PART TWO

OPEN NETWORK ARCHITECTURES

3 CONCEPTS AND TERMINOLOGY OF OPEN SYSTEMS

Introduction

Throughout *Open Systems Networking: TCP/IP and OSI*, the terminology and concepts of OSI are used to describe open systems architectures. Although its terminology is certainly original, most of the basic concepts of OSI—the layered model, service definitions, hierarchical naming and addressing, internetworking, and subnetworks—are derivative, having been derived from principles that were established by other architectures, especially TCP/IP, and documented in different ways. The important contribution of OSI is not the concepts but the way in which they have been expressed in the form of a comprehensive "reference model" of open systems interconnection—the *Basic Reference Model of Open Systems Interconnection* (ISO/IEC 7498: 1993).[1] The use of the descriptive tools of the OSI architecture as the basis for describing the general open systems principles of layering, naming and addressing, protocol specification, and service definition throughout this book recognizes not that OSI as a whole is worthier than TCP/IP or other protocol suites but that the OSI architecture is widely known, and its terminology and concepts are readily accepted as the basis for architectural descriptions.

The terminology and specification of TCP/IP present a striking contrast to OSI. The ARPA researchers (at the time they built it, DARPA was actually ARPA) who built the first TCP/IP networks freely admit that they were not terribly concerned with defining an architectural model. Of course, over time, the urgency of formally documenting the

1. The new second edition of ISO/IEC 7498 will be published in 1993. It has already been published by CCITT as Recommendation X.200-1992.

TCP/IP architecture has waned; unlike OSI, TCP/IP is defined by the real networks (including the vast Internet) that implement it, not by the relatively few documents that describe it. Thus, although there are several landmark journal articles that describe the architecture of TCP/IP (Cohen and Postel 1983; Cerf and Cain 1983) and a retrospective RFC (RFC 871), much of what is known about the architecture of TCP/IP remains folklore.[2]

This chapter avoids attributing basic architectural principles to any particular open systems architecture, focusing instead on the way in which those general principles are expressed by OSI and by TCP/IP. It examines the pleonastic[3] terminology of OSI and compares it to the blue-collar language of TCP/IP and identifies the core set of terms and definitions that are used throughout this book. Readers will note that the authors owe no strict allegiance to either OSI or TCP/IP. Although OSI and TCP/IP terms are used when protocol-specific terminology is necessary, the authors use what they believe to be the best terminology from the entire field of networking for the general discussion of the characteristics and principles that are common to both protocol suites.

Architectures

An *architecture* is an abstract model of some part of the real world—in this case, a model of the organization and behavior of networks consisting of interconnected, communicating computer systems and applications. Because it is an abstraction, it is a useful device for describing concepts and relationships in a clear and concise fashion, without cluttering the description with references to the characteristics of specific systems or applications. The utility of such an architectural description depends on the power of the abstraction (how successfully the architecture ex-

2. In an electronic-mail exchange, Jon Postel assisted the authors' archaeological dig for TCP/IP architectural artifacts but warned us that "any writing about the ARPANET protocol architecture is after the fact (probably revisionist) history." The authors interpret Jon's comment not as an implication of TCP/IP writers in an Orwellian scheme to present a deliberately spin-controlled version of the architectural origins of TCP/IP but rather as a recognition that, for example, a paper written seven to ten years after the fact may perceive an "architectural principle" in what was really just good fortune or the result of a series of hits and misses. In fact, of course, Jon Postel *is* the TCP/IP architecture; any attempt to "improve" the documentation of TCP/IP by replacing Jon with a document, however well constructed and thorough, would be an enormous step backward.

3. At the risk of being accused of pedantry, the authors feel the word *pleonastic*, derived from *pleonasm*, or "the use of more words than are necessary to express an idea," is simply too accurate to eschew.

presses important concepts and relationships and suggests new ones) and on its relevance (how useful the architecture proves to be in the development of real systems and networks).

For designers and builders of networks and the distributed applications that use them, the development of an architectural description of the environment in which the many individual components of their designs and implementations will interact serves two essential purposes. First, it creates a global conceptual framework within which the relationships among individual components can be studied and explored at a common level of abstraction. This framework encourages broadly based solutions to problems, since it places each component in an abstract context that illuminates its interactions with other components. Second, it serves as the basis for formal descriptions of the characteristics of individual components, becoming a "global functional specification" for the distributed environment. Like any functional specification, it establishes a common reference point for the behavior of the designs and implementations that follow from it.

