NETWORK MANAGEMENT

The term *network management* is often used in an imprecise way to capture multiple meanings. The first part, *network,* can mean the entire range of network communications and computing systems and services, or just a subset of these associated with the physical and network layers; in the latter case one distinguishes network management from system management. *Management* means both a collection of operations tasks handled by network and system administrators and support staff, as well as technologies and software tools intended to simplify these tasks. This article uses the term *network management* in its broadest sense. *Network* here means any system and service, whether associated with communication or with computing functions of a network; in practical terms, the entire range of systems and services provided by an information system. *Management* means both operations and administration tasks as well as technologies and tools to support them.

This article is organized as follows: The first section describes and illustrates the central operational problems that network management technologies seek to resolve. The second section describes the architecture, operations, and protocol standards underlying current network management systems. The final two sections describe emerging technologies used in network management.

CHALLENGES AND PROBLEMS

Several factors render network management an area of increasing importance. First, with rising scale, the complexity and rate of changes of network information systems increase the difficulties associated with their operations management. Operations staff confront increasingly more complex challenges in configuring underlying network elements. Failure modes can escalate rapidly among multiple components in a manner that is unpredictable; and their diagnosis can involve complex analysis requiring substantial knowledge about the large variety of components composing a network. Frequent changes in configuration, components, and applications often introduce significant performance inefficiencies and increased exposure to failures. With growing dependence on networks to deliver mission-critical functions, organizations are increasingly exposed to such failures and inefficiencies. Network failures can paralyze not only an entire organization, but possibly an entire industry. For example, in 1998 failure in a data network paralyzed retail sales throughout the United States as vendors could not access credit card authorization and billing functions. Similar network failures have paralyzed air traffic through major US airports, Nasdaq trading, and other major national-level network-dependent functions. A 1992 failure in the ambulance dispatch network of London, England, resulted in the loss of over 20 lives.

At present, network management requires a significant level of expert manual labor by operations staff. The complexity and intensity of operations management activities translate into enormous costs; operations management costs are responsible for 65 to 85% of information technology budgets, by far the largest component. In a paradoxical way, discussed in the next section, current management technologies have been responsible for sharply increasing the complexity and costs of operations management rather than reducing them.

The rest of this section describes central network management challenges and illustrates them through respective operations scenarios.

Problem Management

A central concern of network management is to detect, diagnose, isolate, handle, and track network operations problems. Detection is typically handled through alarms, dispatched by network management software, typically embedded in network elements, to a network management system (NMS) at an operations center. These alarms are typically displayed on operations staff consoles. Operations staff must correlate the alarms associated with a particular problem, isolate this root cause problem, and then resolve it. This often requires coordinated activities by multiple staff, possibly including multiple administrations of affiliated network domains as well as the vendor's technical support. To track the work flow of these problem resolution activities, one often uses a trouble-ticketing system. A problem instance is described by a trouble ticket, which records the problem's state of resolution and triggers activities by various individuals and organizations involved.

The following scenario, depicted in Fig. 1, illustrates some of the fundamental elements of problem management and the technical challenges involved. The figure depicts a database server (on the left) interacting with a remote client (on the right) over a network of IP routers (marked A, B, C, D). The IP links are layered over underlying Ethernet local area network (LAN) links (domain I, V), or wide area network (WAN) links (Domains II, III, IV). In particular, a T3 link connects router D to the backbone router C and an ATM permanent virtual circuit (PVC) connects the router A to the backbone router B. The database client and server use a typical TCP/IP stack to exchange query-response transactions.

Consider a problem scenario whereby a SONET interface underlying the T3 link loses clock synchronization intermittently; the synchronization problem lasts for a short 1 ms until the equipment automatically recovers from it. This loss is well under the threshold for the WAN operation center to notice, and thus the problem will typically go unnoticed. However, during this 1 ms, the T3 frames are corrupted, resulting in a loss of thousands of bits. As a result, packets transmitted over the IP link between routers C and D are damaged or lost. The routers detect corrupted headers, drop such packets, and increase the value of a discard counter built into their network management instrumentation. However, most packets that are lost or corrupted would be invisible to the router management instrumentation. The first element to detect and respond to this loss is the TCP software at the client and server hosts. TCP responds to packet corruption and loss by retransmissions and reduction in the size of the window controlling the amount of data that it transmits. The result is a significant increase in connection delays and reduction in throughput. This performance degradation of TCP causes the database client-server transactions to hold locks on data for a much longer time. Similar performance degradations are experienced by all transactions between the database server and remote clients. This results in significant degradation of the database server response time, as clients have to wait significantly longer for transactions to complete and release their locks. Many of these transactions will be aborted, and the database server response time will significantly degrade. Thus, minor problems at low network layer equipment often propagate, across numerous operating domains, to end systems and have amplified effects on the performance of applications and services executed by these and systems.

