PART II

SNMP and Network Management

Part II, comprising Chapters 3–9, is devoted to understanding the principles of network and system management. Chapters 3–7 discuss the management of a TCP/IP network using Simple Network Management System (SNMP) versions 1, 2, and 3. Remote monitoring, which is also part of SNMP management, is discussed in Chapter 8.

In Chapter 3, technical foundations on standards, models, and language, which are needed to build network management based on various standards, are introduced. Network management standards that are currently used are SNMP (Internet), CMIP (OSI), TMN, IEEE, and Web-based Management. Of these, SNMP is the most widely deployed management system due to its truly simple architecture and implementation. An overview of models and concepts of network management is covered. Specifications of most protocols are done using ASN.1 syntax, which is discussed in some detail.

There are three versions of SNMP-based protocols that manage TCP/IP networks. SNMPv1 is covered in Chapters 4 and 5. Chapter 4 is devoted to the organization and information models of SNMP in network management. System architecture and SNMP messages are presented. The structure of management information (SMI) is presented using ASN.1 syntax. The definition of SNMP objects and their organization in the structure of management information base (MIB) are described. Chapter 5 covers SNMP communication protocol. Message data structures are presented along with the message protocol operations and SNMP MIB.

Learning Chapters 4 and 5 would help the reader to understand the basic principles behind SNMP network management. Case histories and practical examples punctuate the presentation. When the book is used as a textbook for a course, this should provide adequate background for the student to select a project for the course if the course includes a project as part of its requirements. I strongly recommend it—especially for an undergraduate course. A list of projects, which have been used in senior undergraduate and graduate classes, is presented in Appendix B.

Chapter 6 addresses SNMPv2. SNMPv2 adds several significant enhancements to SNMPv1, including efficient transferring of bulk data between systems. One of the intended major enhancements, namely security considerations, was postponed from SNMPv2 to SNMPv3. In Chapter 7, we cover security and privacy, as well as generalized SNMP architecture and applications, which are part of SNMPv3 specifications.

The SNMP management system is based on polling. Remote monitoring of network components using probes and sending only relevant data to the network management system is the goal of RMON discussed in Chapter 8.

Chapter 9 is devoted to networking tools, management tools, and management systems. You will learn some basic networking tools that are part of the operating system, which would be of immense help in day-to-day use and operations of the network. SNMP command tools are convenient tools that can be used for network management even without having a network management system. We have earlier learned the immense benefits of MIBs in network management. Hence, MIBs have to be designed well, which is covered in MIB engineering. The design of the network management system for various vertical network applications, as well as for performing various functions, is covered in detail. After learning these tools, we will cover network management systems ranging from low-end to high-end solutions. We will dwell in more detail on the mid-range enterprise network management systems with examples of commercial systems that are in widespread use.

Basic Foundations: Standards, Models, and Language

OBJECTIVES

- *Standards, models, and language needed for network management*
- *Network models*
  - *OSI*
  - *Internet*
  - *TMN*
  - *IEEE 802*
  - *Web based*
- *Management communication protocols*
  - *SNMP*
  - *CMIP*
  - *XML*
  - *CORBA*
- *ASN.1 language*
  - *Syntax*
  - *Macro*
- *Basic encoding rule*
- *Management application functions*

In Part I we had an overview of networking and management of network and systems. We learned about network technology and components that need to be managed. There are several management standards and models in existence for managing networks, systems, and services. We can understand and appreciate them better by first looking at the commonality among them, and then the differences that distinguish them. These goals form the objectives of this chapter.

We will consider the foundations that are needed to build various network management models and protocols. We will survey the network management standards and present the general architecture of the network management models in Section 3.1.

The International Standards Organization (ISO) has defined a generalized model that addresses all aspects of network management. We will cover the three models of the architecture in Sections 3.3 through 3.5, which deal with organization, information, and communication. Then we will learn the basics of the formal language, ASN.1, and the data structure that is used for management systems to store management information and communicate with each other in Sections 3.6 through 3.8.

All the above three models are designed for management applications to manage networks, systems, and services. The fourth model, functional model, addresses these in Section 3.9. The applications fall into the categories of fault, configuration, performance, security, and accounting.

In a global perspective, three areas of network need managing. They are network, systems, and services; inter-layer protocols; and intra-layer protocols. In this book our focus will be on network and system management aspects. We define network management as management of the network comprising nodes and links, and system management as managing system resources, such as central processor usage, disk usage, and application processes. Service management deals with services provided by organizations to customers. Service management is an extension of network and systems.

The two leading models of network management are the Internet model and the Open System Interconnection (OSI) model. The Internet model is the most widely used for network management. It is a simple scalar model and hence easy to implement. The OSI model, which is object oriented, is more complex and harder to implement. However, with the matured state of object-oriented technology and the convergence of data and telecommunications technologies, object-oriented implementation of network management has come into vogue. We will address this in Chapter 16. A higher-level management network called the Telecommunications Management Network (TMN) is also based on the OSI model. It addresses all levels of management including service and business aspects. We will study TMN in Chapter 10.

In this book we will be concerned primarily with the study of Internet-based SNMP model. The OSI model is discussed in Appendix A.

3.1 NETWORK MANAGEMENT STANDARDS

There are several network management standards that are in use today.

Table 3.1 lists four standards, along with a fifth class based on emerging technologies, and their salient points. The first four are the OSI model, the Internet model, TMN, and IEEE LAN/MAN. A detailed treatment of the various standards can be found in [Black, 1995]. The first category in Table 3.1, Open System Interconnection (OSI) management standard, is the standard adopted by the International Standards Organization (ISO). The OSI management protocol standard is Common Management Information Protocol (CMIP). The OSI management protocol has built-in services, Common Management Information Service (CMIS), which specify the basic services needed to perform the various functions. It is the most comprehensive set of specifications and addresses all seven layers. OSI specifications are structured and deal with all seven layers of the OSI Reference Model. The specifications are object oriented and hence managed objects are based on object classes and inheritance rules. Besides specifying the management protocols, CMIP/CMIS also address network management applications. Some of the

Table 3.1 Network Management Standards

STANDARD	SALIENT POINTS
OSI/CMIP	• International standard (ISO/OSI) • Management of data communications network—LAN and WAN • Deals with all seven layers • Most complete • Object oriented • Well structured and layered • Consumes large resource in implementation
SNMP/Internet	• Industry standard (IETF) • Originally intended for management of Internet components, currently adopted for WAN and telecommunication systems • Easy to implement • Most widely implemented
TMN	• International standard (ITU-T) • Management of telecommunications network • Based on OSI network management framework • Addresses both network and administrative aspects of management • eTOM industry standard for business processes for implementing TMN using NGOSS framework
IEEE	• IEEE standards adopted internationally • Addresses LAN and MAN management • Adopts OSI standards significantly • Deals with first two layers of OSI RM
Emerging Technologies	• Web-Based Enterprise Management (WBEM) • Java Management Extension (JMX) • XML-based Network Management • CORBA-based Network Management

major drawbacks of the OSI management standard were that it was complex and that the CMIP stack was large. Although these are no longer impediments to the implementation of the CMIP/CMIS network management, SNMP is the protocol that is extensively deployed.

In contrast to the CMIP protocol, the Simple Network Management Protocol (SNMP) presented in Table 3.1 is truly simple as its name indicates. It started as an industry standard and has since become very much like standard specifications of a standards organization. The Internet Engineering Task Force (IETF) is responsible for all Internet specifications including network management. The managed objects are defined as scalar objects in SNMP. It was primarily intended to manage Internet components, but is now used to manage WAN and telecommunications systems. It is easy to implement and is the most widely implemented network management system today. We will discuss SNMP management in more detail in this book.

The third category in Table 3.1, TMN, is designed to manage the telecommunications network and is oriented toward the needs of telecommunications service providers. TMN is ITU-T (International Telecommunications Union—Telecommunications) standard and is based on OSI CMIP/CMIS specifications. TMN extends the concept of management beyond managing networks and network components. Its specifications address service and business considerations (M3000). Chapter 10 is devoted to the discussion of TMN.

Enhanced Telecommunications Operations Map (eTOM) is a guidebook for business processes in the telecommunications industry. It is an extension of TMN. It is being developed by TeleManagement Forum (TM Forum) as component of NGOSS (New Generation OSS) [Reillly and Creaner, 2005]. The main difference between the TMN and eTOM approaches is that the former has been developed starting from networks and network equipment (bottom up), while eTOM is a top–down approach. The eTOM framework has been incorporated within the TMN framework as a set of standards (M.3050.x).

The IEEE standards for Local Area Network (LAN) and Metropolitan Area Network (MAN) specifications shown in Table 3.1 are only concerned with OSI layers 1 (physical) and 2 (data link). Those specifications are structured similar to OSI specifications. Both OSI/CMIP and Internet/SNMP protocols use IEEE standards for the lower layers. The IEEE 802.x series of specifications define the standards for the various physical media and data link protocols. IEEE 802.1 specifications present overview, architecture, and management. The IEEE 802.2 standard specifies the Logical Link Control (LLC) layer. As we saw in Chapter 1 (Figure 1.14), the LLC layer provides transparency of the various physical media and protocols to the network layer. The others in series are for specific media and protocols. For example, 802.3 specifications are for Ethernet LANs.

The last category in Table 3.1 addresses several emerging management technologies. One of them is based on using Web technology, Web server for the management system and Web browsers for network management stations. In Web-based management, the organization model uses Web server–Web browser architecture. Much of the object-oriented technology, such as hypermedia server, CORBA-oriented transportation, and client–server push-technology influence the Web-based management.

The Web-Based Enterprise Management (WBEM) standard is developed by the Desktop Management Task Force (DMTF). It is based on the Common Information Model (CIM) data model transported using CIM Operations over HTTP.

The Java Management Extension [JMX] is an open Java technology for management. It defines management architecture, application programming interfaces (APIs), and management services under a single umbrella specification. It was developed under Sun Microsystems's JMAPI (Java Management API) initiative.

