Chapter 8: Network Management

In this final chapter we provide a brief introduction into network management and firewalls. We begin my motivating the need to provide appropriate tools for the network administrator--the person whose job is to keep the network up and running. These tools are for monitoring, testing, polling, configuring, analyzing, evaluating, and controlling the operation of a network. We discuss the five key components of a network management architecture: (1) a network manager, (2) a set of managed remote devices, (3) management information bases (MIBs), (4) remote agents that report MIB information and take action under the control of the network manager, and (5) a protocol for communicating between the network manager and the remote devices. We then delve into the details of the Internet Network Management Framework, and the SNMP protocol in particular. We also examine ASN.1 in some detail. We conclude this chapter with a discussion on firewalls--a topic that falls within the realms of both security and network management. We discuss how packet filtering and application-level gateways can be used to provide the network with some level of protection against unwanted intruders.

Online Book

8.1: What Is Network Management?

Having made our way through the first seven chapters of this text, we're now well aware that a network consists of *many* complex, interacting pieces of hardware and software--from the links, bridges, routers, hosts, and other devices that comprise the physical components of the network to the many protocols (in both hardware and software) that control and coordinate these devices. When hundreds or thousands of such components are cobbled together by an organization to form a network, it is not surprising that components will occasionally malfunction, that network elements will be misconfigured, that network resources will be overutilized, or that network components will simply "break" (for example, a cable will be cut, a can of soda will be spilled on top of a router). The network administrator, whose job it is to keep the network "up and running," must be able to respond to (and better yet, avoid) such mishaps. With potentially thousands of network components spread out over a wide area, the network administrator in a network operations center (NOC) clearly needs tools to help monitor, manage, and control the network. In this chapter, we'll examine the architecture, protocols, and information base used by a network administrator in this task.

Before diving in to network management itself, let's first consider a few illustrative "real-world" non-networking scenarios in which a complex system with many interacting components must be monitored, managed, and controlled by an administrator. Electrical power-generation plants (at least as portrayed in the popular media in such movies as the *China Syndrome*) have a control room where dials, gauges, and lights monitor the status (temperature, pressure, flow) of remote valves, pipes, vessels, and other plant components. These devices allow the operator to monitor the plant's many components, and may alert the operator (the famous flashing red warning light) when trouble is imminent. Actions are taken by the plant operator to control these components. Similarly, an airplane cockpit is instrumented to allow a pilot to monitor and control the many components that make up an airplane. In these two examples, the "administrator" *monitors* remote devices and *analyzes* their data to ensure that they are operational and operating within prescribed limits (for example, that a core meltdown of a nuclear power plant is not imminent, or that the plane is not about to run out of fuel), *reactively controls* the system by making adjustments in response to the changes within the system or its environment, and *proactively manages* the system (for example, by detecting trends or anomalous behavior, allowing action to be taken before serious problems arise). In a similar sense, the network administrator will actively monitor, manage, and control the system with which she/he is entrusted.

In the early days of networking, when computer networks were research artifacts rather than a critical infrastructure used by millions of people a day, "network management" was an unheard of thing. If one encountered a network problem, one might run a few pings to locate the source of the problem and then modify system settings, reboot hardware or software, or call a remote colleague to do so. (A very readable discussion of the first major "crash" of the ARPAnet on October 27, 1980, long before network management tools were available, and the efforts taken to recover from and understand the crash is RFC 789). As the public Internet and private intranets have grown from small networks into a large global infrastructure, the need to more systematically manage the huge number of hardware and software components within these networks has grown more important as well.

In order to motivate our study of network management, let's begin with a simple example. Figure 8.1 illustrates a small network consisting of three routers, and a number of hosts and servers.

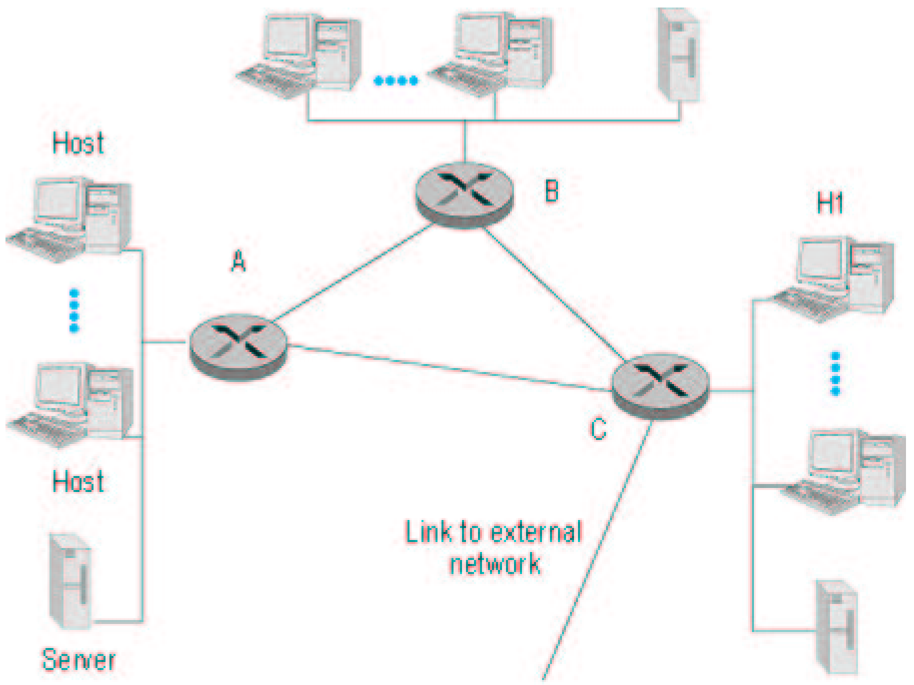

Figure 8.1: A simple scenario illustrating the uses of network management

Even in such a simple network, there are many scenarios in which a network administrator might benefit tremendously from having appropriate network management tools:

- *Failure of an interface card* at a host or a router. With appropriate network management tools, a network entity (for example router A) may report to the network administrator that one of its interfaces has gone down. (This is certainly preferable to a phone call to the NOC from an irate user who says the network connection is down!) A network administrator who actively monitors and analyzes network traffic may be able to *really* impress the would-be irate user by detecting problems in the interface ahead of time and replacing the interface card before it fails. This might be done, for example, if the administrator noted an increase in checksum errors in frames being sent by the soon-to-die interface.

- *Host monitoring.* Here, the network administrator might periodically check to see if all network hosts are up and operational. Once again, the network administrator may be able to really impress a network user by proactively responding to a problem (host down) before it is reported by a user.

- *Monitoring traffic to aid in resource deployment.* A network administrator might monitor source-to-destination traffic patterns and notice, for example, that by switching servers between LAN segments, the amount of traffic that crosses multiple LANs could be significantly decreased. Imagine the happiness all around (especially in higher administration) when better performance is achieved with no new equipment costs. Similarly, by monitoring link utilization, a network administrator might determine that a LAN segment, or the external link to the outside world is overloaded and a higher-bandwidth link should thus be provisioned (alas, at an increased cost). The network administrator might also want to be notified automatically when congestion levels on a link exceed a given threshold value, in order to provision a higher-bandwidth link before congestion becomes serious.

- *Detecting rapid changes in routing tables.* Route flapping--frequent changes in the routing tables--may indicate instabilities in the routing or a misconfigured router. Certainly, the network administrator who has improperly configured a router would prefer to discover the error his/herself, before the network goes down.
- *Monitoring for SLAs.* With the advent of **Service Level Agreements (SLA)**--contracts that define specific performance metrics and acceptable levels of network provider performance with respect to these metrics--interest in traffic monitoring has increased significantly over the past few years [Larsen 1997; Huston 1999a]. UUnet and AT&T are just two of the many network providers that guarantee SLAs [UUNet 1999; AT&T SLA 1998] to their customers. These SLAs include service availability (outage), latency, throughput, and outage notification requirements. Clearly, if performance criteria are to be part of a service agreement between a network provider and its users, then measuring and managing performance will be of great importance to the network administrator.
- *Intrusion detection.* A network administrator may want to be notified when network traffic arrives from, or is destined to, a suspicious source (for example, host or port number). Similarly, a network administrator may want to detect (and in many cases filter) the existence of certain types of traffic (for example, source-routed packets, or a large number of SYN packets directed to a given host) that are known to be characteristic of certain attacks.

The International Organization for Standards (ISO) has created a network management model that is useful for placing the above anecdotal scenarios in a more structured framework. Five areas of network management are defined:

- *Performance management.* The goal of performance management is to quantify, measure, report, analyze, and control the performance (for example, utilization, throughput) of different network components. These components include in dividual devices (for example, links, routers, and hosts) as well as end-to-end abstractions such as a path through the network. We will see shortly that protocol standards such as the Simple Network Management Protocol (SNMP) [RFC 2570] play a central role in Internet performance management.

- *Fault management.* The goal of fault management is to log, detect, and respond to fault conditions in the network. The line between fault management and performance management is rather blurred. We can think of fault management as the immediate handling of transient network failures (for example, link, host, or router hardware or software outages), while performance management takes the longer term view of providing acceptable levels of performance in the face of varying traffic demands and occasional network device failures. As with performance management, the SNMP protocol plays a central role in fault management.

- *Configuration management.* Configuration management allows a network manager to track which devices are on the managed network and the hardware and software configurations of these devices.

- *Accounting management.* Accounting management allows the network manager to specify, log, and control user and device access to network resources. Usage quotas, usage-based charging, and the allocation of resource-access privileges all fall under accounting management.

- *Security management.* The goal of security management is to control access to network resources according to some well-defined policy. The key distribution centers and certification authorities that we studied in Section 7.5 are components of security management. The use of firewalls to monitor and control external access points to one's network, a topic we will study in Section 8.4, is another crucial component.