Propagation of problems among domains and layers is very typical. Often the symptoms for a problem are not directly observable at the operations domain where the problem exists. In the preceding example the alarm events associated with the problem would be observed at the database application management domain. The application manager will get complaints from end users, who see long response times and abortion of their transactions. Diagnosing the source of these symptoms and isolating it to the performance of the IP link from C to D requires complex ad hoc processes that demand the collaboration of multiple experts responsible for different domains. Resolving the problem presently requires coordinated problem management and tracking between the WAN domain managers, the IP routers managers, the LAN managers, the database server administrator, and the application managers.

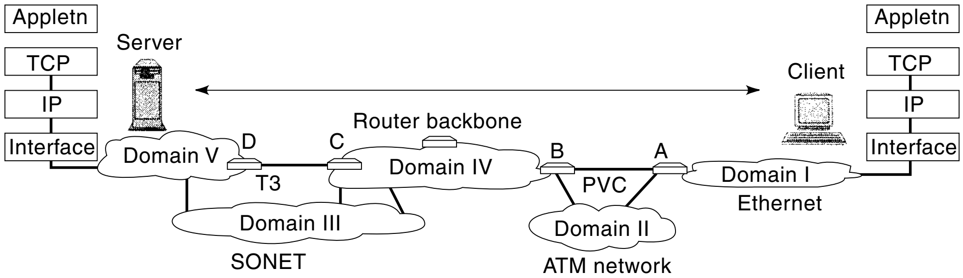

Figure 1. A problem management scenario.

Presently, these different tasks involved in problem management are processed through complex, ad hoc, manual, expert-intensive activities. Often these processes require hours, even days, during which a network may remain nonoperational or only partially operational. Often these processes result in errors that exacerbate the problems rather than resolve them. The central challenge of problem management is how to automate the detection, diagnosis, and resolution of problems.

Configuration Management

Configuration management is concerned with setting or changing configuration parameters of elements to accomplish a desired operational mode and to assure consistent and efficient operations. A typical network goes through multiple changes daily as new systems are installed or existing systems are reconfigured. A typical network system can involve hundreds or thousands of configuration parameters. A single reconfiguration task may involve multiple coordinated changes of configuration parameters of various systems. These changes must be executed to assure consistent operations among the various systems involved.

Consider, for example, an organization that wishes to move a Web server to a different virtual LAN in order to accommodate changes in traffic levels. It is necessary to change the virtual LAN configuration data at underlying Ethernet switches; assign new IP address to the server to reflect its new network affiliation; change address translation tables of other systems attached to the new LAN and of those of the old LAN; change the primary DNS name servers database entries associated with the respective networks and propagate these changes to secondary DNS servers; reconfigure the Web server and respective file servers to reflect the new associations among them; reconfigure directory servers respectively; and possibly reconfigure routers serving the new and old virtual LANs to reflect the new traffic patterns. Each of these different reconfiguration activities involves very different configuration data repositories and change mechanisms and protocols. These changes can result in various forms of operational inconsistencies that can have enormous impact on the overall performance of the network. Should such inconsistency occur, recovering a consistent operational mode is very difficult. Each of these systems may or may not admit backup of its configuration state. Even if it permits simple backup and recovery of a configuration state, it is often difficult to restore the overall network to a consistent operational mode. Various components of the operational mode of a network are constructed to adapt automatically to configuration changes. For example, routers typically adapt dynamically to topology changes. A configuration change in one system often triggers propagated adaptation mechanisms in other systems. Even if one restores the original configuration state of the system, the propagated changes may be irreversible and thus other systems remain in an inconsistent operational mode.

