9 NETWORK MANAGEMENT

Why all the interest in network management, and why all the effort? Part of the answer lies in the ever-increasing importance of networks and internetworking. Today, the phrase *mission critical* applies not just to networks that support space-exploration applications but also to those that support sales, airline and hotel reservation systems, health care, commodity and stock exchanges, commerce, and finance. Nearly every business depends on the health of its network to remain in operation. Like the dial tone in the voice network, the availability of facilities for data communications is now taken for granted, and there is an increased incentive to keep networks healthy.

Part of the answer lies in the actual number and size of networks. Data network operation is no longer a matter of managing one site, with one vendor's equipment and one vendor's proprietary management tools. And increasingly, as enterprises connect their networks to share information, "operations" crosses administrative boundaries as well.

These observations address the question "why all the interest?" The answer to the question "why all the effort?" lies in the fact that until recently, with only a handful of noteworthy exceptions, management and diagnostic tools for data networks were either proprietary or homebrewed. There is thus both opportunity and incentive for those who wish to become involved in a new technology that can be put to practical use in a short time frame.

The OSI and Internet communities have each developed technologies to solve the problem of network management. The Internet approach is based on a management protocol called *Simple Network Management Protocol* (SNMP), designed with the philosophy apparent from its name: keep it simple. The OSI approach is based on a management protocol called *Common Management Information Protocol* (CMIP), also designed with the philosophy apparent from its name: provide a common protocol flexible enough to solve all problems.

This fundamental difference in design philosophies created the predictable result. SNMP was designed and deployed quickly, solving the immediate problem of managing the Internet and then moving beyond to other networking environments. CMIP took longer to design, and its deployment is slowed both by its inherent complexity and by the market success of SNMP. Nevertheless, both SNMP and CMIP technologies exist today and will likely continue to be deployed in future network- and systems-management products. And it is also true that these two management models do not address the entirety of the issues related to network management. It is therefore advisable to explore what each tool has to offer and to construct complete management solutions that solve real-world user needs based on whatever combination of technologies is appropriate.

The Internet Approach: Keep It Simple

While the Internet remained a modest collection of hosts, gateways, and users, network management was an "ad hack" practice, and home-brew management, abetted by the sturdy ICMP "ping" facility, was the norm. As the Internet grew both in size and in importance to its users, and as TCP/IP-based internetworking expanded outside the research community, the need for network management capabilities grew. By 1988, research and experimentation by the Internet community led to the definition of two management systems and associated management protocols: the *High-Level Entity Management System* (HEMS) (Partridge and Trewitt 1988; RFC 1021; RFC 1022) and the *Simple Network Management Protocol* (RFC 1157).

Both of these management protocols were designed to operate over the TCP/IP protocol suite. However, because of a strong desire on the part of U.S. government agencies to have a graceful transition from TCP/IP to OSI, the research community agreed to abandon the longer-term HEMS effort in favor of a management framework based on OSI CMIP standards (the trials, travails, and tragedy of OSI Common Management over TCP/IP are discussed later in this chapter).

SNMP was initially viewed as an interim step on the road to the longer-term OSI solution; of course, nothing is ever truly interim. SNMP proved to be an especially adaptable management framework. Executable code for SNMP fit into literally anything containing an embedded microprocessor and was rapidly implemented into a wide range of networking components, from mainframe host systems to PCs, routers, bridges, data/channel service units, even uninterruptable power supplies. The immediate success and subsequent widespread deployment of SNMP have caused folks to reconsider even mentioning SNMP and the word *interim* in the same breath.

SNMP and the SNMP-based network management framework are described in the following sections.

What Is Managed in an Internet?—Internet SMI and MIBs

In the SNMP management framework, shown in Figure 9.1, there are four basic elements: the managing entity, the managed entity, the management protocol used between the two entities, and the management information base. The *managing entity* is an application residing in a network management station (NMS). The network management application monitors, and in some cases controls, the operation of network resources (e.g., routers, bridges, and hosts). The *managed entity* that resides in the managed network resource is called an SNMP *agent*; it is responsible for responding to management operation requests received from the NMS. In addition, the SNMP agent may asynchronously send messages called *traps* to NMSs when the SNMP agent detects certain predefined events, such as a link failure, in the managed network resource. The *management protocol* used between the NMS and the SNMP agent is, of course, the simple network management protocol. The *management information base,*

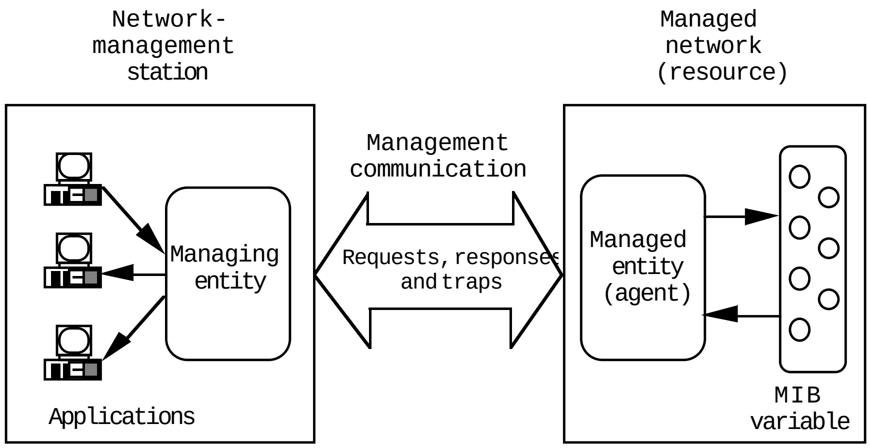


FIGURE 9.1 Internet Management Model

or MIB, is a logical store of information used to support network management (RFC 1213).

SNMP-based network management is based on a philosophy of keeping the required management overhead and functionality at a minimum for both the network and the managed resources (this is referred to in *The Simple Book* as the Fundamental Axiom of Network Management [Rose 1991]). In the SNMP model, the network-management station is expected to perform more network management–related processing than the managed resource since (1) the network management station's primary job is to support network management, and (2) generally speaking, the network management station and its users have a clearer understanding of the "big picture" of the network than any individual managed resource. The NMS usually has applications that may show maps of the managed network, chart various statistics (received/sent packets, error packets, etc.), and notify an operator of errors via audible or visible alarms. The NMS may even perform some rules-based analysis—also hyped as artificial intelligence or applied learning—to help the network manager maintain the network and resolve network problems.

The managed network resource's primary function is not network management: routers should route, bridges should bridge, and hosts should host applications. For these resources, network management should be secondary. In the SNMP model, the agent is expected to devote less to network management processing than the NMS, and typically performs management operations in response to requests from the NMS. With SNMP, network management is based on the NMS *polling* the agents in a network for information; it is expected that generating responses to polls should impose less processing at a managed resource than the often computationally complex process of diagnosing problems.

SNMP-based network management further refines the notion of polling by applying a network management paradigm known as *trap-directed polling.* In trap-directed polling, the NMS polls an SNMP agent in response to an asynchronously generated alarm called a *trap message*[1] it receives from the SNMP agent (as opposed to continually polling for information from the SNMP agent or expecting the SNMP agent to download information on a regular basis). Trap-directed polling relieves the SNMP agent from the burden of monitoring for thresholds or maintaining schedules for downloads, and relieves the network of the traffic caused by frequent polling for information. Trap messages, however, are

1. The trap message indicates that a noteworthy event has occurred; e.g., a link has failed, a link has been restored, or a managed resource has re-booted.

not delivered reliably, so it is not advisable to eliminate polling in the SNMP management model and rely solely on event notifications for direction. A prudent management practice is to poll for information that describes the operational status of all managed resources, in a "round-robin" fashion.

Polling is useful for some aspects of network management—especially fault detection, isolation, and recovery—but there are other occasions when large amounts of network data must be collected for subsequent review and analysis. Frequent requests for large amounts of network data impose a heavy load on managed resources, and are discouraged. Polling can indeed be a two-edged sword; if it is applied incorrectly, by polling too frequently, network managers (especially those with new tools) can figuratively bring a network or managed resource to its knees.

One network manager the authors know tried polling network resources for eight pieces of information on every interface every 30 seconds. The managed network resources were spending more time responding to network management queries than to routing network traffic. At the end of a day of intensive polling, the network manager had a gigabyte of network information, a network with lousy performance, and very irritable users. With a little more experience, the network manager now polls for less information less frequently.

The Management Information Base

The initial Internet efforts focused on three aspects of network management: defining the minimum set of information that is useful and necessary for managing the Internet and its components; identifying how that set of information would be defined; and finally, the management protocol itself.

The management information base (MIB) is the logical store of information used to support network management of the Internet. The goal in designing the MIB was to identify a minimal set of useful information that could be implemented quickly without unduly burdening the managed network resources. The first MIB (or MIB I) was described in RFC 1066 in 1988. MIB I contained 114 *objects,* or types/pieces of information that were viewed as essential for managing TCP/IP-based networks. After additional work and experimentation, some objects were deleted and new objects were added to the MIB, which is now called MIB II (RFC 1213).