XML is a meta-markup language standardized by the Worldwide Web Consortium (W3C) for document exchange in the Web. XML-based network management is based on a network management method, which defines management information by XML and the exchange of data for management in the form of an XML document, and it uses an XML document-processing standard method for processing data.

Common Object Request Broker Architecture (CORBA)-based Network Management is an object-oriented client–server model that uses CORBA. The objects are defined using Interface Description Language (IDL) and uses a distributed managed objects architecture.

With the Web-based management system, not only can object-oriented technology be implemented but also the dedicated workstation constraint is removed by the use of a Web browser. However, which object-oriented technology should an IT manager choose? There is no clear-cut answer to this question, and different vendors have implemented NMSs using different technologies for different applications.

3.2 NETWORK MANAGEMENT MODELS

The OSI network model is an ISO standard and is most complete of all the models. It is structured and it addresses all aspects of management. Figure 3.1 shows an OSI network management architectural model that comprises four models. They are the organization model, the information model, the communication model, and the functional model. Although, the above classification is based on the OSI architectural model, and only parts of it are applicable to other models, it helps us understand the holistic picture of different aspects of network management.

The organization model describes the components of a network management system, their functions, and their infrastructure. The organization model is defined in ISO 10040 OSI Systems Management Overview. It defines the terms object, agent, and manager.

The OSI information model deals with the structure and the organization of management information. ISO 10165 specifies the Structure of Management Information (SMI) and the information database, management information base (MIB). SMI describes how the management information is structured and MIB deals with the relationship and storage of management information.

The third model in OSI management is the communication model. There are three components to this—management application processes that function in the application layer, layer management between layers, and layer operation, which is within the layer. We will focus on the application processes in this book.

The functional model is the fourth component of OSI management, which deals with the user-oriented requirements of network management. As mentioned in Chapter 1, there are five functional application areas defined in OSI, namely configuration, fault, performance, security, and accounting. These are defined as system management functions in OSI.

As mentioned earlier, only OSI presents the most complete model for network management, while the others either deal with only a subset or are still in the process of development of standards. Although a discussion of OSI management is not part of this book, it is briefly covered in Appendix A for completeness of the subject. OSI deals with all seven networking layers. Besides, as we shall see in Chapter 10, it lends itself to addressing service and business management that are more than just networking. The second standard listed in Table 3.1 is SNMP/Internet standard. IETF does not define architecture for the SNMP management model explicitly. However, it does exist implicitly. The organization, information, and communication models are similar to OSI models. The SNMP network management model addresses the functional model in terms of operations, administration, and security. SNMP-based management is widely used for campus-wide networks, although enterprise-wide networks are also managed by using distributed configurations of SNMP-based network management systems (NMSs). SNMP-based management systems, tools, and applications are addressed in Chapter 9. The third standard listed in Table 3.1 is the TMN, which is based on the OSI model. Thus, the four models apply to TMN. The focus of the TMN standard is towards managing telecommunications networks. As mentioned earlier,

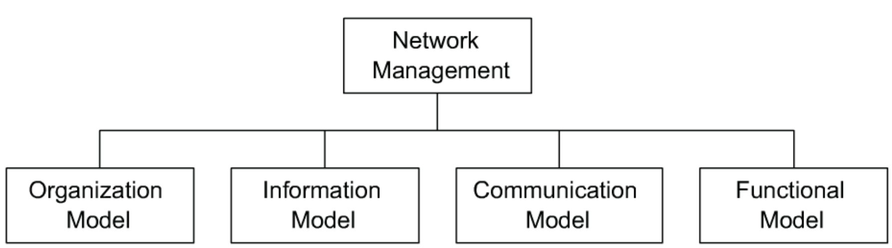

Figure 3.1 OSI Network Management Model

it extends the application functions of OSI further into business and service considerations. Operations systems support service and business management. The fourth standard in Table 3.1 is the IEEE Standard on management and is dedicated to the management of layers 1 and 2 of the OSI Reference Model. It is applicable to LAN and MAN. LAN refers to local area network and MAN (Metropolitan Area Network) refers to metropolitan (intra-city) network. It also addresses standards on broadband network management, which is of great relevance to the current technology. Broadband management is covered in Part IV. Since it deals only with physical and data link layers, it is primarily concerned with the communication model.

In Web-based and object-oriented management, the organization model uses Web server–Web browser architecture. Much of the object-oriented technology, such as hypermedia server, CORBA-oriented transportation, and client–server push-technology are influencing Web-based management. Applications developed under Web-based management could still fall under the OSI functional model. We will now look at each of the models.

3.3 ORGANIZATION MODEL

The organization model describes the components of network management and their relationships. Figure 3.2 shows a representation of a two-tier model. Network objects consist of network elements such as hosts, hubs, bridges, routers, etc. They can be classified into managed and unmanaged objects or elements. The managed elements have a management process running in them called an agent. The unmanaged elements do not have a management process running in them. For example, one can buy a managed or unmanaged hub. Obviously the managed hub has management capability built into it and hence is more expensive than the unmanaged hub, which does not have an agent running in it. The manager communicates with the agent in the managed element.

The manager manages the managed element. As shown in Figure 3.2, there is a database in the manager, but not in the agent. The manager queries and receives management data from the agent, processes them, and stores them in its database. The agent can also send a minimal set of alarm information to the manager unsolicited.

Figure 3.3 presents a three-tier configuration. The intermediate layer acts as both agent and manager. As manager, it collects data from the network elements, processes them, and stores the results in its database. As agent, it transmits information to the top-level manager. For example, an intermediate

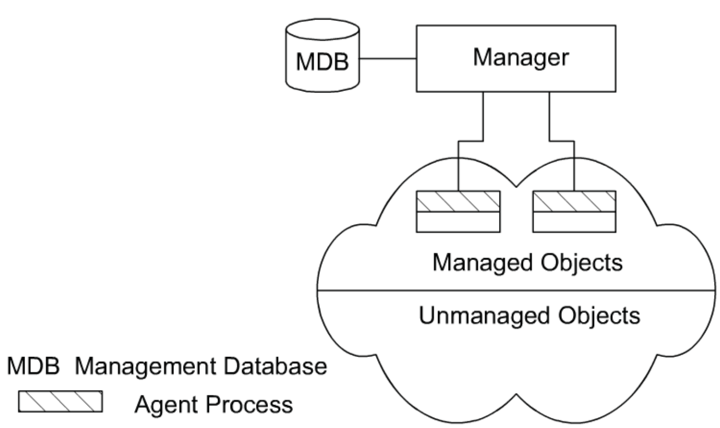

Figure 3.2 Two-Tier Network Management Organization Model

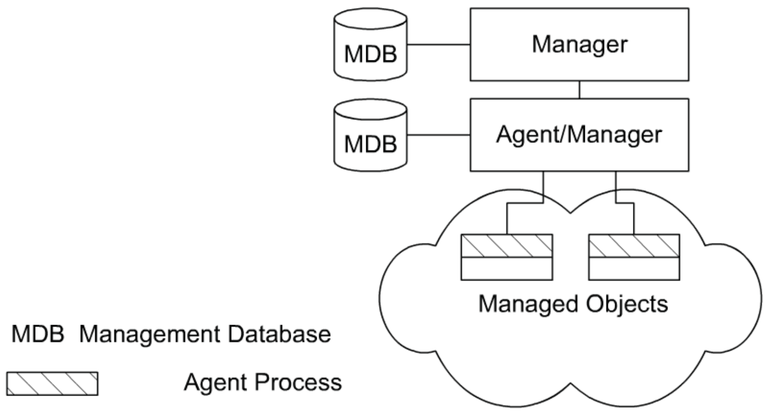

Figure 3.3 Three-Tier Network Management Organization Model

system is used for making statistical measurements on a network and passes the information as needed to the top-level manager. Alternatively, an intermediate NMS could be at a local site of a network and the information is passed on to a remote site.

Network domains can be managed locally; and a global view of the networks can be monitored by a manager of managers (MoM), as shown in Figure 3.4. This configuration uses an enterprise NMS and is applicable to organizations with sites distributed across cities. It is also applicable to a configuration where vendor management systems manage the domains of their respective components, and MoM manages the entire network.

Network management systems can also be configured on a peer-to-peer relationship as shown in Figure 3.5. This is the dumbbell architecture shown in Figure 1.24. We can recognize the similarity between this and the client–server architecture where a host serves as both a client and a server. An example of such a situation would be two network service providers needing to exchange management information

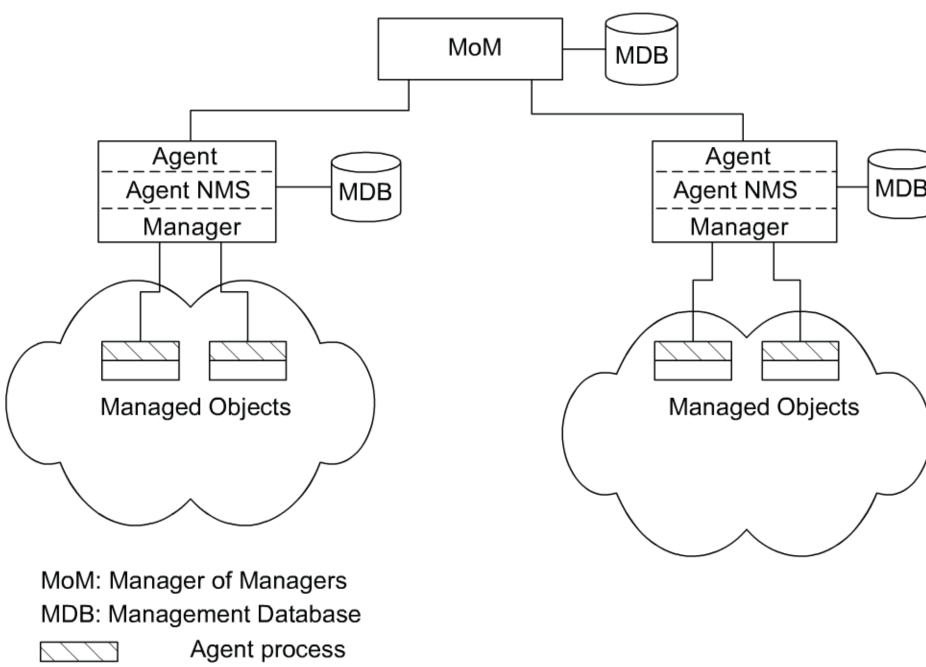

Figure 3.4 Network Management Organization Model with MoM

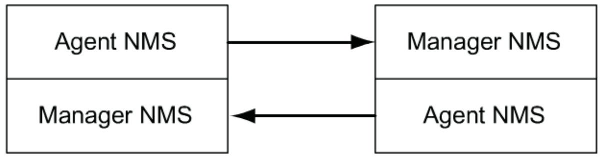

Figure 3.5 Dual Role of Management Process

between them. From the user's point of view, the information traverses both networks and needs to be monitored end-to-end.

