256	256 * 256 * 4	64 * 256	256 * 256 * 256 * 4	64
224	8	32 * 256 * 256 * 256	256 * 8	32 * 256 * 256	256 * 256 * 8	32 * 256	256 * 256 * 256 * 8	32
240	16	16 * 256 * 256 * 256	256 * 16	16 * 256 * 256	256 * 256 * 16	16 * 256	256 * 256 * 256 * 16	16
248	32	8 * 256 * 256 * 256	256 * 32	8 * 256 * 256	256 * 256 * 32	8 * 256	256 * 256 * 256 * 32	8
252	64	4 * 256 * 256 * 256	256 * 64	4 * 256 * 256	256 * 256 * 64	4 * 256	256 * 256 * 256 * 64	4
254	128	2 * 256 * 256 * 256	256 * 128	2 * 256 * 256	256 * 256 * 128	2 * 256	256 * 256 * 256 * 128	2
255	256	1 * 256 * 256 * 256	256 * 256	1 * 256 * 256	256 * 256 * 256	1 * 256	256 * 256 * 256 * 256	1

Figure 34-12. Subnet octet ratios

Using the class C network 192.168.1.0 255.255.255.0 as an example, if you apply a subnet mask of 255.255.255.224, the result is 8 subnets with 32 hosts in each. Look at Figure 34-11, and you'll see that the ratio for the subnet octet 224 is 8:32.

The ratios happen to correlate with the number of subnets and hosts per subnet in a native class C network. If you look at the other columns, you'll notice that the ratio is the same, but it is in a different position in the equation.

Let's look at a single example. Figure 34-13 shows the ratios for the subnet octet of 224.

Mask	First octet		Second octet		Third octet		Fourth octet	
224	8	32 * 256 * 256 * 256	256 * 8	32 * 256 * 256	256 * 256 * 8	32 * 256	256 * 256 * 256 * 8	32

Figure 34-13. 224 ratios

In practice, here's what the chart is saying, using the network 10.0.0.0:

- 10.0.0.0 224.0.0.0 = 8 subnets of 536,870,912 (32 * 256 * 256 * 256) hosts
- 10.0.0.0 255.224.0.0 = 2,048 (256 * 8) subnets of 2,097,156 (32 * 256 * 256) hosts
- 10.0.0.0 255.255.224.0 = 524,288 (256 * 256 * 8) subnets of 8,192 (32 * 256) hosts
- 10.0.0.0 255.255.255.224 = 134,217,728 (256 * 256 * 256 * 8) subnets of 32 hosts

Notice that in each case, somewhere in the equation, the ratio of 8:32 appears.

That's all well and good, but it still looks like a lot of icky math to me, so lets make it even simpler. This leads us to our third rule:

> When you know these rules, all you need to be able to do is to double or halve numbers.

Subnet masks, like all things computer-related, are based on binary. Because binary is based on powers of two, all you really need to be able to do is to double a number, or cut it in half.

Memorize the nine possible values for subnet masks (two are easy—0 and 255), and write them down on a sheet of paper as shown in Figure 34-14. Once you have those down, you've successfully done all the memorization you need.

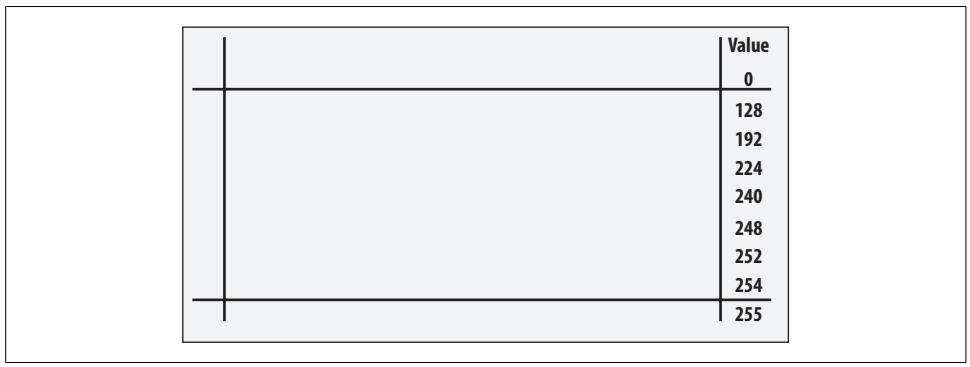


Figure 34-14. Subnet worksheet step #1

You'll fill in the rest of the sheet by using the simplest math possible: halving and doubling. Write the numbers 1:256 next to the 0 at the top. This is easy to remember. How many natural networks are there in a class C? 1. How many hosts are there in a class C? 256. Remember that everything regarding subnets comes back to the number 256.

Now, take the number to the left of the colon, and double it (1 * 2 = 2), and take the number to the right of the colon, and cut it in half (256 / 2 = 128). Repeat this process until you get to the bottom of the sheet. You've just figured out the network: host ratios for a native class C network! Your paper should look like the example in Figure 34-15.

This trick is so simple that when I take mentally exhausting exams like the CCIE lab, I actually write out a sheet like this so I don't have to think at all during the lab. Now that you have all the ratios, you're set for working on a class C network—but what about other networks? Well, you have two choices: always carry this book with you (which certainly would appeal to me), or apply some simple logic to what you've already learned.

As an example, let's use the private network 172.16.0.0 255.255.0.0, which is a class B network. People often get confused when working with ranges other than class C

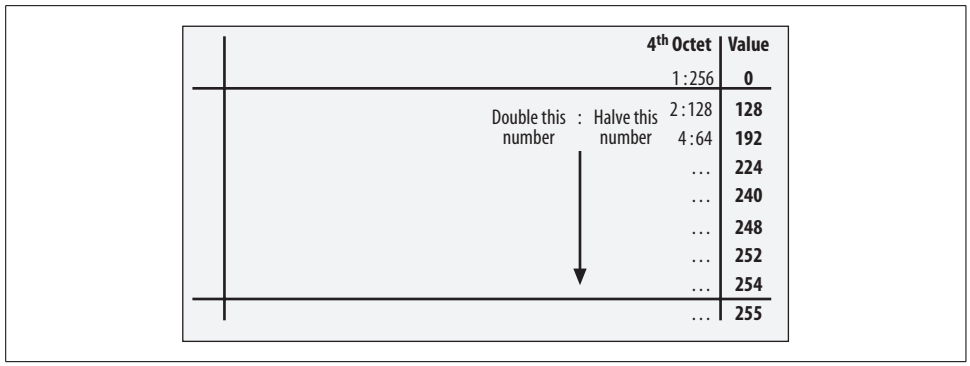


Figure 34-15. Subnet worksheet step #2

because these ranges aren't usually seen in small companies. In practice, using this method, there is no difference. We'll divide the network into 8 pieces. Using your subnet worksheet, which octet value has an 8 on the left of the colon? 224. Replace the leftmost zero in the native subnet mask with this number, and you have your answer: the subnet mask you need is 255.255.224.0. It's as simple as that.

Now, let's try a more complex problem, like one you might see on an exam. With a network of 172.16.0.0 255.255.0.0, what subnet mask would you need to allow a network with 1,200 hosts in it? Here's the easy way to figure this one out. A class C network has 256 hosts in it. Double this number until you get a number larger than 1,200:

256 * 2 = 512
512 * 2 = 1,024
1024 * 2 = 2,048

Let's fill in the sheet some more. Draw a line under the top entry, and another above the bottom entry, as shown in Figure 34-16. The entries between the lines are subnet octet values that fall between classful boundaries. This third column of numbers in Figure 34-16 indicates a further binary progression.

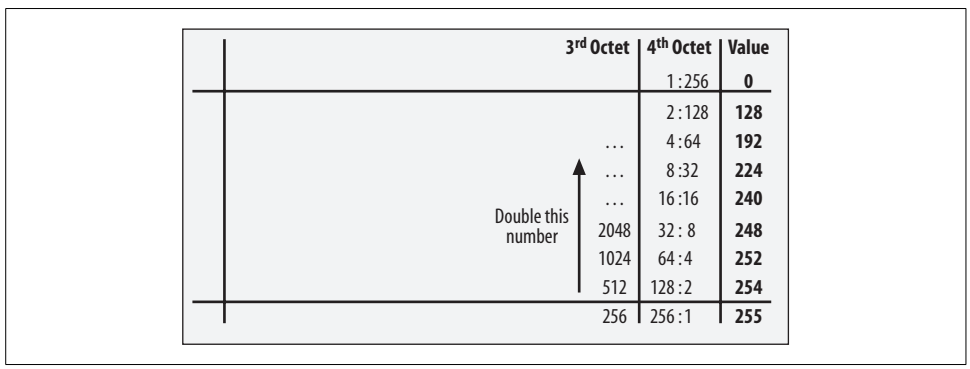


Figure 34-16. Subnet worksheet step #3

These numbers relate to the number of hosts in a subnet. You can keep the progression going as long as you need. When you get to the top of the sheet, start a new column to the left. These will be the number of hosts for your second and first octets.

Looking at this worksheet reveals that the required subnet mask is 255.255.248.0. The network 172.16.0.0 (a native class B network) with a subnet mask of 255.255.248.0 will result in 32 subnets with 2,048 hosts in each. To validate the math, perform this equation: 8 * 256 = 2,048.

A quick word about a trick question you may see on tests is in order. When using this method, you must remember that the first and last subnet may not be used, and the first and last host may not be used. Therefore, if the question, "What would the subnet mask be for a network that required 1,024 hosts to be on the network?" were posed, the answer would not be 252, as it might appear, because two of those hosts are unusable (network and broadcast). If you require 1,024 hosts to be live devices on the network, you must have a network capable of supporting 2,048 hosts. This is one of the limitations of using a binary system.

Similarly, if you need 8 usable hosts on the network, you must provide 16 host addresses, not 8.

Figure 34-17 shows the same information as that presented in the preceding worksheet in a different format. Some people respond better to seeing this information in a horizontal rather than a vertical format—I like this model because it is laid out the same way the bits are laid out in a subnet mask. Use whichever format works for you. There are no tests in this book.

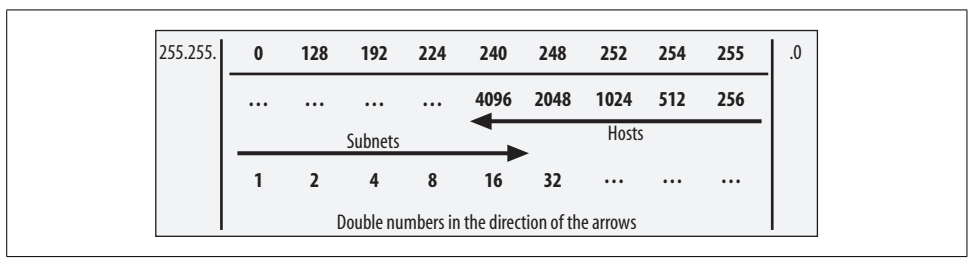


Figure 34-17. Horizontal format of the subnet worksheet

Remember, a proper subnet mask can have only one of nine values in any single octet. If you start with a value of 1 on the bottom left, and a value of 256 on the top right, then double those numbers under each subsequent value, you can quickly figure out what your subnet mask needs to be based on how many hosts it needs to have or how many subnets.