Presently, the tasks of managing configurations are handled by expert operations staff through ad hoc processes. Configuration changes are responsible for some of the most catastrophic network failures. To avoid failures, it is necessary to assure consistency of the changes in multiple systems. The rules governing consistency of configurations can be very complex and are often unknown to the operations staff responsible for the changes. At present this knowledge is acquired through apprenticeship and through trial and error. Therefore, configuration management problems result in enormous exposure to unpredictable and uncontrollable failures and require scarce and very expensive expertise and labor; the expertise and exposure are replicated among enterprises. Furthermore, exposure is rapidly emerging as a central hurdle in deploying new technologies and capabilities in networked information systems.

Clearly, this primitive state of the art of configuration management is fundamentally inadequate. The long-term solution is to create technologies for self-configuring, plug-and-play networks.

Performance Management

Performance management is concerned with planning and allocating network resources to optimize the network's overall performance behavior. A typical network involves resources representing multiple technology generations with a broad spectrum of capacities. The bandwidth of communication links can range over several orders of magnitude, as is the speed of routers and switches; the processing speeds of various attached systems can vary greatly; and applications demands for resources can change dramatically as a result of even minor changes. Furthermore, the relationship between resources of different systems can change dramatically as new components are deployed in the network. As a result, performance bottlenecks are typically formed and change very rapidly. Operations staff must often reconfigure resource allocations to address emergent performance bottlenecks and inefficiencies.

Performance problems can be intimately linked with failures and configuration changes. For example, consider again the network failure described previously. A failure of a SONET layer interface resulted in a performance problem of overlaid TCP links; and these, in turn, resulted in a database-server failure.

In what follows we use a typical network performance problem scenario, depicted in Fig. 2, to illustrate some of the challenges of performance management. Consider a network of a financial enterprise, consisting of a data processing center interconnected with various sites via a private router network. Clients at these sites interact with various applications services offered by the data center. The data center includes a market information system, executing on a very large server system, and a mission-critical trading server. A Web server is introduced to provide novel and very powerful access to the market information system. This Web service generates significant new traffic levels over the WAN links connecting the data center and remote offices. This new Web traffic competes for link bandwidth and router buffer space with traffic of existing applications, such as the trading application, where remote clients transact with the trading server. As the Web traffic grows, it generates frequent congestion and packet losses at backbone and access router.

Congestion and loss at routers result in several performance problems. The performance of TCP connections served by a congested router degrades sharply and results in respective impact on the performance of the applications. In the scenario of Fig. 2, the Web services and the trading server transactions will see reduced throughput and increased delays.

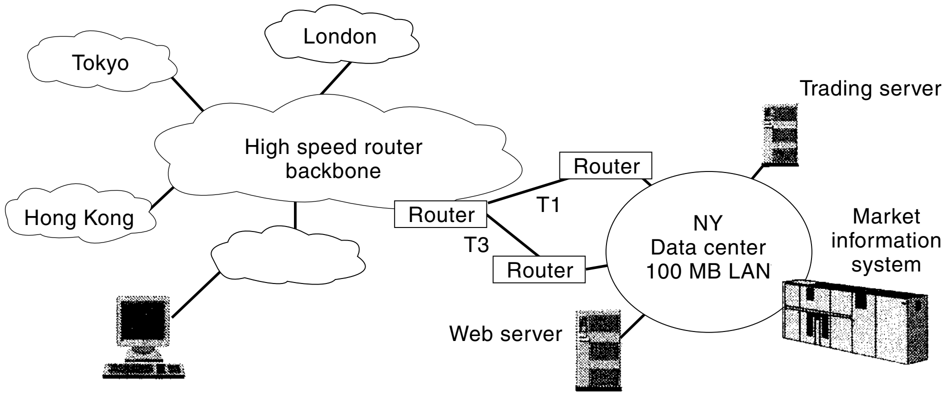

Figure 2. A performance management problem scenario.

Applications may show varied sensitivity to such performance problems. The Web traffic may be somewhat insensitive to increased delay, while the trading center transactions are very sensitive. In general, database transaction traffic as well as emerging real-time continuous media applications can be very sensitive to the quality of services (QoS) delivered by the underlying network. Technologies to manage the QoS delivered to different applications are just emerging.