In the above discussion, we have used the term network management system (NMS) to mean a system that runs a management process, not just a managed object. Thus, the agent and the manager devices are defined as agent NMS and manager NMS, as shown in Figure 3.4 and Figure 3.5.

3.4 INFORMATION MODEL

An information model is concerned with the structure and storage of information. Let us consider, for example, how information is structured and stored in a library and is accessed by all. A book is uniquely identified by an International Standard Book Number (ISBN). It is a ten-digit number identification that refers to a specific edition of a specific book. For example, ISBN 0-13-437708-7 refers to the book "Understanding SNMP MIBs" by David Perkins and Evan McGinnis. We can refer to a specific figure in the book by identifying a chapter number and a figure number; e.g., Fig. 3.1 refers to Figure 1 in Chapter 3. Thus, a hierarchy of designation {ISBN, Chapter, Figure} uniquely identifies the object, which is a figure in the book. "ISBN," "Chapter," and "Figure" define the syntax of the three pieces of information associated with the figure; and the definition of their meaning in a dictionary would be the semantics associated with them.

The representation of objects and information that are relevant to their management forms the management information model. As discussed in Section 3.3, information on network components is passed between the agent and management processes. The information model specifies the information base to describe managed objects and the relationship between managed objects. The structure defining the syntax and semantics of management information is specified by Structure of Management Information (SMI). The information base is called the Management Information Base (MIB). The MIB is used by both agent and management processes to store and exchange management information. The MIB associated with an agent is called an agent MIB and the MIB associated with a manager is designated as the manager MIB. The manager MIB consists of information on all the network components that it manages; whereas the MIB associated with an agent process needs to know only its local information, its MIB view. For example, a county may have many libraries. Each library has an index of all the books in that location—its MIB view. However, the central index at the county's main library, which manages all other libraries, has the index of all books in all the county's libraries—global manager MIB view.

Figure 3.6 expands the network configuration that is shown in Figure 3.2 to include the MIB that is associated with the manager. Thus, the manager has both the management database (MDB) and the MIB. It is important to distinguish between the two. The MDB is a real database and contains the measured or administratively configured value of the elements of the network. On the other hand, the MIB is a virtual database and contains the information necessary for processes to exchange information among themselves.

Let us illustrate the distinction between MIB and MDB by considering the scenario of adding a component to the network. Assume that all the hubs in the network are made by a single vendor,

3.4.1 Management Information Tree

The managed objects are uniquely defined by a tree structure specified by the OSI model and are used in the Internet model. Figure 3.7 shows the generic representation of the tree, defined as the Management Information Tree (MIT). There is a root node and well-defined nodes underneath each node at different levels, designated as Level 1, Level 2, etc. Each managed object occupies a node in the tree. In the OSI model, the managed objects are defined by a containment tree representing the MIT.

Figure 3.8 shows the internationally adopted OSI MIT. The root node does not have an explicit designation. The root has three nodes in the layer beneath it—iso, ccitt (itu), and iso–ccitt, (iso–itu). The iso defines the International Standards Organization and ccitt, or itu, defines the International Telecommunications Union (the old name is ccitt). The two standards organizations are on the first layer defining managing objects under them. The joint iso–itu node is for management objects jointly defined by the two organizations. The number in each circle identifies the designation of the object in each layer. Thus, iso is designated as 1, org as 1.3, dod, Department of Defense, as 1.3.6, and the internet as 1.3.6.1. All Internet-managed objects will be that number followed by more dots and numbers. It is to be noted that the names of the nodes are all in lowercase letters as a convention, which we will formally define in Section 3.6.

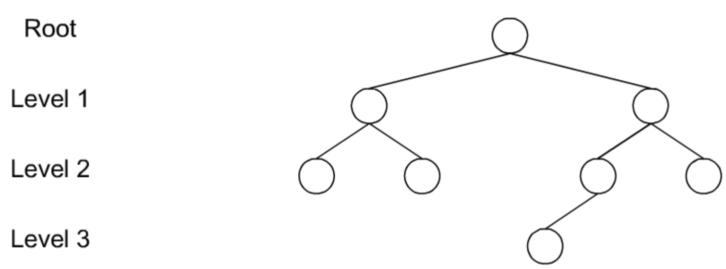

Figure 3.7 Generic Representation of the Management Information Tree

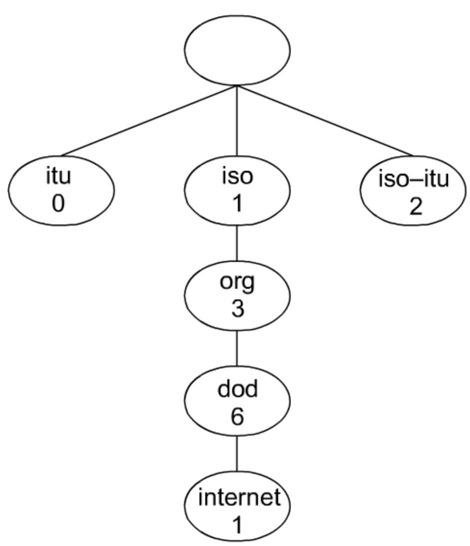

Figure 3.8 OSI Management Information Tree

3.4.2 Managed Object Perspective

Although a managed object need not be a physical object that can be seen, touched, or felt, it is convenient to use a physical representation to understand the characteristics and operations associated with a managed object. Let us consider an object, which is circular in shape. We can define the object in English language syntax as *circle*. To associate a meaning with the object name, *circle*, we can use Webster's dictionary *definition* "a plane figure bounded by a single curved line every point of which is equally distant from the point at the center of the figure." In other words, the definition is a textual description of the object. The object can be viewed and its parameters changed by people who have *access* to it. The *access* privilege could be limited to just accessing it or performing some action on it; for example, resetting a counter value to two. These are all defined as *access* attributes. If we envision a scenario in which the object is used by a nursery school to explain shapes to children, it should at least have some basic shapes, such as a circle, a square, etc. We can define the basic objects that are required of a group (of objects) as the status of the object—whether it is mandatory or optional to have (implement) that object. This attribute of the object is defined as the *status* of the object. There could be many types of objects in the nursery school that are circular in shape. There is a unique identification and name (*object identifier* and *descriptor*) associated with each of them, such as a ring, a donut, etc. There could be many instances of ring and donut; but we are only addressing the types of object, not instances of them here. We have thus defined the five basic attributes of a managed object type from the Internet perspective. They are *name, definition, syntax, access, and status.*

A pictorial view of a circular object in the Internet is shown in Figure 3.9(a).

A managed object in the Internet is defined by five parameters [RFC 1155]. They are:

- *object identifier* and *descriptor* — unique ID and name for the object
- *syntax* — used to model the object
- *access* — access privilege to a managed object
- *status* — implementation requirements
- *definition* — textual description of the semantics of object type

A modification of this is specified in RFC 1212, as we shall see in the next chapter.

The Internet object model is a scalar model and is easy to understand, as seen above. In contrast, the OSI perspective of a managed object is complex and has a different set of characteristics. We will extend the above analogy of the circular object in a nursery to illustrate an OSI perspective.

Figure 3.9(b) presents the conceptual OSI representation of the various characteristics of a managed object. As mentioned earlier, OSI specifications are object oriented, and hence a managed object belongs to an object class. The left side of Figure 3.9(b) presents the same circular object in the OSI model. The definition of an object in an object-oriented perception would include both the shape and values. Thus, the *attribute* of the object is a circle with given dimensions. The *attribute* of an object defines the external perspective of the object. It undergoes an *operation* "push." Push is not really an OSI operational entity, but is used here to illustrate the concept. The *behavior* of the object is to change its shape or attribute from a circle to an ellipse. It then sends *notifications* to the relevant community informing of its change. Thus, the characteristics of an OSI managed object are:

- *object class* — managed object
- *attributes* — attributes visible at its boundary
- *operations* — operations that may be applied to it

