ISO OSI LAYERED PROTOCOL MODEL

In recent years, information technology has become a major part of human civilization. The sheer variety of information technology has created a significant problem when interconnecting computer systems. The Open System International (OSI) standards are a set of international standards prescribing how multivendor computer systems can be interconnected in a consistent fashion. Among these standards, the standard on the OSI Reference Model provides a framework for the development of protocols to interconnect open systems. This article attempts to introduce the OSI Reference Model and the OSI protocols.

INTEROPERABILITY

The purpose of the OSI interconnection standards is to promote interoperability, which will enable open systems to communicate with each other. To appreciate the role of the OSI interconnection standards in promoting interoperability, a three-step strategy to achieve interoperability is described in the following.

The first step is to develop the OSI standards. The International Organization for Standardization (ISO) and the International Electrotechnical Committee (IEC) jointly formed a committee, ISO/IEC JTC (Joint Technical Committee) 1, which is responsible for solving problems that can occur at the interconnection and information processing levels. Within JTC 1 are subcommittees (SC). For example, SC 6 is involved in lower-layer information exchange standards and SC 18 is involved in information processing standards. The OSI Reference Model is covered in ISO/IEC 7498 which is an SC 21 standard.

Another important OSI standard organization is the International Telecommunications Union (ITU). The ITU recommendations provide end-to-end compatibility for international telecommunications. Although the ITU has been more concerned with issues addressing the lower three layers of the OSI Reference Model and with applications using the telecommunications capabilities, there is, however, a great deal of overlap between its work and that of the ISO. Indeed, ISO and ITU publish nearly identical texts as both an International Standard and a Recommendation.

The second step is to establish functional profiles. The difficulty in implementing the open systems standards is the abstract often informal nature of the standards documents, as well as the wide variety of options that can be implemented for a protocol. Furthermore, in a given layer of the model, there may be a variety of similar protocols. To overcome such difficulties in consistency, OSI implementors define *functional profiles* which identify specific protocols and specific choices of permitted options and specific values for parameters in the standards. Functional profiles have been defined primarily in three OSI regional workshops, which are the OSI Implementation Workshop (OIW) in the United States, the European Workshop in Open Systems (EWOS) in Europe and the Asian and Oceanic Workshop (AOW) for Japan, Australia and the Pacific Rim countries. Because functional profiles are developed in three different workshops, there is potential for interoperability problems between systems implemented in different parts of the world. A subcommittee of ISO/IEC JTC 1 was formed in 1987 to define International Standard Profiles (ISPs) which would harmonize potentially divergent regional efforts into common internationally recognized functional profiles.

The third step is to test. There are two kinds of OSI tests—conformance testing and interoperability testing. *Conformance testing* examines a product to determine whether it meets the standard and profile requirements. It is not ensured that two products that have passed conformance testing may interoperate with each other, e.g., in the case where they choose incompatible option values. *Interoperability testing* is done to determine whether two products conform to the same standard and profile requirements. This process is time consuming because a total of $n*(n-1)/2$ interoperability tests are needed to demonstrate full interoperability in an environment consisting of n implementations. Test strategies, concepts and scripts have been developed by ISO and specific industry consortia. Testing houses have been established to conduct manufacturer-neutral conformance and interoperability test campaigns.

OSI REFERENCE MODEL

In 1978, ISO proposed the establishment of a framework for developing standards for the interconnection of heterogeneous computer systems, potentially via a variety of intermediary networking devices. The resulting framework, known as the OSI Reference Model (ISO/IEC 7498 or ITU X.200) was published in the spring of 1983. Since then, it has been extended to cover the connectionless communication mode, security, naming/addressing, and management.

The interconnection requirements can be divided into internetworking requirements and interworking requirements. To meet these requirements, the OSI Reference Model is divided into seven layers (Fig. 1). Layers 1 through 4 deal with the internetworking requirements; Layers 5 through 7 deal with the interworking requirements.

The objective of the internetworking requirements is to provide physical connectivity between systems in different networks in an internetworking environment. Conceptually, the internetwork environment consists of subnetworks that are connected by intermediate systems (IS) and populated by end systems (ES). Applications are found in ESs and ISs provide the glue to interconnect ESs. The objectives behind the internetworking requirement are twofold, specifically, trans-

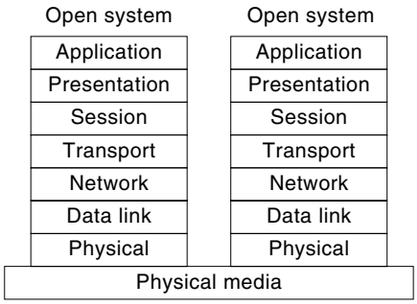

Figure 1. OSI Reference Model.

parency over the topology of the internetwork, and transparency over the transmission media used in each subnetwork.

The internetworking requirements are met by the lower four layers in the following way. Layer 1, the physical layer, is responsible for providing physical connectivity among adjacent systems. The bits in a physical connection may occasionally experience corruption. Thus Layer 2, the data link layer, takes care of error handling as well as flow control for physical connections. Because not all the ESs are necessarily adjacent to each other, Layer 3, the network layer, provides network connectivity among nonadjacent ESs. Although a network connection may require one or more connections to ISs, the Network Layer ensures that the connectivity details are transparent to the two ESs. Layer 4, the transport layer, provides the final touch in meeting the internetworking requirements. Although the lower three layers provide the means for moving data from one ES to another, reliability is still an issue. For example, the aggregated bit-error rate over all the underlying subnetworks may not meet the performance requirements of the application. The purpose of the transport layer is to enhance the transmission quality of the network layer. As a result, users of the transport service are presented with a reliable end-to-end transport pipe.

Even with the task of providing the internetworking service, ESs may not be able to interwork significantly with each other. For instance, an ES may not be able to interpret the received bits although the bits are received correctly. The interworking service is provided by Layers 5 through 7. Layer 5, the session layer, is responsible for dialogue control as well as synchronization in the case of bulk transfer. It achieves its functionalities by adding a structure to the transport pipe.

Layer 6, the presentation layer, provides the environment for applications to determine the application context. The application context, which determines the scope of communication, should identify the syntax of the application information exchanged. These syntax are known as abstract syntax because they do not depend on how their values are represented locally. Values of the abstract syntax are represented in the transfer by a set of rules known as the transfer syntax. The presentation layer provides the capabilities for applications to negotiate on such transfer syntax. Applications can also rely on the presentation layer to perform encryption for secured communication.

Layer 7, the application layer, is responsible for meeting all the interworking requirements that are not met by the lower layers. The application layer model provides a structured approach to build objects in the application layer due to the large number of communication requirements.

OSI APPLICATION STANDARDS

Among the OSI application standards, the following are introduced below: Common Management Information, X.500, X.400, File Transfer Access Management, and Remote Database Access.

Common Management Information Protocol

The OSI management standards are to provide uniformity in the specification of a management communication protocol, uniformity in the definition of managed objects, and uniformity in the definition of systems management functions.

The Common Management Information Protocol (CMIP) is the OSI protocol used to support the exchange of system management messages between a manager and an agent (Fig. 2). *Managed objects* are management views of resources such as network elements. The manager invokes remote management operations such as finding the status of a managed object. After the manager successfully completes these operations, the results are returned to the manager by the agent. The agent can send a notification to the manager as the result of a (e.g., alarm) notification received from a managed object.

Unlike other management protocols, CMIP is connection-oriented. The management operations, which are defined in ISO/IEC 9595, can be summarized as follows:

- M-GET: This operation is invoked to retrieve attribute values from one or more managed objects.
- M-CREATE: This operation is invoked to create a managed object.
- M-SET: This operation is invoked to modify attribute values of one or more managed objects.
- M-ACTION: This operation is invoked to have a defined, ad-hoc action performed on a managed object, e.g., resetting a network connection.
- M-DELETE: This operation is invoked to delete a managed object (e.g., deleting a log record).
- M-CANCEL-GET: This operation is invoked to request the cancellation of an outstanding invocation of the M-GET operation.
- M-EVENT-REPORT: This operation is invoked by the agent to report an event to the manager.