Once you have memorized the nine subnet octet values (remember, two are easy!), and can double and halve the rest in your head, you'll find that the numbers become obvious. You'll see all of these numbers over and over if you're around computers for any length of time. They're all powers of two, and when you start to recognize the patterns I've outlined for you, calculating any subnet mask will be an easy matter of doing some quick, very simple math in your head.

CHAPTER 35

Network Time Protocol

The Network Time Protocol (NTP) is an Internet protocol used for synchronizing a device's clock with a reference source across a network. When accurate time is required on your networking equipment or computers, you should use NTP—in other words, you should always use it.

NTP was originally defined in RFC 958. The last documented version is Version 3, which is defined in RFC 1305. To learn more about NTP, check out its home page: *http://www.ntp.org*.

The interesting stuff relating to how NTP really works is found in RFC 1129, which is available only in PostScript or PDF format. If you get excited when you see the type of math shown in Figure 35-1, RFC 1129 is for you.

$$e_{i+1} = \frac{d_i}{K_g}\frac{q^{n_i}-1}{q-1} + \frac{1}{K_f}\sum_{j=1}^{i}\frac{n_j d_j}{a_{j-1}u_{j-1}}.$$

Figure 35-1. Actual math from RFC 1129

If, like most people, you can't be bothered with the math, and just want to know what you need to do to make NTP work for you, read on.

What Is Accurate Time?

How do we measure time? How long is a second? Who defined how long a second is, and why does it matter? These are questions most people don't think about. Most of us take time for granted.

Every electronic device you own, from your personal computer to your television, relies on time being accurate to a certain degree. The power in your home oscillates at 60 cycles per second (60 Hz). High-definition televisions update their screens at 60 frames per second. Modems, T1s, DS3s, and all other data services send information at a set number of bits per second. When two pieces of equipment communicate, they must agree on the length of a second.

For a long time, a second was defined as 1/86,400th of a sidereal day. 24 hours * 60 minutes * 60 seconds = 86,400 seconds. A *sidereal day* is the time it takes for the Earth to spin once (360 degrees) on its axis. Astronomers will tell you that a sidereal day is measured as the amount of time it takes for a star in the sky to transit the meridian twice. A *solar day* is shorter than a sidereal day by four minutes, which is why we have leap years: the calendar has to reset itself due to the accumulation of these four-minute errors each day. Thus, 1 sidereal second = 1.00278 solar seconds. A solar second is defined with the same formula (1/86,400th of a solar day), but uses the sun as the reference point in the sky instead of a distant star. Because the Earth rotates around the sun, the position of the sun in the sky changes every day.

As you can see, agreeing on the length of a second is not as straightforward as you might think. Currently, the definition of one second is accepted to be the duration of 9,192,631,770 periods of the radiation corresponding to the transition between the two hyperfine levels of the ground state of the cesium 133 atom. That clears things up, doesn't it? This definition refers to the reference used with atomic clocks. Atomic clocks are the most accurate clocks available to the public. They are considered to be the reference by which all other clocks should be set.

If we consider atomic clocks to be the gold standard by which other clocks should be measured, we can also categorize other clocks by how accurate they are compared to the atomic clock standard.

In NTP, each clock exists inside a *stratum*. An actual atomic clock is a stratum-zero time source. A device that serves NTP requests that is directly connected to the atomic clock is considered to be at stratum one. If you set up an NTP server that learns time from a stratum-one NTP server, your server will become a stratum-two server, and so on down the line. Strata in NTP reference how many levels of clocks you are removed from the actual time source. A clock operating in stratum 16 is considered to be unsynchronized.

Strata are defined differently in NTP than they are in telecommunications systems. In telecom, stratum levels reference the holdover performance of an oscillator should synchronization be lost. If an oscillator is accurate to $1.0 * 10^{11}$, it is considered to be stratum-one. Stratum-two oscillators are accurate to $1.6 * 10^{8}$. In other words, the level of accuracy is not tied to the distance from the source clock, as it is with NTP.

NTP Design

NTP is often not designed, but rather implemented in the simplest possible way. Many people mistakenly believe all they need to do is configure a single NTP source and their time problems will be solved. This idea is perpetuated because it usually works. But what would happen to your network if the original time source stopped responding or became inaccurate?

I learned about NTP the hard way. I configured a single time source for the core switches on a large network. I thought I was being clever by having all the other devices on the network get accurate time from my core switches. This meant that only the core switches needed to take up Internet bandwidth with NTP requests, instead of potentially hundreds of other devices.

One day, the time source stopped responding to our requests. However, we never knew of the failure. The core switches (6509s) were still acting as NTP servers, so everyone appeared to have accurate time. In this case, the devices were all close in time to each other, but not to the real time (Coordinated Universal Time, or UTC). Still, the difference between UTC and the time being reported was minor—perhaps a minute different over the course of a few months.

Lesson #1: Always have more than one time source. Not only will NTP failover to another source in the event of a failure, but it will choose the most accurate one available. A minimum of three NTP servers should be configured for core devices.

At some point, we needed to reboot the core switches for some maintenance. The next day was a disaster. Somehow, every device in the network—including all servers and workstations—had decided that it was a day earlier than it actually was.

Lesson #2: Some Cisco devices have a *clock* and a *calendar*. The clock keeps time for the current running session. When the device reboots without an NTP source, it will learn its time from the calendar. Always configure devices that contain calendars to update the calendars automatically from NTP.

When the core switches rebooted, they loaded time from their internal calendars, and used that time to set their clocks. Because I hadn't configured the calendars, they defaulted to the wrong date and time. When NTP initialized, it would have fixed the time had it been able to establish connectivity with an accurate clock. However, it couldn't. Because every device in the company derived its time from the core switches, they all lost a day. This became a problem when programs that duplicated data to backup servers determined that the last save was invalid, as the last save's date was in the future. The company ended up losing an entire day's worth of transactions because of my inability to correctly deploy NTP.

The problem boiled down to my belief that I only needed to configure a single NTP server. I assumed that all time would be accurate from that point on because I had configured the network devices to learn accurate time from a third party.

When designing NTP for a network, I still like to limit the number of devices that will make inquiries from the Internet. Usually, I make the two core switches the NTP servers for the entire network. I configure these two switches (assuming a high-availability pair) to receive time from a minimum of three Internet time servers. The switches also act as NTP *peers*. An NTP peer is considered to be an equal in a stratum. This allows each switch to learn time from the other as an additional source.

I then configure all other networking devices to receive their time from the core switches. Assuming the Internet time servers are all operating at stratum one, the core switches should operate at stratum two. The rest of the switches in the network operate at stratum three. This hierarchy is shown in Figure 35-2.

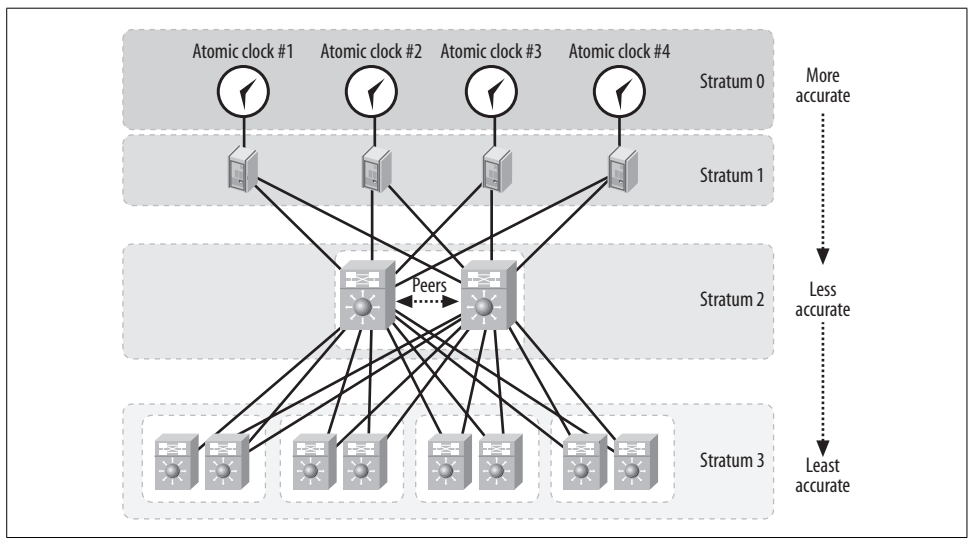


Figure 35-2. NTP hierarchy in a corporate network

Older versions of Windows do not support NTP. Instead, they support Simple NTP (SNTP). Because of this limitation, you cannot use Cisco devices as NTP servers for these older Windows machines without adding a true NTP client to them.

To find publicly available Internet time servers, search the Web for "public NTP servers." Some servers require registration, while others are freely available to the public.

Configuring NTP

NTP is a client/server application. Devices participating in NTP are either NTP servers, which provide time to other devices, or NTP clients, which request time from NTP servers. Servers are also clients, and can be peered with each other as well. Configuring an IOS device as an NTP client is the simpler of the two models, so we'll start there.

NTP Client

To configure an IOS device to request accurate time from an NTP server, use the `ntp server` command. You can list as many NTP servers as you need, each on a separate line. Here, I've listed seven publicly available NTP servers. Using so many will help illustrate how NTP behaves:

```
ntp server 132.163.4.102
ntp server 193.67.79.202
ntp server 152.1.58.124
ntp server 128.118.46.3
ntp server 129.6.15.29
ntp server 64.236.96.53
ntp server 208.184.49.9
```

Once you've configured the NTP servers, you should begin receiving accurate time signals (assuming you can reach them).

To see the statuses of the servers, and the time they are providing, use the command `show ntp associations`:

```
2950# sho ntp associations

      address         ref clock     st  when  poll reach  delay  offset    disp
+~132.163.4.102    .ACTS.            1   373  1024  377   109.0    1.10     0.7
 ~193.67.79.202    .GPS.             1   419  1024  377   282.7   81.38    81.6
 ~152.1.58.124     0.0.0.0          16     -  1024    0     0.0    0.00  16000.
 ~127.127.7.1      127.127.7.1       7    16    64  377     0.0    0.00     0.0
 ~128.118.46.3     0.0.0.0          16     -  1024    0     0.0    0.00  16000.
-~129.6.15.29      .ACTS.            1   539  1024  277    22.4   -4.63     3.0
+~64.236.96.53     .ACTS.            1   255  1024  377    20.9    1.80    24.7
*~208.184.49.9     .ACTS.            1   603  1024  377    17.0    3.61     5.3
 * master (synced), # master (unsynced), + selected, - candidate, ~ configured
```

A lot of information is presented, but you only need to know what some of it means. The address is, obviously, the server you configured. To the left of the address are one or more symbols indicating the status of the server. The key for these symbols appears in the last line of the output; Table 35-1 provides more detailed explanations. The line beginning with an asterisk (just above the key in this example) indicates the server with which the switch has synced its time.