MIB II contains over 170 objects organized into nine groups: *system,* *interfaces, ip, icmp, tcp, udp, egp, transmission,* and *snmp* (RFC 1213). The *system* group includes general information on the managed network resource; examples include system name and location. The *interfaces* group contains information relevant to specific interfaces on a managed resource; examples include interface index, interface type, and interface description. The *ip, icmp, tcp, udp, egp,* and *snmp* groups contain information relevant to the protocols, such as counts of protocol packets sent and received.

The *transmission* group is one of the extension mechanisms that allow graceful addition of network management information to MIB II. It is structured to allow for addition of MIB subsets (called modules) specific to transmission media. For example, MIB modules for access to the digital transmission facilities of the telephony network (DS1- and DS3-based access; see Chapter 15) are defined in separate RFCs but are considered to exist as branches of the MIB II transmission group. Other examples of media-specific transmission MIB modules include the token-ring and token-bus LANs, FDDI, the SMDS interface protocol, and frame-relay modules (again, see Chapter 15). This modularity in the MIB structure is one mechanism for supporting the various, changing technologies that underlie the Internet, while leaving the fabric of network management unchanged.

Even with all the information identified in MIB II and in the transmission MIBs, more information is available for information-hungry network managers. Managed objects that are applicable to a specific vendor's products or protocols may be identified and posted electronically as enterprise-specific MIBs. "It's raining MIBs" is a phrase frequently used to describe the tremendous growth in the number of MIBs available for SNMP-based network management.

AHA *Why are there so many MIBs and so many objects to manage? Engineers rather than network operators have traditionally composed MIB modules. All too often enamored with their (well, our) technologies, they (we) get carried away. As one participant in the IEEE 802.6 subcommittee once suggested, they (we) "count everything that moves, and even things that don't move, just to be sure that they stay in the same place." Staff at the NSF Network Operations Center have often suggested that "any more than six [objects] is too many." A downside of having so many objects to keep track of is that the essence of SNMP management is ultimately compromised; despite good intentions, managed resources are spending too much time counting things!*

The Structure of Management Information

The second aspect of network management addressed by the SNMP designers was the structure and definition of the information. When the designers tackled the "How should information be defined?" issue, they recognized that the basic rule of simplicity was joined by another fundamental need—the need for extensibility. Knowing that the Internet was a growing beast and that initial efforts to manage the beast would also grow, the designers of SNMP were concerned that the information structure accommodate future growth. These efforts resulted in the definition of the *Structure and Identification of Management Information for TCP/IP-based Internets* (RFC 1155).

The information needed for network management is modeled as a collection of *managed objects*. As described in RFC 1155, the managed objects for SNMP-based management are defined using a subset of OSI's abstract syntax notation one (ASN.1; see Chapter 4). Objects are classified into types, with each object type identified by a name, a syntax, and an encoding. The *name* uniquely identifies the object *type*; the representation of the object type name is an OBJECT IDENTIFIER, as discussed in Chapter 5. For example, the Internet subtree (shown in Figure 9.2) is assigned the object identifier `{ iso(1) org(3) dod(6) internet(1) }` —or more descriptively, "ISO, organization, Department of Defense, Internet."

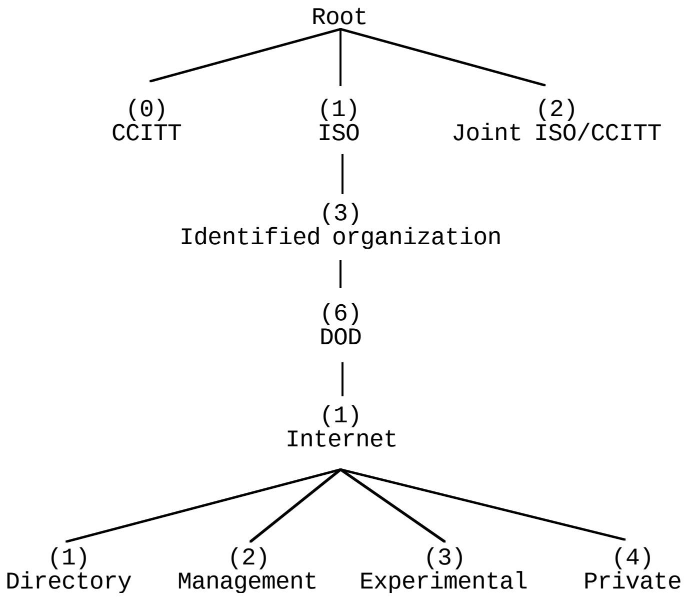


FIGURE 9.2 Internet Object Identifier Tree

Under the Internet subtree, there are four nodes: directory (1), management (2), experimental (3), and private (4). The MIB II standard objects for SNMP-based management are defined in the management branch. Proposed objects are classified under the experimental branch, and objects appropriate for specific products are defined in the private branch. By allocating the fourth branch to "private," the SNMP designers provided a mechanism for extending the naming structure to include information that was pertinent to specific products but not required for general use in the Internet or for network management. It is under the private node that companies attempt to distinguish their agents from their competitors' by identifying even more objects to manage!

The *syntax* is an ASN.1 data structure used to describe a particular object type. Examples of syntax include INTEGER, OCTETSTRING, and NULL. The *encoding* identifies how instances or specific occurrences of an object type are specified using the syntax.

It is the object instances, not object types, that are of interest during network management operations. When an interface is causing networking problems, a network manager needs to know specifically which interface is misbehaving, not just that an interface object type exists. The instance is identified by its object type name and an instance identifier, which is specified in the MIB containing that object. For example, in the *interfaces* table—the `ifTable` group of MIB II (RFC 1213)—instances are identified by `ifIndex`, which is a unique integer value assigned to each interface. The instance is depicted as the object identifier with the generalized format x.y, where *x* refers to the object type identifier and *y* is the instance identifier (Rose 1991). Consider, for example, the object `ifOperStatus` of the interfaces table of MIB II. The `ifOperStatus` value indicates the status—up, down, or testing—of an interface. For this example, assume that the network manager has received a trap message indicating a link-down condition on interface 7 and that the network manager wants to verify the operational status of the interface. The object identifier for `ifOperStatus`—or the "x" part of x.y—is `{ iso(1) org(3) dod(6) internet(1) mgmt(2) mib-2(1) interfaces(2) ifTable(2) ifEntry(1) ifOperStatus(8) }`, numerically represented as "1.3.6.1.2.1.2.2.1.8." The object identifier for the specific instance, interface 7, is "1.3.6.1.2.1.2.2.1.8.7." The NMS polls the router's SNMP agent for the value of this instance using this object identifier; the SNMP agent responds with a value indicating that the interface-7 operational status is down. (Note: The pairing of the name of an object or a variable with the variable's value is called a variable binding, or *VarBind*.) Based on this knowledge, the network manager would begin additional troubleshooting procedures, focusing on interface 7.

How Do I Manage It?—SNMP Protocol

SNMP consists of five message types: `GetRequest-PDU`, `GetNextRequest-PDU`, `SetRequest-PDU`, `GetResponse-PDU`, and `Trap-PDU` messages. There are two basic formats for these message types. The first format is used for the `GetRequest`, `GetNextRequest`, `SetRequest`, and `GetResponse` (see Figure 9.3). The information contained in these messages includes: request ID, error status, error index, and a list of object names and values. A second format is used for the `Trap-PDU` message. The information contained in this message includes: the object identifier for the sending SNMP agent, SNMP agent network address, trap identification, enterprise-specific trap identification, timestamp, and any associated objects and values.

The `GetRequest-PDU` and `GetNextRequest-PDU` are sent by the network management station to the SNMP agent to request retrieval of network management information. The `GetRequest-PDU` includes an object identifier (or a list of object identifiers) that identifies the information needed by the NMS. If the receiving SNMP agent does not have that specific object, the SNMP agent will respond with a `GetResponse-PDU` with a packet format identical to the `GetRequest-PDU`, indicating "noSuchName" in the error-status field. If the SNMP agent does maintain a value for the object named by the object identifier, it populates the value field of the VarBind and returns the packet to the transport address of the originator of the `GetRequest-PDU` (the network-management station).

```
GetRequest-PDU ::=
     [0] IMPLICIT SEQUENCE {
     request-id
     RequestID,
     -- an INTEGER, used to distinguish among outstanding requests
     error-status -- an INTEGER, always 0
     ErrorStatus,
     error-index -- an INTEGER, always 0
     ErrorIndex,
     variable-bindings
     VarBindList
     -- a SEQUENCE of VarBinds, where each VarBind is a name
     -- (objectName) and value (objectSyntax); here, values are always 0  }
```

FIGURE 9.3 `GetRequest-PDU`

The `GetNextRequest` is processed differently by the SNMP agent. The `GetNextRequest-PDU` also includes the object identifier (or list of object identifiers) for the requested information; however, the SNMP agent will respond to a `GetNextRequest-PDU` with the next lexicographical instance in relation to the sent object identifier. The `GetNext Request-PDU` is referred to as the "powerful GetNext operator" (in particular, see Rose [1991]) because of this feature and the relative ease of its implementation.[2]

The `SetRequest-PDU` is used by the network management station to request that the value of an object instance be changed by the SNMP agent. For example, the network management station may use a `Set-Request-PDU` to request an update to a routing table entry or to change the desired status of an interface to "testing." For sets, the SNMP agent designers and implementers must do additional work to identify the correct combination of information needed from the NMS for the set and to ensure that the set is performed correctly in the managed resource. Upon completion of the set operation, the SNMP agent returns a `GetResponse-PDU` with the value in the VarBind indicating the result of the set operation.[3]

There are security concerns related to SNMP; for example, changes to values of configuration-related managed objects, introduced either by malicious intrusion or by error, can disrupt the performance of a network, even bring it to a halt, and monitoring of network management traffic can reveal user behavior patterns and perhaps help an intruder identify critical (frequently used) resources. (The security aspects of SNMP are discussed later in this chapter.) For these reasons, the set operation is considered the "least simple" of the SNMP management operations.