The network managers solved the congestion problem of Fig. 2 by using a separate T1 link to route the transaction traffic between the trading server and the router backbone. This required complex reconfiguration of the LAN switch to relocate the trading server on a separate virtual LAN; reconfiguration of the routers to handle the transaction traffic accordingly; and propagation of these configuration changes to various directories. This approach of solving performance problems by creating separate networks to serve different applications is unscalable, complex, and can lead to intractable configurations that are difficult to manage and change and result in complex failure patterns.

Emerging technologies offer simplified solutions to the problem of QoS management. One such solution is to use a traffic shaper on WAN links. A traffic shaper can be configured to control and assure the bandwidth allocated to different applications served by the WAN link. In the scenario of Fig. 2, a traffic shaper can be installed at the T3 link connecting the data center to the router backbone. This traffic shaper can be configured to assure that trading server transaction traffic obtains the necessary bandwidth, by controlling the amount of Web and other lower-priority traffic admitted into the network. A traffic shaper thus serves to effect and manage site-level admission control policies to assure the QoS delivered to applications. The RSVP protocol allows applications to accomplish more global bandwidth reservations at routers, based on their priorities. However, this requires globally coordinated management of priorities among competing traffic streams.

Performance problems can lead to complex failure patterns. In the scenario of Fig. 2, sustained congestion at a router can lead to loss of network-layer control traffic. For example, routers often use "keep-alive" packets to monitor the links connecting them. When enough of these packets are lost, the link is considered dead and the router switches the traffic to an alternate link. This clears the congestion and allows "keep-alive" packets to reach the router, causing it to recognize the link as having become live again. The router responds by switching traffic back. This instability, known as route flapping, can lead to significant avalanche failures.

In summary, performance problems typically arise due to a congested network-layer or application-layer resource. Performance problems can have a significant impact on applications behaviors and cause failures of applications or network-layer systems. Technologies to manage and assure applications QoS will permit effective solutions of the problems. However, these technologies are in the early stages of development and involve complex technical and operations management challenges that are not yet fully understood.

ARCHITECTURE OF NETWORK MANAGEMENT SYSTEMS

Figure 3 depicts the overall architecture and operations of a typical network management system. A network system (element) is instrumented to monitor and configure the operations of the element. The instrumentation data are organized in a management information base (MIB), and an embedded agent software provides access to this MIB via a respective management protocol. A manager software at an NMS can use the management protocol to access and manipulate MIB data at remote elements. The NMS presents to operations staff status data on the operations of the network and provides tools to configure network elements and tools to monitor and analyze their operations. The management protocol, governing interaction between a manager and element agents, is an application-layer protocol that depends on an underlying transport protocol to move its packets between the element and NMS. This manager-agent architecture has been crystallized and popularized by the Simple Network Management Protocol (SNMP) and adopted by the ISO Common Management Information Protocol (CMIP). For example, the element

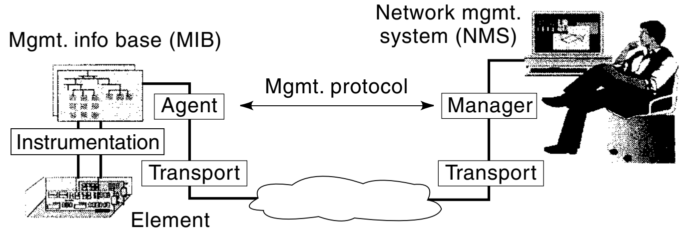

Figure 3. Overall architecture and operations of a network management system.

instrumentation may include various counters for error conditions. The manager software at the NMS may poll these error counters through the management protocol. The data collected by the manager are then used by a monitor application to provide operations staff with graphical depiction of error conditions.