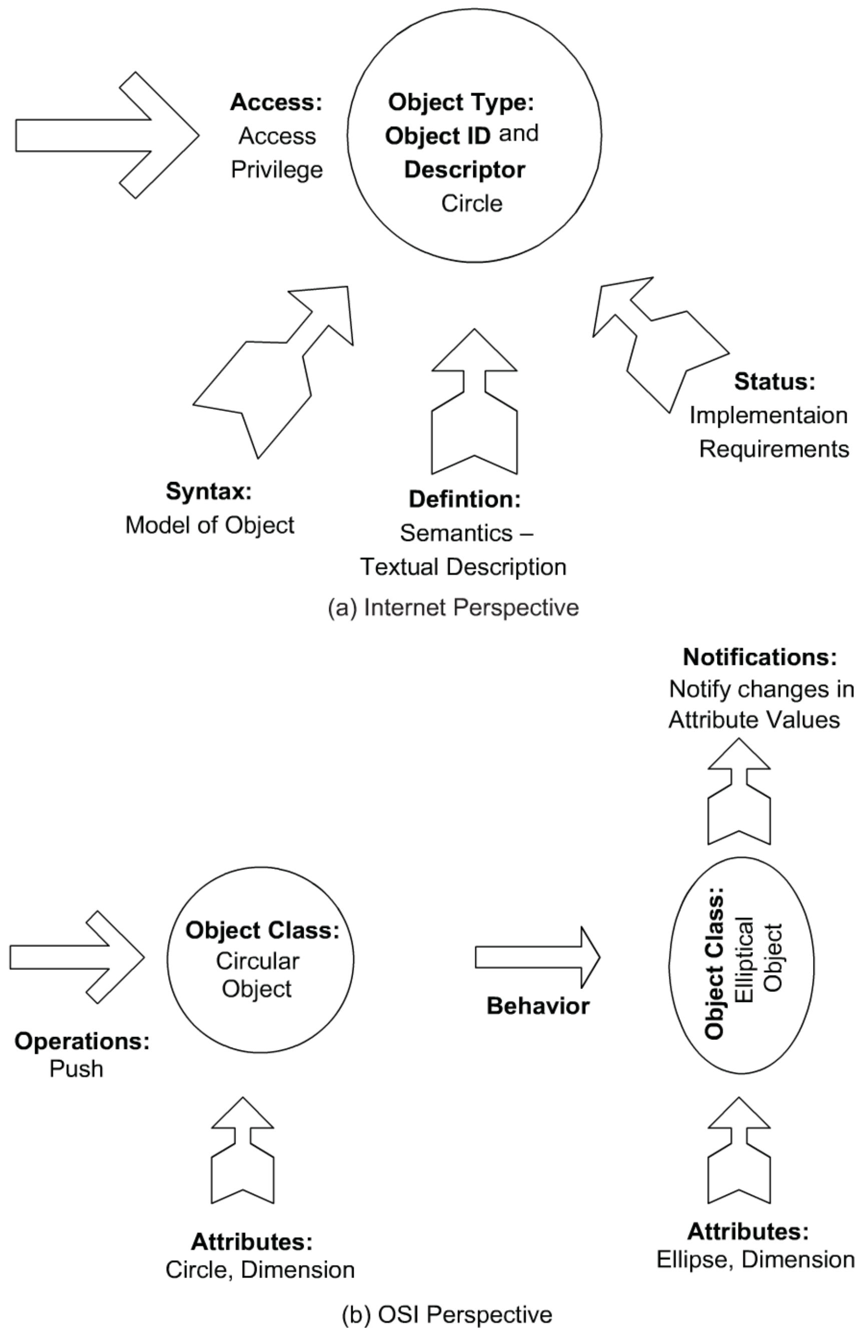

Figure 3.9 Conceptual Views of Managed Object

- *behavior* behavior exhibited by it in response to an operation
- *notifications* notifications emitted by the object

It is hard to compare the characteristics of a managed object in the Internet and OSI models on a one-to-one basis, as they are very much different. However, it can be observed in the conceptual models in Figure 3.9 that the OSI characteristics—*operations*, *behavior*, and *notification*—are a part of the Internet communications model. Operation in the Internet is done by get and set commands. Notification is done by response and alarm messages. The *syntax* characteristic of the Internet is part of OSI *attribute*s.

Characteristics	**Example**
Object type	PktCounter
Syntax	Counter
Access	Read-only
Status	Mandatory
Description	Counts number of packets

(a) Internet Perspective

Characteristics	**Example**
Object class	Packet Counter
Attributes	Single-valued
Operations	get, set
Behavior	Retrieves or resets values
Notifications	Generates notifications on new value

(b) OSI Perspective

Figure 3.10 Packet Counter as an Example of a Managed Object

The *access* characteristic of the Internet is a part of the security function in the OSI functional model. The *status* characteristic of the Internet is handled by conformance as a part of application services in OSI. Further, in OSI we can create and delete objects, while these concepts do not exist in the Internet. Objects in early SNMP management are assumed to exist for management purposes.

Figure 3.10 shows the comparison between Internet and OSI specifications for the object, packet counter. An example of a packet counter as a managed object in the Internet model is given in Figure 3.10(a). The *object type* (we will define *object id* later) is PktCounter. The *syntax* is Counter. The *access* mode is read-only. The *status* implementation is mandatory, which mandates that this object must be implemented if the group it belongs to is implemented. The *description* provides the semantics that the packet counter counts the number of packets.

The example of the same counter as a managed object in the OSI model is given in Figure 3.10(b). The counter is defined as an *object class, Packet Counter*. It could be related to either a sub- or superclass. The *attribute* value is single-valued. We can perform get and set *operations* on its *attribute*. Its *behavior* to a set operation would be to reset the counter, or just retrieve data if the operation is get. The new value is sent out as *notification.*

3.5 COMMUNICATION MODEL

We have discussed in the previous section how information content is defined (SMI) and stored (MIB). We will now address the model associated with how the information is exchanged between systems. Management data are communicated between agent and manager processes, as well as between manager processes. Three aspects need to be addressed in the communication of information between two entities: transport medium of message exchange (transport protocol), message format of communication (application protocol), and the actual message (commands and responses). Let us illustrate this by an example of Azita buying a car from an automobile salesperson, Roberto.

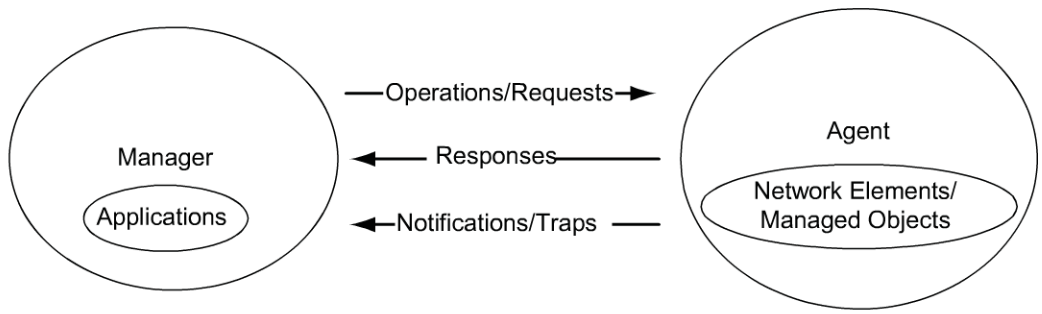

Figure 3.11 Management Communication Model

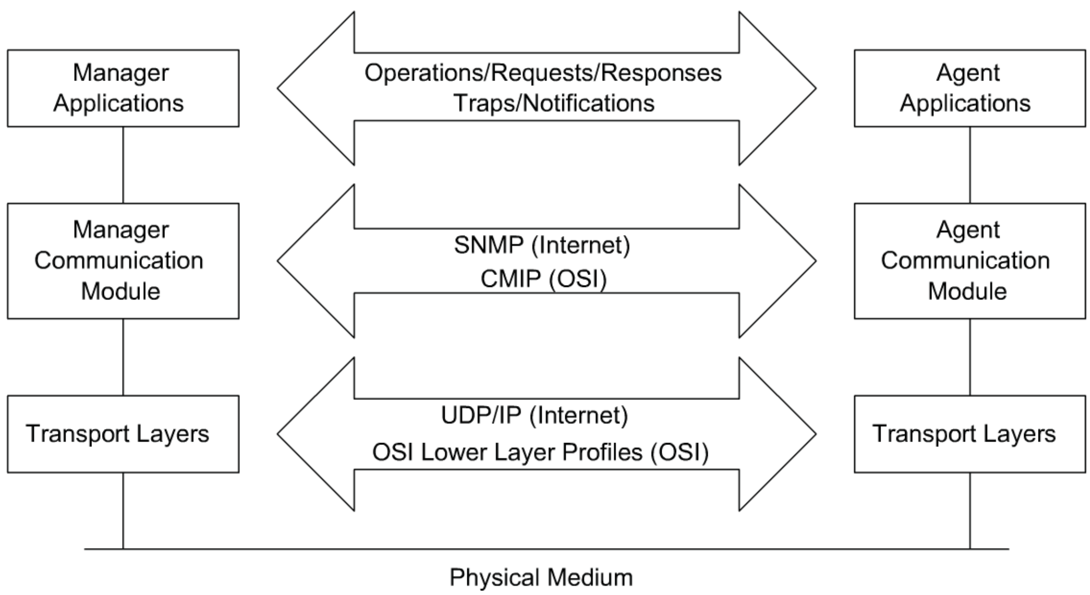

Figure 3.12 Management Communication Transfer Protocols

Azita could go to the automobile dealer and communicate in person with Roberto. Alternatively, she could communicate with Roberto via the Internet. In the former, visual and audio media are the transport mechanisms, and electronic exchange is used in the latter. The communication at the application level could be exchanged in English, Spanish, or any other mutually understandable language between the two. This would be the application-level protocol that is decided between Azita and Roberto. Finally, there are messages exchanged between Azita and Roberto. For example, Azita could request what cars are available and Roberto would respond with the cars that are in stock. Azita could then set a price range and Roberto responds with cars that match the price range. These exchanged messages are the commands/requests/operations and responses/notifications. They can be considered services requested by Azita and provided by Roberto.

Figure 3.11 presents the communication model. The applications in the manager module initiate *requests* to the agent in the Internet model. It is part of the *operations* in the OSI model. The agent executes the request on the network element; i.e., managed object, and returns *responses* to the manager. The traps/notifications are the unsolicited messages, such as alarms, generated by the agent.

Figure 3.12 presents the communication protocol used to transfer information between managed object and managing processes, as well as between management processes. The OSI model uses CMIP along with CMIS. The Internet uses SNMP for communication. The services are part of operations using requests, responses, and alarm notifications.

OSI uses both connection-oriented and connectionless protocols for transportation. For example, the TP4 transport layer protocol riding on top of the x.25 protocol could be used for connection-oriented transporting and application messages. TP4 over Connectionless Network Protocol is used for connectionless transportation. The Internet uses connectionless UDP/IP protocol for transporting messages.