Open Systems

Open systems are "open" by virtue of their mutual adherence to one or more open systems standards, which specify those aspects of the behavior of an open system that are directly relevant to its ability to communicate with other open systems. There are many open system specifications and standards, each of which belongs to a particular architecture (e.g., OSI or TCP/IP). An architecture is typically described by a *reference model*, which expresses the organizing principles of the architecture (the reference points) and provides a framework (a model) within which the various services and protocols, and the relationships among them, may be defined. Thus, for OSI, we have the *OSI reference model* (OSI RM);[4] for TCP/IP, the Internet architectural model or, simply, the Internet architecture.

The term architecture suggests an analogy between a reference model and the elements of the more familiar architecture of buildings. A building-construction manual is concerned with generally applicable truths about building: "The roof goes on the top, and the basement goes

4. Or simply *RM* for short. One often encounters the equivalent acronym *ISORM*, for "ISO reference model" (reflecting the provenance of both the OSI architecture and the associated standards), as in "ISORMites" (devout disciples and defenders of the OSI faith).

on the bottom," or "Plumbing is good." This is just the accumulated wisdom, or perhaps common sense, of the builder's craft. A local building code is much more specific: it specifies the use of a particular grade of 4-inch PVC pipe supported at no less than 1-foot intervals (etc.), or it mandates conformance to an American Society for Testing Materials standard for the flammability of roofing material. The blueprints for an office building are even more specific: they specify, very precisely, every detail of the construction of an actual building, giving the length and placement of every piece of 4-inch PVC pipe and the brand name and stock number of the roofing material.

The principles of construction that are collected in the manual are assumed to be universal. There are, however, a great many local building codes, each of which "conforms" to the generally accepted principles of construction. And within the jurisdiction of a single building code, an almost infinite variety of actual buildings can be constructed.

The OSI reference model describes both the general principles of open systems networking and the specific prescriptions for open systems that follow the OSI architecture. Like the building-construction manual, the OSI reference model collects "universal truths" about open systems: layers are good; internetworking accommodates many different types of real-world networks; addresses must be unambiguous. Like a building code, the OSI reference model also defines an abstract model of an open system: not only are layers good, but OSI has seven of them; internetworking functions are in the network layer; and addresses in OSI are not only unambiguous, they are constructed in a particular (hierarchical) fashion. However, just as a building code doesn't describe any particular building, the OSI reference model doesn't describe any particular implementation of a real open computer or communications system.

In fact, it could be argued that the scope of the OSI reference model is even more limited than that of a building code, since it specifies only the externally visible behavior of an open system and carefully avoids issues that are not directly relevant to the ability of the system to communicate with other open systems. A building code might prescribe PVC pipe for drain lines in residential construction, although the requirement is actually for pipe with certain characteristics (thermal stability, resistance to corrosion, etc.), whether made of PVC or an equally suitable material. The OSI reference model specifically refrains from defining the characteristics of an open system, such as how to manage buffers or pass information from one process to another, that are not relevant to the interconnection of open systems. This restraint is essential, since such restrictions would be based on assumptions about the current state of the art. In addition to reducing opportunities for competitive differentiation among open system vendors, they might easily deter innovation in computer hardware and software design.

Architecture Wars

The "architecture wars" between OSI and TCP/IP involve primarily the building-code aspects of open systems. Very few people get into arguments about whether or not plumbing is a good thing or whether or not it is useful to organize the functions of an architecture into layers; many plumbing contractors, however, are quite willing to debate the merits of copper pipe over PVC, and similarly, network engineers will debate whether security functions belong in the transport layer or the network layer. Anyone associated with the construction business knows how many different variations on the same basic theme can be captured by different codes; nevertheless, buildings constructed according to codes in Massachusetts and Pennsylvania are (presumably) equally habitable. The differences between the OSI and the Internet architectural models are almost entirely variations on themes that are common to all open system architectures: layers, services, protocols, and other generally applicable concepts from the open system cookbook.