Operations such as M-GET, M-SET, M-ACTION and M-DELETE contain scoping and filtering parameters that permit the agent to operate only on the managed objects that have

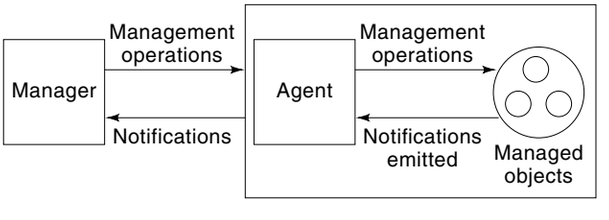

Figure 2. CMIP Functional Model.

passed the filter. Only intelligent agents with the ability to evaluate filters can perform such operations. Therefore, it is not practical to run OSI agents on dumb devices.

To provide uniformity in the definition of managed objects, the OSI management standards use the object-oriented approach and provide Guidelines to Definition of Managed Objects (GDMO) templates to define managed object classes. *A managed object class* is a set of managed objects with similar characteristics. A managed object is an instance of a managed object class. For example, the transport-connection managed object class is an encapsulation of common characteristics of transport connections. When a transport connection is established, a transport-connection managed object can be created to model the management behavior of a transport connection.

The managed object class template forms the basis of the definition of a managed object class. It supports inheritance by identifying the inheritance relationships between the class and other managed object classes. Within the template are place holders for the specification of behavior, attributes, notifications and actions. Attributes are used to specify properties such as states, and notifications are used to specify unsolicited messages that can be emitted by a managed object.

To facilitate rapid development of managed object classes, standard and profile groups have registered managed object classes for the common resources. For example, ITU M.3100 provides definitions of managed object classes for generic telecommunications resources. These definitions can be used to derive (via inheritance) definitions of managed object class for specific telecommunications resources.

System management functions are required to address a variety of management functions for the entire managed system. Management functions can be classified into five categories: fault management; configuration management; performance management; accounting management; and security management. The OSI standards have defined systems management functions to address common problems in these five categories. Among these functions, the object management function, for example, provides services for the reporting of creation/deletion and attribute value change of managed objects; the event reporting management function provides services for event report delivery to managers.

The OSI management standards have found their greatest success in telecommunications applications. ITU M.3010 defined a five-layer model for Telecommunications Management Network (TMN), and recommended the use of OSI management standards. Given the huge number of OSI management standards, it is not easy for a telecommunications manager to identify the OSI management standards that can be used to solve a specific management problem. To address this dilemma, the Network Management Forum (NMF) came up with the idea of a solution set. Similar to the notion of a functional profile, a solution set provides the solution to a specific telecommunications management problem. It identifies the managed object classes and systems management functions which are needed to solve the problem. Through scenarios, it explains how the functions can be applied to meet the requirements of the problem. The NMF has already defined a number of useful solution sets for practical telecommunications problems. A majority of telecommunications managers have committed to deploying NMF solution sets to solve telecommunications problems.

X.500

In a distributed computing environment, the X.500 standard defines a directory service to manage information which is not likely to change often (e.g., e-mail addresses and network addresses). The directory information base (DIB) is the conceptual repository of such directory information. Structured as a directory information tree (DIT), it represents each piece of directory information (known as directory entry) by a tree node.

In the X.500 Functional Model (Fig. 3), the directory user agent (DUA) invokes directory access operations to request directory services on behalf of a directory user. The outcome of the invocation is either a result, a referral or an error. A result is returned if the request has been carried out successfully. A referral is returned if the service is unobtainable at a service access point or more easily reached at another service access point. An error is returned if the request cannot be carried out.

Directory access operations fall into three categories. The read category consists of operations to retrieve information of a directory entry, to compare the value of a directory entry with a submitted value, or to cancel an outstanding interrogation. The search category consists of operations to list "child" subordinates of a directory entry or to search for directory entries satisfying an input filter. The modify category consists of operations to add/remove a directory entry, to modify a directory entry, or to change a directory name. The directory access operations are described in the Directory Access Protocol (DAP).

In a large network, the directory services may be distributed among multiple directory service agents (DSA). The DSAs collaborate to provide the distributed directory service in three modes—chaining, referral and multicasting. In the chaining mode, a DSA continues passing a request to another DSA until one is found that can provide the requested information. In the multicasting mode, a DSA which does not have the requested information chains an identical request in parallel to multiple DSAs. In the referral mode, a DSA which does not have the requested information refers another DSA to the DUA. The directory service protocol (DSP) is used by the DSAs to provide the distributed directory service. The DSP operations are quite similar to the DAP operations.

In the X.500 information model, a directory entry is either an object entry or an alias entry. An object entry is the primary collection of information about a real world object. On the other hand, an alias entry, which points to another directory entry, is used primarily to provide a user-friendly name to the referenced entry. Directory entries of similar characteristics are grouped into directory object classes. Examples of directory object classes are country, organization, person and device. The X.500 standard provides templates for the speci-

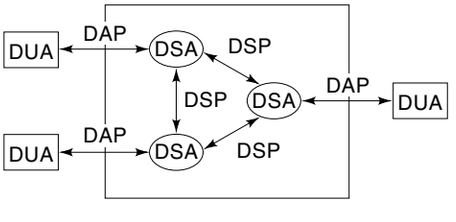

Figure 3. The X.500 Functional Model.

fication of directory object classes as well as attributes used in the definition of directory object classes.

The naming of a directory entry is straightforward. As a tree node, the directory entry is labeled by a set of naming attributes known as the relative distinguished name (RDN). There is a unique path from the node to the root of the DIT. The concatenation of the RDNs of nodes in this path gives the distinguished name (DN) of the directory entry. The X.400 standard relies on DNs for the naming of X.400 users.

X.400

The current trends towards electronic office automation have created a demand for universal message communication. The X.400 standard defines a store-and-forward infrastructure for the transport of interpersonal messages and business documents.

Figure 4 shows the X.400 functional model. An X.400 user is a person or an application process that originates and receives messages. A user agent (UA) is a process which acts on behalf of an X.400 user. It is responsible for the submission and the delivery of messages. The message transfer system (MTS) provides the store-and-forward transfer service for the X.400 systems, and it transfers messages regardless of their content types. However, it does not examine or modify the message content unless content conversion is explicitly requested by the originator.

A message store (MS) provides a secure and continuously available storage mechanism on behalf of a UA. By serving as an intermediary entity between the UA and the MTS, the MS can store incoming messages from the MTS until the UA is ready to process the messages. It can perform autoforwarding and provide a summary of the stored messages. Not every UA has an associated MS. If it does have one, all incoming messages for the UA are delivered first to the associated MS.

The value of the X.400 system can be enhanced if it can be connected to other non-X.400 systems such as postal systems. An access unit (AU) is a functional object to link a non-X.400 system to the X.400 system. There are a number of AU types, for example, telex, teletex and facsimile.

The functionality of the MTS is distributed among a set message transfer agents (MTA). To illustrate how a UA interacts with an MTA, suppose that the originator UA submits a user message to an originator MTA. First, the originator MTA validates the submission envelope and records appropriate administrative information on the envelope. Next, the MTA attempts to deliver the message to the recipient. If the recipient is local to the originator MTA, the delivery is straightforward because it does not involve another MTA. If not, the originator MTA needs to relay the message to another MTA. When there is more than one recipient, the originator MTA needs to create a copy for each MTA to which the message is relayed. An MTA, on receiving the relayed message from the originating MTA, may be held responsible for progressive delivery of the message to the intended recipient(s). An MTA, if held responsible, can discharge its responsibility by either delivering the message if the recipient is local to the MTA or by transferring the message to other MTAs that are closer (measured according to a metric, such as distance) to the recipient. If delivery is unsuccessful, its responsibility ends with the generation of a nondelivery report.