CMIP and SNMP specify the management communication protocols for OSI and Internet management. CMIP is addressed in Appendix A. SNMP is extensively covered throughout the book.

The application processes invoke the management communication layer protocols. OSI deals with messages in the specification of managed objects. Managed objects and their attributes could be manipulated by operations. Basic application service modules are defined by CMIS. In the Internet, operations are executed by SNMP messages.

3.6 ABSTRACT SYNTAX NOTATION ONE: ASN.1

In both the information model and the communication model, discussed in the previous sections, we have addressed functions. In these models, SMI needs to be specified syntactically and semantically, which will be the content of this section.

It is important for communication among systems that a formalized set of rules is agreed upon on the structure and meaning of the language of communication, namely syntax and semantics of the language. There are numerous sets of application and transport protocols. Thus, it is beneficial to choose a syntactical format for the language that specifies the management protocol in the application layer, which is transparent to the rest of the protocol layers. One such format is an old and well-proven format, Abstract Syntax Notation One, ASN.1. We will introduce ASN.1 here to the extent needed to understand its use in network management. The reader is referred to other references [Cassel and Austing, 1996; Larmouth, 1997; Stallings, 1998] for greater depth on the subject.

ASN.1 is more than just syntax. It is a formal language developed jointly by CCITT (now ITU-T) and ISO for use with application layers for data transfer between systems. It is also applicable within the system for clearly separating the abstract syntax and the transfer syntax at the presentation layer. We define the *abstract syntax* as the set of rules used to specify data types and structures for the storage of information. The *transfer syntax* represents the set of rules for communicating information between systems. Thus, the abstract syntax would be applicable to the information model discussed in Section 3.4 and the transfer syntax to the communication model discussed in Section 3.5. The abstract syntax can be used with any presentation syntax, the latter depending on the medium of presentation. The abstract syntax in ASN.1 makes it independent of the lower-layer protocols. ISO 8824/X.208 standards specify ASN.1. The algorithm to convert the textual ASN.1 syntax to machine-readable code is defined in ISO 8825/X.209 standards. It is called Basic Encoding Rules (BER).

3.6.1 Terminology, Symbols, and Conventions

ASN.1 syntax is based on the Backus system and uses the formal syntax language and grammar of Backus–Nauer Form (BNF), which looks like:

<name> ::= <definition>

where the notation "<entity>" denotes an "entity" and the symbol "::=" represents "defined as."

Let us illustrate the Backus system by developing a simple arithmetic expression <SAE> [Maurer, 1977]:

We can define an entity <digit> as

<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

where the symbol "|" represent "or." We can also define an operation entity <op> as

<op> ::= + | - | x | /

The definitions on the right side are called *primitives*. Using these primitives, we can construct more entities. Thus, an entity *number* can be constructed from the primitive, <digit>

<number> ::= <digit> | <digit><number>

For example, the number 9 is the digit 9; the number 19 is the concatenation of the digit 1 and the number 9; and the number 219 is the concatenation of digit 2 with the number 19.

We can now construct a simple arithmetic expression <SAE> from the primitives and the construct <number>. Thus,

<SAE> ::= <number> | <SAE> | <SAE><op><SAE>

The format of each line is defined as a *production* or *assignment*.

Let us consider an example with the following two assignments:

<BooleanType> ::= BOOLEAN

<BooleanValue> ::= TRUE I FALSE

The expression on the left side specifies the name of the type and the right side is the definition or value of the type. Thus, BooleanType is defined as BOOLEAN and BooleanValue is defined as either TRUE or FALSE. The above example illustrates the two basic parameters associated with an entity, namely, *data type* and *value*. The first line is called *data type assignment* and it defines the name of the entity; and the second line, value assignment, specifies the assigned value to the data type. Thus, in the above example the entity BOOLEAN can have assigned values of TRUE or FALSE. Entities that are all in capital letters, such as TRUE and FALSE, are called *keywords*.

A group of assignments makes up an ASN.1 module. For example, a name consists of first, middle, and last names, and they can be specified as:

```
person-name    Person-Name::=
          {
          first      "John",
          middle   "I",
          last      "Smith"
          }
```

Here person-name, beginning with lowercase letters, is the name of the data type. Person-Name is a module and begins with capital letters. The module comprises three assignments, whose names are first, middle, and last with values "John," "I," and "Smith."

Figure 3.13 and Figure 3.14 show two examples of ASN.1 data type definition [Larmouth, 1997]. They are two ASN.1 modules defining data types personnelRecord and trade-Message. Because they are modules, they start with capital letters. PersonnelRecord describes the personnel record of an employee in a global corporation. The Trade-Message is a module specifying a list of invoices defining customer name, part numbers, quantity, charge, and security authentication.

```
PersonnelRecord ::=   SET
                  {                  Name,
                   title             GraphicString,
                   division    CHOICE
                      marketing            [0]      SEQUENCE
                           {Sector,
                            Country},
                  research                 [1]      CHOICE
                           {product -based          [0]      NULL,
                            basic                   [1]      NULL},
                  production               [2]      SEQUENCE
                           {Product -line,
                            Country        }                          }
etc.
```

Figure 3.13 ASN.1 Data Type Definition Example 1

```
Trade -Message ::= SEQUENCE
         {invoice -no              INTEGER
          name                     GraphicString,
          details                  SEQUENCE OF
                                            SEQUENCE
                          {part -no                  INTEGER
                           quantity                  INTEGER },
          charge                    REAL,
          authenticator             Security -Type}

Security -Type ::= SET
         {         ...
                   ...
                   ...       }
```

Figure 3.14 ASN.1 Data Type Definition Example 2

Note that in the examples of Figure 3.13 and Figure 3.14, the data types are built-up from primitive data types: INTEGER, REAL, NULL, and GraphicString. GraphicString is one of several Character-String type primitives. These examples present three kinds of data types, which are built using three construction mechanisms:

alternatives:	CHOICE
list:	SET and SEQUENCE
repetition:	SET OF and SEQUENCE OF

These constructs are used to build structured data types. Just as we saw in the <SAE> example earlier, all data types are built from the ground up using primitive (also called atomic) entities. ASN.1 definition allows both backward and forward references, as well as in-line definition. For instance, in Figure 3.13 the data types Name, Sector, Country, and Product-line are defined externally either before or after the module defining PersonnelRecord. The data type whose name is title is defined in-line as the data type GraphicString. It could have been defined as data type Title as follows:

title   Title::= GraphicString

Let us analyze the three construct types. In PersonnelRecord, the person works in one of the three divisions—marketing, research, or production. This is built using CHOICE construction. Notice that in each of those divisions, research could be either product-based or basic.

The constructs SET and SEQUENCE are list builders. The PersonnelRecord module is a set of data types, Name, GraphicString, Sector, Country, etc., which are all different data types. Since they are different and each is uniquely associated with a name, they can be encoded and transmitted in any order. For example, they could be arranged in any of the following orders:

```
"Smith", "Manager", {"North", "Chile"}
"Manager", "Smith", {"North", "Chile"}
{"North", "Chile"}, "Manager", "Smith"
etc.
```

Notice that "North" and "Chile" are always in the same order. This is because it is a list built with SEQUENCE construction, and the order in the list should be maintained.

The third type of construction is the repetitive types SET OF and SEQUENCE OF. In the example on TradeMessage in Figure 3.14, the SEQUENCE OF construction is shown. The "details" in the invoice are a repetition of data consisting of the ordered list (SEQUENCE construct) of part number and quantity in each invoice. The repetitive records themselves are ordered in a SEQUENCE OF construction. This means that the data will be transmitted in the order in which they are entered. The encoding scheme will preserve that order while transmitting the data from one process to another. For example, if data are entered for *details* in Figure 3.14 as a sequence {part-no, quantity} in the order {1, 5}, {60, 3}, {120, 40}, they will be transmitted in that order by the sending process. If they had been a SET OF construct instead of a SEQUENCE OF construct for *details* in Figure 3.14, the order is irrelevant. The order in this case for the example could be encoded and transmitted by the sending process as any of the combinations, {1, 5}, {60, 3}, {120. 40}; or {60, 3}, {1, 5}, {120, 40}; or {120, 40}, {1, 5}, {60, 3}; etc. without relevance to the order.

The NULL data type used in Figure 3.13, PersonnelRecord, is a placeholder. No value needs to be associated with it except indicating that such a data type exists.

We observe in the PersonnelRecord example in Figure 3.13 that some assignments have integers in square brackets. For instance,

```
{product-based [0] NULL,
basic          [1] NULL}
```

These are called *tags*. The definition of a tag is introduced in ASN.1 to uniquely identify a data type and will be discussed in detail later.

We have used several symbols and primitive data types including *keywords* in the preceding examples. A complete list of ASN.1 symbols is shown in Table 3.2.

Table 3.3 lists some of the frequently used ASN.1 keywords. The reader is directed to the reference [Perkins and McGinnis, 1997] for a more complete list.