Table 35-1. NTP association characters

Character	Description
*	Synchronized to this peer
#	Almost synchronized to this peer
+	Peer selected for possible synchronization
-	Peer is a candidate for selection
~	Peer is statically configured

The next column indicates the reference clock used for the listed server. This can be useful, but is usually only informative. For example, the first entry shows a reference clock of ACTS (Automated Computer Time Service). ACTS is a dial-up service from the National Institute of Standards and Technology (NIST). The second server shows a reference of GPS, indicating that it is receiving time from Global Positioning System satellites, which contain atomic clocks.

The third column shows the stratum of the server listed. Servers that show stratum 16 are unavailable, and are not providing time.

To see whether your clock is synchronized, use the `show ntp status` command. This command will show you the status of NTP, including the stratum within which you're operating:

```
2950# sho ntp status
Clock is synchronized, stratum 2, reference is 129.6.15.29
nominal freq is 249.5901 Hz, actual freq is 249.5791 Hz, precision is 2**18
reference time is C90B6253.35C5562A (18:08:03.210 EST Sun Nov 19 2006)
clock offset is 4.3896 msec, root delay is 22.31 msec
root dispersion is 13.55 msec, peer dispersion is 9.19 msec
```

If your clock is not synchronized, the show `ntp status` command will explain the reasons why. In this example, I've removed all the NTP servers so that there is no reference clock. Notice that the stratum is now 16:

```
2950# sho ntp stat
Clock is unsynchronized, stratum 16, no reference clock
nominal freq is 249.5901 Hz, actual freq is 249.5790 Hz, precision is 2**18
reference time is C90DA30A.C8104777 (11:08:42.781 EST Tue Nov 21 2006)
clock offset is 4.8738 msec, root delay is 20.45 msec
root dispersion is 4.82 msec, peer dispersion is 1.25 msec
```

Another way to show that NTP is driving the system clock is with the `show clock detail` command:

```
2950# sho clock detail
.11:33:43.755 EST Tue Nov 21 2006
Time source is NTP
Summer time starts 02:00:00 EST Sun Apr 1 2007
Summer time ends 02:00:00 EDT Sun Oct 28 2007
```

NTP Server

To configure an IOS device to be an NTP server, enter the command `ntp master`:

```
6509(config)# ntp master
```

When configuring an IOS device as an NTP server, it can take five minutes or more for the clock to become synchronized. You cannot force this process.

On a device that has a calendar, you also need to enter the `ntp update-calender` command:

```
6509(config)# ntp update-calendar
```

To configure another device within the same stratum as an NTP peer, use the `ntp peer` *`ip-address`* command:

```
6509(config)# ntp peer 10.10.10.1
```

This will allow another device in the same stratum to share time with this device.

There is no way to show the status of the NTP server within IOS.

CHAPTER 36
Failures

Outright failures can often be detected easily; when a router fails completely, there are usually some pretty obvious symptoms. When a WAN Interface Card (WIC) starts mangling packets, the problem can be a little harder to diagnose. In this chapter, I'll cover some examples of what can cause failures, and discuss how to troubleshoot them effectively.

Human Error

Human error can be one of the hardest problems to track, and, once discovered, may be almost impossible to prove. Getting people to own up to mistakes they've made can be a troublesome task—especially if the person responsible is you!

Once, I was working on a global network, administering changes to the access lists that allowed or denied SNMP traffic to the routers themselves. We'd just added a new network management server, and we needed to add its address to all the routers so they could be monitored and managed from the central office.

Instead of properly writing the configurations ahead of time, testing them in a lab, and then deploying the proven changes during a change-control window, I decided that because I was so smart, I would apply the changes on the fly. The changes were minuscule—just one line—what could go wrong?

Naturally, I bungled something in one of the routers, and ended up removing an active ACL on the inbound interface. The router was the sole means of entry into the entire continent of Australia for this company. Because nothing simple ever happens to me, the disaster struck on a Friday night (Australia time) on a holiday weekend. No one could be reached on-site until the following Tuesday. For three days, the global financial institution was unable to access any of its servers in Australia due to (my) human error. I like to chalk events like this up to "experience."

A neat way to prevent this sort of disaster is with the `reload` command. With this command, you can make a router reboot in a set amount of time. Because the configuration that caused the router to become unavailable will not have been saved, having the router reboot automatically will bring the router back to life.

To reboot the router in 15 minutes, issue the `reload in` command, as shown here:

```
TS-1# reload in 15
Reload scheduled in 15 minutes
Proceed with reload? [confirm]
```

Alternatively, you can specify a specific time at which to reload the router with the `reload at` command:

```
TS-1# reload at 12:05 July 10
Reload scheduled for 12:05:00 UTC Mon Jul 10 2006 (in 566 hours and 33 minutes)
Proceed with reload? [confirm]
```

Be warned that if you've already started to make your changes, the router will prompt you to save them before you reload.

Remember to cancel the reboot when you've successfully made your change! To cancel an impeding scheduled reload, execute the `reload cancel` command:

```
TS-1# reload cancel
TS-1#

***
*** --- SHUTDOWN ABORTED ---
***
```

Multiple Component Failure

Many networks are designed to avoid single points of failure. What usually brings these networks down is multiple component failure. Multiple component failure can be triggered by a root cause such as dirty or unreliable power, or it can just be a fluke. One device can also sometimes cause failures in other devices.

Sun servers used to be programmed to halt when a break character was sent to them. Many people used Cisco routers as terminal servers that allowed remote connections to the consoles of these servers. One of the issues that was discovered with this combination was that when the Cisco routers were powered off and on, break signals were sent out all of the serial lines connecting the Sun consoles. So, if the terminal server failed, all the Sun servers halted. The parallel discovery was that having all the servers offline was not good for business. Sun's newer servers do not halt on a break signal, but, if you're using older Sun systems, beware of using Cisco routers as remote console devices.

See Cisco field notice #15521, titled "Terminal Server Break Character on Cisco Access Servers," for more information on this problem.

Sometimes, multiple devices fail for reasons known only to them. In one installation, I saw a dual-supervisor 6509 have a compound failure where the primary supervisor failed, but the primary MFSC stayed active. Because the MSFC is tied physically to the supervisor to get connectivity to the networks in the switch, the entire location went down, even though the secondary supervisor was up and had failed over properly. The MSFC that was still active on the failed supervisor had no networks to route. Problems like this don't normally happen when using newer versions of IOS, but then, I would have told you at the time that what we were seeing was not possible, either.

Disaster Chains

Modern commercial aircraft are some of the most amazingly redundant systems on Earth. When one device fails, another device takes over. When two systems fail, a third can take over with a reduced capacity, and so on. Commercial airplanes even have multiple pilots! Still, these testaments to fail-safe engineering can and do crash. When they do, the crash is usually discovered to have been caused by a series of events. These events, all relatively innocuous in and of themselves, spell disaster when strung together. Such a chain of events is called an *accident chain*.

Massively redundant networks can suffer from accident chains, too. Considering the impact such compound networking failures can have on a business (not to mention one's paycheck), I like to use the term *disaster chains* to describe them.

Imagine a network with two Cisco 6509 switches in the core. An outage is planned to upgrade one, then the other. Because they're in a redundant pair, one can be brought down without bringing down the network. The first switch is brought down without incident. But, as I'm working, I manage to get my foot tangled in the power cord of the other 6509, and pull it out of the power supply. Of course, the 6509 AC power supplies allow the power cables to be secured with clamps, but the last engineer to work on the switches forgot to retighten the clamps. Each 6509 has two power supplies, which are connected to different circuits, so pulling one power cord should not be an issue. However, the engineer who forgot to retighten the clamps decided that the cables would look neater if he wire-tied them together. When I tripped over the cable, it was really both cables bound together, which means I ended up unplugging all the power from the 6509 that was supporting the network. The result: complete network failure.

To recap, the following events occurred to cause the outage:

- I was born completely without grace or coordination.
- The last engineer to work on the system tied the power cords together and left them out where I could trip on them.
- The last engineer did not clamp the AC power cords to the power supplies.
- I shut down one 6509.
- I tripped over the cords to the remaining 6509.

Any one of these events in isolation would not have caused an outage. In fact, any two would probably not have caused an outage. If the power cords had been left unclamped, but the cables were not tied together, my big clown-sized feet would probably have only pulled one cable, leaving the switch with one active power supply.

Disaster chains can take some digging to uncover. In the case of me tripping over the power cord, the exec who was grilling me asked, "How could this happen?" Having an answer to a question like this can save your job, especially when the answer appears to be as simple as you being a klutz.

No Failover Testing

I once worked with a team designing a large e-commerce web site infrastructure. When I say large, I mean eight Cisco 6509 switches serving more than 200 physical servers (most with multiple Solaris zones), providing upwards of a gigabit per second of content. Timelines were tight, and everyone was stressed. In all the compression of timelines that occurred during the life of the project, one of the key phases eliminated was failure testing.

After the site went live, a device failed. The site was designed to withstand any single point of failure, yet it stopped functioning properly. It turned out the failover device had been misconfigured in a way that only presented a problem when the active device failed. Because the failure caused a loss of connectivity to the site, we had no way of getting to the failed equipment, except to drive to the collocation facility. This failure, which should not have been possible, resulted in a two-hour outage while someone drove to the facility with a console cable.

Had failover testing been done, the problem would have been found during testing and the outage avoided. The design was correct, but the implementation of the design was not. Always insist on failure testing in high-availability environments. Failure testing should be done on a regular basis, and included in normal maintenance at scheduled intervals as well.

Troubleshooting

Entire books have been written on troubleshooting techniques. I've seen people who are natural troubleshooters, and people who aren't. Some people can seem to smell the source of a complex problem, while others can't figure out what's wrong even when *they* are the cause of the problem.

The most interesting problems are usually the ones that cause the most damage. These are the problems that can make or break your career. I've been in the middle of web site failures where downtime means zero income for the company for the duration of the outage. I've worked through failures in banking networks, where

each minute of outage costs millions of dollars in lost trades. The best resolutions were the ones that happened quickly and weren't necessitated by my mistakes. The ones that were my fault were identified as such as quickly as possible. People make mistakes. When people try to hide their mistakes so they won't be identified as the causes of outages, the outages often last longer than they would have if the people troubleshooting had been properly informed.