In this chapter, we'll cover only the rudiments of network management. Our focus will be purposefully narrow--we'll examine only the *infrastructure* for network management--the overall architecture, network management protocols, and information base through which a network administrator "keeps the network up and running." We'll *not* cover the decision-making processes of the network administrator, who must plan, analyze, and respond to the management information that is conveyed to the NOC. In this area, topics such as fault identification and management [Katzela 1995; Medhi 1997], proactive anomaly detection [Thottan 1998], alarm correlation [Jakobson 1993], and more come into consideration. Nor will we cover the broader topic of service management [Saydam 1996]--the provisioning of resources such as bandwidth, server capacity, and the other computational/communication resources needed to meet the mission-specific service requirements of an enterprise. In this latter area, standards such as TMN [Glitho 1995; Sidor 1998] and TINA [Hamada 1997] are larger, more encompassing (and arguably much more cumbersome) standards that address this larger issue. TINA, for example, is described as "a set of common goals, principles, and concepts that cover the management of services, resources, and parts of the Distributed Processing Environment" [Hamada 1997]. Clearly, all of these topics are enough for a separate text and would take us a bit far afield from the more technical aspects of computer networking. So, as noted above, our more modest goal here will be to cover the important "nuts and bolts" of the infrastructure through which the network administrator keeps the bits flowing smoothly.

An often-asked question is "What is network management?". Our discussion above has motivated the need for, and illustrated a few of the uses of, network management. We'll conclude this section with a single-sentence (albeit a rather long, run-on sentence) definition of network management from [Saydam 1996]:

> *"Network management includes the deployment, integration, and coordination of the hardware, software, and human elements to monitor, test, poll, configure, analyze, evaluate, and control the network and element resources to meet the real-time, operational performance, and Quality of Service requirements at a reasonable cost."*

It's a mouthful, but it's a good workable definition. In the following sections, we'll add some meat to this rather bare-bones definition of network management.

Online Book

8.2: The Infrastructure for Network Management

We've seen in the previous section that network management requires the ability to "monitor, test, poll, configure, . . . and control" the hardware and software and components in a network. Because the network devices are distributed, this will minimally require that the network administrator be able to gather data (for example, for monitoring purposes) from a remote entity and be able to affect changes (for example, control) at that remote entity. A human analogy will prove useful here for understanding the infrastructure needed for network management.

Imagine that you're the head of a large organization that has branch offices around the world. It's your job to make sure that the pieces of your organization are operating smoothly. How would you do so? At a minimum, you'll periodically gather data from your branch offices in the form of reports and various quantitative measures of activity, productivity, and budget. You'll occasionally (but not always) be explicitly notified when there's a problem in one of the branch offices; the branch manager who wants to climb the corporate ladder (perhaps to get your job) may send you unsolicited reports indicating how smoothly things are running at his/her branch. You'll sift through the reports you receive, hoping to find smooth operations everywhere, but no doubt finding problems in need of your attention. You might initiate a one-on-one dialogue with one of your problem branch offices, gather more data in order to understand the problem, and then pass down an executive order ("Make this change!") to the branch office manager.

Implicit in this very common human scenario is an infrastructure for controlling the organization--the boss (you), the remote sites being controlled (the branch offices), your remote agents (the branch office managers), communication protocols (for transmitting standard reports and data, and for one-on-one dialogues), and data (the report contents and the quantitative measures of activity, productivity, and budget). Each of these components in human organizational management has a counterpart in network management.

The architecture of a network management system is conceptually identical to this simple human organizational analogy. The network management field has its own specific terminology for the various components of a network management architecture, and so we adopt that terminology here. As shown in Figure 8.2, there are three principle components of a network management architecture: a managing entity (the boss in our above analogy--you), the managed devices (the branch office), and a network management protocol.

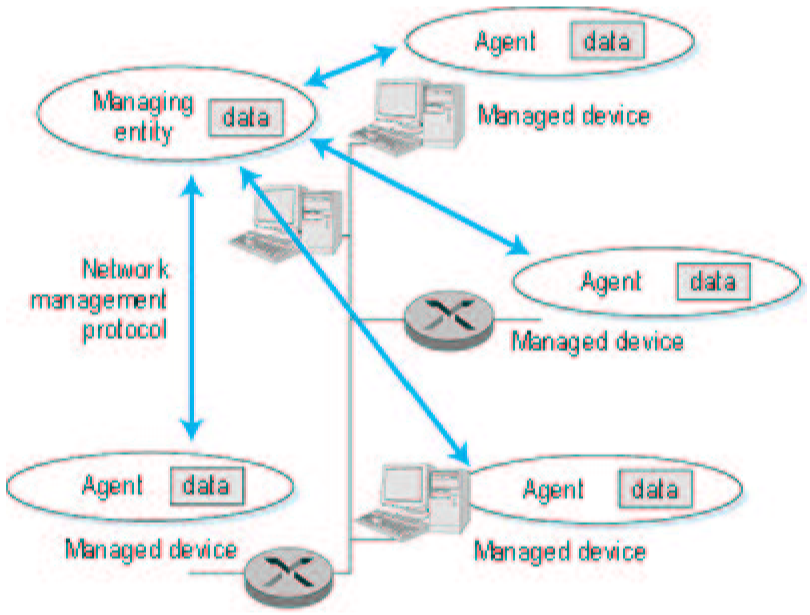

Figure 8.2: Principal components of a network management architecture

The **managing entity** is an application, typically with a human in the loop, running in a centralized network management station in the network operations center (NOC). The managing entity is the locus of activity for network management; it controls the collection, processing, analysis, and/or display of network management information.

It is here that actions are initiated to control network behavior and here that the human network administrator interacts with the network devices.

A **managed device** is a piece of network equipment (including its software) that resides on a managed network. This is the branch office in our human analogy. A managed device might be a host, router, bridge, hub, printer, or modem device. Within a managed device, there may be several so-called **managed objects.** These managed objects are the actual pieces of hardware within the managed device (for example, a network interface card), and the sets of configuration parameters for the pieces of hardware and software (for example, an intradomain routing protocol such as RIP). In our human analogy, the managed objects might be the departments within the branch office. These managed objects have pieces of information associated with them that are collected into a **management information base (MIB);** we'll see that the values of these pieces of information are available to (and in many cases able to be set by) the managing entity. In our human analogy, the MIB corresponds to quantitative data (measures of activity, productivity, and budget, with the latter being setable by the managing entity!) exchanged between the branch office and the main office. We'll study MIBs in detail in Section 8.3. Finally, also resident in each managed device is a **network management agent,** a process running in the managed device that communicates with the managing entity, taking local actions on the managed device under the command and control of the managing entity. The network management agent is the branch manager in our human analogy.

The third piece of a network management architecture is the **network management protocol.** The protocol runs between the managing entity and the managed devices, allowing the managing entity to query the status of managed devices and indirectly take actions at these devices via its agents. Agents can use the network management protocol to inform the managing entity of exceptional events (for example, component failures or violation of performance thresholds). It's important to note that the network management protocol does not itself manage the network. Instead, it provides a tool with which the network administrator can manage ("monitor, test, poll, configure, analyze, evaluate and control") the network. This is a subtle, but important distinction.

Although the infrastructure for network management is conceptually simple, one can often get bogged down with the network-management-speak vocabulary of "managing entity," "managed device," "managing agent," and "management information base." For example, in network-management-speak, our simple host monitoring scenario, "managing agents" located at "managed devices" are periodically queried by the "managing entity"--a simple idea, but a linguistic mouthful! Hopefully, keeping the human organizational analogy and its obvious parallels with network management in mind will be of help as we continue through this chapter.

Our discussion of network management architecture above has been generic, and broadly applies to a number of the network-management standards and efforts that have been proposed over the years. Network-management standards began maturing in the late 1980s, with OSI **CMISE/CMIP (the Common Management**

Service Element/Common Management Information Protocol) [Piscatello 1993; Stallings 1993; Glitho 1998] and the Internet **SNMP (Simple Network-Management Protocol)** [Stallings 1993; RFC 2570, Stallings 1999; Rose 1996] emerging as the two most important standards [Miller 1997; Subramanian 2000]. Both are designed to be independent of vendor-specific products or networks. Because SNMP was quickly designed and deployed at a time when the need for network management was becoming painfully clear, SNMP found widespread use and acceptance. Today, SNMP has emerged as the most widely used and deployed network management framework. We cover SNMP in detail in the following section.

Online Book

8.3: The Internet Network-Management Framework

Contrary to what the name SNMP (Simple Network-Management Protocol) might suggest, network management in the Internet is much more than just a protocol for moving management data between a management entity and its agents, and has grown to be much more complex than the word "simple" might suggest. The current Internet Standard Management Framework traces its roots back to the Simple Gateway Monitoring Protocol, SGMP [RFC 1028]. SGMP was designed by a group of university network researchers, users, and managers, whose experience with SGMP allowed them to design, implement, and deploy SNMP in just a few months [Lynch 1993]--a far cry from today’s rather drawn-out standardization process. Since then, SNMP has evolved from SNMPv1 through SNMPv2 to the most recent version, SNMPv3 [RFC 2570], released in April 1999.

When describing any framework for network management, certain questions must inevitably be addressed:

- What (from a semantic viewpoint) is being monitored? And what form of control can be exercised by the network administrator?
- What is the specific form of the information that will be reported and/or exchanged?
- What is the communication protocol for exchanging this information?

Recall our human organizational analogy from the previous section. The boss and the branch managers will need to agree on the measures of activity, productivity, and budget used to report the branch office’s status. Similarly, they’ll need to agree on the actions the boss can take (for example, cut the budget, order the branch manager to change some aspect of the office's operation, or fire the staff and shut down the branch office). At a lower level of detail, they'll need to agree on the form in which this data is reported. For example, in what currency (dollars, euros?) will the budget be reported? In what units will productivity be measured? While these are trivial details, they must be agreed upon, nonetheless. Finally, the manner in which information is conveyed between the main office and the branch offices (that is, their communication protocol) must be specified.