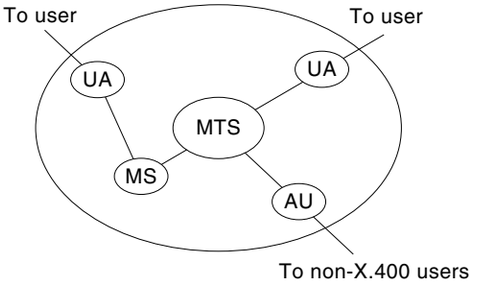

Figure 4. The X.400 Functional Model.

The X.400 protocols are designed to reflect two major levels (i.e., the MTS and the MTS-user). The P1 protocol addresses how an MTA interacts with other MTAs to provide the distributed MTS service. The P3 protocol addresses how a UA accesses the MTS service. The P7 protocol addresses how a MS accesses the MTS service. Similar to the X.500 protocols, each X.400 protocol is modeled by a set of operations. For example, the P3 protocol specifies operations for an originator UA to submit/receive a message, and to manage messages in a mailbox.

The X.400 information model deals with how a message is structured. An interpersonal message (IPM) consists of an envelope and an IPM content. The IPM content consists of a header and one or more body parts. Each body part has an encoded information type (EIT), e.g., text, facsimile or telex. If conversion is explicitly requested by the originator UA, an MTA can transform an EIT to another EIT. In addition to IPMs, thre are also interpersonal notifications (IPNs). An IPN conveys a receipt/nonreceipt notification. If the originator UA requests a receipt notification, an IPN signifying receipt is sent by the recipient UA to the originator UA.

The X.400 standard relies on the X.500 standard in a number of ways. In naming, for example, an X.400 user is named by an O/R (i.e., Originator/Recipient) name which is a two-slot data structure—DN and O/R address. The DN provides a user-friendly naming of the X.400 user. For example, the following DN may be assigned to Adrian Tang: {C = US, ORG = UMKC, PersonalName = Adrian Tang}.

The O/R address component identifies the management domains (MD) to which an X.400 user belongs. The X.400 standard defines a management domain to be a set of messaging systems owned by an administration or an organization. An MD managed by an administration (e.g., a public carrier) is called an administration management domain (ADMD), while an MD managed by an organization (e.g., a private company) other than an administration is called a private management domain (PRMD). For example, the following O/R address may be assigned to Adrian Tang: {C = US, ADMD = ATTMAIL, PRMD = UMKC, PersonalName = Adrian Tang}.

File Transfer Access Management

Existing file protocols are concerned primarily with moving complete files. The file transfer access management (FTAM) standard (ISO/IEC 8571) has broadened the scope of these protocols by offering three modes of file manipulation, i.e., file transfer, file access and file/filestore management. File transfer is the movement of a complete file between two filestores in different open systems. File access enables reading, writing or deletion of selected parts of a remote file. File/filestore

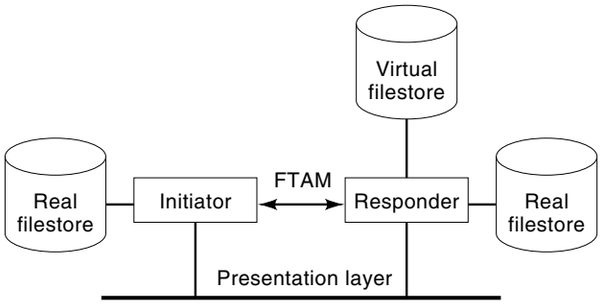

Figure 5. The FTAM Functional Model.

management refers to the management of a remote file/filestore. The FTAM standard has found popularity in the banking industry.

Figure 5 shows the FTAM functional model. Initiators and responders correspond to file clients and file servers, respectively. A virtual file/filestore is the abstraction of a real file/filestore presented by the responder to an initiator. Before the initiator can read, write, access or manage a virtual file/filestore, it must establish an FTAM dialogue with the responder.

A virtual file is characterized by a document type which specifies the structure and content of a file. In the FTAM information model, the file structure is a hierarchical access structure. Each node of the structure may contain structural information (such as its name and level) and content information called *data unit* (DU). File access is directed to a *file access data unit* (FADU) which is a subtree of the structure. As a special case of a hierarchical access structure, the *unstructured file structure* is a one-level structure consisting of only one node, and the *flat file structure* is a two-level structure in which the nodes are either named (*ordered flat structure*) or unnamed (*sequential flat structure*).

A virtual file is described by two classes of attributes: file attributes and activity attributes. The values of a *file attribute* are supposed to remain constant throughout the lifetime of a file unless specifically modified. For example, file name, date/time of creation, document type and access control are file attributes. An *activity attribute* describes a file relative to a particular FTAM dialogue in progress. It is dynamic in nature and has no meaning outside the dialogue. For example, current file position, current access request and initiator identity are activity attributes.

The FTAM standard introduces many service elements. FTAM functional units are used to group the functional related FTAM service elements. Some examples of FTAM functional units are given in the following.

- Kernel: This supports the basic functions for establishing and releasing an FTAM association, selecting and deselecting of a file for further processing.
- Read: This supports data transfer from a responder to an initiator.
- Write: This supports data transfer from an initiator to a responder.
- File access: This is used to locate a specific FADU for subsequent file operations.
- Limited file management: This supports file management operations such as creating and deleting files and interrogating the attributes of files.
- FADU locking: This supports concurrency control on either the file basis or the FADU basis.
- Recovery: This allows an initiator to perform recovery actions when a failure occurs after a file is opened.
- Restart: This allows a data transfer to be interrupted and restarted at some checkpoint.

Remote Database Access

Today, multivendor database interoperability is limited. A database client can only access a limited set of remote database servers. The remote database access (RDA) standard enables multivendor database interoperability. The RDA is premised on a client-server relationship between communicating end systems (Fig. 6). The RDA client is a process desiring the remote database access service supported by RDA. The RDA server is a process running on a remote end system that provides the RDA-based remote database access service to RDA clients.

The RDA standard is composed of a Generic Standard (ISO/IEC 9579-1), which defines the common aspects of the RDA protocol independent of specific database models and query languages, and a Specialization Standard, which specifies the part of the RDA protocol specific to a particular database model and query language. The SQL (Structured Query Language) Specialization is specified in ISO/IEC 9579-2.

The RDA standard defines two modes of operation—basic and transaction processing. The RDA basic mode is intended to be used for basic tasks such as data retrievals from a centralized database. The transaction processing mode is intended to be used for complex actions such as updates to a distributed database.

The RDA standard defines services for managing an RDA dialogue, managing a transaction, controlling outstanding operations, controlling the availability of database resources, and executing database language commands. For example, the Database Language group enables an RDA client to execute database language commands which can be either executed as soon as it is issued, or defined first and then later invoked. The latter mechanism would typically be employed for efficiency purposes in the case where the same command is used repeatedly within a given RDA dialogue.

The RDA standard describes procedures for handling errors and recovering from failures occurring during an RDA dialogue. It specifies how an RDA server must react to the failure of an RDA dialogue. When an RDA dialogue failure occurs, the RDA server deletes all state information for that

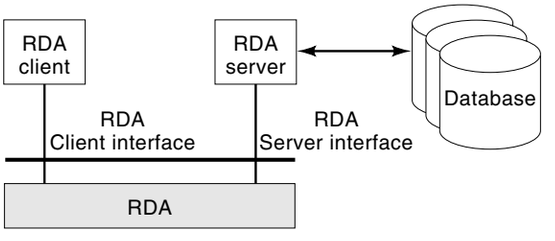

Figure 6. The RDA Functional Model.

dialogue and rolls back all uncommitted transactions. The actions of the RDA client following an RDA dialogue failure are not addressed in the RDA standard.

The RDA standard has been found useful in manufacturing and telecommunications environments. In the telecommunications environment, for instance, the Revenue Accounting Office can act as an RDA client to retrieve telecommunications usage information from a remote RDA server before applying a rate algorithm to the retrieved usage information.