The `Trap-PDU`, unlike the `GetRequest-PDU` and `GetNext Request-PDU` messages, is initiated by the SNMP agent. A small number of predefined events, which are identified in MIB modules, are triggers for the SNMP agent to send the `Trap-PDU` message to authorized network management stations. RFC 1157, "SNMP," defines five trap messages: cold start, warm start, link up, link down, and authentication failure (i.e., attempts by an unrecognized NMS to get or set values of managed ob-jects); others have been identified in enterprise-specific MIBs. After the event is detected by the SNMP agent, it makes a "best-effort"

2. If the SNMP agent does not have a "next lexicographical instance" to return, then a `GetResponse` message indicating the "noSuchName" error will be returned.

3. Note that since SNMP is datagram-based, the `GetResponse-PDU` may be lost; i.e., the NMS may receive no confirmation of the completion of the set operation. To determine the results of a set operation, implementations are likely to run a timer following an attempt at a set operation; if the time expires and no `GetReponse-PDU` has arrived, the NMS will issue a `GetRequest-PDU` to determine the value of the object upon which the set operation was performed.

attempt to notify the NMS. The underlying transport protocols (e.g., UDP, TCP) determine whether the network will attempt resends, etc., to transport the message to the destination. (Note that the recommended transport for SNMP, User Datagram Protocol [UDP], is an unreliable datagram protocol and that, typically, there are neither retransmissions nor notifications of loss of a Trap message.)

Examples of Trap-Directed Polling When an SNMP agent in a router detects that it can no longer send or receive traffic across a physical interface, it sends a link-down trap to the network management station. After receiving the trap at the NMS, a network operator will initiate trouble-resolution steps—or follow a "recipe" of actions—to identify the source of the problem. First, the operator may use the NMS application to poll for the interface's operational status to verify that the status of the interface is indeed "down" (i.e., the NMS will send a `GetRequest-PDU` requesting the value of the object `ifOperStatus` in MIB II and may receive a `GetResponse-PDU` with the value of `ifOperStatus` set to "down"). If the operator elects to diagnose the fault in a "bottom-up" fashion, she may poll for physical-layer information from the transmission MIB for this particular interface type. For example, if this were a 1.544-Mbps link based on the DS1 signal, the operator might request values from the *DS1 configuration table* (RFC 1232), which contains information on the alarm and loopback states of an interface, and the *DS1 current table*, which contains counts of DS1-related errors (RFC 1232). If these polls yielded information that isolated the fault to the satisfaction of the network operator, the NMS might proceed by calling in a trouble report to the telephone company; if not, the NMS might continue polling for errors until the reason for the error condition is found.

Note that although fault isolation can be trap-directed, it need not be so; in fact, since there are no guarantees that Trap messages will be delivered, NMS applications often rely on repeated polling of one or more specific objects—e.g., the `ifOperStatus`—and monitor for a change in value. In practice, more than a handful of NMS applications still rely on an echo packet or a ping facility to determine loss of reachability to a managed resource; if the `ifOperStatus` changes, or the echo requests fail to elicit an echo reply after a given number of attempts, the NMS application informs the operator that the status of a managed resource has changed or cannot be determined, and the operator will then proceed with the troubleshooting previously described.

The network-management applications in the NMS may automate some of the protocol exchanges with the SNMP agent so that the network operator does not have to manually initiate polling for trouble resolution. Also, some applications automatically monitor the status of the managed network resources. For example, some SNMP-based applications can display the managed network as a diagram of links and nodes. The links and nodes are shown in different colors to indicate their status; some applications show links that have gone "down" in red and links that are "up" in green. These applications generally poll the SNMP agents for the `ifOperStatus` values of the relevant interfaces on a regular basis (or alternatively, use ping). The `ifOperStatus` values are then used as the variable to determine the color changes for the maps. This allows for proactive monitoring of the network while minimizing the traffic exchange required across the network. Other applications allow the network manager to select statistics to be displayed in a graph. The output graphically depicts trends in error and usage counts, allowing the network manager to develop a clearer notion of trends.

SNMP and the Protocol Stack

SNMP is an application-level protocol. To avoid forced retransmissions and to minimize the use of network bandwidth, SNMP was originally specified to run over the user datagram protocol, a connectionless transport protocol in the TCP/IP suite (see Chapter 12). Use of SNMP has expanded outside of TCP/IP-based networks. Standards for its use over other transport protocols such as AppleTalk, IPX, and OSI's connectionless transport protocol—the so-called SNMP "over foo"[4] effort—have been developed and are discussed later in this chapter.

Security and SNMP

The current, "community-string–based" SNMP offers what is known as a trivial authentication service. A *community string* is an OCTETSTRING that provides an authentication service similar to passwords. If the receiving SNMP entity "knows" the community string received in an SNMP message, the message is considered authentic. Implementations are also encouraged to check that the IP address received in the SNMP message is the correct IP address for that community string.

There are four main security concerns with community-string–based SNMP: message modification, unauthorized monitoring of messages, masquerades, and replay. Each of these concerns is related to the results of unauthorized access and possible changes to network-management information that may harm the functions of the network or a managed network resource.

To address these concerns, the Internet community has specified an

4. Literally, "SNMP over anything."

enhancement to the SNMP protocol that provides for additional security features (RFC 1351; RFC 1352; RFC 1353). The *secure SNMP* described in this enhancement provides four security services: data-origin authentication, data-integrity verification, privacy, and protection from replay.

One of the challenges in defining the secure SNMP protocols was to leave the basic SNMP messages unchanged. The designers accomplished this by identifying procedures that left the basic packets intact and added "wrappers" containing the enhanced security-related information to the message formats. Obviously, even with the basic packet headers left unchanged, there are some changes to SNMP to accommodate the security features. These changes involve the use of SNMP parties, a data-authentication algorithm, a data-privacy algorithm, and loosely synchronized clocks. Instead of using community strings, secure SNMP uses *SNMP parties*. An SNMP party represents a role that an SNMP entity takes when performing SNMP management operations. Associated with each SNMP party is a set of information specific to that party; this information includes SNMP party identifier, transport address, acceptable message size, and secret keys for the protocols. The secure SNMP protocol provides for use of both an authentication protocol and a privacy protocol for protecting SNMP messages. Currently, the message digest 5 (MD5) algorithm (RFC 1321) is suggested for the authentication protocol, and the U.S. national data-encryption standard (DES) is suggested for use as the privacy protocol. However, the structure of the protocols is modular, and alternate authentication and privacy protocols may be used. The secure SNMP documents outline a system using private keys with placeholders established for future public-key system use. The secure SNMP documents also outline a mechanism for key management using SNMP.

OSI Common Management: Flexibility, At A Price

ISO began developing a network management extension to the OSI reference model in the early 1980s, but did not complete this initial design phase until almost a decade later. Many factors contributed to this delay, but perhaps the single biggest factor was that—unlike the authors of SNMP, who had a mutual, clearly defined objective ("manage the Internet")—the many, varied ISO members who developed CMIP had no single, common objective. As is the case with so many ISO standards, the OSI Network Management Framework (ISO/IEC 7498-4: 1989) describes a model that is incredibly general and can essentially mean almost anything to anyone.

This model subdivides the network management problem space into five *systems management functional areas* (SMFAs)—fault, performance, configuration, security, and accounting—and enumerates tasks that are relevant to each area. This model was originally intended to organize development of individual standards that would define technologies to implement and automate each identified task. In practice, however, it was found that most tasks are common to several areas, and the concept of developing standards for each functional area has since been abandoned. The current approach is to develop standards representing *systems management functions* (SMFs), each of which defines a self-contained tool that can be used by a number of management applications. For example, the event-report management function (ISO/IEC 10164-5: 1991) defines a mechanism that can be used to forward an identified set of event reports to a given destination under specified conditions; the events themselves might be relevant to any area of management.