As we said earlier, we can group assignments that are related to each other; this group is called a module. A formal definition of a module is as follows:

```
<module name> DEFINITIONS ::= BEGIN
          <name> ::= <definition>
          <name> ::= <definition>
END
```

Table 3.2 ASN.1 Symbols

SYMBOL	MEANING
::=	Defined as or assignment
I	Or, alternatives, options of a list
-	Signed number
--	Following the symbol are comments
{ }	Start and end of a list
[ ]	Start and end of a tag
( )	Start and end of a subtype
..	Range

Table 3.3 ASN.1 Keywords

KEYWORD	BRIEF DESCRIPTION
BEGIN	Start of an ASN.1 module
CHOICE	List of alternatives
DEFINITIONS	Definition of a data type or managed object
END	End of an ASN.1 module
EXPORTS	Data types that can be exported to other modules
IDENTIFIER	A sequence of non-negative numbers
IMPORTS	Data types defined in external modules
INTEGER	Any negative or non-negative number
NULL	A placeholder
OBJECT	Used with IDENTIFIER to uniquely identify an object
OCTET	Unbounded 8-bit bytes (octets) of binary data
OF	Used with SET and SEQUENCE
SEQUENCE	Ordered list maker
SEQUENCE OF	Ordered array of repetitive data
SET	Unordered list maker
SET OF	Unordered list of repetitive data
STRING	Used with OCTET for denoting a string of octets

For example, a MIB definition module will look like:

```
RFC1213-MIB DEFINITIONS ::= BEGIN
…
…
…
END
```

The terms DEFINITIONS, BEGIN, and END are primitives and are called keywords in ASN.1. They are built-in expressions and have special meaning. The DEFINITIONS indicate that the named module,

Table 3.4 ASN.1 Data Type Conventions

DATA TYPES	CONVENTION	EXAMPLE
Object name	Initial lowercase letter	sysDescr, etherStatsPkts
Application data type	Initial uppercase letter	Counter, IpAddress
Module	Initial uppercase letter	PersonnelRecord
Macro, MIB module	All uppercase letters	RMON-MIB
Keywords	All uppercase letters	INTEGER, BEGIN

RFC 1213-MIB, is being defined. The body of a module always starts with BEGIN and ends with END. Grouping assignments into modules has the great advantage that modules can be imported into and exported from other modules. Thus, they are reusable.

We notice in the examples described so far in this section that we have used both lowercase and uppercase letters. There are ASN.1 conventions to designate the data. These are shown in Table 3.4.

3.6.2 Objects and Data Types

We will now use ASN.1 notation to define the various data types and apply them to describe objects in the context of SMI and MIB.

We observed in Section 3.6.1 that the data type could be either a simple type (also called primitive, atomic, or basic), or it could be structured. In addition, we talked about tag designation, which uniquely identifies the data type irrespective of the syntax version. In general, data types are defined based on structure and tag. The structure is subdivided into four categories. The tag is subdivided into class and tag number. This is shown in Figure 3.15. An object can be uniquely defined by its tag, namely class and tag number. For exchange of information between systems, the structure information is also included.

The four categories of data type structure, shown in Figure 3.15 are, *simple type*, *structured type*, *tagged type*, and *other type*.

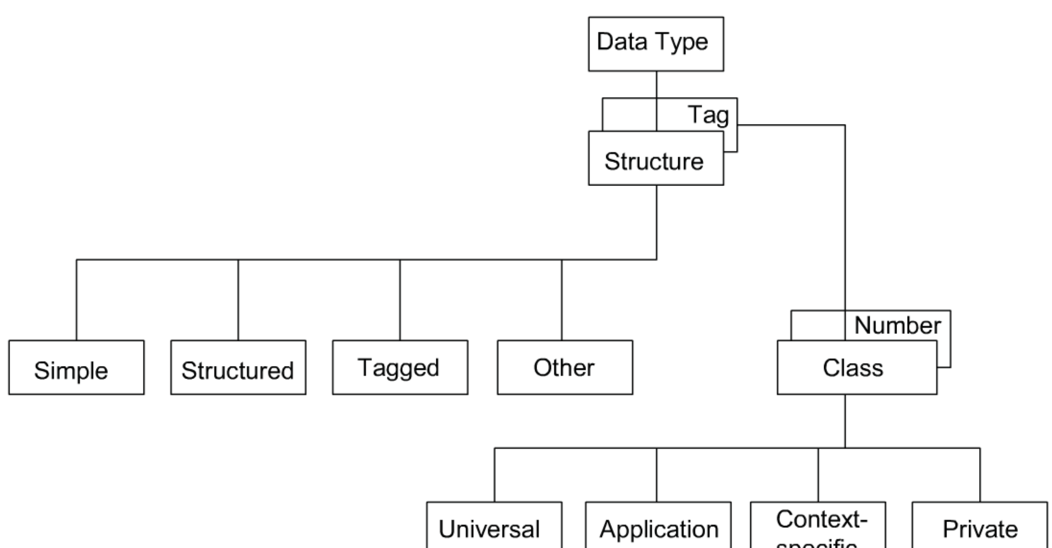

Figure 3.15 ASN.1 Data Type Structure and Tag

A *simple type* is one for which the values are specified directly. For example, we can define a page of a book as PageNumber of simple type, which can take on any integer value. INTEGER is a *simple type*. Thus,

PageNumber ::= INTEGER

Similarly, we can define the chapter number of the book as

ChapterNumber ::= INTEGER

Values for PageNumber can be specified as 1, 2, 3,… and for ChapterNumber as 1, 2, 3,…

A data type is defined as a *structured type* when it contains other types. Types that are within a *structured type* are called *component types*. In the above example, a page number of a book could be defined as a *structured type* by a SEQUENCE construction of ChapterNumber and PageNumber component data types. Let us call it BookPageNumber.

```
BookPageNumber ::= SEQUENCE
                   {ChapterNumber, Separator, PageNumber}
```

where Separator is a data type with value "-." BookPageNumber is a *structured type*. Values for BookPageNumber would then be like 1-1, 2-3, or 6-25.

We can define all the pages of the book as a collection of individual pages. If we want to define them in a sequential order from the first page of the first chapter to the last page of the last chapter, we would use a SEQUENCE OF construction. Let us call it BookPages.

BookPages ::= SEQUENCE OF { BookPageNumber}

We could define the same in an alternative manner as

```
BookPages ::= SEQUENCE OF
                   {
                   SEQUENCE
{ChapterNumber, Separator, PageNumber}
                   }
```

The above two definitions have identical meaning. Values for BookPages would then be 1-1, 1-2, 1-3, …, 2-1, 2-2, 2-3,… The ordering of the values is by the order in which the data are specified and not by sorting of the component data types in the structured construct.

The pages of a book could also be specified as a collection of individual pages in random order. The *structured type* for BookPages would then be constructed with the SET OF data type construct:

```
BookPages ::= SET OF

                   {

                   SEQUENCE

{ChapterNumber, Separator, PageNumber}

                   }
```

Note that we could not have used a SET OF construct for BookPageNumber as the order of the chapter number, separator, and page number is important to keep. However, we could have used the SET construct to define BookPages as

BookPages ::= SET {ChapterNumber, Separator, PageNumber} and assigned values 1-2, 2-3, 1-1, … in a random order. The order of the values in the transmission of data between the sender and the receiver is unimportant. Thus, SET is distinguished from SEQUENCE in two respects. First, the data types should all be distinct; and second, the order of values in SET is of no consequence, whereas it is critical in a SEQUENCE construct. It is also worth noting that the component data types in a SEQUENCE construct need not be distinct since the order is preserved.

Tagged type is a type derived from another type and is given a new tag id. Although a data type has a unique tag associated with it, a tag data type is defined to distinguish types within an application. For instance, in Figure 3.14 although "invoice-no" is an INTEGER type, which we will soon learn as a universal class with a tag number [1], it could have been assigned a local tag id. This is sometimes done to improve the efficiency of encoding.

The fourth and last category of structure is *other type*, which is a data type that is not pre-defined. It is chosen from CHOICE and ANY types, which are contained in other types. Type CHOICE defines the selection of one value from a specified list of distinct types. Thus, in Figure 3.13, "research" uses a CHOICE construct to select one of the two alternatives between "product-based," and "basic." We can represent them with specific values instead of NULL, as follows:

```
research        Research ::= CHOICE
                        {
                            product-based       ProductType,
                            basic               VisibleString
                        }
                        ProductType ::=         VisibleString
```

Type ANY is always supplemented with any valid ASN.1 type defined in another module. We have given two representations for Research, the one above and the other in Figure 3.13. We could give a definition of these two options by defining Research as follows:

```
Research ::= CHOICE
        {
        product-based   ANY,
        BASIC           ANY
        }
```

This definition using ANY specifies that the "product-based" entity could be either a NULL or a ProductType data type, and similarly "basic" could be either VisibleString or NULL.

Figure 3.15 shows two perspectives of data type—*structure* and *tag*. The structure that we have so far described addresses how the data type is constructed. On the other hand, *tag* uniquely identifies the data type. It is required for encoding the data types for communication. Every data type except CHOICE and ANY has a tag associated with it. Tag has two components—*class* and *tag number*. There are four classes of tag. They are: *Universal*, *Application*, *Context-specific*, and *Private*; and each data type belonging to each class is assigned a unique number.

The *universal class* is the most commonly used class, and the ASN.1 list of *universal class* assignments is given in Table 3.5. A core set of assignments is used in all applications. Data types belonging to the *universal class* are application-independent. It is similar to the use of a global variable in a software program, and is applicable anywhere in a program. It need not be defined repeatedly in the subroutines of the program. BOOLEAN and INTEGER are examples of a *universal class*, whose tag numbers are [1] and [2], respectively.

Table 3.5 Universal Class Tag Assignments

TAG	TYPE NAME	SET OF VALUES
Universal 1	BOOLEAN	TRUE or FALSE
Universal 2	INTEGER	0, Positive and negative numbers
Universal 3	BIT STRING	A string of binary digits or null set
Universal 4	OCTET STRING	A string of octets or null set
Universal 5	NULL	Null, single-valued
Universal 6	OBJECT IDENTIFIER	Set of values associated with the object
Universal 7	Object description	Human readable text describing the object
Universal 8	EXTERNAL	The type is external to the standard
Universal 9	REAL	Real numbers, expressed in scientific notation Mantissa × Base^{exponent}
Universal 10	ENUMERATED	Specified list of integers
Universal 11	ENCRYPTED	Encrypted information
Universal 12–15	Reserved for future use	
Universal 16	SEQUENCE and SEQUENCE OF	Ordered list of types
Universal 17	SET and SET OF	Unordered list of types
Universal 18	NumericString	Digits 0–9, space
Universal 19	PrintableString	Printable characters
Universal 20	TeletexString	Character set specified by CCITT Recommendation T.61
Universal 21	VideotexString	Character set specified by CCITT Recommendation T.100 and T.101
Universal 22	IA5String	International Alphabet 5, which is equivalent to ASCII
Universal 23	UTCTime	Time format YYMMDDHHMM[SS][local time differential from universal standard time]
Universal 24	GeneralizedTime	Time format YYYYMMDDHHMM[SS][local time differential from universal standard time]
Universal 25	GraphicString	Graphic character set specified by ISO 8824
Universal 26	VisibleString	Character set specified by ISO 646, equivalent to ASCII
Universal 27	GeneralString	General character string
Universal 28	CharacterString	Character set
Universal 29–	Reserved for future use	

Tags belonging to the *application class* are specific to applications. Examples of application-specific tag numbers are used in examples in Figure 3.13. A universal class tag number can be overridden with an application-specific *tag number*. Types in two different applications can have the same application-specific tag, but carry two different meanings.

```
ErrorStatus ::=
        INTEGER{
                NoError(0)
                tooBig(1)
                noSuchName(2)
                badValues(3)
                readOnly(4)
                genErr(5)
                }
```

A subtype data type is derived from a parent type. For example, in the PageNumber example, if we limit the maximum page number to 255 (based on 2^8), then the assignment would read

PageNumber ::= Integer (0..255)

The parenthesis indicating that it is a subtype expression (see Table 3.2), where the integer range is from 0 to 255.