Regardless of the problem or the situation, there are some things to remember when troubleshooting an outage. Here's my short list.

Remain Calm

I once worked with a former Marine Sergeant. He had been in combat, and had lived though months of rehabilitation after a gunshot wound to his shoulder. He was now working for me as a senior network engineer supporting a global network that included more than 10,000 nodes.

One day we were in the middle of an outage that had the vice presidents standing over his shoulder while he worked to isolate and resolve the issue. He was getting more stressed by the minute, and I could see that it was hampering his ability to troubleshoot effectively. As his boss, the first thing I did was to usher away the executives with the promise that I would give them updates every 10 minutes. The second thing I did was to write on his whiteboard, "Relax—no one is shooting at you." He smiled at me, and became visibly more relaxed. We had the problem isolated and resolved in time for the next executive update.

Network outages usually do not put lives at risk—including your own. In extreme cases, they may put your job at risk, but you'll still be able to go home and hug your kids, your spouse, your dog, or your Xbox, if that's what makes you happy.

The more stressed you allow yourself to become, the longer the outage will last.

Log Your Actions

When troubleshooting an outage, every time you do something, write it down and record the time. Eventually, you'll need to document what happened in a post-mortem report. Keeping a log of your actions will prove invaluable later. It will also serve as a way for you to keep track of what you've already done so you don't waste time repeating steps that were ineffectual.

Find Out What Changed

Outages do not happen without cause. Even if there has not been an obvious device failure, if everything has been working—and now it's not—something has changed. In some environments, people will confess, and in others, they won't. Either way, the

first question you should ask is, "What changed?" If something changed just before the outage, there is a good chance that the change is related to the outage. Figure out if you can reverse the change, but make sure you know before you act what the impact of your reversal will be.

Outages can be caused by seemingly unrelated changes. When I used to work with mainframes, I witnessed some pretty strange outage scenarios. One customer complained that his mainframe was crashing every night between midnight and 1:00 a.m. There were no obvious causes for the crashes, but they happened every weeknight, and had only started about a month earlier. While the customer was very helpful and understanding, backups were not completing because of the crashes, and this was causing problems. Finally, we sent someone out to the site to literally sit with the system for a week to see what was going on.

The person we sent to babysit the system noticed a police officer parked in the company's parking lot every weeknight at around midnight. When the officer left the parking lot, he called the dispatcher on his radio. As soon as the officer pushed the button on his radio microphone, the system crashed.

We determined that the system administrator had disconnected a group of terminals from the mainframe about a month prior to our visit. He had removed the terminals, but had left the serial cables connected to the mainframe. These cables were acting as antennas. When the police officer keyed up his radio, the serial cables received the signal, and the resulting electrical impulses entering the mainframe caused it to crash.

If you can't figure out the cause of a problem or outage, look for any changes, no matter how inconsequential they may seem.

Check the Physical Layer First!

My boss taught me this rule when I worked at an ISP. He used to teach troubleshooting to telecom field engineers for one of the RBOCs. The idea here is that most failures are caused by physical faults. Cabling or hardware should be suspected first, and, between the two, look at the cabling first. Once you plant this idea in your mind, you'll be amazed how often it's borne out. To this day, I can hear him barking "Physical layer first!" at me during outages.

Assume Nothing; Prove Everything

When you assume something, it will return to bite you. As soon as you hear someone (including yourself) say, "It can't be that because…," set out to prove that statement true or false. During outages, there seems to be a tendency for engineers to convince themselves that something must be true because they think it is. Don't assume anything is true unless you've proven it.

Isolate the Problem

Problems are often represented by multiple symptoms. Problems are also sometimes caused by other problems. For example, you may think you have a complex routing problem because packets are flowing in a way you think is wrong. However, the problem might be caused by an interface failure. You could spend hours trying to figure out a routing problem in a complex network, while the root cause is a simple interface failure. Try to isolate problems to root causes. Though compound failures can and do happen, single failures are far more common. Rule things out systematically and logically, and you should eventually discover the root cause.

Don't Look for Zebras

I once had a doctor who told me, "If you hear thundering hooves, don't assume they're zebras." This statement is similar to Occam's Razor, which is usually paraphrased as, "With all things being equal, the simplest answer is usually the correct answer." Just because I was getting a lot of headaches didn't mean I had a brain tumor.

The same principle can be applied to any problem you're trying to solve. If you're seeing packets that have bits transposed, it's probably not due to aliens altering your data in an effort to destroy the world. I'd check the cabling first. NORAD won't take my calls anymore, anyway.

Do a Physical Audit

If you've been going round and round in circles, and nothing is making sense, do a physical audit. Chances are that assumptions are being made, and an audit will disprove those assumptions. Sometimes, the documentation is outdated, or simply wrong. When in doubt, redraw the networking diagrams from scratch. Using a whiteboard or a legal pad is fine for this exercise. You're not looking for beautiful documents, but rather, factual data. In the mainframe example I described earlier, a physical audit would have solved the problem because we had standing orders to remove unterminated serial cables.

I once got involved in a network problem after another team had worked on it for more than a week. The network was deceptively simple: the nodes involved included two T1s, four routers, and two servers. A team of people could not find the problem, no matter what they tried. When I got involved, I looked over their notes, and instructed them to do a complete audit of the network, down to every device, interface, and IP address. I also instructed the team to test connectivity from every node to every other node in the network. Within an hour, I was informed that they had discovered the problem. A T1 WIC had gone bad and was deforming packets. The problem did not become obvious until they'd run through the process of auditing the network in detail. By testing connectivity between all points in the simple network, they soon discovered the root cause.

Escalate

If you can't figure out what's wrong, escalate to someone else. Usually, your boss will want you to escalate to Cisco if the network is the problem (and Cisco is your vendor). If you have an internal department you can call first, by all means, do that. If you feel the problem is beyond you, don't waste any time—call for reinforcements.

Troubleshooting in a Team Environment

When there is a team of people troubleshooting the same problem, someone needs to be the leader. I could write an entire book on this subject alone. If you find yourself troubleshooting as part of a team, work only on the piece of the puzzle you've been assigned. If you try to fix someone else's piece, you're wasting your time as well as his.

If someone else is trying to solve a problem, standing over his shoulder and yelling out ideas will not help. If you're sure of the answer, but no one is listening, find a way to prove your solution, and push to be heard.

The Janitor Principle

The Janitor Principle states that explaining your problem to someone who doesn't understand it will cause you to make connections you've previously missed. This is an amazingly powerful tool. To explain a complex problem to someone who doesn't understand it, you need to reduce the problem to its simplest elements. This action forces you to think about your problem from a different viewpoint. Looking at a problem differently is often all it takes to find a solution.

CHAPTER 37

GAD's Maxims

Maxim #1

Over the years I've been in the industry, it has become apparent to me that there are certain driving forces in the IT universe. These forces are evident in just about all aspects of life, but their application is never more evident than it is in IT.

In every situation where an engineer does not get to do what she wants to do—or worse, in her eyes, to do what she believes is right—these forces come into play. I believe that if more people understood them, there would be less conflict between engineers and their superiors.

The driving forces of network design are summarized here:

- Politics
- Money
- The right way to do it

GAD's Maxim #1: Network designs are based on Politics, Money, and The right way to do it—in that order.

Figure 37-1 shows it all in a nutshell. The idea is simple, really. Engineers want to "do it the right way," which is usually considered best practice. To do whatever it is "the right way," money will be required. To get money, someone will have to be adept at politics. To put it another way, if you want to do it the right way, you need money, and the only way to get money is through politics. I can hear your voices in my head as you read this, groaning, "I hate politics." The truth is, you don't hate politics—you hate *dirty* politics. There is a distinct difference between the two.

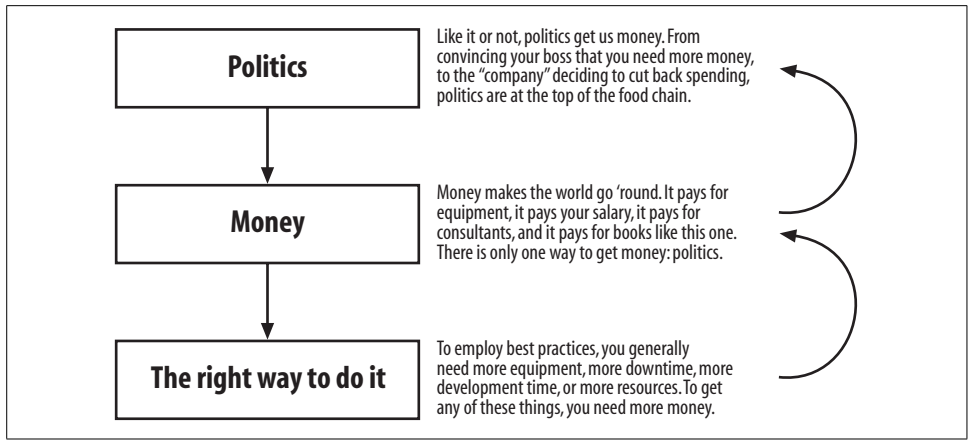


Figure 37-1. GAD's three rules of network design

Let's take a closer look at the three elements and how they're interrelated:

Politics

Politics isn't just about the president stumping for votes, or the vice president hiring his brother rather than the best person for the job. Politics is, among other things, *intrigue or maneuvering within a political unit or group to gain control or power*. A group, like it or not, is two or more people. Every conversation you have with another person is political in some form. Every time you say hello to someone, your attitude and your demeanor help shape that person's opinion of you. This is politics. If you have the best idea the world has ever seen, and you go in to tell it to your boss so she can get the funding for you, how supportive do you think she'll be if only yesterday you called her a pencil-necked knucklehead? Would she be acting politically if she turned you down? Maybe you were being political when you assaulted her with badly formed insults.

My point is that any time there are two people in a room, politics comes into play. If you have a reputation for being difficult, when you ask for something, that will be considered (whether consciously or not). Conversely, if you have a reputation for being levelheaded and easy to work with, that can work in your favor.

Now that you understand that politics are not always bad, let's look at the realities of how politics may affect you. I've already stated that without politics, there will not be money, but what does that really mean? Here's an example.

A mid-sized company is considering a new network, and the vice president of technology, who was a developer for his whole working life until getting into management, is in charge of the project. As the lead engineer, you propose a completely fault-tolerant design consisting of $2 million worth of equipment, which will require six months of work to implement. The plan is perfect. You know every detail of the hardware, and how it will all work. It will be a masterpiece.

The VP takes a look at your proposal and says simply, "It's too expensive, and it will take too long. The budget is only $1 million for equipment and consulting."

You feel as though the wind has been knocked out of you. Your first response is probably something along the lines of, "But this is what we need," or "But we can't do it for less than this!"