Like SNMP, the OSI management framework and a model known as the *systems management overview* (ISO/IEC 10040: 1991) identify a number of components that together provide management capabilities (see Figure 9.4). The components include a *managing system* (which plays the *manager role,* similar to an SNMP network management station), a *managed system* (which plays an *agent role,* similar to an SNMP network element), a *management communications protocol* (CMIP), and *management information*. An OSI *managing system* serves as the interface between management applications with which network administrators deal, and the network or system to be managed. On the other side, an OSI *managed system* provides an access point to the resources to be managed, receiving

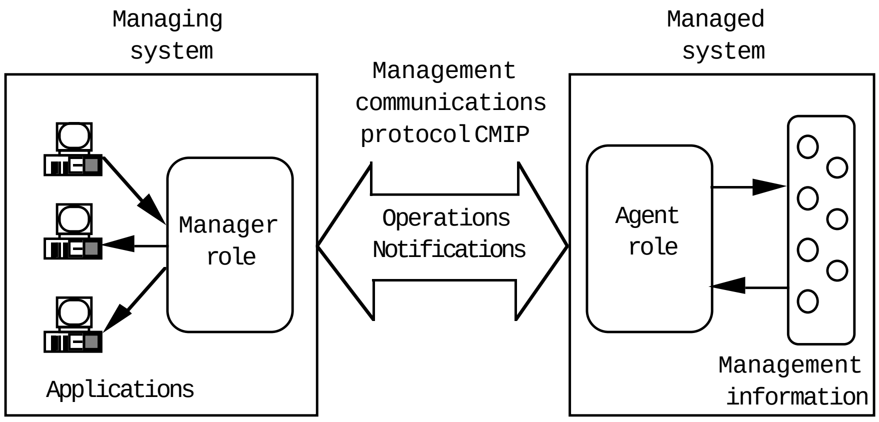


FIGURE 9.4 OSI Management Model

and carrying out management *operations* directed at specific resources and forwarding *notifications* that indicate events pertaining to those resources. The resources themselves are modeled as *managed objects*; the collection of all managed objects (conceptual) is known as the *management information base* (MIB).

AHA *Although these OSI concepts are similar to those described previously for SNMP, the details and usage can be quite different, as the authors show in this chapter. In fact, most of the difference—and hence, the debate between "simple" and "common" management—lies neither in the concepts nor in the protocol bits; rather, it lies in the management paradigm.*

What Do I Manage?—OSI Management Information Model and GDMO

Like the designers of SNMP, the architects of OSI management also recognized that the resources to be managed were many and varied and that a common representation would be necessary to manage them in a multivendor environment. The challenge was again to find a method of abstractly representing resources in a manner that would allow for common understanding but also facilitate vendor extension and future enhancement. The resulting approach, defined in the *management information model* (ISO/IEC 10165-1: 1991) and based on object-oriented technology and techniques, is extremely flexible; flexibility, however, does not come without cost.

Using the management information model, resources such as systems, protocol layer entities, and devices are modeled as *managed object classes.* Each managed object class has a number of properties that are made visible over the management interface. Properties include:

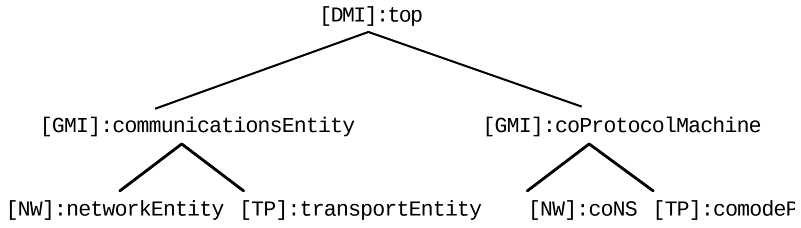
- *Attributes:* detailed information that is known (or that might be configured) about the resource
- *Notifications:* significant events that might occur during the lifetime of the resource
- *Operations:* management requests that can be performed on the resource
- *Behavior:* rules that describe the way the resource can be managed

These properties are encapsulated in a single, self-contained definition that is assigned an object class identifier. This identifier—which may be an OBJECT IDENTIFIER, as described in Chapter 5, or a simple integer—serves as the target or source of all CMIP data units. Each property—attribute, notification, or operation—is also assigned a unique identifier to be included in CMIP data units.

For example, a managing system sends a CMIP packet to retrieve the `octetsReceived` counter attribute for all resources that are protocol layer entities. However, there are many kinds of protocol layer entities, and it would be impractical to represent them all as the same object class with exactly the same set of properties. Since all protocol layer entities receive octets of incoming data, it would be nice to be able to capitalize on this so that you could develop one piece of code capable of handling this property. Thus, what is really needed is one object class that represents the properties that are common between all protocol layer entities and many specific object classes that contain both that which is common and that which is unique to a given layer. This can be accomplished using a technique known as *inheritance*, in which what is common is represented as a *superclass* and what is specific/unique is represented as one or more *subclasses.* This technique can be applied iteratively, refining and combining superclasses into more complex and specialized subclasses. Inheritance can also be handy for adding vendor extensions to a standard object class or for enhancing object classes of an old product with new features added in the next release. Even when inheritance is not used, properties that have previously been defined can be reused in other managed-object class definitions.

Another tool—*allomorphism*—can be used to take advantage of commonality between object classes. Allomorphism allows an actual resource (known as a managed-object instance)[5] to behave as several managed object classes. For example, a product that provides OSI class 4 transport service (see Chapter 12) might be viewed as a generic communications entity and connection-oriented protocol machine or as a specialized transport-layer entity and connection-mode protocol machine (see Figure 9.5). These specialized views correspond to object class definitions that are subclassed from the generic views. The view taken in a particular management interaction depends on the level of detail in which an administrator is interested. Using inheritance when defining managed object classes makes allomorphism easier but is not required—any classes that share common properties can take advantage of allomorphism, so long as the class designers reused the same attribute, notification, operation, and/or behavior definitions.

5. A managed-object class defines the type of resource to be managed; a managed-object instance is a particular resource of that type. For example, all OSI transport connections can be considered instances of the same OSI transport connection class.



[DMI] = ISO/IEC 10165-2: 1991, Definition of Management Information
[GMI] = ISO/IEC 10165-5: 1992 Generic Management Information
[NW] = ISO/IEC 10737-1: 1993 OSI Network Layer Management Information
[TP] = ISO/IEC 10733: 1993 OSI Transport Layer Management Information

FIGURE 9.5 Example of an Inheritance Tree

AHA *The term* object *is used very differently in the OSI and Internet models. OSI* managed object classes *are similar to Internet* object groups*, whereas OSI* attributes *are similar to Internet* object types. *These "similar but different" terms can be confusing and misleading. For example, it is not uncommon to see a comparison between the number of "objects" defined in OSI and Internet MIBs: an OSI MIB may define only a dozen or so object classes but probably contains attributes numbered in the hundreds—in the same ballpark as the number of object types defined by the Internet MIB II. However, beneath this terminology lies a fundamental difference between the two models—in the Internet model, individual variables are addressable units of information; in the OSI model, variables are encapsulated within classes that are addressed as a whole.*

To illustrate, consider the difference between "flat" programs, which contain a simple list of statements, and structured programs, which contain a nested hierarchy of function calls or modules. Modular programs can be more complicated to develop but can be easier to update and reuse. The same can be said of Internet and OSI "objects."

Managed object class designers have a number of tools available to them. The management information model also defines structuring techniques like *containment* (a method of hierarchically organizing class definitions; refer to the discussion of naming in the following section) and *conditional packages* (a method of clumping together related properties and assigning a condition that indicates when the properties will be present—for example, OSI transport class 4 properties are contained in a conditional package that is present only if class 4 operation has been negotiated for a transport connection). Each class definition is documented using a format called *Guidelines for the Definition of Managed Objects*, or GDMO (ISO/IEC 10165-4: 1991).

GDMO contains a number of *templates*—blank forms that are filled in to represent specific managed object classes and their properties. The OBJECT CLASS template lists the packages that make up the class. The PACKAGE template lists the properties included in the package. Each property is defined using an ATTRIBUTE, ATTRIBUTE GROUP, NOTIFICATION, ACTION, PARAMETER, or BEHAVIOR template. Containment relationships between classes are represented using a NAME BINDING template. The filled-in GDMO templates define how the object class and its properties are represented in CMIP data units. For example, the ATTRIBUTE template that defines `octetsReceived` refers to an ASN.1 type called "Count," which is defined as an ASN.1 INTEGER (ISO/IEC 10165-2: 1991). A CMIP request packet that retrieves the `octetsReceived` attribute (identified by its OBJECT IDENTIFIER) causes the agent to return a CMIP response packet containing the `octetsReceived` attribute, the OID, and an INTEGER value. Unlike the Internet structure of management information, the OSI guidelines allow any ASN.1 type to be used when defining managed object class properties.