This manager-agent architecture rigidly separates computational and communications functions between elements and centralized NMS. Embedded management software in elements is responsible for a minimal role of instrumenting monitoring and configuration functions and providing access to a repository of this data. The tasks of monitoring, analyzing, controlling, and handling problems and configuration management are entirely processed by NMS software. The manager-agent protocol is thus responsible for allowing the NMS applications to access and manipulate remote element instrumentation. This division of labor among element and NMS software has been inspired by the needs, realities, and opportunities of networks of the middle to late 1980s. At the time, network elements had limited software processing power compared with the NMS; management functions were mostly simple to allow occasional interaction among an NMS and elements; networks were often small enough to admit a centralized management paradigm; and the focus of management was on individual elements rather than on the end-end functions that they support. These needs, realities, and opportunities have changed dramatically, requiring and stimulating new technologies and paradigms currently in early stages of development.

It is therefore useful to consider the architecture of network management systems from a broader functional perspective that is independent of the specific division and organization of functions under the manager-agent architecture. Furthermore, with the broadening expansion of the term *network management* to include management of systems, applications, and services and with the boundaries among these different operations management tasks and technologies all but disappearing, it is necessary to develop a common architectural framework that can capture operations management activities, based on SNMP, as well as those based on system-level directories such as NIS+, NDS, and the Registry, or application-level management functions involving specialized configuration and problem management software.

Such architecture is depicted in Fig. 4. At the bottom layer reside elements and systems to be managed. The instrumentation layer, above the element layer, includes software that instruments monitoring and configuration of the underlying elements and data structures to organize it. The instrumentation access layer includes software, API, and protocols to access and manipulate this instrumentation. The modeling layer includes data models of managed components and subsystems that enable management applications to handle uniformly the tasks of simplifying and automating problem, configuration, and performance management.

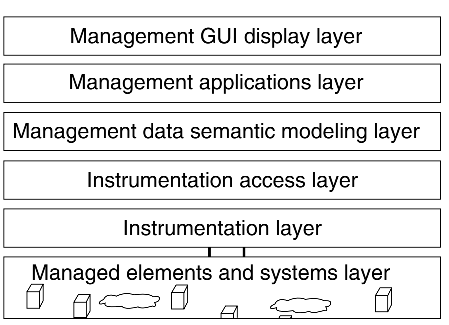

Figure 4. Functional architecture of a management system.

For example, in the manager–agent architecture of SNMP and CMIP, the instrumentation layer includes instrumentation data and procedures to monitor and configure elements and hierarchical MIB data structures to organize naming and access to this instrumentation. The instrumentation access layer includes manager and agent software, at the NMS and elements, respectively, to handle access to MIB data; and a management protocol to support their interactions. The management data semantics modeling layer is implemented in various NMS databases, where data about network elements and their connectivity and other relationships are maintained. The management applications layer includes various NMS tools to monitor and configure elements, and the GUI display layer provides user interfaces to these tools.

The Element Instrumentation Layer

Element instrumentation typically includes the following categories of data:

1. *Operating Statistics.* These include routines to monitor and collect operational statistics and status data of network elements and their operating components. For example, a switch can include instrumentation counters to monitor the number of bytes processed by a given port over its input or output streams; the number of packets or cells handled by the port; the number of packets discarded due to various error conditions; or the operational status of the port (e.g., disabled, testing).
2. *Configuration.* This includes routines to set configuration data of network element. For example, a LAN switch can include configuration data that disconnects ports, assigns a port to a virtual LAN, creates a new virtual LAN, or deletes a virtual LAN.
3. *Identification.* This includes various identifying system data. For example, a router can include data about the model, version of components, and contact information to reach its administrators.
4. *Operating Events.* These include event generators that create spontaneous notifications when certain conditions arise in an element. For example, a router port can generate events when a link is detected to be lost, when the port hardware is experiencing a failure, or when buffers overflow due to high-level of congestion.

Operating statistics and configuration data are accessed synchronously by processes that monitor and change them. The instrumentation software is passive, and its access derives from synchronous activities by its consumers who pull it to support their computational needs. In contrast, events are generated and notified asynchronously to processes that wish to process them. The event notifications instrumentation software is active, and its access thus requires subscription by its consumers who gets the events pushed to them.

The instrumentation layer can be very rich. The instrumentation of a typical router or a LAN switch involves several thousand various statistics and configuration data items and hundreds of potential events. Consistent access to and naming of these instrumentation data require an organizational scheme. This function is provided by a MIB data structure.