This is politics. The VP may have the power to get the additional funding. In fact, that's usually what VPs are best at. If you can explain to the vice president all of the benefits of the design, and why you feel the $2 million in hardware alone will be well spent, she may listen. Better yet, if you can explain how the company will get a return on its investment in only six months because of your design, you've got a fighting chance!

If instead you sulk and whine about how the company is mismanaged and only cares about money, and how the VP doesn't understand the technology, the VP will likely think you don't understand reality, and are incapable of working within real constraints. You may both be wrong, but either way, you won't get your funding. This happens every day all around the world.

Here's a news flash: if you're going to present ideas to VPs and expect to gain their respect and backing, you need to learn to play on their field. Like it or not, they have the money, the power, and the influence. If you don't know what an ROI is, or how to talk about depreciation of equipment, don't expect VPs to give you more money.

Money

Money is what we all want. As I've already said, you need to understand politics to get the money you want or need. There is no way around it. If you're getting money, and you're not involved in the politics, someone must be rooting for you behind the scenes.

If there is not enough money, chances are you will not be able to do what you want (i.e., do it "the right way").

The right way to do it

This is the goal engineers pursue. "It" is whatever they happen to be working on at the time. Engineers know what "the right way" is, and they are often frustrated because they're "not allowed" to do "it" the right way.

If you're not willing to play the game, my advice here is, "Get over it." You simply cannot do anything "the right way" without money, and you will not get the money you need without politics.

If your company is short on cash, you're out of luck. If your company is in the middle of political turmoil that is preventing you from getting money or approval, you are out of luck. If you want to change things, learn politics. It's that simple.

Some of the best advice I ever received on the subject of politics came from the book *How to Win Friends and Influence People*, by Dale Carnegie. I recommend this book for anyone who's ever been frustrated by human interaction in any form.

Maxim #2

Many years ago, I was a manager at a large company. The network was in trouble: there were constant failures, due to a poor existing design, as well as political issues galore. I had a group of very smart engineers working for me, and they all had ideas about how to make things better. The engineers all believed their changes should be implemented, solely because they thought their ideas were sound (some were; others weren't). My problem was how to get the engineers to understand that they could not simply make any changes they wished.

In an effort to get the engineers to understand that not all changes are good changes, and to encourage them to think about the implications of the changes they were proposing, I came up with three rules for them to follow. The rules were simple—each only a word. In order for a change to be considered by me, it had to follow one of the three rules. The rules were:

- Simplify
- Standardize
- Stabilize

GAD's Maxim #2: The only valid reasons to change a properly sized production network are Simplification, Standardization, and Stabilization.

If the proposed change did not accomplish one of the three stated goals, I would not even consider the change request. At first, the engineers were constantly frustrated. They wanted to effect change, but they were not being allowed to do what they thought was needed. Over the course of about a month, however, they started to catch on. Within six months, we had so greatly improved the network that it was no longer the source of any problems. With a stable network came stable services.

Long-term thinking is the only way to accomplish stability in an unstable environment. Enforcing these three simple rules caused the network to go from being a point of ridicule for upper management to being a source of pride.

Here's a closer look at the three rules:

Simplify

This one is pretty straightforward. To comply with this rule, a proposed change must simplify the network or its operation in some way. Examples might include removing equipment, or replacing a complex IP scheme with one that's easier to understand or maintain. Perhaps the change will cause all of the routers to have the same revision of code. Anything that makes the operation or troubleshooting of the network less complex saves time and money. Simple is good. If you can easily explain how a change will enhance the simplicity of the network, chances are the change is a valid one.

Standardize

Standardization can also make networks more cost-effective and easier to maintain. If you have 200 routers, and they're all the same model, you will know what to expect from each router when you connect to it. While that may not be realistic, having a standard device for every function, and standard ways to deploy services, is. You can make all of your edge routers, the core switches in every building, and so on, the same type. If you deploy NTP, it should be deployed the same way everywhere.

Standardization allows for spares to be more readily available, and it allows for devices to be swapped in from less critical places in an emergency.

Stabilize

How to enhance stability is not always obvious. Perhaps the revision of IOS code that has been standardized on all the VoIP gateways has been reported to have an issue that might affect your environment. That code should no longer be considered stable, and should be updated. Likewise, if there is a piece of hardware that has caused problems in the past, it should be replaced. These sorts of changes are valid, and will increase the network's stability.

Maxim #3

There are a finite number of reasons why a company will fund changes to an existing network. The reasons are surprisingly simple and fairly obvious:

- Lower costs
- Increase performance or capacity
- Increase reliability

Unless you can prove that your idea will meet one of these goals, chances are, you will be denied the funds you need.

GAD's Maxim #3: Companies only spend money on IT projects that lower costs, increase performance or capacity, or increase reliability.

Let's explore the three goals:

Lower costs

I once worked for a large telecom company that had DS3s crisscrossing the globe. The network was perfection in terms of design. Every node had multiple network entry points, latency was acceptable, resiliency was excellent, and downtime was extremely rare. The network was as close to perfect as we could make it, and my team and I were very proud of the work we had done to make it so.

One day, word got to me that the network needed to be changed. More specifically, I was told we needed to cut $1 million from the budget. The choices were telecom or personnel. Welcome to the world of business—I was told to either destroy my perfect network, or choose who should be let go.

My team and I went to work, and we managed to redesign a network with a $12 million annual cost into one that was almost as good, but cost only $10 million annually. We were never as proud of this network because it involved a lot of compromises that we didn't believe in, but the company saw this as a victory.

The point of this story is that the changes we made did not benefit the company in terms of resiliency or performance. Instead, the changes were approved because they saved the company money. In the eyes of the company, *good enough* was better than *perfect* because it saved $2 million a year.

If you have an idea for a network change, one of the best ways to get it approved is to show how it can save money—especially recurring expenses, such as telecom costs, maintenance contract costs, and consulting fees.

While my example was one of a forced change rather than a desired change, the point is that cost is a high priority for a business. If you have an idea that will save the company money, chances are, management will listen. From the point of view of most managers, all engineers want to do is spend money. Producing an idea that will actually save money will make you stand out from the crowd.

Remember that there are different types of costs that a business must deal with. Examples of these costs are:

Equipment costs

Hardware, cabling, and the like are examples of equipment costs. If you can replace three routers in a planned design with two, you've reduced equipment costs.

Recurring costs

Maintenance contract costs and telecom costs such as T1 bills, telephone bills, and the like are all recurring costs. Cutting recurring costs can have a dramatic impact on a company's bottom line.

Human resource costs

Your salary is a cost for the company. If there are 10 people in your department making $100,000 each, and you design a change that costs $200,000, but lowers your staffing needs by 2 people, after less than a year, the company will have made a return on its investment.

Engineers are not often aware of what it costs to keep them. For example, did you know that the taxes taken out of your salary are only a portion of those required by the government? If you have a salary of $100,000/year, the company might pay as much as $25,000 a year in taxes above and beyond what you pay. Suddenly, you're costing the company $125,000 a year. And let's not forget other things we all take for granted, such as 401k contributions and healthcare. Those two alone could total another $25,000 a year. Now you're costing the company $150,000 a year. Do you get a bonus? Add that to the total. And let's not forget that it costs money for you to have a cube or office (rent). The company also has to pay to cool and heat the air you breathe (facilities), and for you to have a computer, a telephone, and probably a cellular phone. When all is said and done, you may easily cost the company twice what you consider to be your annual salary.

Consulting costs

Consulting costs are often reported differently from HR costs (this is why sometimes salaried employees are let go, and consultants are brought in to do the same work for the same or more pay). If you can redesign a network that currently is supported by three consultants so it only needs one, you've lowered consulting costs.

Increase performance or capacity

Increasing performance is rarely a bad thing. The exception might be improving performance where no increase is necessary. Remote branches with 10 users probably don't need PIX 535 firewalls capable of a gigabit of throughput.

On the other hand, devices that are near their limits will soon become problems if normal growth continues. Telecom links that are running in excess of 70 percent utilization should be upgraded. Routers, switches, and firewalls that are regularly running at more than 50 percent CPU utilization should be considered for replacement with larger devices.

Voice over IP is an excellent example of how performance bottlenecks can be serious. If your gateways can only support 50 concurrent calls, and you suddenly become busy due to growth and start dropping customer-service calls, you can bet that the money will be found to upgrade those gateways so they can support more calls.

The smarter engineers will watch the network and spot trends. If you can quantify a trend that will translate into a required upgrade in the next budgetary cycle, you will be very well received because the change can be planned for and budgeted for ahead of time. Remember, managers don't like surprises. Be careful to try and spot these opportunities before there's a problem. Telling your boss the network should be improved after an outage has occurred will not make you a hero.

Increase reliability

Chassis-based switches are more reliable than 1-RU access switches. Why? Because they usually have dual power supplies, and can often support dual processors. Reliability is an important consideration in any network design. Reliable components such as chassis-based switches are usually more expensive than their less reliable counterparts, though, so smaller companies usually do not employ them.

As smaller companies grow into mid-sized and larger companies, however, they often discover that unplanned outages cost them more money than they would have spent on more reliable equipment. At this point, they start to spend money in an effort to increase reliability.

Another example of increasing reliability would be replacing a single PIX firewall with an active/standby pair.

Reliability can be an easy sell if your company has experienced an outage due to a single point of failure, or your management has experience with such a failure. Sure, there are those who understand that single points of failure are bad, but those who have lived through the experience of an outage caused by lack of foresight generally are more receptive to spending money to avoid having such an experience again.

While citing any one of the above reasons will make your change more desirable, combining the first (lower cost) with one of the others will make your idea as close to a sure thing as possible. If you can manage to come up with an idea that combines all three, you will be lifted upon the shoulders of management and paraded through the halls like a conquering hero.

If, on the other hand, you know what needs to be done to increase performance or reliability, but you don't let anyone else know, when the failures happen they will likely be viewed as your fault. Most engineers know what needs to be done to ensure a healthy network. Using these tips, you should be able to convince your boss that the changes are worth implementing.

CHAPTER 38

Avoiding Frustration

I've been working in the computer and networking industries for more than 20 years, and in that time I've witnessed people getting frustrated about the same things again and again. I've learned a lot in that time, and the lessons haven't always come easily. My aim in this chapter is to try to help you avoid some of the frustrations that you're likely to encounter in the course of your work, and thereby make your job more enjoyable.

Why Everything Is Messed Up

I can't tell you how often I've heard this question: "Why is everything so messed up?" Of course, the language is often more colorful that that, but decorum prohibits me from including examples here. Suffice it to say that the phrase "messed up" is often replaced with something more satisfying when used in moments of frustration and anger.

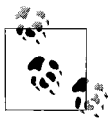
The situation is a common one. A company has grown to the point where it needs a reliable network. Management decides, for whatever reason, that the in-house staff is not able to get the network where it needs to be, so they call in consultants. That's when I usually arrive and am greeted by the question above. Sometimes it comes from employees at the company, and sometimes it's posed by the junior consultants working with me.