The object-oriented philosophy of subclassing/allomorphism and reuse described in the OSI management information model has been used to define many resource-specific MIBs in GDMO format. Unlike the Internet community, which initially developed MIB I to manage the resources of the Internet, ISO has not focused its MIB[6] development efforts on managing OSI stack resources. Initial MIBs, such as the OSI Network Management Forum Library, published in 1990, were developed based on preliminary versions of the *Guidelines for the Definition of Managed Objects*. The *Definition of Management Information* (DMI; ISO/IEC 10165-2: 1991) and *Generic Managed Information* (GMI; ISO/IEC CD 10165-5: 1992), published in 1992, were the first MIBs based on the final (IS) version of the GDMO; these might be considered the ISO equivalent of Internet MIB II (RFC 1213) because they provide the foundation on which all other resource-specific MIBs are built. For example, the *Definition of Management Information* defines an object class called "Top," which is the ultimate superclass—it contains attributes that are inherited by all other managed object classes. The "definitions" and generic man-

6. In this chapter and throughout the industry, the term *MIB* is often used to refer to a collection of definitions. ISO standards have no single term for this concept, although *management information library* is often used. ISO/IEC 7498-4: 1989 defines the term *MIB* as a conceptual collection of management information.

aged information model also define object classes such as "System" and "Network," which are positioned at the top of the managed object tree visible to each agent. Now that the underlying management information model and GDMO standards have been published, dozens of resource-specific MIBs are expected to appear by the end of 1993. The first of these—the OSI transport- and network-layer MIBs (ISO/IEC 10737-1: 1992 and ISO/IEC 10733: 1992)—have already been completed.

As might be expected with a general model and format, the management information model and GDMO are being used by a wide variety of organizations throughout the industry, including standards bodies (ISO, CCITT, ANSI, IEEE, even the IETF), consortia (OSF, UI, X/Open), implementers' workshops (OIW), procurement specifiers (NIST, U.S. Air Force), and vendors (telecom, datacom, systems). In the absence of a single governing body like the IETF, it is more difficult to keep track of MIBs under development, where they are published, how they are registered, etc. To address this problem, *management information catalogs* are being developed to provide both paper and on-line listings for publishing organizations, MIBs, and object classes. Also, since any registration authority can provide object identifiers for GDMO-based MIBs, most organizations act as their own registration authority; other MIB definers use a *public registration service,* such as that provided by the OSI/NM Forum.

AHA *ISO GDMO-based MIB development got off to a slow start, but all signs indicate that an "MIB explosion" will occur in the 1993–94 time frame as the industry enters the "publish or perish" phase. Because MIBs to some extent dictate the choice of management protocol, availability of Internet MIBs for many technologies may already have locked OSI management out of some potential markets (most notably, management of devices critical for internetworking, such as routers and bridges). Thus, the eventual market success of OSI management rides largely on the ability of MIB definers to publish a comprehensive set of implementable GDMO-based MIBs—quickly!*

How Do I Manage It?—CMIP/CMIS and SMFs

Given an MIB that defines what you want to manage, the next obvious question is "How do I manage it?" In the traditional ISO style, the authors of the *Common Management Information Service* (ISO/IEC 9595: 1990) answered this question by defining a set of abstract service primitives and parameters. CMIS can be thought of as a collection of function calls or methods that can be invoked to perform operations on or receive notifications from managed-object classes. The CMIS services are: M-GET, M-SET, M-EVENT-REPORT, M-CREATE, M-DELETE, M-ACTION, and M-CANCEL-GET.

- The CMIS *M-GET* service can be used to retrieve the attributes of network and system resources—for example, to get the `octets-Received` counter attribute. This is similar to the "service" that is provided when the SNMP `Getrequest-PDU`, `GetResponse-PDU`, and `GetNextRequest-PDU` messages are used.
- The CMIS *M-SET* service can be used to modify attributes—for example, to configure OSI transport layer timer values. This is similar to the service provided via the SNMP `SetRequest-PDU` (and corresponding) `GetRequest-PDU`, `GetResponse-PDU.`
- The CMIS *M-EVENT-REPORT* can be used to signal a noteworthy occurrence—for example, an excessive number of retransmissions occurring in the OSI transport layer. This is similar to the SNMP trap message.
- The CMIS *M-CREATE* and *M-DELETE* services allow the manager to request that managed-object instances be added or removed—for example, an event-forwarding discriminator might be created to control the flow of events from the agent to the manager (more on this subject later in the chapter). There are no comparable SNMP messages, although MIB designers may define tables in which rows can be added or deleted using the SNMP `SetRequest-PDU` message.
- The CMIS *M-ACTION* service allows a manager to request that any arbitrary operation be performed on or by a resource—for example, an ACTION might be used to activate or deactivate protocol layer entities. There are no comparable SNMP messages, although, again, MIB designers might provide a variable that, when set, would initiate an operation.
- The CMIS *M-CANCEL-GET* service allows a previously requested M-GET to be aborted; the reason for this will become obvious later in the chapter, when multiple replies are discussed. There is no comparable SNMP message.

Each service primitive is converted into protocol data units by the *common management information protocol* specification (ISO/IEC 9596-1: 1990);[7] Figure 9.6 illustrates the CMIP GET packet, which is used to convey the semantics of the M-GET service.

7. CMIS and CMIP version 1 were published in 1989, vendor experimentation began, a few show demos were held, and defects discovered during this process were reflected in

```
CMIP-1 {joint-iso-ccitt ms (9) cmip (1) modules (0) protocol (3)}
DEFINITIONS ::=
BEGIN
m-GET OPERATION
  ARGUMENT     GetArgument
  RESULT       GetResult
  ERRORS       { accessDenied, classInstanceConflict,
               complexityLimitation, getListError, invalidFilter,
               invalidScope, noSuchObjectClass, noSuchObjectInstance,
               operationCancelled, processingFailure, syncNotSupported }
  LINKED       { m-Linked-Reply }
::= localValue 3

GetArgument ::= SEQUENCE { COMPONENTS OF BaseManagedObjectId,
                      accessControl    [5] AccessControl OPTIONAL,
                      sychronization   [6] IMPLICIT CMISSync
                                           DEFAULT bestEffort,
                      scope                [7] Scope DEFAULT baseObject,
                      filter               CMISFilter DEFAULT and{},
                      attributeIdList  [12] IMPLICIT SET OF AttributeId
                                           OPTIONAL }

BaseManagedObjectId ::= SEQUENCE { baseManagedObjectClass     ObjectClass,
                                   baseManagedObjectInstance ObjectInstance }

AccessControl ::= EXTERNAL
CMISSync ::= ENUMERATED { bestEffort (0), atomic (1) }

Scope ::= CHOICE {
              INTEGER { baseObject (0), firstLevelOnly (1), wholeSubtree (2) },
              individualLevels        [1] IMPLICIT INTEGER,
              baseToNthLevels         [2] IMPLICIT INTEGER }

CMISFilter ::= CHOICE {
              item   [8] FilterItem,
              and    [9] IMPLICIT SET OF CMISFilter,
              or  [10] IMPLICIT SET OF CMISFilter,
              not [11] CMISFilter }

AttributeId ::= CHOICE { globalForm [0] IMPLICIT OBJECT IDENTIFIER,
               localForm [1] IMPLICIT INTEGER }
ObjectClass ::= CHOICE { globalForm [0] IMPLICIT OBJECT IDENTIFIER,
               localForm [1] IMPLICIT INTEGER }

ObjectInstance ::= CHOICE { distinguishedName [2] IMPLICIT DistinguishedName,
              nonSpecificForm [3] IMPLICIT OCTET STRING,
              localDistinguishedName [4] IMPLICIT RDNSequence }
```

version 2, published in November 1990. At this point, to eliminate churn and provide a stable base, ISO “froze” CMIS/CMIP for the next four years. Because version 2 is the basis for most existing and future CMIS/CMIP-based products, this chapter discusses only version 2.

```
FilterItem ::= CHOICE {
    equality   [0] IMPLICIT Attribute,
    substrings [1] IMPLICIT SEQUENCE OF CHOICE {
                      initialString    [0] IMPLICIT SEQUENCE {
                                       attributeId AttributeId,
                                       string      ANY DEFINED BY attributeId },
                      anyString             [1] IMPLICIT SEQUENCE {
                                       attributeId AttributeId,
                                       string      ANY DEFINED BY attributeId },
                      finalString           [2] IMPLICIT SEQUENCE {
                                       attributeId AttributeId,
                                       string      ANY DEFINED BY attributeId }},
    greaterOrEqual    [2] IMPLICIT Attribute,
    lessOrEqual       [3] IMPLICIT Attribute,
    present           [4] AttributeId,
    subsetOf          [5] IMPLICIT Attribute,
    supersetOf        [6] IMPLICIT Attribute,
    nonNullIntersection [7] IMPLICIT Attribute }

Attribute ::= SEQUENCE { attributeId   AttributeId,
                      attributeValue   ANY DEFINED BY attributeId }
-- The following are IMPORTed from X.500 Info Framework...
DistinguishedName ::= RDNSequence
RDNSequence ::= SEQUENCE OF RelativeDistinguishedName
RelativeDistinguishedName ::= SET OF AttributeValueAssertion
AttributeValueAssertion ::= SEQUENCE { AttributeType, AttributeValue }
AttributeType ::= OBJECT IDENTIFER
AttributeValue ::= ANY


END
```