The Internet Network-Management Framework addresses the questions posed above. The framework consists of four parts:

- Definitions of *network-management objects* known as MIB objects. In the Internet network-management framework, management information is represented as a collection of managed objects that together form a virtual information store, known as the Management Information Base (MIB). An MIB object might be a counter, such as the number of IP datagrams discarded at a router due to errors in an IP datagram header; or the number of carrier sense errors in an Ethernet interface card; descriptive information such as the version of the software running on a DNS server; status information such as whether a particular device is functioning correctly or not; or protocol-specific information such as a routing path to a destination. MIB objects thus define the management information maintained by a managed node. Related MIB objects are gathered into so-called **MIB modules.** In our human organization analogy, the MIB defines the information conveyed between the branch office and the main office.

- *A data definition language,* known as SMI (Structure of Management Information) that defines the data types, an object model, and rules for writing and revising management information. MIB objects are specified in this data definition language. In our human organizational analogy, the SMI is used to define the details of the *format* of the information to be exchanged.

- A *protocol, SNMP,* for conveying information and commands between a managing entity and an agent executing on behalf of that entity within a managed network device.

- *Security and administration capabilities.* The addition of these capabilities represents the major enhancement in SNMPv3 over SNMPv2.

The Internet network-management architecture is thus modular by design, with a protocol-independent data-definition language and MIB, and an MIB-independent protocol. Interestingly, this modular architecture was first put in place to ease the transition from an SNMP-based network management to a network-management framework being developed by the International Organization for Standardization (ISO), the competing network-

management architecture when SNMP was first conceived--a transition that never occurred. Over time, however, SNMP's design modularity has allowed it to evolve through three major revisions, with each of the four major parts of SNMP discussed above evolving independently. Clearly, the right decision about modularity was made, if even for the wrong reason!
In the following four subsections, we cover the four major components of the Internet network-management framework in more detail.

8.3.1: Structure of Management Information: SMI

The **Structure of Management Information, SMI** (a rather oddly named component of the network-management framework whose name gives no hint of its functionality), is the language used to define the management information residing in a managed-network entity. Such a definition language is needed to ensure that the syntax and semantics of the network-management data are well-defined and unambiguous. Note that the SMI does not define a specific instance of the data in a managed-network entity, but rather the language in which such information is specified. The documents describing the SMI for SNMPv3 (which rather confusingly, is called SMIv2) are [RFC 2578; RFC 2579; RFC 2580]. Let us examine the SMI in a bottom-up manner, starting with the base data types in the SMI. We'll then look at how managed objects are described in SMI, and then how related managed objects are grouped into modules.

SMI Base Data Types

RFC 2578 specifies the basic data types in the SMI MIB module-definition language. Although the SMI is based on the ASN.1 (Abstract Syntax Notation One) [ISO 1987; ISO X.680 1998] object-definition language (see Section 8.4), enough SMI-specific data types have been added that SMI should be considered a data-definition language in its own right. The 11 basic data types defined in RFC 2578 are shown in Table 8.1. In addition to these scalar objects, it is also possible to impose a tabular structure on an ordered collection of MIB objects using the SEQUENCE OF construct; see RFC 2578 for details. Most of the data types in Table 8.1 will be familiar (or self-explanatory) to most readers. The one data type we will discuss in more detail shortly is the OBJECT IDENTIFIER data type, which is used to name an object.

Table 8.1: Basic data types of the SMI

Data type	**Description**
INTEGER	32 bit integer, as defined in ASN.1, with a value between -2^31 and 2^31-1 inclusive, or a value from a list of possible named constant values
Integer 32	32 bit integer with a value between -2^31 and 2^31-1 inclusive
Unsigned 32	Unsigned 32 bit integer in the range 0 to 2^23-1 inclusive

OCTET STRING	ASN.1-format byte string representing arbitrary binary or textual data, up to 65535 bytes long
OBJECT IDENTIFIER	ASN.1-format administratively assigned (structured name); see Section 8.3
IPaddress	32-bit Internet address, in network byte order
Counter32	32-bit counter that increases from 0 to 2^32-1 and then wraps around to 0.
Counter64	64-bit counter
Gauge32	32-bit integer that will not count above 2^31-1 nor decrease beyond 0 when increased or decreased
TimeTicks	Time, measured in 1/100th of seconds since some event
Opaque	Uninterrupted ASN.1 string, needed for backward compatibility

SMI Higher-Level Constructs

In addition to the basic data types, the SMI data-definition language also provides higher-level language constructs.

The OBJECT-TYPE construct is used to specify the data type, status, and semantics of a managed object. Collectively, these managed objects contain the management data that lie at the heart of network management. There are nearly 10,000 defined objects in various Internet RFC's [RFC 2570]. The OBJECT-TYPE construct has four clauses. The SYNTAX clause of an OBJECT-TYPE definition specifies the basic data type associated with the object. The MAX-ACCESS clause specifies whether the managed object can be read, be written, be created, or have its value included in a notification. The STATUS clause indicates whether object definition is current and valid, obsolete (in which case it should not be implemented, as its definition is included for historical purposes only) or deprecated (obsolete, but implementable for interoperability with older implementations). The DESCRIPTION clause contains a human-readable textual definition of the object; this "documents" the purpose of the managed object and should provide all the semantic information needed to implement the managed object.

As an example of the OBJECT-TYPE construct, consider the ipInDelivers object type definition from RFC 2011. This object defines a 32-bit counter that keeps track of the number of IP datagrams that were received at the managed node and were successfully delivered to an upper-layer protocol. The final line of this definition is concerned with the name of this object, a topic we will consider in the following section.

```
ipInDelivers OBJECT-TYPE
  SYNTAX Counter32
  MAX-ACCESS read-only
  STATUS current
  DESCRIPTION
        "The total number of input datagrams
         successfully delivered to IP user-protocols
         (including ICMP)."
  ::= { ip 9 }
```

The MODULE-IDENTITY construct allows related objects to be grouped together within a "module." For example, RFC 2011 specifies the MIB module that defines managed objects (including ipInDelivers) for managing implementations of the Internet Protocol (IP) and its associated Internet Control Message Protocol (ICMP). RFC 2012 specifies the MIB module for TCP, and RFC 2013 specifies the MIB module for UDP. RFC 2021 defines the MIB module for RMON remote monitoring. In addition to containing the OBJECT-TYPE definitions of the managed objects within the module, the MODULE-IDENTITY construct contains clauses to document contact information of the author of the module, the date of the last update, a revision history, and a textual description of the module. As an example, consider the module definition for management of the IP protocol:

```
ipMIB MODULE-IDENTITY
   LAST-UPDATED "941101000Z"
   ORGANIZATION "IETF SNMPv2 Working Group"
   CONTACT-INFO
        "       Keith McCloghrie
        Postal:  Cisco Systems, Inc.
                170 West Tasman Drive
                San Jose, CA 95134-1706
                US

        Phone:   +1 408 526 5260
        E-mail:  kzm@cisco.com"

   DESCRIPTION
        "The MIB module for managing IP and ICMP
        implementations, but excluding their
        management of IP routes."
   REVISION "9103310000Z"
   DESCRIPTION
        "The initial revision of this MIB module was
        part of MIB-II."
   ::= { mib-2 48}
```

The NOTIFICATION-TYPE construct is used to specify information regarding "SNMPv2-Trap" and "InformationRequest" messages generated by an agent, or a managing entity; see Section 8.3.3. This information includes a textual DESCRIPTION of when such messages are to be sent, as well as list of values to be included in the message generated; see RFC 2578 for details.

The MODULE-COMPLIANCE construct defines the set of managed objects within a module that an agent must implement.

The AGENT-CAPABILITIES construct specifies the capabilities of agents with respect to object- and event-notification definitions.

8.3.2: Management Information Base: MIB

As noted above, the **Management Information Base, MIB,** can be thought of as a virtual information store, holding managed objects whose values collectively reflect the current "state" of the network. These values may be queried and/or set by a managing entity by sending SNMP messages to the agent that is executing in a managed node on behalf of the managing entity. Managed objects are specified using the OBJECT-TYPE SMI construct discussed above and gathered into **MIB modules** using the MODULE-IDENTITY construct.

The IETF has been busy standardizing the MIB modules associated with routers, hosts, and other network equipment. This includes basic identification data about a particular piece of hardware, and management information about the device's network interfaces and protocols. As of the release of SNMPv3 (mid-1999), there were nearly 100 standards-based MIB modules and an even larger number of vendor-specific (private) MIB modules. With all of these standards, the IETF needed a way to identify and name the standardized modules, as well as the specific managed objects within a module. Rather than start from scratch, the IETF adopted a standardized object identification (naming) framework that had already been put in place by the International Organization for Standardization (ISO). As is the case with many standards bodies, the ISO had "grand plans" for their standardized object identification framework--to identify every possible standardized object (for example, data format, protocol, or piece of information) in any network, regardless of the network standards organization (for example, Internet IETF, ISO, IEEE, or ANSI), equipment manufacturer, or network owner. A lofty goal indeed! The object identification framework adopted by ISO is part of the ASN.1 (Abstract Syntax Notation One) [ISO 1987; ISO X.680 1998] object-definition language that we'll discuss in section 8.4. Standardized MIB modules have their own cozy corner in this all-encompassing naming framework, as discussed below.