OSI CONCEPTS

Layering

The basic structuring technique used by the OSI Reference Model is layering. Layering divides the overall communication functions of an open system into a succession of smaller subsystems. Subsystems of the same rank (N) form the *(N)-layer* of the OSI reference model. Objects in the (N)-layer are called *(N)-entities.* Collectively, they provide the *(N)-service* that is specified in terms of *service elements.* The (N)-service is always an enhancement of the ($N-1$)-service. Before an ($N+1$)-entity acquires any service from the (N)-layer, it must be bound to one or more *(N)-service access-points* (SAPs). At any time, no more than one ($N+1$)-entity can be bound to the same (N)-SAP. The service at an (N)-SAP is supported by a unique (N)-entity.

Communication Modes

OSI offers three communication modes: connection-oriented; connectionless; and store-and-forward. *Connection-oriented communication* requires an *(N+1)-association* to be set up between the two communicating ($N+1$)-entities. The establishment of the ($N+1$)-association in turn requires an *(N)-connection* to be set up between two (N)-SAPs, i.e., the (N)-SAPs to which the ($N+1$)-entities are bound.

The lifetime of a connection has three distinct phases: connection establishment; data transfer; and connection release. Once the connection is established, a *connection endpoint identifier* is assigned at each end to its local service user. During the data transfer phase, requests to data transfer phase are logically related within the context of the connection addressed by the two connection endpoint identifiers.

The establishment of an (N)-connection requires the availability of an ($N-1$)-connection. This means that, in the worst case, when none of the lower layers have an established connection, a connection request from a higher layer would trigger connection establishments from the lower layers. As a result, connection establishment can be time consuming. There are at least two methods of optimizing the cost incurred at connection establishment. One method is to assign reasonably permanent connections at the layers where the cost is low, regardless of whether a higher-layer connection request has been issued. The other method, embedding, attempts to establish multiple connections simultaneously. Embedding is used for connection establishment in Layers 5 and 6.

The release of an (N)-connection is initiated by either an ($N+1$)-entity associated with the connection or the (N)-layer supporting the connection. There are orderly release, negotiated release and destructive release. In an orderly release, there is no loss in transit for data sent before the release request. In a negotiated release, which is a special case of orderly release, an ($N+1$)-entity can reject a release service request issued by its peer. In a destructive release, the release of an (N)-connection may disrupt the procedures of service requests issued earlier, implying potential loss of data in transit in both directions.

In the connection-oriented mode, the mapping of ($N+1$)-connections to (N)-connections can be one-to-one, many-to-one, or one-to-many; *(N+1)-multiplexing* means that more than one ($N+1$)-connection is mapped to the same (N)-connection; *(N+1)-splitting* means that an ($N+1$)-connection is split into several (N)-connections, e.g., when the bandwidth of an (N)-connection is less than that of the ($N+1$)-connection.

In *connectionless communication,* there is neither connection establishment nor connection release. The transmitted data units contain all the control information (such as addresses and quality of service) necessary for the transfer. Furthermore, they are transmitted independently of each other, meaning that there is no defined context. Data unit independence has the advantage that the data transfer service can be robust.

Store-and-forward communication is a mixture of connection-oriented communication and connectionless communication. A connection between the two communicating end systems is not required, although connections are established with an intermediate system on a hop-by-hop basis. Communication between communicating UAs in an X.400 system uses this mode.

Data Units

While providing the (N)-service, the cooperating (N)-entities exchange *(N)-protocol-data-units* (PDUs) with each other. An (N)-PDU has two components—data and control. The control component, which is known as the *(N)-protocol control information* (N-PCI), contains control information such as name/version of the (N)-protocol, type of the (N)-PDU, and addresses of the communicating ($N+1$)-entities. The data component, which is known as the *(N)-service-data-unit* (SDU), contains user information which is meant to be interpreted by the receiving (N)-entity.

When an (N)-entity sends an (N)-PDU to its peer, the (N)-PDU is first transported to the lower layers in the source system through encapsulation. Each time the PDU is passed to the layer below, the layer below adds a PCI prefix to the PDU. Thus, when the PDU reaches the lowest layer (i.e., the Physical Layer), it has been encapsulated with one or more PCIs. Next, the encapsulated PDU at the Physical Layer is transmitted across one or more transmission media to the Physical Layer of the target system. In the target system, the encapsulated PDU is transported to the receiving (N)-entity through decapsulation. Each time the encapsulated PDU is passed to the layer above, the layer above strips off a PCI from the PDU. A layer in the target system always strips off the PCI added earlier by the same layer in the source system. When the receiving (N)-entity receives the PDU, all the PCIs which were added by the source system should have been removed.

While (N)-PDUs are messages passed between two open systems, *(N)-interface-data-units* (IDUs) are messages passed between two adjacent layers in the same open system, i.e., the ($N+1$)-layer and the (N)-layer. An (N)-IDU is passed either because of a local management need or because a ($N+1$)-

entity wants to send an (N+1)-PDU to its peer. In the latter case, the (N)-IDU is constructed with two components—an (N+1)-PDU (which is treated as an (N)-SDU by the (N)-layer) and an *(N)-interface-control-information* (ICI). The (N)-ICI contains control information (e.g., address of the sending (N+1)-entity), which is to be interpreted by the (N)-layer in the source system. In some cases, the sending (N+1)-entity may compose more than one (N)-IDU in order to send an (N+1)-PDU to its peer. This happens when the (N)-layer imposes a size constraint on an (N)-IDU. Since (N)-IDUs are messages passed between adjacent layers in an open system, the design of their structures is a local implementation issue. OSI standards, which are only concerned with interconnection matters, do not define (N)-IDUs.

When the (N)-layer receives an (N)-IDU, it will separate the (N)-ICI from the (N)-SDU. From the control information in the (N)-ICI, an (N)-PCI is built which is then concatenated with the (N)-SDU to form an (N)-PDU (Fig. 7). The (N)-PDU is subsequently passed to the (N−1)-layer through the use of one or more (N−1)-IDUs.

Naming and Addressing

The OSI environment is rich in objects. Some OSI objects require a global identification so that they can be referenced unambiguously by applications. They include abstract/transfer syntax, application contexts, FTAM document types and managed object classes.

Attribute-based names and object-identifier-based names are found in the OSI environment. An attribute-based name is made up of a set of naming attributes, where each attribute can be represented by a pair, such as (STATE, Missouri). For example, X.500 names are attribute-based names.

Object-identifier-based names are given by object identifiers. To explain object identifiers, the object identifier tree (OIT) is described, which is defined to provide global naming. In the OIT, leaf nodes represent objects or object classes, and nonleaf nodes represent administrative authorities. Each arc of the OIT is labeled by an integer and occasionally a mnemonic name for descriptive purpose. An object identifier is an ordered sequence of integer labels for the unique path from the root to a node in the OIT.

Let us examine the arcs of the OIT in more detail. The OIT begins with three numbered arcs emanating from the root:

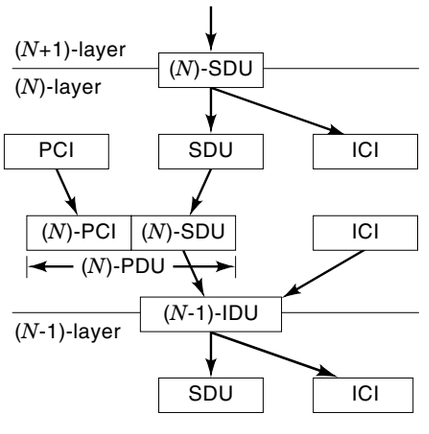

Figure 7. Building an (N)-PDU and an (N−1)-IDU.

arc 0 for ITU; arc 1 for ISO; and arc 2 for joint ISO-ITU. Below ITU there are arcs leading to recommendations (0), questions (1), administrations (2), and network operators (3). Below ISO there are arcs leading to standard (0), registration-authority (1), member-body (2) and identified-organization (3). The arcs below standard (0) shall have the number of an International Standard. Consider the FTAM standard, i.e., ISO/IEC 8571. An arc with the arc number 8571 can be created for the FTAM standard. In this way, the object identifier, {1(ISO) 0(standard) 8571}, can be used to name the FTAM standard.