Let us conclude this section with a real-life example in network management of a data type, which is the address translation table in SNMP IP MIB. An entry in the table is of data type IpNetMediaEntry, which is a sequence of four managed objects with associated data types as shown below. Each of the four objects starts with a lowercase letter, and the associated data type with either a capital letter or is all capital letters.

```
IpNetMediaEntry ::=SEQUENCE {
        ipNetToMediaIfIndex          INTEGER
        ipNetToMediaPhysAddress      PhysAddress
        ipNetToMediaNetAddress               IpAddress
        ipNetToMediaType             INTEGER}
```

3.6.3 Object Name

In a MIB, there is an identifier for each occurrence of an object. In the ASN.1 notation, it is the OBJECT IDENTIFIER. The object identifier for the Internet shown in Figure 3.8 is

internet OBJECT IDENTIFIER ::= {iso(1) org(3) dod(6) internet(1)}

Thus, the object identifier for the Internet has the value 1.3.6.1, which we discussed in Section 3.5.1. The MIT shown in Figure 3.8 has been extended to include the class private type in the MIB and is shown in Figure 3.16. Thus, the object identifier for private enterprise IBM is 1.3.6.1.4.1.2.

3.6.4 An Example of Use of ASN.1 from ISO 8824

Figure 3.17 shows the ASN.1 structure for a personnel record. Part (a) shows the informal description, part (b) shows the ASN.1 description of the record, and part (c) shows the description of the record value. There are several salient points to note in this example. First, there are no simple types in this example such as the page number defined in Section 3.6.3. The data type, Name, does not have an associated object name, although we could define one, for example, personnel-name. In such a case, the second line in Figure 3.17(b) would read

personnel-name   Name

PersonnelRecord is a structured data type, SET with the basic component types Name, EmployeeNumber, Date, Name (nameOfSpouse), and ChildInformation. ChildInformation itself is a

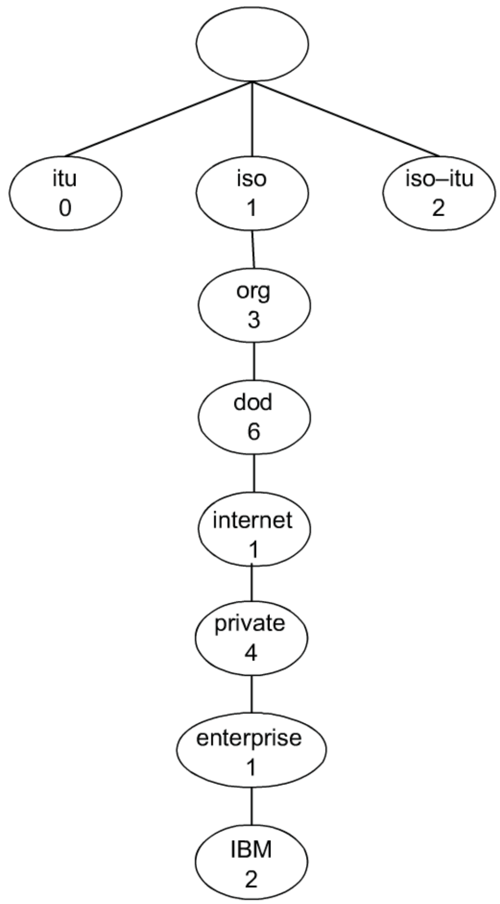

Figure 3.16 IBM as an Example of a Private Class in MIT

structured data type, a SET consisting of Name and Date as component types. A third structured data type that we notice is SEQUENCE for the data type Name with VisibleString as the component types.

The SEQUENCE type is used for Name and the SEQUENCE OF type is used for children, which contains component type SEQUENCE. Thus, the first occurrence of Name in PersonnelRecord is a SEQUENCE construct, and the same construct is embedded in children, which is a SEQUENCE OF construct. Thus, we see a nested structure in this example.

The structure for PersonnelRecord is a structured type and it could have been defined without the data designation IMPLICIT as well as the local tag [APPLICATION 0]. However, as mentioned in Section 3.6.2, the local tag type has been used to improve the efficiency of coding. Further use of the IMPLICIT designation makes the coding more efficient in that it will be encoded with the [APPLICATION 2] tag and not the UNIVERSAL tag, which is also applicable. In this situation, it would not be encoded as UNIVERSAL type 1.

3.7 ENCODING STRUCTURE

The ASN.1 syntax containing the management information is encoded using the BER defined for the transfer syntax. The ASCII text data are converted to bit-oriented data. We will describe one specific encoding structure, called TLV, denoting Type, Length, and Value components of the structure. This is shown in Figure 3.18. The full record consists of type, length, and value.

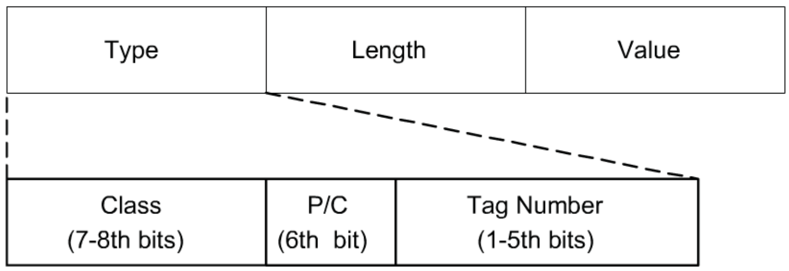

Figure 3.18 TLV Encoding Structure

defined in Table 3.6. The value of P/C is 0 for Primitive and 1 for Construct. The lowest 5 bits (1–5) designate the tag value in binary. For example, INTEGER, from Table 3.5, belongs to a universal class with a tag value of 2 and is a primitive data type. Hence, the type is 00000010.

The length specifies the length of the value field in the number of octets. The length is defined as a series of octets. It is either one octet (short) or more than one octet (long). The most significant bit (8th bit) is set to 0 for a short length with the low 7 bits indicating the length of the value. If the value field is longer than 127 (maximum specified by 7 bits), then the long form is used for length. The 8th bit of the first octet is marked as 1 and the rest of the seven bits of the first octet indicate how many octets follow to specify the length. For example, a value length of 128 would look like

10000001 10000000

The value field is encoded based on the data type. It is a multiple number of octets. The simplest data type value to encode is an OCTET STRING. An octet string of ‘0C1B’H (the string is designated with apostrophes on both sides and an H denoting hexadecimal notation) would look like

00001100 00011011

The complete TLV for the string of octets ‘0C1B’H is made up from universal (00) Primitive (0) data type of a tag value of 4 with a one-octet length field to indicate that there are two octets of value field. It is

00000100 00000010 00001100 00011011

The integer value is encoded using a twos-complement form. For a positive value, the actual value is the binary representation with the most significant always being 0 to indicate a positive sign. If the integer exceeds 127, an additional octet of 0s is prefixed. Thus, a value of 255 is written as 00000000 11111111, with the leading 0 indicating the positive sign bit. For a negative integer, the absolute value of the integer is written in a binary form. The leading sign bit should be 0 to indicate the positive sign. Invert all the 1s to 0s and all the 0s to 1s. Then add 1 to the inverted binary digits. The leading sign bit will automatically become 1, indicating a negative integer. For example, a −5 will start as 00000101.

Table 3.6 Value of Class in Type

CLASS	8TH BIT	7TH BIT
Universal	0	0
Application	0	1
Context-specific	1	0
Private	1	1

Inverting the bits and adding 1, it becomes 11111011. Refer to Perkins and McGinnis [1997] for the encoding of other values.

3.8 MACROS

The data types and values that we have so far discussed use ASN.1 notation of syntax directly and explicitly. ASN.1 language permits extension of this capability to define new data types and values by defining ASN.1 macros. The ASN.1 macros also facilitate grouping of instances of an object or concisely defining various characteristics associated with an object.

The structure of a macro takes the form shown in Figure 3.19.

As can be observed from Table 3.4, the keyword for a macro is all in capital letters. TYPE NOTATION defines the syntax of the new types and VALUE NOTATION defines the syntax of the new values. The auxiliary assignments define and describe any new types identified.

The OBJECT-IDENTITY macro is used to define information about an OBJECT IDENTIFIER assignment. Figure 3.20 shows an example from RFC 2578 of creating an Internet object using an OBJECT-IDENTITY macro. The two syntactical expressions STATUS and DESCRIPTION are mandatory and the type ReferPart is optional. The value in VALUE NOTATION defines the object identifier.