As shown in Figure 8.3, objects are named in the ISO naming framework in a hierarchical manner. Note that each branch point in the tree has both a name and a number (shown in parentheses); any point in the tree is thus identifiable by the sequence of names or numbers that specify the path from the root to that point in the identifier tree. A fun, but incomplete and unofficial, Web-based utility for traversing part of the object identifier tree (using branch information contributed by volunteers) may be found in [Alvestrand 1997].

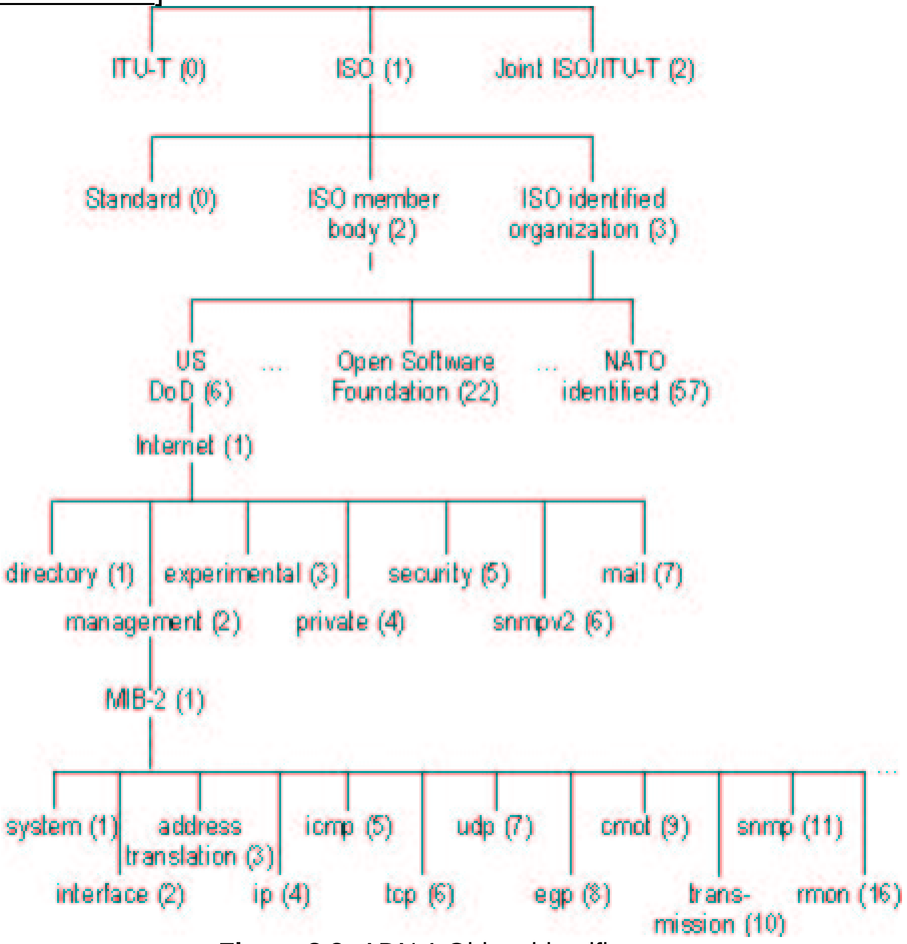

Figure 8.3: ADN.1 Object identifier tree

At the top of the hierarchy are the International Organization for Standardization (ISO) and the Telecommunication Standardization Sector of the International Telecommunication Union (ITU-T), the two main standards organizations dealing with ASN.1, as well as a branch for joint efforts by these two organizations. Under the ISO branch of the tree, we find entries for all ISO standards (1.0) and for standards issued by standards bodies of various ISO-member countries (1.2). Although not shown in Figure 8.3, under (ISO ISO-Member-Body, a.k.a. 1.2) we would find USA (1.2.840), under which we would find a number of IEEE, ANSI, and company-specific standards. These include RSA (1.2.840.11359) and Microsoft (1.2.840.113556), under which we find the Microsoft File Formats (1.2.840.113556.4) for various Microsoft products, such as Word (1.2.840.113556.4.2). But we are interested here in networking (*not* Microsoft Word files), so let us turn our attention to the branch labeled 1.3, the standards issued by bodies recognized by the ISO. These include the U.S. Department of Defense (6) (under which we will find the Internet standards), the Open Software Foundation (22), the airline association

SITA (69) and NATO-identified bodies (57), as well as many other organizations.

Under the Internet branch of the tree (1.3.6.1), there are seven categories. Under the private (1.3.6.1.4) branch, we will find a list [IANA 1999b] of the names and private enterprise codes of more than 4,000 private companies that have registered with the Internet Assigned Numbers Authority (IANA) [IANA 1999]. Under the management (1.3.6.1.2) and MIB-2 branch (1.3.6.1.2.1) of the object identifier tree, we find the definitions of the standardized MIB modules. Whew--it's a long journey down to our corner of the ISO name space!

Standardized MIB Modules

The lowest level of the tree in Figure 8.3 shows some of the important hardware-oriented MIB modules (system and interface) as well as modules associated with some of the most important Internet protocols. RFC 2400 lists all of the standardized MIB modules. While MIB-related RFCs make for rather tedious and dry reading, it is instructive (that is, like eating vegetables, it is "good for you") to consider a few MIB module definitions to get a flavor for the type of information in a module.

The managed objects falling under system contain general information about the device being managed; all managed devices must support the system MIB objects. Table 8.2 defines the objects in the system group, as defined in RFC 1213.

Table 8.2: Managed objects in the MIB-2 system group

Object Identifier	**Name**	**Type**	**Description (from RFC 1213)**
1.3.6.1.2.1.1.1	sysDescr	OCTET STRING	"full name and version identification of the system's hardware type, software operating-system, and networking software
1.3.6.1.2.1.1.2	sysObjectID	OBJECT IDENTIFIER	Vendor assigned object ID that "provides an easy and unambiguous means for determining 'what kind of box' is being managed."
1.3.6.1.2.1.1.3	sysUpTime	TimeTicks	"The time (in hundredths of a second) since the network management portion of the system was last re-initialized."
1.3.6.1.2.1.1.4	sysContact	OCTET STRING	

"The contact person for this managed node, together with information on how to contact this person."

1.3.6.1.2.1.1.5
sysName
OCTET STRING
"An administratively assigned name for this managed node. By convention, this is the node’s fully qualified domain name."

1.3.6.1.2.1.1.6
sysLocation
OCTET STRING
"The physical location of this node."

1.3.6.1.2.1.1.7
sysServices
Integer32
A coded value that indicates the set of services available at this node: physical (for example, a repeater), datalink/subnet (for example, bridge), Internet (for example, IP gateway), end-end (for example, host), applications

Table 8.3 defines the managed objects in the MIB module for the UDP protocol at a managed entity.

Table 8.3: Managed objects in the MIB-2 udp module
Object identifier
Name
Type
Description (from RFC 2013)

1.3.6.1.2.1.7.1
udpInDatagrams
Counter32
"total number of UDP datagrams to UDP users"

1.3.6.1.2.1.7.2
udpNoPorts
Counter32
"total number of received UDP datagrams for which there was no application at the destination port"

1.3.6.1.2.1.7.3
udpInErrors
Counter32
"number of received UDP datagrams that could not be delivered for reasons other than the lack of an application at the destination port"

1.3.6.1.2.1.7.4
udpOutDatagrams
Counter32
"total number of UDP datagrams sent from this entity"

1.3.6.1.2.1.7.5.
udpTable

SEQUENCE of UdpEntry
"a sequence of UdpEntry objects, one for each port that is currently open by an application, giving the IP address and the port number used by the application"

8.3.3: SNMP Protocol Operations and Transport Mappings

The Simple Network Management Protocol Version 2 (SNMPv2) [RFC 1905] is used to convey MIB information among managing entities and agents executing on behalf of managing entities. The most common usage of SNMP is in a **request-response mode** in which an SNMPv2 managing entity sends a request to an SNMPv2 agent, who receives the request, performs some action, and sends a reply to the request. Typically, a request will be used to query (retrieve) or modify (set) MIB object values associated with a managed device. A second common usage of SNMP is for an agent to send an unsolicited message, known as a **trap message,** to a managing entity. Trap messages are used to notify a managing entity of an exceptional situation that has resulted in changes to MIB object values. We saw earlier in Section 8.1 that the network administrator might want to receive a trap message, for example, when an interface goes down, congestion reaches a predefined level on a link, or some other noteworthy event occurs. Note that there are a number of important tradeoffs between polling (request-response interaction) and trapping; see the homework problems.
SNMPv2 defines seven types of messages, known generically as Protocol Data Units--PDUs, as shown in Table 8.4.

Table 8.4: SNMPv2 PDU types

SNMPv2 PDU Type	**sender-receiver**	**Description**
GetRequest	manager-to-agent	get value of one or more MIB object instances
GetNextRequest	manager-to-agent	get value of next MIB object instance in list or table
GetBulkRequest	manager-to-agent	get values in large block of data, for example values in a large table
InformRequest	manager-to-manager	inform remort managing entity of MIB values remote to its access
SetRequest	manager-to-agent	set value of one or more MIB object instances

Response
agent-to-manager or manager-to-manager
generated in response to GetRequest, GetNextRequest, GetBulkRequest, SetRequestPDU, or InformRequest

SNMPv2-Trap
agent-to-manager
inform manager of an exceptional event

The format of the PDU is shown in Figure 8.4.