Both SNMP and CMIP pursue a hierarchical directory naming structure for their MIB—called Management Information Tree (MIT), in the case of CMIP. In what follows we describe briefly the structure of managed information (SMI) model pursued by these protocols.

SNMP Managed Information Model. SNMP establishes a static tree-structure naming scheme for instrumentation data. Specific instrumentation data are located at the leaves of a hierarchical naming tree. A given instrumentation is named by a string of numbers along the path from the root of the tree. For example, the string 1.3.6.1.2.1.1 can be the name of data describing the system being managed. This name is static in that its binding to a specific data set is entirely determined at MIB design.

A static naming structure has multiple benefits. First, designers of management applications software can encode with ease the name of data that they need to access and manipulate. The binding of MIB naming to respective data is known to the designers of the software and does not require complex management during runtime. When the naming structure can change dynamically, the application software must first resolve what is the name of the data that it needs to access and bind with it. This renders the task of building, deploying, using, and maintaining management applications more complex. Second, a static naming structure facilitates simpler coordination of management instrumentation names among vendors and standard committees. With static names, each participating entity can be allocated its own subtree and managed its own naming conventions. Thus, for example, the SNMP naming directory allocates a subtree to a common standard MIB, called MIB-2, and to numerous other standards MIBs while allocating similarly private subtrees to each vendor desiring one; each of the owners of a subtree can allocate and manage its name space independently.

However, not all instrumentation at a given element can be known at MIB design time. For example, one cannot know at MIB design how many interface objects will be incorporated in a given switch. Furthermore, this information can change dynamically when one configures new interfaces into the switch. Therefore, even a static MIB must include provisions to accommodate dynamic changes in instances of similar instrumentation data. SNMP solves this problem by organizing such multi-instance data in tabular form. Each instance is described by a row of instrumentation data. These rows are stacked into a table. Rows can be added or deleted dynamically. This overall organization of an SNMP MIB tree is depicted in Fig. 5.

The organization of dynamic instances in tables requires a naming scheme that can distinguish different rows in a given table. This naming scheme is depicted in Fig. 5 and explained here. The instrumentation data are depicted in the figure as gray cells at the bottom of the MIB tree. The name of a single-instance variable is the path leading to it from the top of the tree. For example, the name of attribute 1 of subsubgroup1 of subgroup2 is 1.2.1.1. The name of a table cell is identified by a static column identifier *X* followed by a dynamic row (instance) identifier *Y*. Columns have static identifiers defined by their unique tree path labels. For example, the identifier for the attribute2 column of the table on the left of Fig. 5 is 1.1.2. How can rows be uniquely identifier, even when they can be dynamically added or deleted to the table? SNMP solves this by selecting a collection of columns (key) whose contents uniquely identifies the respective rows. For example, the first column of the Interface table of MIB-2 is an index attributes (ifIndex) that assigns a unique integer to each row of the table. Suppose, for example, that the table in Fig. 5 has attribute1 serve as a key for the table. Suppose also that the values of attribute1 for the four rows depicted in the figure are 5,12,31,54. SNMP assigns names for cells of a table by concatenating the column name with the row name. The identifiers of the cells in the attribute2 column are, therefore, 1.1.2.5, 1.1.2.12, 1.1.2.31, 1.1.2.54.

SNMP also provides unified mechanisms to associate syntactic data type with the contents of cells (instrumentation data). A small number of data types, originally 9, are provided. An ASN.1 specialized syntax notation, called the structure of managed information, is used to declare formally the data types and names of cells. These SMI notations are compiled by an MIB compiler to appropriate MIB data structures that are used to organize and access the instrumentation of managed information at elements.

CMIP Managed Information Model. CMIP, like SNMP, uses a hierarchical tree structure to associate unique names with instrumentation of managed information. Names are defined. However, unlike SNMP, this MIT is dynamically structured and can change among elements of the same type and in the same element over time.