The answer to the question is usually based in a set of simple truths.

Truth #1: Companies that start small and experience intense growth often end up with complicated networks that cannot scale to the new demands placed upon them.

Small companies that grow to be large companies often do so at a fast pace. Companies that are quick to solve problems are often the ones that do well in today's economy. When these companies are small, they hire computer professionals with server experience. Networking professionals are not usually required at smaller companies, and the server people do a great job of keeping the place running.

As the small companies grow, the networks grow with them. The skilled server engineers, who are experts at server scalability and design, may not be experts at network scalability and design. This often leads to networks being pieced together to solve immediate problems, rather than being planned carefully and built with an end state design in mind. This in no way should be held against the server engineers. They have risen to the occasion, and performed miracles above and beyond the call of duty. The problem usually lies in the fact that they are server engineers and not network engineers. Remember that small companies don't tend to require network engineers at first. Many find it hard to accept that they need them as they grow larger.



Truth #2: Networks are often pieced together instead of designed.

Networks are often viewed as nothing more than a means to get the workstations talking to the servers and the Internet. Especially in smaller environments, the network takes a backseat to server and workstation issues. This is related to Truth #1, but is more a case of management not understanding the value of the network.

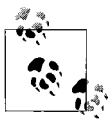
Network infrastructure is often viewed in the same category as buildings, chairs, and such. That is to say, the network is not perceived to be a competitive advantage, but rather, a necessity that consumes capital. Because it's not a profit center, why waste money on it? That is the perception a lot of management has regarding networking equipment and staffing. This attitude is usually prevalent in smaller, nontechnical companies, though it is certainly found elsewhere as well. As the network is often an afterthought, training is usually not a high priority, either. And people who have devoted their careers to servers are likely to choose further server training over network training.

Truth #3: Network design is not high on upper management's list of priorities.

The upper management of many small- and mid-sized companies often doesn't understand the impact a poorly built network can have. While news of the latest viruses and server security holes is plastered on the front pages of CNN and the *New York Times*, the networks usually chug along without any fanfare. When the networks fail, these managers usually wake up, at which point it's too late and major changes must be made. Again, this is when the consultants show up.

I have spent weeks designing elegant IP schemes for large companies. The scheme for one client was so impressive as to be described only as magnificent. It was a masterpiece. The scalability was measured in decades, and the flexibility was unheard of. It was so impressive that only the two of us who designed it understand it to this day. As far as I know, it is still in place. The principle of the design, like gravity (if I may be so bold), did not need to be understood to be implemented.

The moral here is that we spent weeks on the scheme, which cost the client quite a bit in billable hours. The benefits were not obvious, and the time spent was questioned. The benefits were not obvious because they were intangible. Measuring the amount of time that will be saved by never needing to renumber servers is impossible. How much future time and effort will be saved because servers of similar types can be referenced by a single access list entry? What about locations? Countries? We designed the scheme so that every component of the network could be aggregated into a larger chunk. We designed it so that entire locations could be moved, and only the second octet would change. But all of the time saved was *potential*. You cannot bank potential time saved. This makes IP scheme design a hard sell to management. But for the staff who work with the network on a daily basis, a great IP scheme can make life substantially easier.



Truth #4: Engineers are not always adept at communicating needs to management.

Management's view of the network is typically based on a combination of personal experience and input from the engineers. If the engineers are not skilled at communicating with their managers, perceptions are affected accordingly. This really complements all the other issues I've listed. If an engineer knows what needs to be fixed, but can't get his manager to understand the need, the manager won't get the funding for the resolution.

I once went to a customer's site to investigate client/server problems. The customer was an advertising agency that used very large images, and there were complaints that it was taking too long for the 80 MB graphics files to move from the server to the workstations (this was in the late 1990s, when 80 MB graphics files were unheard of). The problem was a result of having too many devices on a network that was based on hubs. I advised the replacement of at least the central hub with a switch. But back then, switches were expensive, and only the largest of companies could afford to make such a change.

The management refused to replace anything on the network, and insisted that I look further into the server. They felt that something had to be wrong with the server because that was what they understood. They had no idea how networks operated, and that there were limits to how much traffic they could handle.

I gave them a small lesson on networking, and explained why they were having problems. I offered them a guarantee that if the introduction of a switch did not resolve their problems, we would take the switch back at no cost. The switch did solve their immediate problems, and I made a regular customer into a loyal customer.

Small companies like that one focus on what they're good at. In this case, the client was good at advertising. They didn't care about computers or networks, but they understood computers because they used them on a daily basis. They had no daily interaction with the networking equipment; it just worked, until it started to become a problem.

Truth #5: Sometimes, the problem is simply that no one knows or even cares that there are problems.

People in advertising should not need to learn how a switch works. The network is there simply to facilitate the work they need to do.

How to Sell Your Ideas to Management

You've got a great idea that you believe will change something for the better. Maybe it's the network, or maybe it's the way the company hires and interviews people, but you know it's worth doing. The problem is that you can't get anyone in management to listen to you. In your mind, they're all idiots. In their minds, you're wasting their time. Who's right? You both are.

Good managers want to know what you're thinking and will listen to you. If your idea does not have merit, they will give you meaningful feedback, and help you refine it. Other managers worry only about rocking the boat, and will not entertain any notion that upsets the status quo. If your manager is of the second type, you can either work to convince him that change is good (at least, as it pertains to your idea), or bypass him altogether. In the latter case, my advice is to ask permission, or at least inform your boss, if you plan to go over his head. No one likes to be blindsided. If you sneak around, you'll only make an enemy.

Engineers are full of great ideas. Making things better is what we're built to do. However, when it comes to engineers, there seems to be an inverse relationship between great ideas and great communication skills. This is what gets us into trouble.

While sitting in a meeting, I once got everyone's attention by announcing that the entire network was about to fail. We were paying out millions in penalties for service-level agreements that we were not meeting. The room was tense, and I had just raised the tension with my statement. The focus was on me, and I was ready to make my move. I explained to the long table of executives how their failure to make the right decisions had caused the mess we were in. I pointed out specific individuals,

and made it clear how they were to blame. I was on a roll, and I let it all out. I was their superhero, and I could save their world. That was my message, delivered from on high as the mighty engineer-god, and woe betide anyone who dared not listen.

My view of the world was absolute. Certain people had been shortsighted, and thus, cheap, which had caused failures, which in turn had caused us to have angry customers. I could fix it all if they'd just shut up and let me do what they'd hired me to do.

Does any of this sound familiar?

I had trouble understanding why my recommendations weren't taken as gospel, and why the executives didn't throw themselves at my feet and lavish me with gifts (I like Ferraris). It wasn't until I'd finished my rant that one of the executives pulled me aside and explained that I had not only made a complete fool of myself, but also destroyed any credibility I might have had with every person in the room.

I've learned a lot since then. Hopefully, I can help others learn from my mistakes. What did I do wrong? *Everything*. First and foremost, I did not speak with respect. Forget about respecting people because of their positions. I don't believe in that. I mean respecting people because they are human beings. If anyone ever talked to me the way I talked to that roomful of executives, I'd get up and walk out. Yet somehow I felt justified in speaking to them like I was their father. I also got emotional. I looked down on the whole group and then started ranting. That will never get you where you want to go in the business world.

So, how do you talk to management? How do you translate your internal engineer's rage into useful business-speak that will get the results you want? Here are some tips:

Document your idea

If your idea is sound, your boss will want you to give him something that he can show his boss. This should be a small document outlining your idea, along with the benefits of its implementation. It should begin with an executive summary containing a minimum of technical jargon, followed by a few pages of concise technical explanations relating the details. Include timelines and budgets—even estimates are better than nothing. If you don't know how to do these things, enlist the help of someone who does.

Be grammatically correct

When producing documentation, make sure that everything you write is grammatically correct. Make sure you use a spellchecker on your documents and proofread them *at least* twice. Have someone else proofread them as well, then proofread them again yourself. You'll be amazed at how many mistakes you'll find on even the third reading.

I've had the pleasure of working with some very smart people. I've worked with CCIEs, PhDs, teachers, doctors, lawyers, scientists, and many other people with far more education than I have. I am constantly amazed by the unreadable documentation produced by people of all ranks. If you want to be taken seriously, make sure your documentation is grammatically sound and looks professional.

Take emotion out of the equation

Any emotion involved in the delivery of your message will be held against you. The only possible exception to this rule is passion. Feeling passionate about your ideas is a good thing. Keep an eye on your passion, though, as it can easily turn into anger or frustration.

If you're like me, you feel passionate about the things you believe in. As an engineer, it's very easy to believe in an idea or concept. We all know that following certain best practices can make our lives easier, but I've seen engineers lose their cool because a VP approved only one DNS server instead of two. A calm engineer should be able to discuss the benefits and drawbacks of redundant DNS servers, and perhaps convince the VP to change his mind. An emotional engineer will simply be ignored. As soon as you inject emotion into your presentation, you will lose credibility.

I like to argue. I don't mean that I like to scream at my wife while she hurls cooking utensils at me. I like to argue in the strict definition of the term. I like to debate a proposition with an opponent whose premise differs from mine. If I'm arguing with you, and I can get you to become emotional, I've practically won. Why? Because when emotion swells, reason abates. The more emotional you become, the less reasonable you will be.

If you become emotional while presenting an idea, everyone around you will know you're not being rational. Irrational people are, at best, ignored. Passionate people are revered, though, so where do you draw the line? Being excited about your idea is a good thing. Being belligerent when offered a counterpoint is not.

Be polite

Above all, be polite. If you are anything less than polite, it's probably because you've allowed yourself to get emotional. Regardless of the reason, a failure to be polite is a failure to be respectful. If you don't treat others with respect, chances are, others will not treat you with respect, either. Politeness is a form of respect you should practice with everyone, but if you fail to practice it with your superiors, especially when presenting a new idea, chances are you won't get the results you desire.

Be succinct

Managers don't have a lot of free time, and any time they devote to you and your ideas should be used thoughtfully. They'll be looking for a summary of your idea and the benefits it offers. Once they're interested, you will be able to go into detail, but you have to start with the sales pitch. If your idea if valid, you will have plenty of time to expound on the details later.

Understand the shortcomings of your ideas

No ideas are perfect. If you don't know the possible arguments against your idea, someone else will produce them. Chances are they'll assault you with them during your presentation. A professional knows the limitations and shortcomings of his ideas, and has answers prepared when questioned about them.

This point is so often missed that you can often make a big impact with executives just by addressing their questions before they have a chance to ask them.

Accept "no" as an answer

Don't whine or complain if the answer is "no." This would be unprofessional, and would further discourage management from considering your ideas in the future. Sometimes there is not enough funding, and sometimes the timing is not right. Sometimes the answer will be "no" for reasons that you cannot be told. Regardless of the reasons, accepting defeat gracefully will add credence to your next request. It is perfectly permissible to ask why your idea was rejected, but don't press the point.