As an example of the usage of the OBJECT-IDENTITY macro, let us consider a registration authority that registers all computer science courses that are offered in the College of Computing. Suppose we

```
<macroname> MACRO ::=
BEGIN
        TYPE NOTATION ::=  <syntaxOfNewType>
        VALUE NOTATION ::= <syntaxOfNewValue>
        <auxiliaryAssignments>
END
```

Figure 3.19 Structure of an ASN.1 Macro

```
OBJECT-IDENTITY  MACRO
BEGIN
TYPE NOTATION ::=
“STATUS” Status
“DESCRIPTION”Text
ReferPart
VALUE NOTATION ::=
value(VALUE OBJECT IDENTIFIER)
Status ::= “current” | “deprecated” | “obsolete”
ReferPart ::= “REFERENCE” Text I empty
Text ::= “value (IASString)”
END
```

Figure 3.20 OBJECT-IDENTITY Macro [RFC 1902]

```
cs8113          OBJECT-IDENTITY
        STATUS                  current
        DESCRIPTION             "A graduate-level network management course
                                 offered every fall by College of Computing in
                                Georgia Institute of Technology."
                ::= {csclasses 50}
```

Figure 3.21 Example for OBJECT-IDENTITY Macro

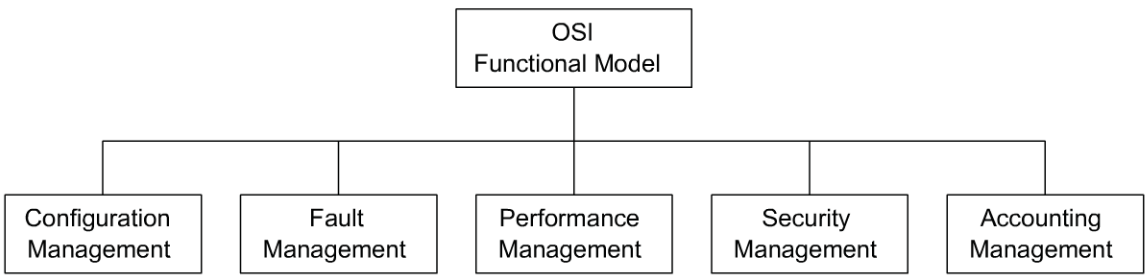

Figure 3.22 Network Management Functional Model

want to formally register the network management course *cs8113* under the object descriptor *csclasses* as the 50th subnode. We can specify an ASN.1 OBJECT-IDENTITY macro shown in Figure 3.21. The object identifier *cs8113* has a value {csclasses 1}. Its status is current and has a description explaining the course offering.

3.9 FUNCTIONAL MODEL

The functional model component of an OSI model addresses user-oriented applications. They are formally specified in the OSI model and are shown in Figure 3.22. The model consists of five models: configuration management, fault management, performance management, security management, and accounting management. Part III of the book is devoted to the application aspects of network management.

Configuration management addresses the setting and changing of configurations of networks and network components. Relevant management information is embedded in managed objects, such as switches, hubs, bridges, and routers. Configuration management involves setting up these parameters. For example, alarm thresholds could be set to generate alarms when packet loss exceeds a defined value. Information on the object name and contact person to be contacted when the component fails could be entered in the management agent. The configuration data are gathered automatically by, and are stored in, the NMS at the network operations center (NOC). NMS displays in real-time the configuration of the network and its status.

Fault management involves detection and isolation of the problem causing the failure in the network. An NMS constantly monitors and displays in real-time major and minor alarms based on the severity of failures. Restoration of service is done as soon as possible and it could involve reconfiguration of the network, which is part of configuration management. In several failure situations, the network could do this automatically. This network feature is called self-healing. In other situations, restoration of service does not include fixing the cause of the problem. A trouble ticket is generated and followed up for resolution of the problem using a trouble ticket administration system.

This is the trouble ticket administration of fault management and is used to track problems in the network. All problems—including non-problems—are to be tracked until resolved. Periodic analysis of the data, which are maintained in a database, is done to establish patterns of the problems for follow-up action. There are trouble-tracking systems to automate the tracking of troubles from the automatic generation of a trouble ticket by an NMS to the resolution of the problem.

Performance management is concerned with the performance behavior of the network. The status of the network is displayed by a network-monitoring system that measures the traffic and performance statistics on the network. Network statistics include data on traffic volume, network availability, and network delay. Traffic data can be captured based on the traffic volume in various segments of the network. Data need to be gathered by the NOC and updated in a timely fashion in order to administer performance management. Any configuration changes needed to relieve temporary congestion in traffic are made by the NOC. Permanent relief is engineered by the addition of equipment and facilities as well as policy changes. Performance-monitoring tools can gather statistics of all protocol layers. We can analyze the various application-oriented traffic such as Web traffic, Internet mail, file transfers, etc. The statistics on applications could be used to make policy decisions on managing the applications. Performance data on availability and delay are useful for tuning the network to increase the reliability and to improve its response time.

Security management covers a broad range of security aspects. It involves physically securing the network, access to the network resources, and secured communication over the network. A security database is established and maintained by the NOC for access to the network and network information. Any unauthorized access to the network resources generates an alarm on the NMS at the NOC. Firewalls are implemented to protect corporate networks and network resources from being accessed by unauthorized personnel and programs, including virus programs. Secured communication is concerned with the tampering of information as it traverses the network. The content of the information should neither be accessed nor altered by unauthorized personnel. Cryptography plays a vital part in security management.

Accounting management administers cost allocation of the usage of network. Metrics are established to measure the usage of resources and services provided. Traffic data gathered by performance management serve as input to this process.

Another dimension of application management is concerned with service and business management, which we discuss in Chapter 7. Service and business management is directed toward service providers, in order for them to accomplish customer satisfaction and to ensure the profitability of business. The traffic statistics, trouble ticket administration data, and accounting management results are inputs to service and business management.

Summary

The foundations of standards, models, and language needed to delve into the study of network management have been addressed in this chapter. These are the four network management models—OSI, Internet, Telecommunications Network Management, and IEEE 802—and a fifth emerging one using Web technology.

The OSI management model categorizes the four functions of network management into four models. They are configuration, information, communication, and application functions. Each of these has been addressed in detail. Some parts of the OSI model are applicable to the other three management models.

The organization model describes the management process in the network element, called the agent process, and the management process in the manager. We presented the two-tier and three-tier architectural models and the relationship between them.

The information model addresses the SMI that enables processes running in different components in the network to exchange management data. We defined the management object for both OSI and Internet/SNMP management models.

The two primary communications protocols are CMIP in OSI and SNMP in the Internet.

We discussed the syntactical format, Abstract Syntax Notation One, and how it is applied to defining managed objects. We presented the terminology, symbols, and conventions used in ASN.1, and then defined the various categories and structure of data types. We defined the managed objects in OSI and SNMP/Internet management models in adequate detail so that we should be prepared to study SNMP management in the next two chapters. We briefly covered how ASN.1 is applied to specifying the management information tree and MIB by giving some specific examples.

The text-oriented ASN.1 specifications need to be encoded for transmission of data between systems. We discussed the most widely adopted encoding scheme, the Basic Encoding Rules.

We defined the extension to ASN.1 in defining an ASN.1 macro and presented an example from the SNMP management model used to create a new object.

The application functions are subdivided into five categories of management: configuration, fault, performance, security, and accounting. We have addressed each function briefly in this chapter.

Exercises

1. What are the standards used for the various layers in an Ethernet-based network that is managed by the Internet management protocol? Assume that Ethernet runs on 10 Mbps on an unshielded twisted pair cable.
2. Consider a network of multivendor network components. Hubs are made by Cabletron and are managed by Cabletron's Spectrum NMS (network management system). Routers are made by Cisco and are managed by CiscoWorks NMS. The entire network is managed by general-purpose NMS such as HP OpenView Network Node Manager. Draw a two-tier management network that performs configuration and fault management. Explain the rationale for your configuration.
3. Redraw the management network configuration of Exercise 2 as a three-tier configuration. What are the requirements on the three-tier network management system?
4. Explain succinctly the difference between the database of a network management system and its MIB. How do you implement each in a network management system?
5. You have been assigned the responsibility of adding a new vendor's components with its own network management system to an existing network managed by a network management system. Identify the three sets of functions that you need to do to fulfill your task.
6. Write an ASN.1 module that specifies DaysOfWeek as SEQUENCE type with each day of the week (day1, day2, …) as the type VisibleString. Write the ASN.1 description
   (a) for the structure and
   (b) for the value
7. Repeat Exercise 6 defining DaysOfWeek as an ENUMERATED data type, with values from 0 to 6.
8. The following is the informal record structure of my home address:

Name	Mani M. Subramanian
Address	1652 Harts Mill Road
City	Atlanta
State	GA
Zip Code	30319

Write for your record:

(a) the informal record structure
(b) ASN.1 description of the record structure
(c) the record value for your home address

9. Given the definition

```
class ::= SET {
        name            VisibleString
        size            INTEGER
        graduate                BOOLEAN
        }
```

which of the following set of values is (are) compatible with the above ASN.1 record structure?

(a) "CS4803B," FALSE, 28
(b) CS8113B, TRUE, 28
(c) "CS4803B," 28, TRUE
(d) CS4803B, 28, TRUE

10. (a) Describe a list and an ordered list in ASN.1 syntax
(b) Identify the differences between them
(c) Differentiate between list construction and repetitive construction using examples

11. In a ballroom dance, the conductor asks the guests to group themselves into couples made up of a male and a female (order does not matter) for a dance. Write an ASN.1 module for danceGroup with data type DanceGroup that is composed of data type Couple; couple is constructed using male and female.

12. A high school class consists of four boys and four girls. The names of the boys with their heights are Adam (65"), Chang (63"), Eduardo (72'), and Gopal (62"). The names of the girls are Beth (68"), Dipa (59"), Faye (61"), and Ho (64"). For each of the following cases, write an ASN.1 description for record values by selecting appropriate data types. Start with data type **Studentinfo**, listing information on each student.

(a) A random list of the students
(b) An alphabetized list of students
(c) Sorted line up of students with increasing height
(d) Any one student to be a class representative to the faculty meeting
(e) Two groups, each of boys and girls only

13. In Section 3.6.2, we defined the tag for Chapter-number type as APPLICATION 2. Encode this chapter (3) in TLV format.

14. You are establishing a small company with your network managed by a network management system. Give an example of each of the five functional applications that you would implement.