The CMIP model of instrumented managed information is based on an extended object model. Related instrumentation data and methods are aggregated to form managed objects. For example, a port object may include various data attributes of a port (e.g., traffic and error statistics of the port,

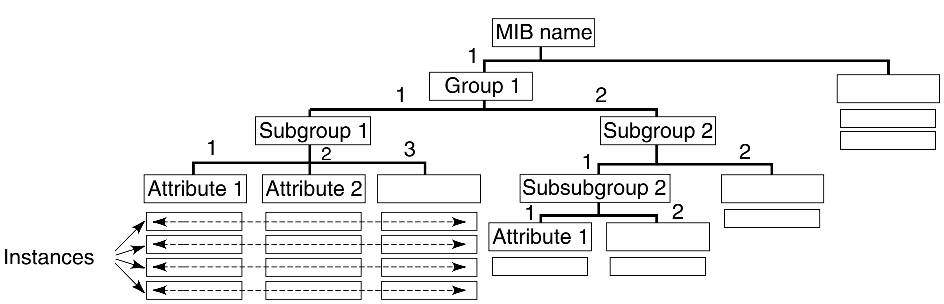

Figure 5. An SNMP MIB tree and its respective naming scheme.

status of the port, configuration data of the port) but also methods to handle management tasks associated with the port (e.g., a procedure to attach it to a virtual network, a procedure to disconnect the port, or a procedure to configure the port operating parameters). In SNMP such procedures must be activated as implicit side effects of changing a setting of port data. In CMIP management procedures are explicit integral components of the managed information instrumentation model.

CMIP's managed information model goes a step beyond traditional object-oriented software. An object can have, in addition to data attributes and methods, event notifications. Event notifications are active elements of an object. Their access and manipulations obey a completely different communications and processing model than the synchronous pull model of accessing object attributes and methods.

CMIP uses, like SNMP, a formal ASN.1-based language to specify the structure of managed information. This language, called Guidelines for the Definitions of Managed Objects (GDMO), is substantially more complex than the SNMP SMI language; unlike the SMI, the GDMO language does not lend itself in its most general form, as defined by ISO, to machine compilation.

Managed objects are organized in a hierarchial directory tree, the MIT. Unlike SNMP MIBs, managed objects of CMIP are located at each tree node, not just leaves. Each managed object has attributes that uniquely identify it among its tree siblings; these attributes are called the relative distinguishing name (RDN). The name of a CMIP object is constructed by concatenating the RDN along the path to it from the root of the MIT. This name is called the unique distinguishing name (UDN) of the object. To identify a specific attribute of a managed object, one uses the UDN concatenated by the name of the attribute. The naming scheme defined by the UDN is based on a dynamic MIT tree structure. When the tree structure changes, so will the respective names. This means that management software must first identify the current name of the objects that it needs to manipulate. Therefore, management software must concern itself with the complexity of managing a dynamically changing name space. Furthermore, the MIT associated with similar elements (say, two routers of the same type) can be very different depending on various aspects of the router configuration. As a result, the software to manage specific elements cannot assume a particular unified structure of the MIT associated with these elements and must be designed to handle possible heterogeneity.

Management Protocols: The Manager-Agent Paradigm

Management protocols, as depicted in Fig. 4, are used by manager software to access and manipulate MIB instrumentation data at an element agent. The fundamental constructs of these protocol must enable a manager to retrieve (GET) data from a MIB, change (SET) data of the MIB, and obtain event notifications (TRAP) from the agent.

SNMP Manager-Agent Protocol. SNMP supports a very simple manager-agent protocol to access and manipulate instrumented management information. The protocol consists of several primitives:

GET: This operation specifies the name of MIB data to be retrieved by the manager.

GET-NEXT: This operation specifies the name of a table row to be traversed and retrieved by the manager.

SET: This operation specifies the name of a MIB data and a value to be written into it.

GET-RESPONSE: This operation retrieves the MIB data requested by a GET/GET-RESPONSE request; it also acknowledges execution of a SET request.

TRAP: This operation is initiated by the agent to provide asynchronous event notification to a manager.

The SNMP protocol is carefully designed so that all commands and responses fit in a single UDP frame. The parameters of a command are coded as variable-binding pairs. Each such pair consists of a name and a respective value. For example, a GET command can specify several MIB variables to be retrieved. The GET packet will carry a variable binding pair for each of these desired data. The name of each pair is included in the pair, while the value part is left as an empty container for the agent to fill in. The agent reads the name of each pair, retrieves the data, codes it in the value part, and then packages it as a GET-REPONSE frame and sends it to the manager. Both the name and the value have statically known lengths, and thus the total size of the data can be determined statically and ascertained by the manager code with ease to fit in a UDP frame.