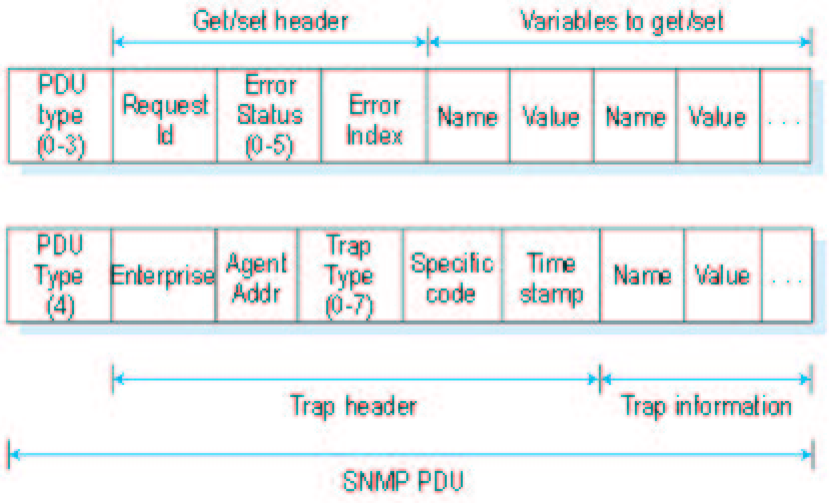

Figure 8.4: SNMP PDU format

- The GetRequest, GetNextRequest, and GetBulkRequest PDUs are all sent from a managing entity to an agent to request the value of one or more MIB objects at the agent's managed device. The object identifiers of the MIB objects whose values are being requested are specified in the variable binding portion of the PDU. GetRequest, GetNextRequest, and GetBulkRequest differ in the granularity of their data requests. GetRequest can request an arbitrary set of MIB values; multiple GetNextRequests can be used to sequence through a list or table of MIB objects; GetBulkRequest allows a large block of data to be returned, avoiding the overhead incurred if multiple GetRequest or GetNextRequest messages were to be sent. In all three cases, the agent responds with a Response PDU containing the object identifiers and their associated values.

- The SetRequest PDU is used by a managing entity to set the value of one or more MIB objects in a managed device. An agent replies with a Response PDU with the 'noError' Error Status to confirm that the value has indeed been set.

- The InformRequest PDU is used by a managing entity to notify another managing entity of MIB information that is remote to the receiving entity. The receiving entity replies with a Response PDU with the ’noError’ Error Status to acknowledge receipt of the InformRequest PDU.

- The final type of SNMPv2 PDU is the trap message. Trap messages are generated asynchronously, that is, *not* in response to a received request but rather in response to an event for which the managing entity requires notification. RFC 1907 defines well-known trap types that include a cold or warm start by a device, a link going up or down, the loss of a neighbor, or an authentication failure event. A received trap request has no required response from a managing entity.

Given the request-response nature of SNMPv2, it is worth noting here that although SNMP PDUs can be carried via many different transport protocols, the SNMP PDU is typically carried in the payload of a UDP datagram. Indeed, RFC 1906 states that UDP is "the preferred transport mapping." Since UDP is an unreliable transport protocol, there is no guarantee that a request, or its response will be received at the intended destination. The Request ID field of the PDU is used by the managing entity to number its requests to an agent; an agent’s response takes its Request ID from that of the received request. Thus, the Request ID field can be used by the managing entity to detect lost requests or replies. It is up to the managing entity to decide whether to retransmit a request if no corresponding response is received after a given amount of time. In particular, the SNMP standard does not mandate any particular procedure for retransmission, or even if retransmission is to be done in the first place. It only requires that the managing entity "needs to act responsibly in respect to the frequency and duration of re-transmissions." This, of course, leads one to wonder how a "responsible" protocol should behave!

8.3.4: Security and Administration

The designers of SNMPv3 have said that "SNMPv3 can be thought of as SNMPv2 with additional security and administration capabilities" [RFC 2570]. Certainly, there are changes in SNMPv3 over SNMPv2, but nowhere are those changes more evident than in the area of administration and security. The central role of security in SNMPv3 was particularly important, since the lack of adequate security resulted in SNMP being used primarily for monitoring rather than control (e.g., SetRequest is rarely used in SNMPv1).

As SNMP has matured through three versions, its functionality has grown but so too, alas, has the number of SNMP-related standards documents. This is evidenced by the fact that there is even now an RFC [RFC 2571] that " describes an architecture for describing SNMP Management Frameworks"! While the notion of an "architecture" for "describing a framework" might be a bit much to wrap one’s mind around, the goal of RFC 2571 is an admirable one--to introduce a common language for describing the functionality and actions taken by an SNMPv3 agent or managing entity. The architecture of an SNMPv3 entity is straightforward, and a tour through the architecture will serve to solidify our understanding of SNMP.

So-called **SNMP applications** consist of a command generator, notification receiver, and proxy forwarder (all of which are typically found in a managing entity); a command responder and notification originator (both of which are typically found in an agent); and the possibility of other applications. The command generator generates the GetRequest, GetNextRequest, GetBulkRequest, and SetRequest PDUs that we examined in Section 8.3.3 and handles the received responses to these PDUs. The command responder executes in an agent and receives, processes, and replies to (using the Response message) received GetRequest, GetNext Request, GetBulkRequest, and SetRequest PDUs. The notification originator application in an agent generates Trap PDUs; these PDUs are eventually received and processed in a notification receiver application at a managing entity. The proxy forwarder application forwards request, notification, and response PDUs.

A PDU sent by an SNMP application next passes through the SNMP "engine" before it is sent via the appropriate transport protocol. Figure 8.5 shows how a PDU generated by the command generator application first enters the dispatch module, where the SNMP version is determined. The PDU is then processed in the message-processing system, where the PDU is wrapped in a message header containing the SNMP version number, a message ID, and message size information. If encryption or authentication is needed, the appropriate header fields for this information are included as well; see RFC 2571 for details. Finally, the SNMP message (the application-generated PDU plus the message header information) is passed to the appropriate transport protocol. The preferred transport protocol for carrying SNMP messages is UDP (that is, SNMP messages are carried as the payload in a UDP datagram), and the preferred port number for the SNMP is port 161. Port 162 is used for trap messages.

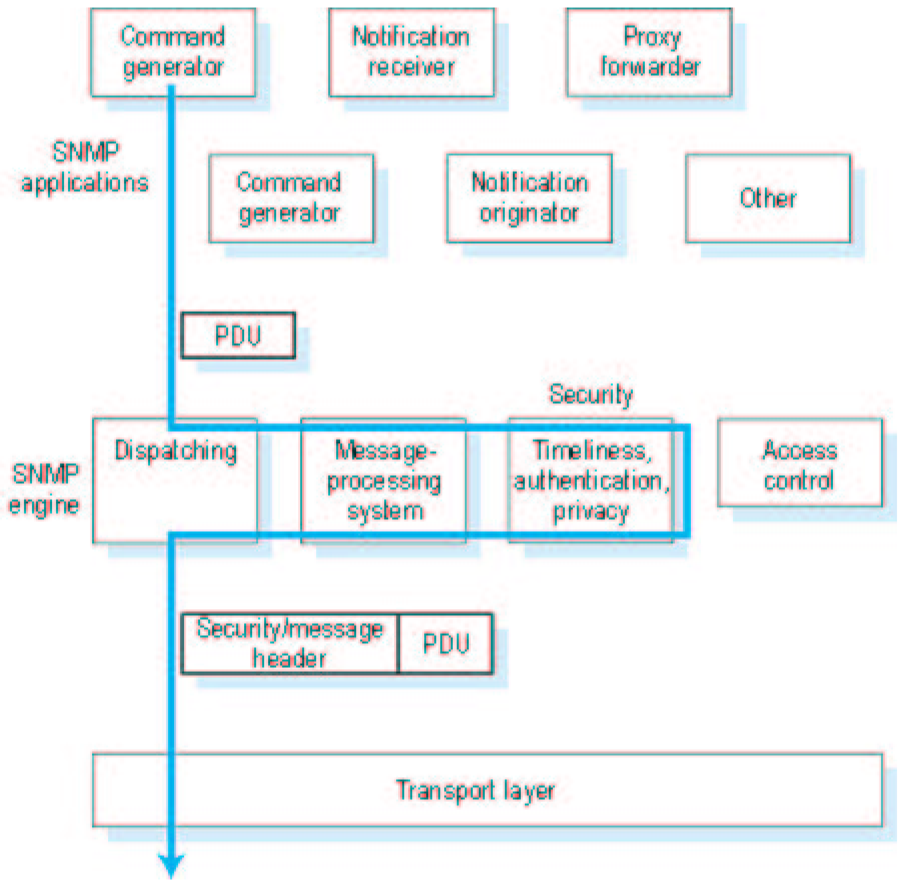

Figure 8.5: SNMPv3 engine and applications

We have seen above that SNMP messages are used to not just monitor, but also to control (for example, through the `SetRequest` command) network elements. Clearly, an intruder that could intercept SNMP messages and/or generate its own SNMP packets into the management infrastructure could wreak havoc in the network. Thus, it is crucial that SNMP messages be transmitted securely. Surprisingly, it is only in the most recent version of SNMP that security has received the attention that it deserves. SNMPv3 provides for encryption, authentication, protection against playback attacks (see Sections 7.2 and 7.3), and access control. SNMPv3 security is known as **user-based security** [RFC 2574] in that there is the traditional concept of a user, identified by a user name, with which security information such as a password, key value, or access privileges are associated.

- *Encryption.* SNMP PDUs can be encrypted using the Data Encryption Standard (DES) in cipher block chaining mode; see Section 7.2 for a discussion of DES. Note that since DES is a shared-key system, the secret key of the user encrypting data must be known by the receiving entity that must decrypt the data.