An address is used to locate an object, e.g., an (N+1)-entity. Suppose that an (N+1)-entity is bound to one or more (N)-SAPs. Any one of these (N)-SAPs can be used to locate the (N+1)-entity. Hence, an address of the (N+1)-entity can be given by a name identifying the set of (N)-SAPs to which the (N+1)-entity is bound. Such an address is called an *(N)-address,* or an *(N)-SAP address* when there is only one (N)-SAP in the set.

To locate an (N+1)-entity, we need lower-layer addressing information to identify a path from the lower layers all the way up to the target (N+1)-entity. An (N)-address (for N greater than 3) has two components, i.e., an (N−1)-address of a supporting (N)-entity, and an (N)-suffix, known as an (N)-selector. The (N)-selector is used to identify the set of (N)-SAPs to which the (N+1)-entity is bound. Accordingly, a presentation (i.e., Layer 6) address is given by a session address and a presentation selector, a session address is given by a transport address and a session selector, and a transport address is given by a network address and a transport selector. In short, a presentation address can be represented by a quadruple consisting of a presentation selector, a session selector, a transport selector and a network address.

OSI LAYERS

Physical Layer

The purpose of the physical layer is to hide the nature of the physical media from the data link layer in order to maximize the transportability of higher-layer protocols. It provides mechanical, electrical, functional, and procedural means to activate, maintain, and deactivate physical connections for serial bit streams between data link entities. The common functions found in the physical layer are synchronization and multiplexing. The physical layer standards should be distinguished from the physical interface standards (e.g., X.21) which define the boundary or interface between the physical layer and the physical transmission medium.

Data Link Layer

The data link layer is responsible for error-free data transmission over a data link. It creates data packets, synchronizes the data packets, detects and corrects errors, and controls the flow of the packet stream.

The data link service definition of the ISO high-level data link control (HDLC) protocol covers both connection mode operation and connectionless mode operation. While HDLC is seen as a superset of data link procedures, many interesting subsets are defined out of it. For example, the HDLC LAP B subset is adopted by ITU as part of the X.25 packet-switched

network standard; HDLC LAP (Link Access Procedure) D, a data link standard developed as part of the ISDN standardization, is a subset of LAP B.

Network Layer

The Network Layer provides the service for network service users to exchange information without being concerned with the topology of the network and the characteristics in each constituent subnetwork. It is perhaps the most complex of the seven OSI layers due to the fact that many existing subnetwork types use different network addressing schemes, network protocols and communication modes.

There are two modes of network service—*connection-oriented network service* (CONS) and *connectionless network service* (CLNS). The specification of CONS has six service elements: N-CONNECT to set up a network connections; N-DATA to send normal user data; N-DATA-ACKNOWLEDGE to acknowledge the receipt of normal user data; N-EXPEDITED to send expedited data; N-RESET to reset a network connection; and N-DISCONNECT to release a network connection. In CLNS, there is only one service element, i.e., N-UNIT-DATA, which is used to send data.

In the Network Layer, there are two kinds of network protocols—routing protocols and interconnection protocols. Routing protocols are used to maintain a routing information base, to collect routing information, to distribute routing information to other nodes and to calculate the metrics of a route. Interconnection protocols are used to support the integration of subnetworks of different types.

A routing framework should be in place before routing protocols are introduced. ISO/IEC TR (Technical Recommendation) 9575 defines a routing framework, essentially partitioning the global network into administrative domains and routing domains. An administrative domain is an autonomous set of ISs and ESs running under a single administration. Within an administrative domain, there may be one or more routing domains. Each routing domain runs the same IS-IS routing protocol among the ISs.

Of all the OSI routing protocols, only the IS-IS routing protocol used within a routing domain is discussed here. This protocol has heavily influenced the design of the Open Shortest Path First protocol which is an Internet routing protocol. In the past, IS-IS routing protocols were primarily based on the vector state algorithm, requiring an IS to periodically send its routing table to its adjacent neighbors. Because the routing information propagates from one link to another, it may take considerable time before a remote IS can receive the update. Thus, the major drawback of the vector state algorithm is slow convergence, resulting in the receipt of an occasional stale update. The link state algorithm, which is adopted by ISO for the IS-IS routing protocol, requires each IS to maintain a complete topology map. Instead of sending a global routing table to its adjacent neighbors, an IS periodically broadcasts the status of its adjacent links. On receipt of link information from other ISs, an IS can build an up-to-date topology map which is modeled as a weighted graph. Using this graph, the IS can apply Dijkstra's Shortest Path First (SPF) algorithm to compute the shortest distance to a destination. The SPF algorithm uses the well-known minimum spanning tree algorithm to convert a weighted graph into a tree. Because the link state algorithm requires each IS to broadcast its link state status, every IS can maintain a consistent view of the entire topology. The slow convergence drawback of the vector state algorithm is precluded.

Interconnection protocols are used to interconnect subnetworks, which may use different network address schemes, network protocols and communication modes. The current solution is to use interworking units (IWU) which perform relaying. Relaying involves the use of convergence protocols which can adapt non-OSI network protocols to OSI protocols. ISO/IEC 8648 defines a framework of the Network Layer for the introduction of interconnection protocols. In the Internal Organization of Network Layer (IONL) model (Fig. 8), there are three sublayers of the Network Layer.

- Subnetwork access sublayer. This sublayer provides the attachment point of a subnetwork. A *subnetwork access protocol* (SNAcP) operating at this sublayer is a protocol associated with an underlying subnetwork. This protocol may or may not conform to the OSI network service requirement.
- Subnetwork dependent sublayer. This sublayer is responsible for augmenting the service offered by a subnetwork technology into something close to the OSI network service. A *subnetwork-dependent convergence protocol* (SNDCP) is used for this sublayer. The operation of an SNDCP depends on the network service of a particular subnetwork. A common function of an SNDCP is to map between OSI network addresses and addresses specific to the subnetwork.
- Subnetwork independent sublayer. This sublayer provides the OSI network service over a well-defined set of underlying capabilities, which need not be based on the characteristics of any particular subnetwork. When a *subnetwork-independent convergence protocol* (SNICP) is used, it is defined to require a minimal set of services from a subnetwork. *Connectionless Network Protocol* (CLNP), which is the OSI interconnection protocol providing CLNS, is an example of an SNICP.

Using the IONL model, there are three basic strategies to interconnect subnetworks.

- Interconnection of subnetworks which support OSI network services. In this strategy, all the subnetworks involved fully support the OSI network service. There is no need for an enhancement protocol.
- Hop-by-hop enhancement. This approach is used in an environment containing at least one subnetwork type,

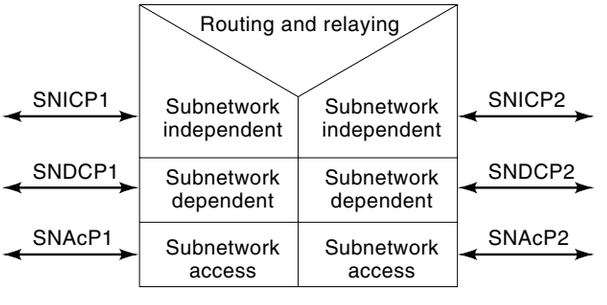

Figure 8. The IONL Model.

which does not provide the OSI network service. It takes each of these subnetworks individually and enhances its subnetwork service to the level of the OSI network service. Different SNDCPs may be required on different subnetworks.

- Internet approach. This approach is used in an environment containing at least one subnetwork type which does not provide the OSI network service. The SNICP (e.g., CLNP), which assumes the minimal set of network services from the underlying subnetworks, would operate on top of the SNAcP in all the systems attached to the subnetworks.