CMIP Manager-Agent Protocol. CMIP, like SNMP, supports capabilities to GET and SET managed object data and to notify events. It offers some additional capabilities not available or needed in SNMP. The protocol consists of the following primitives:

CREATE: Create an object and place it on the MIT.

DELETE: Delete an object from the MIT.

GET: Retrieve data associated with a subset of the MIT.

CANCEL-GET: Stop retrieving the data.

SET: Change the value of objects attributes.

ACTION: Invoke a method of the respective objects of the MIT.

EVENT-REPORT: Notify manager of an event and provide its respective parameters.

The first two commands are provided to allow managers to configure and manipulate the structure of the MIT. In SNMP this structure is static and not controlled by managers. CMIP managers create managed objects to subscribe to events, or associate managed objects with dynamic managed components. It is interesting to note that SNMP has incorporated capabilities for manager-directed creation/deletion of dynamic instances of managed entities. This was required first in the design of the RMON, remote monitoring MIB. RMON controls the functions of a remote monitor. A manager must be able to configure the remote monitor to filter certain traffic statistics of interest. Each such configuration is defined in RMON as a managed entity in a control table. The manager can cause creation or deletion of such monitoring configurations as side effects of a respective SET command. With the growing management functionality required by emerging systems, manager-directed creation/deletion of managed objects has been increasingly incorporated in SNMP MIB designs, pursuing this RMON methodology. This is used, for example, in creating permanent virtual circuits in ATM switches and in creating and configuring virtual LANs.

CMIP GET is substantially different from an SNMP GET. Unlike SNMP, CMIP managers cannot possibly know the precise location or even number of instances of managed objects of interest on the MIT at a given time. Therefore, retrieval actions are applied to all objects that meet certain filtering criteria and belong to entire subtrees of the MIT. CMIP GET, therefore, does not include variable binding parameters, as SNMP GET. Instead, a CMIP GET includes specifications of a subtree and respective filter to be applied in searching and retrieving data of interest. The result of a CMIP GET is therefore unpredictable. Any amount of data can be generated in response to a GET request. The CANCEL-GET command has been included to abort such a response stream of excessive size. CMIP, therefore, cannot depend on a datagram transport protocol as SNMP does. Instead, it requires a full-fledged stream-transport protocol, as provided by the OSI transport stack.

CMIP, unlike SNMP, provides an explicit construct—ACTION—to invoke remote management procedures. In SNMP remote invocations are accomplished as implicit side-effects of a SET command. Suppose one wishes to activate a configuration procedure to set up a permanent virtual circuit through an ATM switch with appropriate operational parameters. In SNMP one needs to first apply SET to set the appropriate operational parameters and then invoke a SET that activates a remote configuration procedure that uses these parameters to establish the desired circuit. In CMIP the entire task can be accomplished by a single invocation of a remote action to which parameters are passed explicitly. The SNMP implicit and fragmented method of remote procedure invocation presents difficult software architecture problems that are resolved by the explicit ACTION construct of CMIP.

Finally, CMIP EVENT-REPORT offers richer event notification services than SNMP. SNMP originally assumed a management operations model based on trap-directed polling. That is, management software should be activated in response to traps, and then it should pursue polling to determine the details of a problem generating the traps. SNMP version 1, therefore, provided a very minimal event reporting facility intended to be used only for major coarse-grained events, such as complete failure of a system. It became clear that this trap-directed polling was an inadequate assumption of how users and vendors structure operation management. Indeed, typically traps play a central role in management, while polling is rarely conducted. As a result, vendors have incorporated a substantial range of highly informative traps, typically in the hundreds, associated with elements. The primitive trap reporting mechanism of SNMP thus became an obstacle in efficient processing of such rich event systems. This was recognized by the designers of SNMP v.2, who included substantial support of richer event notifications. CMIP incorporated rich event notification mechanism from the start. A manager can subscribe to event notifications and obtain respective managed object information based on configurable event discrimination filters.

YECHIAM YEMINI
Columbia University

NETWORK, NONLINEAR. See NONLINEAR NETWORK ELEMENTS.