- *Authentication.* SNMP combines the use of a hash function, such as the MD5 algorithm that we studied in Section 7.4, with a secret key value to provide both authentication and protection against tampering. The approach, known as HMAC (Hashed Message

Authentication Codes) [RFC 2104] is conceptually simple. Suppose the sender has an SNMP PDU, *m*, that it wants to send to the receiver. This PDU may have already been encrypted. Suppose also that both the sender and receiver know a shared secret key, *K*, which need not be the same key used for encryption. The sender will send *m* to the receiver. However, rather than sending along a simple Message Integrity Code (MIC), *MIC*(*m*), that has been computed over *m* (see Section 7.4.2) to protect against tampering, the sender appends the shared secret key to *m* and computes a MIC, *MIC*(*m*,*K*) over the combined PDU and key. The value *MIC*(*m*,*K*) (but not the secret key!) is then transmitted along with *m*. When the receiver receives *m*, it appends the secret key *K* and computes *MIC*(*m*,*K*). If this computed value matches the transmitted value of *MIC*(*m*,*K*), then the receiver knows not only that the message has not been tampered with, but also that the message was sent by someone who knows the value of *K*, that is, by a trusted, and now authenticated, sender. In operation, HMAC actually performs the append-and-hash operation twice, using a slightly modified key value each time; see RFC 2104 for details.

- *Protection against playback.* Recall from our discussion in Chapter 7 that nonces can be used to guard against playback attacks. SNMPv3 adopts a related approach. In order to ensure that a received message is not a replay of some earlier message, the receiver requires that the sender include a value in each message that is based on a counter in the *receiver.* This counter, which functions as a nonce, reflects the amount of time since the last reboot of the receiver's network management software and the total number of reboots since the receiver's network-management software was last configured. As long as the counter in a received message is within some margin of error from the receiver's actual value, the message is accepted as a nonreplay message, at which point it may be authenticated and/or decrypted. See RFC 2574 for details.

- *Access control.* SNMPv3 provides a view-based access control [RFC 2575] that controls which network-management information can be queried and/or set by which users. An SNMP entity retains information about access rights and policies in a Local Configuration Datastore (LCD). Portions of the LCD are themselves accessible as managed objects, defined in the View-based Access Control Model Configuration MIB [RFC 2575], and thus can be managed and manipulated remotely via SNMP.

Principles in Practice

There are hundreds (if not thousands) of network management products available today, all embodying to some extent the network management framework and SNMP foundation that we have studied in this section. A survey of these products is well beyond the scope of this text and (no doubt) the reader’s attention span. Thus, we provide here pointers to a few of the more prominent products. A good starting point for an overview of the breadth of network management tools is Chapter 12 in [Subramanian 2000]

Network management tools divide broadly into those from network equipment vendors, that specialize in the management of the vendor’s equipment, and those aimed at managing networks with heterogeneous equipment. Among the vendor- specific offerings is CiscoWorks2000 [Cisco CiscoWorks 2000], for the management of LANs and WANs built on a Cisco device foundation. 3Com’s Transcend network management system [3Com Transcend 2000] is SNMP-compliant and provides "3Com SmartAgent intelligent agent" technology to aid in network management. Nortel’s Optivity Network Management System [Nortel 2000] provides for network management, service management and policy management (bandwidth management, QoS, application-level security, and IP/address).

Among the popular tools for managing heterogeneous networks are Hewlett-Packard’s Openview [Openview 2000], Aprisa’s Spectrum [Aprisa 2000], and Sun’s Solstice network management system[Sun 2000]. All three of these systems adopt a distributed system architecture in which multiple servers gather network management information from their managed domain. The network management station can then gather results from these servers, display information, and take control actions. All three products support the SNMP and CMIP protocols, and provide automated assistance for event/alarm correlation.

Online Book

8.4: ASN.1

In this book we have covered a number of interesting topics in computer networking. This section on ASN.1, however, may not make the top-10 list of interesting topics. Like vegetables, knowledge about ASN.1 and the broader issue of presentation services is something that is "good for you." ASN.1 is an ISO-originated standard that is used in a number of Internet-related protocols, particularly in the area of network management. For example, we saw in Section 3.2 that MIB variables in SNMP were inextricably tied to ASN.1. So while the material on ASN.1 in this section may be rather dry, the reader will hopefully take it on faith that the material *is* important.

In order to motivate our discussion here, consider the following thought experiment. Suppose one could reliably copy data from one computer’s memory directly into another remote computer’s memory. If one could do this, would the communication problem be "solved"? The answer to the question depends on one’s definition of "the communication problem." Certainly, a perfect memory-to-memory copy would exactly communicate the bits and bytes from one machine to another. But does such an exact copy of the bits and bytes mean that when software running on the receiving computer accesses this data, it will see the same values that were stored into the sending computer’s memory? The answer to this question is "not necessarily"! The crux of the problem is that different computer architectures, different operating systems, and compilers have different conventions for storing and representing data. If data is to be communicated and stored among multiple computers (as it is in every communication network), this problem of data representation must clearly be solved.

As an example of this problem, consider the simple C code fragment below. How might this structure be laid out in memory?

```
struct {
  char code;
  int x;
  } test;
  test.x = 259;
  test.code = 'a';
```

The left side of Figure 8.6 shows a possible layout of this data on one hypothetical architecture: there is a single byte of memory containing the character ’a’, followed by a 16-bit word containing the integer value 259, stored with the most significant byte first. The layout in memory on another computer is shown in the right half of Figure 8.6. The character ’a’ is followed by the integer value stored with the least significant byte stored first and with the 16-bit integer aligned to start on a 16-bit word boundary. Certainly, if one were to perform a verbatim copy between these two computers’ memories and use the same structure definition to access the stored values, one would see very different results on the two computers!

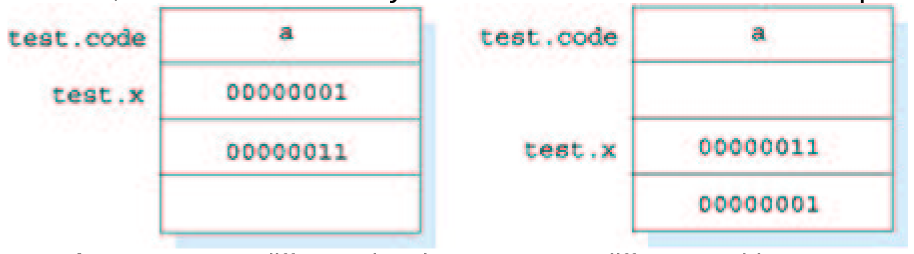

Figure 8.6: Two different data layouts on two different architectures

The fact that different architectures have a different internal data format is a real and pervasive problem. The particular problem of integer storage in different formats is so common that it has a name. "Big-endian" order for storing integers has the most significant bytes of the integer stored first (at the lowest storage address). "Little-endian" order stores the least significant bytes first. Sun SPARC and Motorola processors are big-endian, while Intel and DEC Alpha processors are little-endian. As an aside, the terms "big-endian" and "little-endian" come from the book, *Gulliver’s Travels* by Jonathan Smith, in which two groups of people dogmatically insist on doing a simple thing in two different ways (hopefully, the analogy to the computer architecture community is clear). One group in the land of Lilliput insists on breaking their eggs at the larger end ("the big-endians"), while the other insists on breaking them at the smaller end. The difference was the cause of great civil strife and rebellion.

Given that different computers store and represent data in different ways, how should networking protocols deal with this? For example, if an SNMP agent is about to send a Response message containing the integer count of the number of received UDP datagrams, how should it represent the integer value to be sent to the managing entity--in big-endian or little-endian order? One option would be for the agent to send the bytes of the integer in the same order in which they would be stored in the managing entity. Another option would be for the agent to send in its own storage order and have the receiving entity reorder the bytes, as needed. Either option would require the sender or receiver to learn the other's format for integer representation. A third option is to have a machine-, OS-, language-independent method for describing integers and other data types (that is, a data-description language) and rules that state the manner in which each of the data types are to be transmitted over the network. When data of a given type is received, it is received in a known format and can then be stored in whatever machine-specific format is required. Both the SMI that we studied in Section 8.3 and ASN.1 adopt this third option. In ISO parlance, these two standards describe a **presentation service**--the service of transmitting and translating information from one machine-specific format to another. Figure 8.7 illustrates a real-world presentation problem; neither receiver understands the essential idea being communicated--that the speaker likes something. As shown in Figure 8.8, a presentation service can solve this problem by translating the idea into a commonly understood (by the presentation service), person-independent language, sending that information to the receiver, and then translating into a language understood by the receiver.

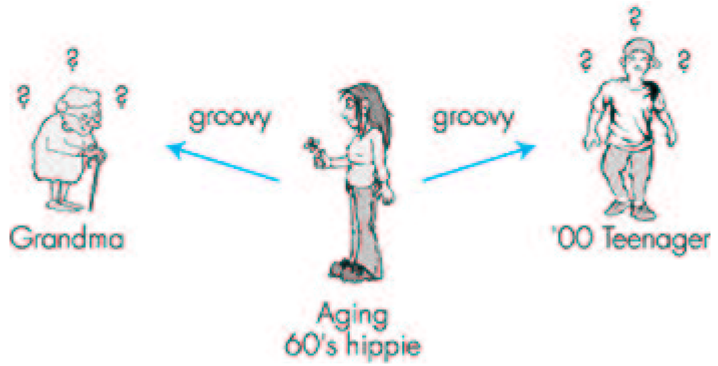

Figure 8.7: The presentation problem

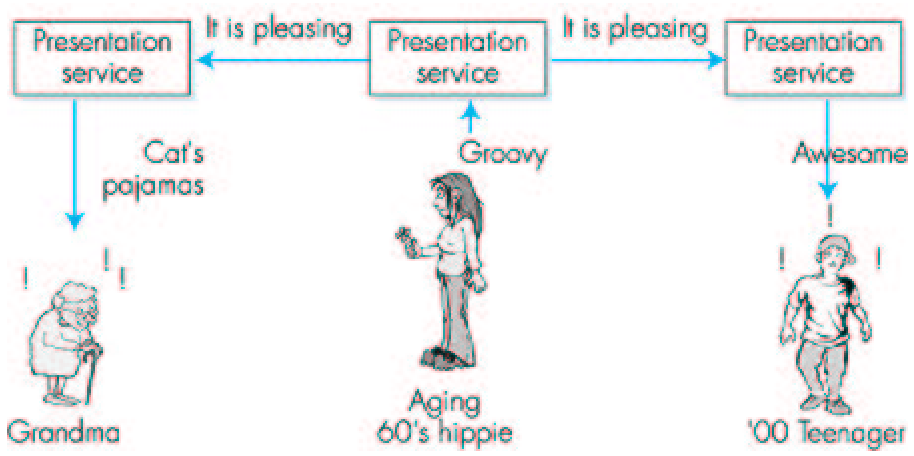

Figure 8.8: The presentation problem solved

Table 8.5 shows a few of the ASN.1-defined data types. Recall that we encountered the INTEGER, OCTET STRING, and OBJECT IDENTIFIER data types in our earlier study of the SMI. Since our goal here is (mercifully) not to provide a complete introduction to ASN.1, we refer the reader to the standards or to the printed and online book [Larmouth 1996] for a description of ASN.1 types and constructors such as SEQUENCE and SET that allow for the definition of structured data types.