The last topic in this section is OSI network addressing. The Network Layer must provide a global addressing scheme so that ESs in different subnetworks can be addressed unambiguously. An NSAP address is an address of an SAP of the Network Layer. To cope with the multitude of NSAP addresses in the global environment, NSAP addresses are partitioned into network addressing domains in a hierarchical fashion. Each network addressing domain has its own network addressing format and is administered by an address registration authority. An address registration authority may further suballocate its addressing space to another address registration authority. On the whole, the network addressing domains are structured as a tree where the root has seven top-level network addressing domains (e.g., X.121, E.164 and ISO 6523-International Code Designator) as children.

An NSAP address consists of an initial domain part (IDP) and a domain specific part (DSP). The IDP, in turn, consists of an authority and format identifier (AFI) and an initial domain identifier (IDI). The AFI specifies one of the seven top-level addressing domains as well as the abstract syntax of the DSP (e.g., binary octets, decimal digits, characters). For example, an AFI value of 47 implies that the format is ISO 6523-International Code Designator (ICD) and the DSP abstract syntax is binary. The IDI component is used to specify an addressing registration authority, e.g., US Government OSI Profile (0005). The DSP is the part of an NSAP address assigned by an addressing registration authority specified in the IDI component. This is the place where one can put information on the routing domain and an ES.

Transport Layer

The transmission quality of the network service may not meet the requirement of an application because of possible signaled and residual errors. By performing transmission quality enhancements such as error detection and recovery, the transport layer (ISO/IEC 8072/8073) provides a reliable end-to-end service.

There are two modes of transport service: connection-oriented transport service (COTS) and connectionless transport service (CLTS). When operating in COTS, the transport layer provides a full-duplex transmission between the communicating transport service users. The following discussion focuses on COTS.

The transport functions invoked by the transport layer depend on the underlying network type. If the underlying network is reliable, only a simple transport protocol is needed. On the other hand, if the underlying network is unreliable, a sophisticated transport protocol involving elaborate transport mechanisms is needed. For this reason, the following five connection-oriented transport protocols have been defined.

- TP 0: This is designed to operate over a reliable network. It is a simple protocol.
- TP 1: This is designed to operate over an X.25-like network, which may have an unacceptable signaled error rate. It is capable of resynchronization upon reset of network signaled failures. It is also able to reassign a new transport connection in the event of a network failure.
- TP 2: Similar to TP 0, this is designed to operate over a reliable network. It adds the multiplexing capability to TP 0 so that multiple transport connections can share a single network connection.
- TP 3: Designed to operate over an X.25-like network, this is basically a combination of TP 1 and TP 2.
- TP 4: This is designed to operate over an unreliable network. It is the most sophisticated transport protocol.

The TP 4 protocol procedures are complicated. To set up a transport connection, three protocol messages are needed to avoid the processing of connection establishment PDUs for transport connections which have been released. Despite the complexity of the transport protocols, the specification of COTS is straightforward. There are four service elements: T-CONNECT to set up a full duplex transport connection; T-DATA and T-EXPEDITED-DATA to deliver normal data and expedited data, respectively, and T-DISCONNECT to release a transport connection. The T-DISCONNECT service is disruptive because data sent before the service request may be lost.

Session Layer

The transport layer provides a full duplex and unstructured pipe between two communicating application processes. In many cases, this pipe is sufficient. In other cases where bulk data transfer is necessary, it is desirable to add structure to the transport pipe so that the application processes can synchronize the data stream and perform recovery after a failure. The session layer (ISO/IEC 8326/8327) enhances the services of the transport layer by enabling application processes to synchronize their dialogues and to manage their data transfer.

There are two methods to structure the transport pipe as a *session dialogue.* One uses the notion of an activity and the other does not. In the latter method, major synchronization points are inserted into the transport pipe to subdivide the pipe into *dialogue units.* A major synchronization point marks the end of a dialogue unit and no recovery is permitted back to that dialogue unit. Within each direction of a dialogue unit, minor synchronization points can be also added to facilitate the process of recovery. Whenever resynchronization is needed during recovery, a session service user can resynchronize the dialogue to the previous confirmed minor synchronization point within the current dialogue unit.

The other method of organizing a session dialogue involves the use of activities. Conceptually, an activity represents a logical piece of work such as a file. It can be dynamically activated, interrupted, resumed or even discarded, thereby supporting parallel tasking and data recovery. The use of activities gives a three-level structure to a session dialogue. At the

topmost level, the dialogue is structured into activities. At the second level, each activity is structured into dialogue units. At the third level, each dialogue unit is structured using minor synchronization points. Note that an activity is a logical concept. As such, an activity may span several session connections, and several overlapping activities may be contained in a session connection.

A session connection has four token attributes: data token; release token; minor synchronize token; and major synchronize/activity token. Upon negotiation, these tokens are assigned to one of the two session service users during either the connection establishment phase or the data transfer phase. The owner of the data token can send data to its peer during a half-duplex data transfer. The use of the release token allows a session service user to refuse the release of a session connection (e.g., if it has data to send) when its peer, the owner of the release token, requests the release. The owner of the minor synchronize token (e.g., the sender of a file) can insert minor synchronization points. Finally, the owner of the major synchronize/activity token can insert major synchronize points to mark the beginning of a dialogue unit or an activity.

The S-CONNECT service element is used to establish a session connection. When the session connection is established, either an existing transport connection is used or a new transport connection is established. When the session connection is released, the associated transport connection does not need to be released and thus can be saved in a reservation pool for future session connections.

The session service elements available for the data transfer phase are used for dialogue control, synchronization and resynchronization. The full-duplex mode, which permits both session service users to send data simultaneously, does not require dialogue control and hence no data token is needed. The half-duplex mode, which enables only one session service user at a time to send data, requires dialogue control and hence the use of the token management capability. There are three session service elements for token management: S-TOKEN-GIVE to surrender a token; S-CONTROL-GIVE to surrender the entire set of available tokens; and S-TOKEN-PLEASE to request a peer to relinquish the ownership of one or more tokens.

The transport layer provides only two types of data transfer facilities compared to the session layer, which provides four types of data transfer facilities. Normal data are sent using the S-DATA service element. When the data token is available, only the owner can invoke S-DATA. Expedited data are sent using the S-EXPEDITED-DATA service element. Due to limitations of the underlying expedited transport data facility, a maximum of 14 octets of expedited data can be transferred. The S-TYPED-DATA service element allows a session service user to send data outside the normal data stream independent of the availability and the assignment of the data token. When used properly, this facility gives the session service users a mixed half/full duplex mode which is useful in many applications. The S-CAPABILITY-DATA service element is used to send limited data in between two activities. At the end of an activity, for example, the session service users can use capability data to decide which activity to start next.

Synchronized data transfer is intended for a bulky data transfer to facilitate error or crash recovery. It is achieved by inserting minor or major synchronization points. The owner of the minor synchronization token can use the S-SYN-MINOR service element to insert a minor synchronization point. In practice, the two session service users have agreed on a window size during initialization, with the understanding that the receiver should acknowledge all the previously unconfirmed minor synchronization points before the two users exceed the window size. Unlike S-SYN-MINOR, S-SYNC-MAJOR is a confirmed service element. Once a session service user invokes a S-SYNC-MAJOR request, a confirmation must be received. The receipt of a confirmation acknowledges not only the major synchronization point but also all previously unconfirmed minor synchronization points. It marks the end of the current dialogue unit; thus the sender can discard all the data associated with the dialogue unit.

Resynchronization is normally triggered by a notification which signals a possible failure. The notification is initiated by either a session service user or a session entity. The S-U-EXCEPTION-REPORT service element is used by a session service user to report a user error (e.g., failure to hand over the data token) to its peer. The S-P-EXCEPTION-REPORT service element is used by a sesson entity to indicate an internal error (e.g., session protocol error) to the session service users. Typically, following an S-U-EXCEPTION-REPORT or S-P-EXCEPTION-REPORT indication, a session service user would initiate resynchronization by invoking the S-RESYNCHRONIZE service element. During the request, it must specify the resynchronize type, such as abandoning the current dialogue unit or resynchronizing the session to an unacknowledged checkpoint within the current dialogue unit. If necessary, the available tokens may be reassigned.