Escalate

Sometimes, when you propose an idea to your manager, you get the brush-off. Maybe your boss doesn't like you, or he has a preconceived notion that your idea will fail. Regardless of the reasons, you may still feel that the idea should be heard, and want the opportunity to present it. In this case, the proper thing to do is to ask permission from your direct manager to propose the idea to his boss. You may be surprised at the response—he may be happy to be able to palm you off on a superior. If he says "no," you may be suffering under a boss who doesn't want to rock the boat. Going around your boss to talk to his boss is not politically smart, but if you believe strongly in your idea, and feel that he is squelching it, this may be the only available path.

No surprises

Do not spring the fact that the network will fail in 18 minutes in a meeting. Tell your boss ahead of time. In fact, tell your boss, and let him tell the group in the meeting. If there is one sure way to make enemies in the business world, it's to blindside people in front of their peers.

What is it that managers, directors, VPs, and officers of the company need to know? More often than not, it's how much you need to spend, how long your idea will take to implement, and your justification for the expense. If you need to spend money that wasn't budgeted, you'd better have a good explanation, because if you don't you won't get a dime.

Vice presidents don't want to hear that you can't do your job because you don't have a Cisco Gigabit Switch Router next to your desk. They want to know how to keep their bosses (often the stockholders) happy, what you're doing to make their lives easier, and that you're not just being difficult.

I once had a boss whose rule for the engineers was, "If it's cool, you can't have it." His reasoning (which was usually dead-on) was that if we thought it was "cool," it didn't have a real business purpose. Sadly, I still use that rule for myself to this day.

Usually, when I ask engineers why they need the new things they've asked for, the answer is either emotional or technical. Upper management won't care if a new $400,000 pair of routers makes you happy, no matter what problems they solve.

However, if you can explain how the $400,000 investment will make the network more resilient, and thus save the company from paying any more penalties, you'll have a captive audience.

So, what is it that your boss wants to hear? Consider these questions:

- How will your idea save money? Examples include:

  Capital expenditure (Capex)
  : It will replace multiple other pieces that need to be replaced anyway.

  Operating expenditure (Opex)
  : It will reduce circuit costs or personnel costs.

  Return on Investment (ROI)
  : Spending the money now will produce a return on investment over the next X months.

- How will your idea increase revenues?
- How will your idea increase profits?
- How will your idea make the department (and thus your boss) look good?

If you can answer several of these questions effectively, your presentation will build momentum quickly.

When to Upgrade and Why

Upgrading your routers and switches is not a fun job, but it's got to be done at some point. Some people like to be on the bleeding edge of technology, and run the latest versions of software and operating systems. For networking equipment, that's not such a good idea, for a variety of reasons that I'll cover here.

It may seem obvious to you that if a new feature is announced, you should upgrade immediately to get that new feature. But this is rarely how things work in the real world. The only time that happens is in small networks without change control, and with no restrictions or penalties relating to downtime.

Upgrading should not be done lightly. The code that runs your switch or router is just software, and software can have bugs, including security vulnerabilities. While the major vendors do a good job of version control and beta testing, there are plenty of bugs listed on any vendor's site to prove my point.

What about upgrading something like memory? Easy enough, right? It might be as simple as swapping out a DRAM SIMM or even just adding a new one, but you can never rule out complications.

The Dangers of Upgrading

Here are some reasons to be wary about upgrading:

Introduction of new bugs

Upgrading might resolve the bug that's causing you grief, but the new code revision might have another bug you didn't know about. The best way to prevent this is through due diligence. Cisco can help you cross-reference bug lists and determine what code release you should be running on your equipment. If you have a problem after getting Cisco's OK, at least you can present your boss with an email from Cisco recommending the change. Beats saying, "Wow, I should have checked into that!"

Hardware problems

Upgrading a network device is not necessarily hard work, but it does involve rebooting the device in most cases. Rebooting is (hopefully) something that does not happen often, and there's always room for error and unforeseen disasters.

I once observed a simple upgrade of memory for a major e-commerce web site that went horribly wrong. A new SIMM was swapped with an old SIMM in a 6506 chassis switch. The change-control request was approved, and the operation began. Total estimated downtime: 15 minutes. Change-control window (the time allotted for us to do the work): four hours. Actual downtime: *seven hours*. Apparently, a small chunk of dust landed in the memory socket when we pulled out the old SIMM. When the new SIMM was installed, the dust bunny got mashed into the socket, preventing the SIMM from contacting all the pins. Reseating the chip didn't resolve the problem. Backing out the change didn't solve the problem either because the old chip used the same socket. The moral: any hardware change should be considered a major change, and should not be undertaken lightly.

Human error

If people are involved, problems are more likely. Human error, whether it's down to simple typos, fatigue, carelessness, or clumsiness, causes more failures than just about anything else. Once I sweated through three hours of a four-hour change control. The time to back out all changes was coming, and nothing was working. We were installing a new T1 link, which was not a complicated task, but it wasn't going right. The problem turned out to be human error. My instructions clearly stated that the T1 was to be in the port marked 4 in a panel with eight ports. What the documents didn't say was that there were two panels marked the same way, and I was in the wrong panel. I managed to bring down the entire operation's phone system, and for three hours, they took no calls. This is called "a learning experience." My hope is that you will learn from mine, but most people need to learn these lessons on their own.

Change of default behavior in software
: Over the years, Cisco has decided (rightfully) that certain commands should be the default behavior of Cisco routers—for example, `ip subnet-zero` and `ip classless`. If you designed your network without these features turned on, and somehow relied on the fact that they were turned off, you'll be in for a surprise if you upgrade and find that some of your networks are now not functioning.

Valid Reasons to Upgrade

The following are legitimate reasons to upgrade:

To resolve a problem
: If you have a router that crashes every time you type the `show arp` command, chances are you've got a bug that can be fixed by upgrading that router. Determining whether you truly have a bug should be done by tech support, but bug listings are available online, and can be searched by anyone with Internet access.

Software or hardware is end-of-life
: Vendors stop supporting products. If your software or hardware is end-of-life, it won't be supported anymore, and if you have any problems, you will not be able to get them resolved. Unsupported hardware is a risk to the ongoing operation of your network, and should be upgraded.

A new feature is needed
: Notice I didn't say, "A new feature is wanted." Sometimes a new feature is a necessity. I once worked on a large network with DS3s crossing the globe. We had almost 40 of them. When one bounced a few times within a few minutes, it wreaked havoc with our VoIP environment. This network supported upwards of 300 calls per second, so bouncing links was a severe problem. The answer was IP route dampening, which was in a cutting-edge version of the code (note that this was EIGRP and not BGP, so route dampening was not inherent in the protocol). Normally, I would never advocate using a cutting-edge release, but in this case, the feature was so necessary to the continued successful operation of the network that we decided to go ahead. Luckily, the feature solved our problems without adding any new ones.

A technology change
: If you had a T1, and now you have a DS3, you probably need to add a DS3 interface card to support the DS3.

To increase performance
: Sometimes, companies simply outgrow their hardware. I've seen web sites go from PIX 515s, to 520s, to 525s, to Firewall Services Modules for no other reason than that they were exceeding the aggregate bandwidth that the devices could handle.

To increase simplicity

I'm a big fan of integrated CSU/DSU WICs. If you have T1s that are still using CSU/DSUs with V.35 cables, you can eliminate connection points and separate devices by integrating the CSU/DSU into the router. This is a single example, but many similar situations exist.

To increase reliability

If you have one firewall, adding a second firewall so they act as a failover pair increases the reliability of the network. High availability is a good thing, and should be implemented whenever your budget allows.

To lower costs

If you can prove that upgrading some hardware or software will lower costs, upgrading is a viable option. An example might be collapsing a router and a switch into a single 3750 layer-3 switch. When you combine the two devices, you only need to have a maintenance contract on one, which might lower yearly maintenance costs.

Why Change Control Is Your Friend

If you've been working in a small company that's grown into a larger company, or you've moved from a smaller to a larger company, you've probably run into change control. *Change control* is the means whereby a company limits the changes that are made to the network (or anything, for that matter) until they are understood and scheduled. If you wanted to upgrade a router with change control active, you would need to submit a request. The request would then need to be approved, at which point the change might be scheduled. In some companies, a single person approves or denies change requests. In other companies, committees review every request. I've seen companies where the change-control committee only meets on, say, Tuesdays. If your change is denied, you have to wait until the next Tuesday for an alternative change to be considered.

Scheduling changes seems to be one of those things that make engineers a bit nutty. Why should you have to wait for a team of people who don't understand what you're doing to tell you it's OK to do it? Why waste time if you know you can fix a problem with a single command that won't hurt anything? To engineers, change control can seem like a waste of time and energy that does nothing but interfere with their ability to get work done.

Over the years, I've held many positions, ranging from junior engineer to director of a large consulting company to running my own business. During those years, I've learned a few things about how businesses operate and how engineers think. Running a business while also being an engineer has given me a unique insight into both worlds.

Change control is important for any company for the simple reason that failures cost money. Take a large e-commerce web site, for example. If an engineer makes a change that causes the site to become unavailable, until it's fixed, the company is losing income. What's more, the reputation of the company may be affected when users try to make purchases online only to find the site unavailable. Most users probably won't wait more than 10 seconds for a page to load if they can get the same information or products elsewhere. Imagine the impact of having a site offline for hours because of a simple mistake!

Change control aims to prevent these types of outages through careful planning and acceptance methods. Achieving availability metrics such as the coveted five-nines (99.999 percent uptime) is simply not possible without change control.

Human error is probably the number one cause of outages today. Change control minimizes human error in a number of ways. Embracing change control will also make you a better engineer who is better able to communicate with management. Let's take a look at some of the benefits:

Change control teaches forethought
: A good change-control program will insist that you lay out, in detail, every step of the work you would like to do, including every command you will enter into any device you will touch. Doing this is an excellent way to think through what you will be doing. It is also an excellent way to understand how a router/switch/firewall works, because you need to think through how your changes will affect everything else.

Change control improves documentation skills
: Change-control programs generally involve documentation of your proposed changes before you complete them. Many engineers dislike documentation and avoid it at all costs. The best way to get over something like this is to do it regularly. A team of people, ranging from engineers to managers and directors, will probably review your change-control documentation. If it's not concise and clearly written, you'll end up having to answer a lot of questions and probably rewrite it anyway.

Change control improves communication skills
: Change control usually includes a review process where your proposed change is evaluated, commented upon, then approved or refused. You must be able to communicate your technical ideas to people who may not have a technical background. Failure to accomplish this may result in a denial of your change request.