Table 8.5: Selected ASN.1 data types

Tag	**Type**	**Description**
1	BOOLEAN	value is "true" or "false"
2	INTEGER	can be arbitrarily large
3	BITSTRING	list of one or more bits
4	OCTET STRING	list of one or more bytes
5	NULL	no value
6	OBJECT IDENTIFIER	name in the ASN.1 standard naming tree, see Section 8.2.2
9	REAL	floating point

In addition to providing a data description language, ASN.1 also provides **Basic Encoding Rules (BER)** that specify how instances of objects that have been defined using the ASN.1 data-description language are to be sent over the network. The BER adopts a so-called **TLV (Type, Length, Value) approach** to encoding data for transmission. For each data item to be sent, the data type, the length of the data item, and then the actual value of the data item are sent, in that order. With this simple convention, the received data is essentially self identifying.

Figure 8.9 shows how the two data items in a simple example would be sent. In this example, the sender wants to send the character string ’smith’ followed by the value 259 decimal (which equals 00000001 00000011 in binary, or a byte value of 1 followed by a byte value of 3) assuming big-endian order. The first byte in the transmitted stream has the value 4, indicating that the type of the following data item is an OCTET STRING; this is the ’T’ in the TLV encoding. The second byte in the stream contains the length of the OCTET STRING, in this case 5. The third byte in the transmitted stream begins the OCTET STRING of length 5; it contains the ASCII representation of the letter ’s’. The T, L, and V values of the next data item are 2 (the INTEGER type tag value), 2 (that is, an integer of length 2 bytes), and the two-byte big-endian representation of the value 259 decimal.

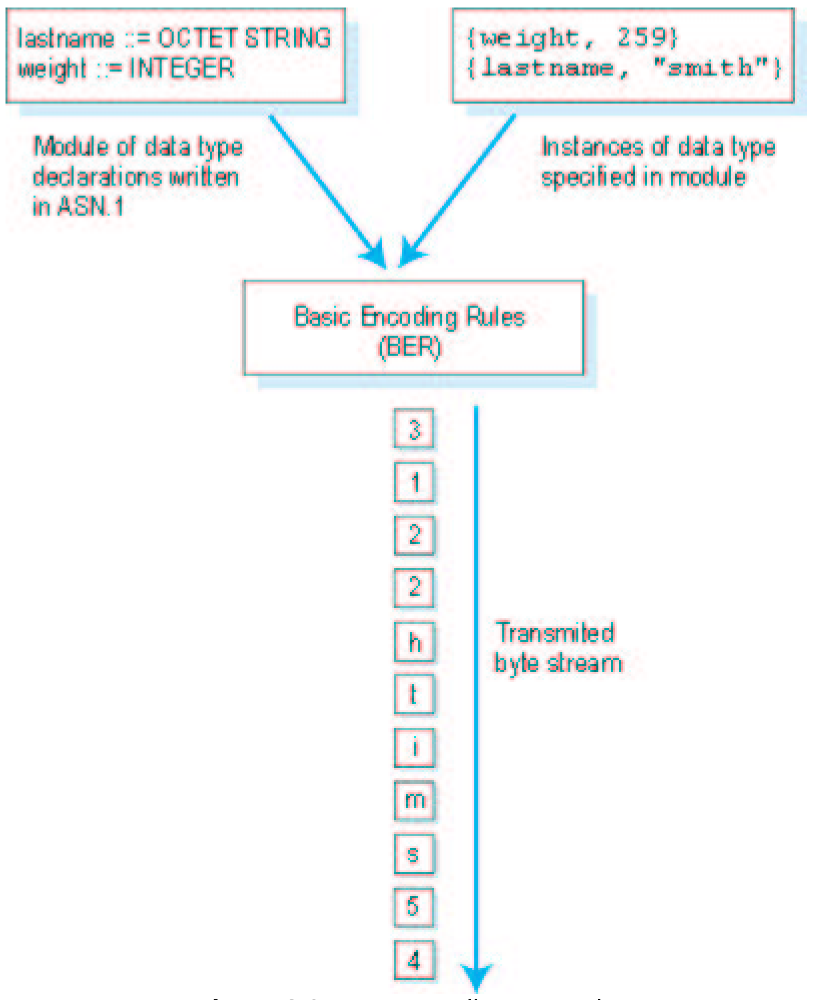
lastname ::= OCTET STRING
weight ::= INTEGER
{weight, 259}
{lastname, "smith"}
Module of data type declarations written in ASN.1
Instances of data type specified in module
Basic Encoding Rules (BER)
3
1
2
2
h
t
i
m
s
5
4
Transmited byte stream

Figure 8.9: BER encoding example

In our discussion above, we have only touched on a small and simple subset of ASN.1. Resources for learning more about ASN.1 include the ASN.1 standards document [ISO 1987, ISO X.680 1998], Philippe Hoschka's ASN.1 homepage [Hoschka 1997], and [Larmouth 1996].

Online Book

8.5: Firewalls

In motivating the need for security in Chapter 7, we noted that the Internet is not a very "safe" place--ne'er-do-wells are "out there" breaking into networks at an alarming rate. (For a summary of reported attacks, see the CERT Coordination Center [CERT 1999]. For a discussion of nearly 300 known attacks that firewalls, the topic we consider here, are designed to thwart, see [Newman 1998]. As a result, network administrators must be concerned not only with keeping the bits flowing smoothly through their network, but also with securing their network infrastructure from outside threats.

We've seen that SNMPv3 provides authentication, encryption, and access control in order to secure network management functions. While this is important (certainly, the network administrator does not want others to gain access to network-management functionality), it is only a small part of the network administrator's security concerns. In addition to monitoring and controlling the components of one's network, a network administrator also wants to exclude unwanted traffic (that is, intruders) from the managed network. This is where firewalls come in. A **firewall** is a combination of hardware and software that isolates an organization's internal network from the Internet at large, allowing some packets to pass and blocking others. Organizations employ firewalls for one or more of the following reasons:

- *To prevent intruders from interfering with the daily operation of the internal network.* An organization's competitor--or just some Internet prankster looking for a good time--can wreak havoc on an unsecured network. In the denial-of-service attack, an intruder monopolizes a critical network resource, bringing the internal network (at its network administrator) to its knees. An example of a denial-of-service attack is so-called **SYN flooding** [CERT 2000] in which the attacker sends forged TCP connection-establishment segments to a particular host. The host sets aside buffer space for each connection, and within minutes there is no TCP buffer space left for "honest" TCP connections.
- *To prevent intruders from deleting or modifying information stored within the internal network.* For example, an attacker can attempt to meddle with an organization's public presence on a Web server--a successful attack may be seen by thousands of people in a matter of minutes. Attackers may also be able to obtain customer purchase-card information from Web servers that provide Internet commerce (see Section 7.7).
- *To prevent intruders from obtaining secret information.* Most organizations have secret information that is stored on computers. This information includes trade secrets, product-development plans, marketing strategies, personal employee records, and financial analysis.

The simplest firewall consists of a packet filter. More sophisticated firewalls consist of combinations of packet filters and application gateways, topics we cover in the following two subsections.

8.5.1: Packet Filtering

An organization typically has a router that connects its internal network to its ISP (and hence to the larger public Internet). All traffic leaving and entering the internal network passes through this router. Most router manufacturers provide options for filtering; when these options are turned on, the router becomes a filter in addition to a router. As the name implies, a **filter** lets some datagrams pass through the router and filters out other datagrams. Filtering decisions are typically based on:

- The IP address the data is (supposedly) coming from.
- IP destination address.
- TCP or UDP source and destination port.
- ICMP message type.
- Connection initialization datagrams using the TCP SYN or ACK bits.

As a simple example, a filter can be set to block all UDP segments and all Telnet connections. Such a configuration prevents outsiders from logging onto internal hosts using Telnet, insiders from logging onto external hosts using Telnet, and "weird" UDP traffic from entering or leaving the internal network. The router filters the UDP traffic by blocking all datagrams whose IP protocol field is set to 17 (corresponding to UDP); it filters all Telnet connections by blocking all TCP segments (each encapsulated in a datagram) whose source or destination port number is 23 (corresponding to Telnet). Filtering of UDP traffic is a popular policy for corporations--much to the chagrin of leading audio and video streaming vendors, whose products stream over UDP in the default mode. Filtering Telnet connections is also popular, as it prevents outside intruders from logging onto internal machines.