Source: ISO/IEC 9596-1: 1990

FIGURE 9.6 ASN.1 Definitions for the CMIP M-GET.request PDU

AHA *CMIS services and SNMP messages, at the surface level, are not all that different. CMIS provides a few additional services built into the protocol, whereas these operations can (mostly) be accomplished using SNMP and a cleverly defined MIB. One must dig deeper yet to see where the true differences lie . . .*

CMIP and the Protocol Suite

The OSI common management application service elements use CMIP, a connection-oriented application-layer protocol, as the means to distribute management function and information. Like other ISO-defined application service elements, the CMIP ASE uses association control (see Chapter 10) to establish, release, and abort the underlying association. CMIP defines a small amount of user information that gets carried in the association-control protocol to allow the common management service user to negotiate what the association is to be used for, but otherwise relies on the normal association-control routine to set up management dialogues. The *Systems Management Overview* (ISO/IEC 10040: 1991) defines a single application context that is used between management application service elements (see Chapter 10). When used over a full OSI stack, CMIP assumes reliable data transfer, allowing managers and agents to go about their business without worrying about detecting loss or retransmission. Insulation from the transmission characteristics of the underlying network has many benefits—management applications and agents don't need to be fine-tuned to "behave nicely" in each installation, and alternative transport mechanisms can be used with minimum disruption (for example, vendors deploy CMIP over proprietary transports as well as OSI). However, as noted in the discussion of SNMP and UDP, connection-oriented transport implies unsolicited retransmission, which can be costly—perhaps even fatal in times of network crisis, and more overhead for brief management exchanges or interrogations (i.e., one must establish a connection, exchange requests and responses, and tear down a connection, as opposed to "simply" using datagrams).

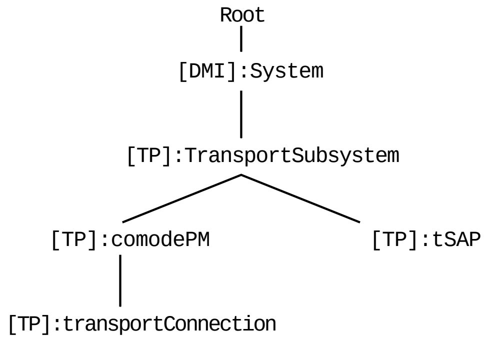
CMIP and Remote Operations

OSI common management uses the invoke/result paradigm of remote operations; the CMIP protocol relies on an application service element called *remote operations* (see Chapter 10). Every CMIS request maps to a remote operations service. The CMIP data units also map into remote operations protocol data units. These can be thought of as "envelopes" into which CMIP data units are inserted for transmission. CMIS also makes use of the linked-reply facility defined by remote operations to cover the case in which multiple replies are generated in response to a single invoke (see Chapter 10). Because CMIP is built on top of remote operations, CMIS can be used in synchronous or asynchronous modes, as described in Chapter 10.

All CMIS operations are directed to a particular resource by including in the request a managed object class ID and a managed-object instance name. The managed object class can be an OBJECT IDENTIFIER (global form) or a simple integer (local form), although the "local form" can only be used when the manager and agent have a common understanding—for example, to reduce overhead when using CMIP to manage a private or proprietary network. The managed object instance name can be either an X.500-style *distinguished name*, a *local distinguished name* (LDN), or a simple OCTETSTRING (limited to use in a well-defined context). Recall from Chapter 7 that distinguished names (DNs) are composed of a sequence of relative distinguished names (RDNs); each RDN consists of an *attribute value assertion* (AVA) containing an attribute ID and value. This syntax allows unambiguous names to be constructed in any multivendor environment by using something called a *containment tree,* an example of which is shown in Figure 9.7.

The containment tree organizes managed-object classes into a hierarchy, with the containing class being called the "superior" and the contained class being called the "subordinate." Each instance in the tree has its own name, unique within the context of its superior—this is the RDN. By concatenating all RDNs in a branch of the tree, you can derive a globally unique distinguished name that identifies one and only one managed object instance. A local distinguished name is the name derived from concatenating all RDNs in a branch of the subtree contained by the "System" managed object class.

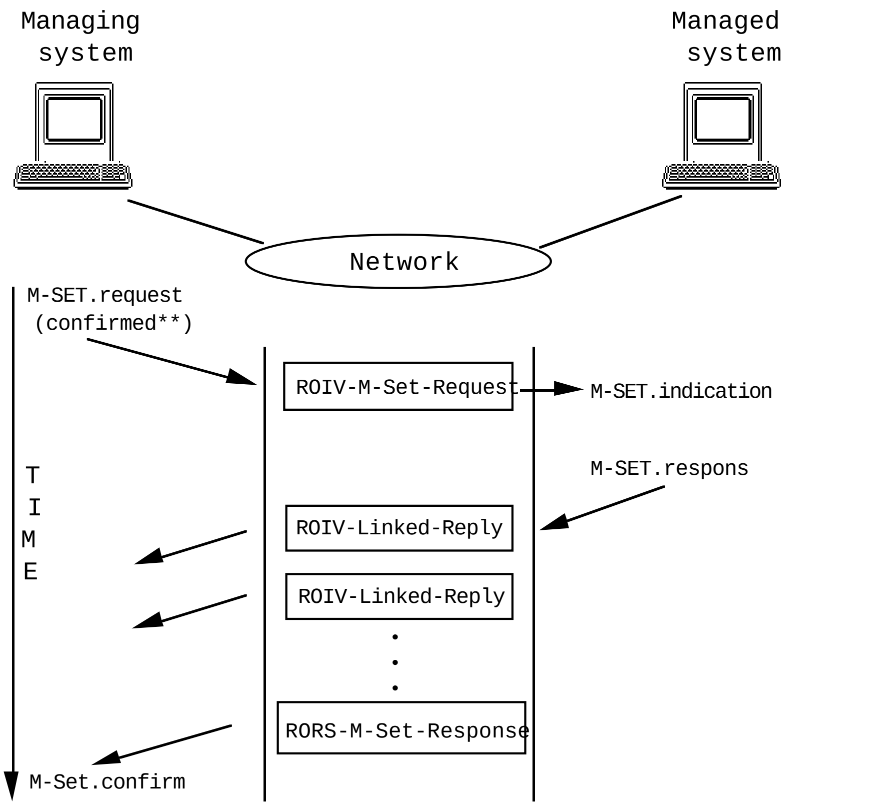
AHA *Internet naming is based on* statically defined *object identifiers, whereas OSI naming is based on* dynamically generated *distinguished names. Internet naming is always* local *to the agent, whereas OSI naming may be local to the system (LDN) or* global *(DN). As might be expected, OSI naming is more general and flexible than Internet naming, but at a cost—DNs and LDNs are typically longer than OIDs and, hence, more expensive to construct, process, and transmit.*



```
Note:Names for an instance of the transportConnection managed object class:
 RDN = { Id = "1," Value = "A" }
 LDN = { TransportSubsystem RDN + comodePM RDN + transportConnection RDN }
 DN  = { System RDN + LDN }
```

FIGURE 9.7 Example of a Containment Tree

CMIS allows, as an option, *scoping* and *filtering* capabilities, which enable the same operation to be performed on several resources by issuing a single request. The scope parameter identifies a part of the containment tree (a subtree)—for example, all instances contained by system X. The filter parameter carries a conditional statement or predicate that can be tested against attribute values to further limit the operation—for example, all instances of class "transport connection" with operational state "disabled." A *synchronization* parameter allows the operation to be treated as *atomic* (do it all or do nothing) or *best effort* (perform the operation on as many instances as you can, and inform me of the result). These optional fields allow for the composition of very powerful CMIS requests, such as "Delete all transport connections in system X that are disabled." Figure 9.8 demonstrates how remote operations linked replies



**CMIS M-SETs can be either confirmed or non-confirmed, at user discretion

FIGURE 9.8 Sample M-SET Operation

might be used by CMIP to return the results of such an operation, one reply per managed object instance (linked replies and parent-child operations are discussed in Chapter 10). In the figure, a managing system calls upon a managed system to perform the sequence of change operations associated with the M-SET.request "delete transport connections in your system that are disabled." A single CMIS request is transferred by invoking remote operations. The M-SET.request is conveyed in a remote operations packet (in the figure, the `ROIV-M-Set-Request` packet). As the agent at the managed system deletes each transport connection (each managed-object instance), a successful result is returned using remote operations linked replies (in Figure 9.8, one `ROIV-Linked-Reply` packet would be returned for each deleted managed-object instance). The completion of the management operation would be signaled back to the managing system by the arrival of a remote operations (successful) result packet (in Figure 9.8, the `RORS-M-Set-Response`).