Change control helps you find mistakes before they occur
: Change control is designed so that any mistakes in the thought process or configuration process can be discovered before they are made. Usually, it only takes a couple of times of someone else finding your mistakes before you learn to be more careful.

The biggest advantage of change control is that it can help you look good. If you make some change on the fly, and it causes hours or days of downtime and possibly lost revenue, what do you suppose the upper echelon of your company will think of you? Do you think you'll be the first on their list for promotion or a raise? Probably not.

Now, think how management will view you if you have a reputation for always following change-control procedures, always presenting well-documented change-control requests, and always performing successful change controls.

Change control is about protecting the company's interests. Like it or not, the company's interests are more important to the company than your disinclination to follow change-control processes. The long and short of all this is that you should learn to love change control. If you accept it as a learning experience, and understand that all successful large-scale businesses employ some version of change control, you'll be happier in the long run.

How Not to Be a Computer Jerk

The computer industry is known for attracting a certain type of individual: the archetypal "computer guy" who thinks he's smarter than everyone around him, and talks to people like they're idiots because they don't know what the fuser does in a laser printer. To be fair, most people in the industry are not like this—and not all of the ones who are this way are guys!

What is it that makes these people the way they are, and why are they attracted to the computer industry? While I don't have any studies to back me up, I have made some observations over the years, both of myself and of others. Yes, I'll admit I was one of these annoying people in a past life, and I'll share with you how I recovered to become a successful professional.

There are a couple of things that contribute to the *computer jerk* phenomenon. Some of them are self-induced, and some are environmental. Some computer jerks are actually nurtured (usually unconsciously) by peers and leaders, though people in these positions often have the power to turn them around. By examining these traits and influences, my hope is that you'll be able to help someone you know—or perhaps even yourself—to become a more balanced, useful computer person, rather than a computer jerk.

In my opinion, the primary influences fall into three principal categories:

Behavioral

Everything we do is based on habit, including our patterns of interaction. If you're used to dealing with people a certain way, you will tend to stick to these patterns. However, that's not to say they cannot be broken. We deal with different types of people in different ways. For example, we typically treat women differently from men, and we treat people in positions of power differently from

our peers. While many people will argue that they treat everyone the same, the simple truth is that they do not. The very fact that you have friends indicates that you somehow treat them differently from others.

One of the ways in which computer jerks operate is through constant attempts to let other people know how smart they are. These people usually are very intelligent, but for some reason, they seem to need to prove it to everyone around them. Unfortunately, they go about it the wrong way.

There are two ways to look smarter than other people:

Be smarter than the people around you
: Knowledge is different from intelligence. Knowing a lot of things is not the same as being able to troubleshoot a problem. Memorizing a book on anatomy is not the same as being a surgeon. People who are naturally smarter than the people around them—and who don't need to flaunt it—are widely known to be smart people. The people who are the smartest are often the most humble, for they have nothing to prove.

Make the people around you look stupid
: This is the way the computer jerk likes to operate. He believes that if he makes everyone around him look stupid, those people will see how smart he is. Sadly, what the computer jerk misses is the fact that all of those people will come to dislike the person who made them look stupid. If computer jerks could stop doing this, many things would change in their lives.

Everyone you meet is good at something. I learned this the hard way all those years ago. Remember that there will always be someone smarter than you, and there will always be someone better than you.

The smartest person in a computer department often suffers from what I call *alpha-geek syndrome*. Alpha-geeks need everyone around them to know that they are the smartest and the best. But often, these people are only the "best" within their small circles, and have no real view into their ranks within the wider world of professional IT consulting. When a consultant is brought in, or a new person with a wide breadth of skills is hired, the alpha-geek will try very hard to discredit this person in the hopes of retaining his alpha-geek status. When faced with a confident, intelligent adversary, however, the alpha-geek will usually fail to discredit the interloper and end up looking foolish.

Another problem that computer people often have is delivery. Remember that you're not the smartest person on Earth, and that everyone should be treated with respect, and you'll go far. People don't need to be told the mistakes they've made (particularly in front of an audience), and they don't need to be told what they've done wrong.

Tell people that *you've* done something wrong, and list the reasons why, and they'll respect you for it.

Tell people that *they've* done something wrong, and list the reasons why, and they'll think you're obnoxious.

Sharing knowledge is good; withholding it is bad. People seem to think that they should never share what they know because they'll lose their jobs. This could not be farther from the truth, unless you're working for a despot. Teams work better together, and respect is formed when information is shared. Trouble-shooting is more efficient when more than one person knows how something works. Think about who you admire in the computer industry. Chances are you admire those people because they've taught you something.

Lastly, if you know something is catastrophically wrong with a project, don't wait until a meeting to blindside everyone with it. This tactic is a hallmark of the computer jerk. Bringing up bad news in a meeting adds shock value to his statement and ensures that everyone will listen to him. What's more, the computer jerk believes that doing this shows everyone how smart he really is. Managers in particular don't like surprises, especially during meetings. The proper way to deal with the delivery of bad news is to quietly inform your manager so he can spread the word. Then the *problem* can be dealt with in the meeting, instead of the reactions to the announcement.

Environmental

I believe that computer jerks are the way they are because they are allowed, and, in some cases, encouraged to be this way. One of the key influencing factors is lack of self-esteem. People want to be liked, or, even better, respected. The problem with computer jerks is that they think the way to gain respect is to show people how smart they are.

I used to work in a company that reinforced the alpha-geek personality in someone who exhibited the archetypal alpha-geek tendencies. The problem was that this guy *was* good—he just wasn't as good as he'd made everyone believe. This led to no end of bad designs and problems that the management teams supported, directly fueling the personality traits that were causing the problems in the first place.

When I sat with this person, and explained that his ideas were valuable—but needed some tweaking to be better—he was interested. When I sat with him and explained that his personality was holding him back, he was also interested. Within weeks of me working with him, he changed for the better. With leadership and mentoring, I was able to break the environmental reinforcements.

Leadership and mentoring

In my experience, people who behave like computer jerks can be changed. Usually, the instrument of change is someone who they respect and who has authority over them. In my case, I was working in a chemical manufacturing plant and thought I was the king of the world. I ran the minicomputer, managed all the PCs, and even managed the network, which at the time was a Santa Clara Systems Version 1.1 network with 10Base-2 coax cabling throughout the plant. (For those of you who don't know Santa Clara Systems, they merged with a company named Novell in 1986.)

On one of the days when I was feeling particularly smug, I managed to offend my boss by speaking to him in a condescending tone when he needed help with something. This was a man whom I liked quite a bit, and admired a great deal, and from whom I had learned a lot. Still, at the age of 22, I thought I was smarter than he was.

My boss called me into his office and calmly asked me what I knew about toluol, methyl methacrylate, and the dispersion process. When I said that I didn't know anything, he proceeded to explain to me that he knew quite a bit about all of these things, and then asked me why someone like me, who was probably a third his age, would think that I had the right to speak to him the way I had. He was right, and I knew it. I was being a jerk.

The point of this story is that he took the time to tell me I was being a jerk, and did it in such a way that I listened. The fact that I respected him already helped a great deal, but the fact that someone pulled me aside and explained to me that I was out of line made all the difference in the world.

I believe that most of the people out there who habitually tread on others in an effort to prove how smart they are simply need a lesson from a leader or mentor to help them understand how they should change.

Index

Numbers

A

We'd like to hear your suggestions for improving our indexes. Send email to *index@oreilly.com*.

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z

About the Author

Gary A. Donahue (GAD) is a working consultant who has been in the computer industry for 25 years as a programmer, mainframe administrator, technical assistance center engineer, network administrator, and network architect. He has been the director of network infrastructure for a national consulting company and is the president of his own New Jersey consulting company, GAD Technologies.

Colophon

The animal on the cover of *Network Warrior* is a German boarhound. More commonly known as the Great Dane, the German boarhound is an imposing yet elegant and affectionate dog that usually weighs between 100 and 130 pounds and measures between 28 and 32 inches in height. German boarhounds range in color from brindle to light grayish brown to harlequin and have a lifespan of 7 to 10 years.

A bit of controversy surrounds the German boarhound's background, with some claiming the dog originates from Denmark, and others, Germany. However, over time, breeders in Germany have made the dog what it is today.

The name German boarhound comes from the breed's ability in its hunting years to pull boars, wolves, and stags to the ground. The kings of Denmark and England often thought of the hound as holy, and at one time it was said that boarhounds lived in every castle in Germany.

Paintings of the German boarhound can be found on the walls of Egyptian tombs. In *Beowulf*, the boarhound makes an appearance as the hunting dog Dene. During the Middle Ages, the dogs were buried alongside their owners, as they were thought to be spirit guides to the afterlife. But their spirit selves were not always welcomed—the dog was sometimes thought of as a hellhound, called Black Shuck, a wraith-like black dog that was most likely the inspiration for Sir Arthur Conan Doyle's third Sherlock Holmes novel, *The Hound of the Baskervilles*.

The cover image is from *Lydekker's Library of Natural History*. The cover font is Adobe ITC Garamond. The text font is Linotype Birka; the heading font is Adobe Myriad Condensed; and the code font is LucasFont's TheSans Mono Condensed.

O'REILLY®

Network Warrior

This book begins where certification exams leave off, and equips you to work in the real world. A thorough and practical guide to network infrastructure, *Network Warrior* helps you deal with *real* Cisco networks, rather than the hypothetical situations presented on exams like the CCNA.

Network Warrior guides you step-by-step through the world of routers, switches, and firewalls, and includes ways to troubleshoot a congested network, and when and why to upgrade. Along the way, you'll gain a historical perspective of various networking features, such as the way Ethernet evolved. Based on the first-hand experience of the author, who uses real-world examples from his time in the trenches, this book also focuses on how complex technologies work, and how you can configure them in your network. Other topics include:

- The types of networks now in use—LANs, WANs, MANs, and CANs
- Auto-negotiation, and why it's a common cause of network slowdowns
- Telecom nomenclature, and why it's different from the data world
- Firewall theory, design, and configuration, using Cisco PIX firewalls
- Designing route maps and access lists in Cisco devices
- Planning and deploying a network
- Using change control to manage your network as it grows
- Selling your ideas to management
- IP design and subnetting made easy
- Server load-balancing technology
- How QoS *really* works (and what it can and cannot do)
- T1 and DS3 explained for the networking professional

The strategies and examples outlined in this book will give you the tools you need to do your job well. With *Network Warrior*, you can win the complex battles you face everyday.

Gary A. Donahue (GAD) is a working consultant who has been in the computer industry for 25 years as a programmer, mainframe administrator, technical assistance center engineer, network administrator, network architect, and consultant. He has been the director of network infrastructure for a national consulting company, and is the president of his own New Jersey consulting company, GAD Technologies.

www.oreilly.com

US $44.99 **CAN $58.99**
ISBN-10: 0-596-10151-1
ISBN-13: 978-0-596-10151-0