A filtering policy can also be based on the combination of addresses and port numbers. For example, the router can forward all Telnet packets (port 23) except those going to and coming from a list of specific IP addresses. This policy permits Telnet connections to and from hosts on the list. It is highly recommended to reject all datagrams that have internal source IP addresses--that is, packets that claim to be coming from internal hosts but are actually coming in from the outside. These packets are often part of address spoofing attacks, whereby the attacker is pretending to be coming from an internal machine. Unfortunately, basing the policy on external addresses provides no protection from an external host claiming to be a different external host.

Filtering can also be based on whether or not the TCP ACK bit is set. This trick is quite useful if an organization wants to let its internal clients connect to external servers, but wants to prevent external clients from connecting to internal servers. Recall from Section 3.4 that the first segment in every TCP connection has the ACK bit set to 0 whereas all the other segments in the connection have the ACK bit set to 1. Thus, if an organization wants to prevent external clients from initiating connections to internal servers, it

simply filters all incoming segments with the ACK bit set to 0. This policy kills all TCP connections originating from the outside, but permits connections originating internally.
Now suppose an organization doesn’t want to block all connections originating from outside; instead it just wants to block only the Telnet connections originating from outside. This can be done by blocking inbound packets with destination port 23, or outbound packets with source port 23.

8.5.2: Application Gateways

Filters allow an organization to perform coarse-grain filtering on IP and TCP/UDP headers, including IP addresses, port numbers, and acknowledgment bits. We saw that filtering based on a combination of IP addresses and port numbers can allow internal clients to Telnet outside while preventing external clients from Telneting inside. But what if an organization wants to provide the Telnet service to a restricted set of internal users? Such a task is beyond the capabilities of a filter. Indeed, information about the identity of the internal users is not included in the IP/TCP/UDP headers, but is instead in the application-layer data.
In order to have a finer-level security, firewalls must combine pack et filters with application gateways. Application gateways look beyond the IP/TCP/UDP headers and actually make policy decisions based on application data. An **application gateway** is an application-specific server through which all application data (inbound and outbound) must pass. Multiple application gateways can run on the same host, but each gateway is a separate server with its own processes.
To get some insight into application gateways, let us design a firewall that allows only a restricted set of internal users to Telnet outside and prevents all external clients from Telneting inside. Such a policy can be accomplished by implementing a combination of a packet filter (in a router) and a Telnet application gateway, as shown in Figure 8.10. The router’s filter is configured to block all Telnet connections except those that originate from the IP address of the application gateway. Such a filter configuration forces all outbound Telnet connections to pass through G. When an internal user wants to Telnet to the outside world, it first sets up a Telnet session with the application gateway. An application running in the gateway, which listens for incoming Telnet sessions, prompts the user for its user id and password. When the user supplies this information, the application gateway checks to see if the user has permission to Telnet to the outside world. If not, the Telnet connection from the internal user to gateway is terminated by the gateway. If the user has permission, then the gateway (1) prompts the user for the hostname of the external host to which the user wants to connect, (2) sets up a Telnet session between the gateway and the external host, (3) relays to the external host all data arriving from the user, and relays to the user all data arriving from the external host. Thus the Telnet application gateway not only performs user authorization but also acts as a Telnet server and a Telnet client. Note that the filter will permit step (2) because the gateway initiates the Telnet connection to the outside world.

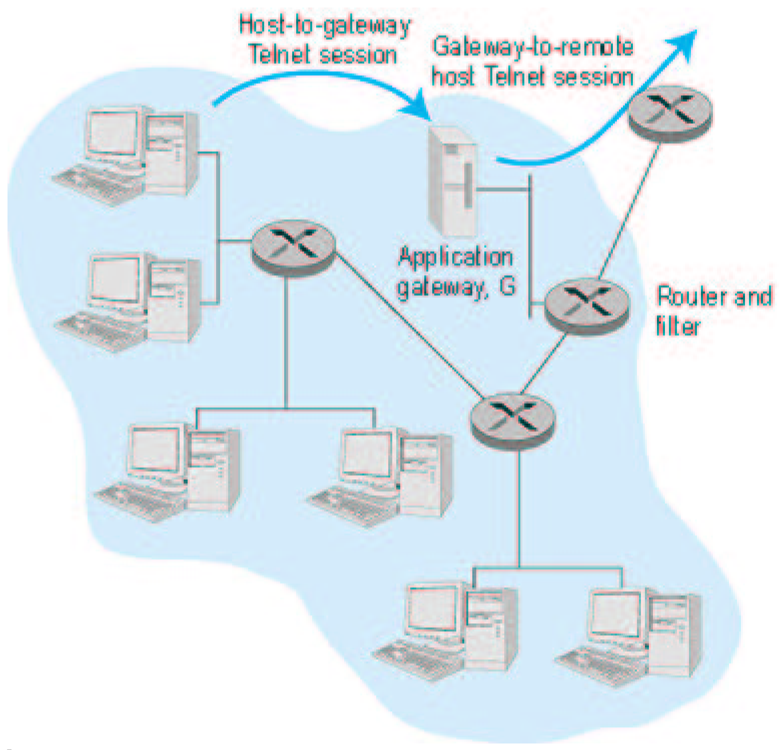
Host-to-gateway Telnet session
Gateway-to-remote host Telnet session
Application gateway, G
Router and filter

Figure 8.10: Firewall consisting of an application gateway and a filter

Internal networks often have multiple application gateways, for example, gateways for Telnet, HTTP, FTP, and e-mail. In fact, an organization's mail server (see Section 2.4) and Web cache (see Section 2.6) are application gateways.

Application gateways do not come without their disadvantages. First, a different application gateway is needed for each application. Second, either:

- the client software must know how to contact the gateway instead of the external server when the user makes a request, and must know how to tell the application gateway what external server to connect to,
- or the user must explicitly connect to the external server through the application gateway.

We conclude this section by mentioning that firewalls are by no means a panacea for all security problems. They introduce a tradeoff between the degree of communication with the outside world and level of security. Because filters can't stop spoofing of IP addresses and port numbers, filters often use an all or nothing policy (for example, banning all UDP traffic). Gateways can have software bugs, allowing attackers to penetrate them. Also, firewalls are even less effective if the internal users have wireless communication with the external world.

Case History

The Limitations of Firewalls

In February 2000, a number of major Internet commerce sites were brought to their knees by a *distributed denial-of-service attack*. As of this writing (March 2000), neither the perpetrators of the attack nor their motives are known. The attackers first hit Yahoo!, and then spread their offensive to other major sites, including Amazon.com, eBay, CNN.com, and Buy.com. In the case of Yahoo!, at the worst moment, less than 10 percent of Yahoo's customers could access a page.

A denial-of-service attack is an attack in which the aggressor swamps a host or a set of hosts with incoming packets--a kind of packet blitzkrieg. For a Web site, the aggressor most easily does this by sending massive numbers of HTTP requests, using destination port 80, to the Web site. The Web site then becomes bogged down in serving the bogus requests, causing the TCP connections carrying the bona fide requests to time out. Firewalls can provide limited protection from a denial-of-service attack--by identifying the source IP address of the perpetrator and filtering out all packets with that IP address.

But the perpetrators of the February 2000 attack used some simple (and well-known!) tricks to break the superficial defenses of a firewall. First, they planted programs called "zombies" in at least 50 innocent hosts, most of which were residing at universities and research institutions. At a given time, they then commanded the zombies to attack the Yahoo site--the zombies swamped Yahoo!, and then the other sites, with TCP connections. Whoever was behind the attacks didn't gain root access on any targeted machine, and no proprietary information was stolen. But the attackers did succeed at bringing many major sites to their knees.

What can be done to prevent such distributed denial-of-service attacks? There doesn't appear to a clear and easy answer to this question. One approach is to find the perpetrators and prosecute them--thereby discouraging other attackers. But it appears the attackers have left few electronic traces for determining their identities. Investigators are therefore taking a more traditional approach, using informants in the digital underground to try to gain information on who might be behind the attacks.

Online Book

8.6: Summary

Our study of network management, and indeed of all of networking, is now complete!

In this final chapter on network management, we began by motivating the need for providing appropriate tools for the network administrator--the person whose job it is to keep the network "up and running"--for monitoring, testing, polling, configuring, analyzing, evaluating, and controlling the operation of the network. Our analogies with the management of complex systems such as power plants, airplanes, and human organization helped motivate this need. We saw that the architecture of network-management systems revolve around five key components--(1) a network manager, (2) a set of managed remote (from the network manager) devices, (3) the management information bases (MIBs) at these devices, containing data about the device's status and operation, (4) remote agents that report MIB information and take action under the control of the network manager, and (5) a protocol for communicating between the network manager and the remote devices.

We then delved into the details of the Internet Network Management Framework, and the SNMP protocol in particular. We saw how SNMP instantiates the five key components of a network management architecture, and spent considerable time examining MIB objects, the SMI--the data-definition language for specifying MIBs, and the SNMP protocol itself. Noting that the SMI and ASN.1 are inextricably tied together, and that ASN.1 plays a key role in the presentation layer in the ISO/OSI seven-layer reference model, we then briefly examined ASN.1. Perhaps more important than the details of ASN.1 itself, was the noted need to provide for translation between machine-specific data formats in a network. While some network architectures explicitly acknowledge the importance of this service by having a presentation layer, this layer is absent in the Internet protocol stack. Finally, we concluded this chapter with a discussion of firewalls--a topic that falls within the realms of both security and network management. We saw how packet filtering and application-level gateways can be used to provide the network with some level of protection against unwanted intruders, perhaps allowing the network manager to sleep better at night, knowing the network is relatively safe from these intruders.

It is also worth noting that there are many topics in network management that we chose *not* to cover--topics such as fault identification and management, pro active anomaly detection, alarm correlation, and the larger issues of service management (for example, as opposed to network management). While important, these topics would form a text in their own right, and we refer the reader to the references noted in Section 8.1.