Five session service elements are available for the management of activities. They are: S-ACTIVITY-START to initiate a new activity; S-ACTIVITY-INTERRUPT to interrupt an activity; S-ACTIVITY-RESUME to resume an activity; S-ACTIVITY-DISCARD to discard an activity; and S-ACTIVITY-END to end an activity.

Two kinds of orderly release are made available to session service users—negotiated release and nonnegotiated release. The distinction between the two forms of release is based on the use of the release token. If this token is not available, the release cannot be negotiated. If the release token is available and the owner invokes the S-RELEASE service, the owner's peer may choose to reply negatively to the release request when data has to be sent. For destructive release, the Session Layer provides an abortive service which can be either user-initiated (S-U-ABORT) or provider-initiated (S-P-ABORT).

Presentation Layer

The presentation layer (ISO/IEC 8822/8823) provides a pass-through capability that makes the entire set of session services visible to application processes as presentation services. In addition, it is responsible for handling the representation of application information which is exchanged for the communication. While the representation of information in an end system is a local issue, the two communicating application processes must agree upon what type of information is exchanged and how such information is represented during the transfer. During presentation connection establishment, the two application processes must identify the abstract syntax for the information to be exchanged for the connection. An

abstract syntax can be represented by one or more transfer syntax. Therefore, the two application processes must also agree upon a common transfer syntax for every abstract syntax. Once the abstract syntax and the corresponding transfer syntax have been established, mapping between local syntax (i.e., syntax used for the local representation of information) and transfer syntax can be initiated.

A presentation context is a pair consisting of abstract syntax and transfer syntax. The objective of the presentation layer is to establish a set of presentation contexts for the communication. This set is known as the defined context set (DCS). Each presentation context in the DCS is named by an integer-valued presentation context identifier (PCI). The DCS can be the empty set if the two application processes have previously agreed upon a default context.

A presentation data value, which is passed to the presentation layer by an application process, may be composed of values from one or more abstract syntax. There are two ways to encode a presentation data value by the presentation layer—full encoding and simple encoding. Full encoding encodes a presentation data value as a presentation data value (PDV) list. Each component in the list is a PCI followed by a value encoded using the appropriate transfer syntax. The PCI value is always encoded using basic encoding rules (BER), which is one of the transfer syntax defined by ISO 8824. Simple encoding is used when the DCS is empty or when the DCS contains only one presentation context. In the simple encoding, the presentaiton data value is given simply by the encoded value; the PCIs are missing since they are not necessary.

As far as the presentation services are concerned, the presentation layer makes the session services directly accessible by the application processes; it offers only a few services of its own. In fact, it offers only one service element which is not related to any session service element. This service element, P-ALTER-CONTEXT, provides the capability for the presentation service users to modify the DCS such as adding/deleting a presentation context. For example, an FTAM initiator, which may not know the abstract syntax of a file that it wants to open when the FTAM dialogue is established, can use this service element to add a presentation context for the abstract syntax of the file once it is known.

A presentation connection is established using the P-CONNECT service element. The following explains how the initial DCS is established during connection establishment. When a presentation service user makes a P-CONNECT request, it passes the presentation context definition parameter. This parameter specifies a partially filled DCS where each item in the list contains two components—PCI and name of an abstract syntax. On receiving the P-CONNECT request, the local presentation entity first determines which transfer syntax it can use to represent each abstract syntax. It then creates a presentation context definition result list, where each abstract syntax is mapped to the set of transfer syntax, which the local presentation entity can support. The presentation context definition result list is passed to the peer presentation entity by means of a presentation CP (Connect Presentation) PDU. The peer presentation entity passes the presentation context definition result list to the called presentation service user. If the called presentation service user accepts the request, a possibly modified result list is returned. On receiving the response, the local presentation entity has a chance to modify the presentation context definition result list before it returns the modified result list to the presentation entity of the initiating presentation service user. At this stage, the initial DCS is established.

The remaining presentation service elements are almost identical to the session service elements. Since the presentation layer reproduces the services of the session layer in a pass-through manner, it is more efficient to implement the two layers in a single implementation module. By sharing the global data structures defined for the two layers, expensive copying can be avoided.

For the rest of this section, examples are given as a brief introduction of abstract syntax notation one (ASN.1), and BER. ASN.1 is similar to the data declaration part of a high-level programming language. It provides language constructs to define types and values. Types correspond to structures and values correspond to content. Unlike any programming language, ASN.1 types, which are meant to be machine-independent, need not be implemented by any machine. For example, the ASN.1 INTEGER type allows all integers as values.

An abstract syntax is a named group of ASN.1 types and values. It can be defined by a standard group, a profile group or a user group. One of the reasons why the types are grouped into an abstract syntax is that values of these types are meant to be encoded by the same transfer syntax. Thus an abstract type can be viewed as a unit for transfer encoding. An ASN.1 module is the ASN.1 notation to define an abstract syntax:

```
ModuleExample DEFINITIONS ::=
    BEGIN
        TypeA ::= INTEGER
        TypeB ::= BOOLEAN
        valueA TYPEA ::= 10
        valueB TypeB ::= TRUE
    END
```

The foregoing module is named by Module Example. It has two ASN.1 types and two values.

An ASN.1 type is either simple or structured. Simple ASN.1 types include INTEGER, REAL, BOOLEAN, CHARACTER STRING, BIT STRING, OCTET STRING, NULL and OBJECT IDENTIFIER. Structured types are built from simple types:

```
Person ::= SEQUENCE{
    name        IA5String(SIZE(0..64),
    phone       IA5String(SIZE(0..64))OPTIONAL,
    email       SET OF IA5String OPTIONAL}
```

The ASN.1 type in the foregoing can be used to represent a person. The following observation about the type is made.

- The keyword OPTIONAL means that the corresponding component of the sequence can be omitted when a value is specified.
- The IA5String type is a CHARACTER STRING type consisting of characters taken from IA5 (International Alphabet number 5).
- IA5String(SIZE(0.64)) is a subtype of IA5String where the allowed strings have a maximum size of 64.
- SET OF is a structured type to represent an unordered list.

ISO has defined a number of transfer syntax, including Basic Encoding Rules, Distinguishing Encoding Rules and Packed Encoding Rules. The following gives a brief introduction of BER which is by far the most popular transfer syntax. Every BER encoded value has three fields: a tag (identifier) field that conveys information on the type and the encoding form; a length field that defines the size of the value in octets; and a content field that conveys the actual value. A BER encoded value is sometimes called a Type-Length-Value (TLV) triple. Each ASN.1 type has an associated encoded tag which is used for the tag field in a TLV triple.

The following gives an instance of a SEQUENCE type and its BER encoding:

```
Constructed ::= SEQUENCE{
      name        OCTET STRING
      place       INTEGER {room1(0), room2(1), room3(2)}
      persons     INTEGER OPTIONAL   }
meeting Constructed ::=
   {  name        '1AA2FFGH',
      place       room3}
```

The TLV encoding (in hex) of meeting, which is constructed, is

```
30 09
   04 04 1A A2 FF GH
   02 01 02
```

The following explains how the encoded value is derived:

- The 30 in the first row is the encoded tag value for SEQUENCE.
- The 09 in the first row means that the length of the value field is 9 octets.
- The second row gives the encoding of the octet string "1AA2EFFGH" where the encoded tag for octet string is 04.
- The third row gives the encoding of the integer 2 (which is an abbreviation for room3).

Application Layer

The application layer provides all the communication support to application processes. Hence, if the lower six layers do not provide the required communication support, the application layer has to provide it. A framework to build the objects in the application layer is needed. The application layer structure standard (ISO/IEC 9545) defines a framework around which application standards can be developed.