AHA

CMIS scoping and filtering provide capabilities that go beyond the "all-powerful" SNMP GetNext but have been criticized as costly to implement and support. This is one example of a key philosophical difference between the authors of SNMP and CMIP—SNMP places the burden of management operations on the network management station, whereas CMIP allows for distribution of the load between manager and agent. It is important to note that CMIP does not require *load distribution—it can be implemented without scoping or filtering capability. Further, it can be argued that scoping and filtering must be provided for somewhere and that providing this capability as part of the management protocol simplifies management application development and reduces network management traffic. With SNMP, the management application issues many GetNext messages, sifts through the results, and figures out what is really of interest; with CMIP, the management application issues a single scoped M-GET, and the agent does the sifting, returning only the interesting stuff. Finally, it is impossible to provide the same semantics without scoping and filtering—attribute values change, and testing them as close to the resource as possible minimizes (but does not eliminate) temporal flux.*

System Management Functions

System management functions (SMFs) provide tools—protocols and managed object classes—to implement routine management tasks common to many applications. The initial batch of SMFs, published in late 1991, provide the following services:

- The *Object-Management Function* (ISO/IEC 10164-1: 1991) provides

pass-through services corresponding to all CMIS operations,[8] and a few common notifications: `attributeValueChange`, `objectCreation`, and `objectDeletion`. These notifications are defined in ISO/IEC 10165-2: 1991 and appear in many GDMO-style managed object classes.
- The *State Management Function* (ISO/IEC 10164-2: 1991) provides a few common state attributes and a common notification that can be used to signal changes in state. For example, most resources can be operational or disabled, some allow administrative control over use of the resource, and some can tell whether or not they are in use. The SMF allows this information to be represented in the same way, in any managed object class, allowing common management applications to be developed to monitor and manage the status of any resource.
- The *Attributes For Representing Relationships Function* (ISO/IEC 10164-3: 1991) provides common relationship attributes and a change notification that allow the interrelationship between managed object classes and instances to be captured in a consistent manner.
- The *Alarm-Reporting Function* (ISO/IEC 10164-4: 1991) defines common notifications to signal faults detected by the agent system. Again, common representation of this information facilitates development of common fault monitoring applications.
- The *Event Report Management Function* (ISO/IEC 10164-5: 1991) and *log control function* (ISO/IEC 10164-6: 1991) allow managers to control which notifications will be sent as CMIS M-EVENT-REPORTs or logged by the agent system.

Many other SMFs are currently available or under development, including those aimed at security, performance, and accounting management. For example, *Objects and Attributes for Access Control* (ISO/IEC 10164-9: 1991) allows security to be provided at the management object class, instance, and even operation level.

The event report and log control functions enable manager control by using object-oriented techniques. For example, the event report management function allows a manager to turn event forwarding on or off or temporarily suspend it, to specify primary and backup destinations to

8. Basing management applications and agents on the object management function (OMF) pass-through services is intended to insulate them from the underlying protocol (CMIP) and therefore allow substitution (for example, to use a local RPC mechanism instead). In practice, it turns out that there is another big advantage to using OMF instead of CMIS—OMF allows an implementation to conform to only the agent "half" of CMIS, whereas CMIP requires support for both roles.

which the events should be sent, to indicate dates and times during which forwarding should be enabled, and to specify detailed filters that can be used to selectively forward only a few events. Similarly, the log control function allows control over logging of notifications at the agent system, for subsequent retrieval by the manager.

AHA *Another key philosophical difference between OSI and Internet management is event handling. As mentioned previously, SNMP uses (but does not rely upon)* trap-directed polling, *in which only a few critical events are detected by the SNMP agent, to be supplemented by periodic or trap-initiated polling by the SNMP NMS. On the other hand, OSI might be considered* event-driven *(recall that CMIP operates over a transport connection; hence, the delivery of events is reliable). GDMO-based managed object classes tend to be defined with a number of notifications intended to signal changes in the resource. The event report and log control functions provide management control over forwarding and logging so that every little notification doesn't end up being transmitted or stored. The end result is enormous flexibility for managers and thus for the management application—network or system administrators can pick and choose what they want to poll for and what they want to be notified about. And—you guessed it—an enormous cost for the agent, which must detect, filter, throw away, store, and/or forward these notifications. Cost can be reduced by clever implementations that propagate filter criteria back into the resources themselves and pragmatic MIB design (use conditional packages to allow implementation flexibility, avoid notifying about anything and everything you can think of). Finally, it is possible to use CMIP in a polling mode, without being event-driven. In fact, most initial implementations of CMIP did just this, for the sake of simplicity—and because the event report and log control functions (ISO/IEC 10164-5: 1991; ISO/IEC 10164-6: 1991) were completed after CMIP (ISO/IEC 9596-1: 1990).*

Profiles

It is impossible to conclude any discussion of OSI without mentioning *international standardized profiles* (ISPs) for CMIP. International standardized profiles identify a collection of base standards, select options contained within the standards, and specify pragmatic constraints (for example, limiting field lengths; see Chapter 2). Because OSI standards are layered and have so many options, ISPs are necessary to enable vendors and users to build and buy products that can interoperate. For OSI management, final ISPs are available for CMIP, and draft ISPs are available for the system-management functions. For example, the *Enhanced Management Communication* profile (ISO/IEC 11183-3: 1992) specifies detailed requirements for CMIP, ROSE, ACSE, presentation, and session protocols. The *Basic Management Communication* profile (ISO/IEC ISP 11183-3: 1992) specifies similar requirements but excludes scoping and filtering features. Any vendor providing a CMIP-based product today can be expected to claim conformance to one or both of these profiles.

Putting It All Together

Having discussed the capabilities that Internet and OSI management ASEs have to offer, it is certainly appropriate to mention that, by and large, network and systems administrators really don't care a whit about management paradigms, robustness, elegance, or which "ASE" they use (if, in fact, they even know what an ASE is). They care about managing their enterprise. Today more than ever before, it is vital to provide "plug-and-play" technologies that work together to solve customer problems. Some of the strategies that have thus far been developed for integration, coexistence, and migration of management technologies are mentioned in the following subsections.

Mix-and-Match Protocols: SNMP over Foo, CMIP for the Internet

Over the past few years, there have been several attempts to "bridge the gap" between CMIP and SNMP, including the following:

- *CMIP for TCP/IP-based internets* (RFC 1095) was originally intended as a long-term management solution for the Internet. CMOT defined the use of CMIP/ROSE/ACSE over a specialized version of the OSI presentation layer (RFC 1085) over TCP/IP. This approach has since been discarded by the Internet community following a network-management "bake-off."

AHA

At a point during the parallel development of SNMP and CMOT at which no further progress could be made in committee, the SNMP and CMOT camps decided to end the debate with a bake-off. It was jointly agreed that following several months of development, the implementation worthiness of the two proposals would provide the final input. According to Internet folklore, the SNMP camp was able to demonstrate multiple, interoperable implementations, and the CMOT camp had none.

- More recently, the IETF considered a full-blown CMIP for the Internet (RFC 1189), which would eventually obsolete the original

CMOT effort. This approach glued the entire OSI CMIP upper-layer stack (down through session) onto TCP/IP using RFC 1006. A GDMO-based version of the Internet MIB II, called the *OIM MIB II* (RFC 1214) was to be developed. This MIB was intended to allow management of Internet resources using CMIP protocols. This activity came under IESG review in the spring of 1992, and the working group was disbanded due to inactivity. Members of the X/Open and OSI Network Management Forum expect to produce a CMIP over TCP/IP using RFC 1006 as described earlier; however, RFC 1214 will be abandoned in favor of the CMIP profiles mentioned in the preceding subsection.

- *"SNMP over OSI"* (RFC 1418) and, more recently, *"SNMP over Foo"* (read "anything") RFCs have been developed by the IETF to glue SNMP over non-UDP/TCP transport technologies. This approach allows a single management protocol (SNMP) to be used in virtually any networking environment. The MIBs are different from those developed for CMIP, but for multiprotocol networks, and for those configurations where SNMP agents cannot be reached using UDP, these extensions are expected to serve the Internet community quite well.
- Although little standards activity is involved, CMIP network management systems vendors, cognizant of the market availability of communications equipment that supports SNMP agents, have developed proxy elements in their CMIP-based products to gather information and alarms from SNMP agents and integrate them into the management capabilities they offer.

Dual-Stack Approach: Hide It Under the API

It is possible, and in some circumstances, quite pragmatic, to offer "dual-stack" products, which provide for concurrent application use of more than one technology (OSI and TCP/IP). In fact, "dual stack" is increasingly becoming "multistack," in which supporting technologies are not limited to Internet and OSI protocols but include RPC-based communication environments and the like. When more than one technology is used, applications can be insulated from this fact by using a common, transparent *application programming interface* (API). For example, the X/Open management protocols API provides programmatic interfaces to both SNMP and CMIP and makes some attempt at limited transparency, at least for services that are common to both management protocols—for example, an `mp_get_request` "C" function call is provided that can initiate an SNMP Get message, a CMIS M-GET.request, or perhaps even a local RPC query.