Conceptually, an application process can be divided into communication objects and noncommunication objects. The communication objects (i.e., objects which provide communication capabilities to the application process), are called *application entities* (AE). An application process may have one or more AEs. For example, a business application may consist of an AE containing X.400 capabilities and an AE containing FTAM capabilities. The division is only conceptual, so an actual implementation of an application process may not follow such a division.

The structure of an AE can be complex. To understand how one AE communicates with another AE, it is necessary to refine the AE into granular components and analyze how these components communicate with their peers. In this way, the design of an application protocol between two communicating AEs can be reduced to the design of an application protocol between two communicating components of less complexity.

The application layer structure standard proposes structuring an AE in a recursive manner, starting with atomic components called *application service elements* (ASE). One or more ASEs can be combined to form an *application service object* (ASO). An ASO can be combined with one or more ASEs or ASOs to form another ASO. Continuing this recursively, the outermost ASO, which is the AE, is derived.

Every ASO contains a *control function* (CF). The CF acts as a traffic cop to coordinate the activities of the ASEs and ASOs within the outermost ASO. In particular, the CF unifies the services of the various components of an ASO. The CF may add temporal constraints on the use of the combined service.

Before two AEs can communicate with each other, they must first establish an application association which is an association between two AE-invocations (i.e., invocations of an AE). An application context, which is the most important attribute of an application association, defines the rules to be enforced during the lifetime of the application association. In particular, it specifies the required ASOs and ASEs, the abstract syntax that may be referenced, and the binding/unbinding information that needs to be exchanged before an application association is established/released. In short, an application context defines the working environment or knowledge that is shared by the AEs for the duration of the application association. The ASEs and ASOs can be viewed as workers at work in the constraint of the application context. Each worker communicates with a peer worker (of the same type) using a specialized protocol (e.g., an application protocol of an ASE).

The decomposition of an AE into ASOs and ASEs is only static. It does not mean that all the ASOs and ASEs in an AE are always involved in an AE-invocation. For example, an AE may have five ASEs and three ASOs, but a particular AE-invocation may only involve three ASEs and two ASOs in an application association.

To understand the dynamic behavior of an AE, one should examine the structure of an AE-invocation. By interacting with its peers, an AE-invocation can be involved in multiple application associations. Conceivably, the application contexts for these application associations may differ from each other. Therefore, one can refine an AE-invocation into components, with one component for each application association. These component objects are called *single association objects* (SAOs). They are active objects since they maintain states. Every SAO contains a *single application control function* (SACF) which acts as a coordinator of the ASEs and the ASOs which are involved in an application association.

When the AE-invocation contains several SAOs, there may be a need for a *multiple association control function* (MACF) which is used to coordinate the SAOs. An MACF is similar to an executive manager. In some cases, an MACF is not needed because the SAOs do not need coordination.

The structure of the application layer can reduce the design of an application protocol between two AEs to that of an application protocol between two ASEs. Since ASEs are the basic building blocks of an AE, we should first standardize the common ASEs and the associated application protocols. Common ASEs provide generic communication capabilities to a number of applications. Examples include the Application

Control Service Element (ACSE), the Remote Operation Service Element (ROSE) and the Reliable Transfer Service Element (RTSE). The use of common ASEs ensures that applications can be built in a consistent manner. In addition to the common ASEs, there are specific ASEs that provide specific capabilities to applications. The FTAM ASE defined in the FTAM standard is an example of a specific ASE.

The ACSE standard is defined for the purpose of establishing and releasing application associations. An application association is a presentation connection with additional application layer semantics, e.g., application context negotiation and peer-to-peer authentication. Currently, there is a one-to-one mapping between application associations and presentation connections. Future versions of the ACSE standard might permit a presentation connection to be reused for a new application association or multiple application associations to be interleaved onto a single presentation connection.

There are four ACSE service elements. The A-ASSOCIATE service element is used to establish an application association between two AE-invocations. The A-RELEASE service element is used to release an application association in an orderly manner. The A-ABORT is used by an AE-invocation to abort an application association with possible loss of transit data. The A-P-ABORT service element is used by the ACSE service provider to notify the abortion of an application association.

The ACSE service elements are mapped onto the presentation service elements in a straightforward manner. Because all the presentation parameters are supplied by the ACSE users, there are over 30 A-ASSOCIATE parameters. Additional A-ASSOCIATE parameters may be added in the future, whenever there is a need to provide additional semantics of an application association. Of the A-ASSOCIATE parameters which are application-specific (i.e., not specific to the Presentation Layer), only the application_context_name parameter which specifies the application context is mandatory.

In a typical interactive environment, an AE-invocation requests a remote AE-invocation to perform an operation. The remote AE-invocation executes the operation and returns either an outcome or an error. Because many distributed applications are written in this kind of interactive environment, it is useful to provide an ASE to provide such interactive communication support. The remote operation service element (ROSE) standard is written for this purpose. It is used by application protocols such as CMIP and DAP/DSP.

The ROSE standard defines a model for remote operations. A remote operation is requested by an invoker. The performer attempts to execute the operation and reports the outcome, which is either a normal outcome or an exception. Every invocation is identified by an invocation identifier, which is used to differentiate this invocation from other invocation(s) of the same operation. In addition, it may have a linked invocation identifier, indicating that the operation is part of a group of linked-operations formed by a parent-operation and one or more child-operations. The performer of the parent-operation may invoke zero or more child-operations to be performed by the invoker of the parent-operation.

There are five ROSE service elements. An invoker uses RO-INVOKE to request a remote operation to be performed. After execution, a positive result is returned using RO-RESULT while a negative result is returned using RO-ERROR. When an ROSE user detects a problem with the invocation, it can use RO-REJECT-U to reject the request or the reply. The ROSE service provider uses RO-REJECT-P to inform ROSE users of problems such as a badly structured application PDU.

ROSE is not meant to be used as a standalone ASE in an application context. In any application context using ROSE, there must be at least one or more ASEs which supply the remote operations for the application needs. ROSE only acts as a courier for such remote operations. To facilitate the specification of remote operations by ROSE users, the ROSE standard provides templates for the definition of remote operations.

CONCLUSIONS

The OSI Reference Model is a very carefully designed model that provides the framework for the development of protocols to interconnect open systems. By providing a rich set of communication functionalities, it meets all the conceivable interconnection requirements. A seven-layer implementation necessitates good software engineering techniques and sound understanding of OSI concepts. Independently manufactured OSI implementations exist and have proven interoperability. Most of them are based on a profile that meets the requirements of a specific application. A reduced profile, known as Minimal Open System Interconnection (MOSI), was proposed by the OSI Regional Workshop. This profile would meet the requirements of most of the networking applications that exist today.

Opponents of OSI believe that the functionalities of the OSI Reference Model are overkill. For instance, very few applications would require most of the session functions. Instead of requiring every open system to implement the session layer, the session functions could have been imbedded in only those applications that require them. This concern of these opponents has been addressed by the MOSI profile, the performance of which is comparable with that of the existing non-OSI stacks.

The presentation layer addresses the situation where a negotiation of transfer syntax is necessary. Such situations arise in wireless communications when there is a need for encryption and compression. The growth of personal communication system (PCS) will cause the appreciation of the incorporation of the presentation layer in the communication stack.

The success of the OSI Reference Model is clearly illustrated in the deployment of OSI application protocols such as CMIP, X.400 and X.500; CMIP, for example, has been unanimously chosen by telecommunications managers to be the management protocol in the lower layers of the TMN model.

The OSI Reference Model is appreciated by practitioners who have a need for the rich functions it provides. It is appreciated by protocol designers (such as the present author) who can learn good protocol design principles from reading the OSI standards. It is certainly a sound protocol model to guide the development of a protocol stack.

ADRIAN TANG
University of Missouri-Kansas City

IT INDUSTRY. See INFORMATION TECHNOLOGY INDUSTRY.