Unfortunately, the way in which a MIB is defined often precludes application transparency—for example, a MIB that uses CMIS M-ACTION cannot be managed using SNMP unless it has been translated to an Internet SMI-style MIB and a gateway function is built to map between the two MIBs below the level of the API. This approach is currently undergoing specification and development by consortia such as X/Open and the OSI/NM Forum.

On the Horizon: Simple Management Protocol (SMP) . . . or Is That "SNMPv2"?

Like most tools, network management isn't very useful if it's not deployed. As a practical consideration, the authors and advocates of SNMP carefully guarded the protocol and management framework against the sort of meddling that caused the OSI common management effort to become bogged down in definition so that it could be widely deployed as quickly as possible. The advent of the secure SNMP enhancements, however compatible with the present SNMP standards, clearly called for revisions to existing implementations. This posed a problem of no small measure to the SNMP community: there were other minor deficiencies that the Internet community would eventually wish to address, and the thought of two revisions and accompanying transitions was, to say the least, troublesome. Following a March 1992 IESG call for contributions on the future of network management for the Internet, a proposal for a *Simple Management Protocol* was presented to the IETF in August 1992. The proposal, currently a set of seven Internet drafts, provides a strategy for coexistence with the current SNMP, addresses some of the identified weaknesses of the SNMP framework, and claims to improve SNMP performance. SMP uses the secure SNMP enhancements but, as an outgrowth of implementation experience, it has exposed and consequently solved some operational problems encountered. (The result is that SMP implementations deviate from the secure SNMP RFCs, and the changes the SMP introduces to secure SNMP are now under consideration in the IETF; once approved, the secure SNMP RFCs will be made historical, and there will be only one transition for internetwork management; i.e., from SNMP version 1 to the eventual SMP/SNMPv2 RFCs.)

The more visible SMP enhancements to the SNMP include a change in the composition of the Trap message (it is now identical in structure to the other SNMP data units). A new message type, dubbed the "awesome" GetBulk operator, improves the way an SNMP manager retrieves large numbers of managed objects. A large set of error codes has been identified to improve exception reporting. Transport mappings of the SMP onto OSI, AppleTalk, and Novell/IPX as well as the recommended UDP are provided. There are also a considerable number of changes to the structure of management information, including some new data types (enumerated BITSTRING and 64-bit counters, an OSI network address data type). Finally, and notably, an *Inform Request* message has been introduced, with an accompanying MIB, to be used for manager-to-manager communications; the `InformRequest-PDU` can be used as an acknowledged event notification and to transfer information between network management stations. Network management stations can thus operate in a dual role—manager of resources as well as a resource or subordinate to other NMSs. A very fortuitous side effect of the timing of the introduction of SMP is that extensions for security, as well as improvements both desired and necessary, can be incorporated into existing implementations in a single version change, and with a well-defined coexistence plan.

AHA *During recent deliberations about SMP, an IETF member preferring anonymity suggested that SMP "is kinda like a datagram version of CMIP, but without scoping and filtering." Indeed, SMP seems to have addressed many of the much-ballyhooed shortcomings and deficiencies, and it will likely refuel the SNMP versus CMIP fire. If the existing base of SNMP agent implementations can be upgraded quickly, and the claims of improved and more robust functionality are warranted, SMP as SNMP version 2 will extend the market lead it currently holds over CMIP. Given that transport mappings are provided to allow SNMP to operate over OSI, AppleTalk, and Novell/IPX, and taking into consideration the practical difficulties of managing a multiprotocol Internet, the argument that there exists a single management framework to manage* all *aspects of a diverse Internet may be a bit premature and somewhat overstated, but it remains compelling.*

Where To from Here?

Most of the attention paid to network management is focused on the management protocols—SNMP and CMIP—and frameworks. However, if you ask a network administrator about products based on these protocols, you will find that although having standard protocols for network management is important, daily operations staff are still concerned that many aspects of network operations remain unattended, and network administrators will quickly tell you that reachability programs like ping, traceroute, netstat, and similar utilities continue to be important components of daily network operations.

The UNIX *ping* command uses the echo request datagram of the Internet control message protocol (ICMP; see Chapter 13) to evoke an echo reply datagram from a designated host on a TCP/IP network. It is by far the most commonly used fault-isolation tool. Ping provides a means of determining whether a host is "IP-reachable" and is useful for obtaining (a coarse measure of) round-trip times; many network management stations use ping in an automated fashion to monitor network connectivity.[9] An OSI version of echo/ping is described in RFC 1139[10] and will be incorporated as two new packet types (request and reply) in the second edition of ISO/IEC 8473, the *OSI Connectionless Network Layer Protocol* (CLNP; see Chapter 13), providing OSI network operations folks with a tool whose features are equivalent to the ICMP-based ping.

Traceroute is a program that determines the route an IP packet would follow from a source to a designated host by issuing a series of user datagram protocol (UDP) packets directed to a bogus destination port and with the internet protocol "time to live" set to intentionally small values, to elicit error messages from gateways that are attempting to forward the UDP packet. A gateway that receives the UDP packet but cannot forward it because its time to live has expired must return an ICMP error message "time exceeded." From these messages, the traceroute program constructs a list of gateways that comprise the route between the sender and the destination. The "probes" begin with a time to live of 1 and increase by 1 until an ICMP message "port unreachable" is returned, indicating that the host has been reached, but the specified port is an incorrect one. Traceroute is an effective diagnostic tool for network operations; a companion OSI traceroute program has been developed by members of the Network OSI Operations (NOOP) group within the IETF.

Network administrators are also quick to point out that there is too much effort directed at "managing everything that moves . . . ," and MIB mania has left vendors with little alternative but to continue to focus their efforts on the development of new MIBs. At the agent level, MIB mania increases the management burden; equally problematic is that additional MIB development in network management applications is performed at the expense of creating better diagnostic aids and applica-

9. It is interesting to note that in the manual pages of some UNIX OSs, automating ping is discouraged; for example, the manual page for ping that accompanies the MACH^{Ten} UNIX® system from Tenon Intersystems says ping "should be used primarily for manual fault isolation. Because of the load it could impose on the network, it is unwise to use ping during normal operations or from automated scripts."

10. This RFC is soon to be obsoleted; the replacement RFC will eliminate one of the two elective ways to ping and align exactly with the ISO standard.

tions that "heal" networks without manual intervention. There is also altogether too much effort directed at simply presenting information to network administrators in the jazziest manner; graphic user interfaces aside, most of the network management systems today offer little to simplify the task of managing. SNMP and CMIP can provide solicited events or traps, but all too often, a human must still intervene to identify and solve the underlying real-world problem. Tools now exist to measure network utilization, but more and better tools are needed to assist in network analysis and capacity planning and provisioning.

If you are in network operations, before getting all worked up about which protocol you use, be forewarned: you won't be giving up your pager any time soon.

AHA *A closing commentary on SNMP versus CMIP: It should be apparent that entirely different perspectives and mind-sets drove and still drive SNMP and CMIP. CMIP is to a large extent telephony-driven and based on a telco view of the composition of packet and fast-packet networks. This view is inherited in part from telephony operations and managing voice connections. In this view, switching systems are comprised of two components: the switching or network elements and computers that are used to manage groups of network elements, called* supervisory systems. *Supervisory systems are themselves managed by yet another and typically more centralized tier of* operations systems. *In this environment, the OSI common management framework is imposed between the supervisory and operations systems (the management framework and protocol used between supervisory systems and network elements is often proprietary). The popular term for this is* management of managers. *In theory, the network elements—the equivalent of routers and bridges of the public networks—don't bear the burden of CMIP/CMIS (in practice, counting and reporting often take their toll). From this example, it can be seen that advocates of OSI common management may have based their framework on the assumption that the managed resources themselves do an awful lot of management. In SNMP, which was developed with a datagram internetworking paradigm in mind, there is an assumption that management is not the principal function of the managed resource—routers, bridges, and hosts all have better things to do with their CPU/links/memory than manage themselves, and so management overhead for these resources should be small (in fact, it can be argued that left to manage themselves, they would inevitably miss the bigger picture of the network and screw up). Is this insight enough to appreciate the differences? Hardly enough to fully appreciate them, but enough to start.*

Conclusion

This chapter has compared the "simple" network management framework for the Internet to OSI's "common" management approach. Both SNMP and CMIP base services on interactions between a manager and its agents and, by extension, interactions between managers. Both model management information in an object-oriented fashion and use abstract syntax notation one for object typing, identification, and encoding. The similarities end there. Through examples, the authors demonstrated that the management paradigms of SNMP and CMIP are vastly different and showed how SNMP attempts to place the burden of network management on the managing side, while CMIP distributes the burden between a manager and its agents. It was suggested that this difference is partly attributable to what the Internet (traditionally and decidedly a *data* community) and OSI (strongly influenced by the *telephony* community) interpret as managed network resources. Finally, the chapter called readers' attention to the fact that even after one decides which management framework is best for his or her networking environment, there are other, equally important management tools to consider and briefly described what TCP/IP and OSI offer as management instrumentation to complement network management protocols.