Layers

Many functions must be performed above the transmission media (the "wires") to support useful communication between computer systems. For example, it is often necessary to ensure that the information sent from one computer to another arrives in order, uncorrupted, and without loss or duplication. If the two computers are physically attached to different transmission media (e.g., one to an Ethernet local area network and another to a public, packet-switching network), it is also necessary to define a function that selects a route and forwards data over multiple "hops" from source to destination. Additional functions encode and preserve the semantics and context of information as it is understood by a distributed application (a networked file service, electronic mail, or a directory service) running on several communicating computer systems.

As an example, consider this partial list of the functions that might be performed by communicating computers (real-world open systems) to accomplish the transfer of a file from one system to another:

- Access to the local file transfer service
- Identification of the application (and destination computer) to which the file is to be transferred
- Establishment of communications between the peer applications that will be involved in the file transfer
- Determination of a common representation of the information to be transferred, including both file and data structure
- Access via the local file storage facility to the contents of the file to be transferred
- Selection of the network service and/or transmission media through which the contents of the file must travel (routing and forwarding)
- Data fragmentation and reassembly
- Physical-level signaling and bit transmission
- Reliable transportation of the contents of the file from source to destination, resulting in an exact duplicate of the original file at the destination

Although it would be possible to do so, specifying one humage[5] protocol to deal with all these functions would be inefficient, inflexible, and inordinately complex; in fact, just plain silly. Imagine, for a moment, what the state machine would look like!

The principle of *layering* solves this problem by collecting functions into related and manageable sets. For example, the functions associated with reliable data transport (detection and correction of lost, misordered, duplicated, or corrupted packets) logically form one set; those that handle routing and forwarding, another; and those that handle data representation, a third. In the process, the sets of functions are organized in a hierarchy. Data representation, typically viewed as an application-oriented or "end-user" function, sits on top of the reliable delivery or "end-to-end transport" functions, so that the end-user functions can make use of the end-to-end functions (i.e., they won't have to duplicate them). Both the OSI reference model and the Internet architectural model call the related and manageable sets *layers.*

Layers are good, but how many layers? The OSI reference model specifies seven (Figure 3.1), which have gradually assumed an almost

5. The origin of this word is attributed to Matthew Piscitello (age 4 at the time), who could not discriminate between daddy's use of humongous and mommy's use of huge, so he created his own.

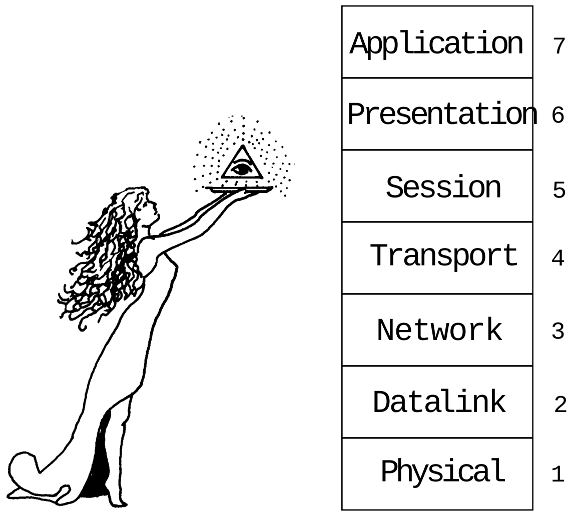

FIGURE 3.1 OSI Reference Model

holy significance and to which other, less sacred architectures are often compared.

The Internet architecture specifies five layers (Figure 3.2), combining the functions of the OSI application, presentation, and session layers into a single application layer.

AHA *One might suppose that the people responsible for the OSI reference model believed that seven is, in fact, the "right" number of layers for an open systems architecture or that the Internet architects, after careful analysis, determined that five is the "right" number. Precedent, of course, argues for the mystical properties of the number 7; there are, after all,* seven *dwarfs:*

7	*Sneezy*
6	*Sleepy*
5	*Dopey*
4	*Doc*
3	*Grumpy*
2	*Bashful*
1	*Happy*

. . . and seven *deadly sins:*

7	*Wrath*
6	*Sloth*
5	*Lust*
4	*Avarice*
3	*Gluttony*
2	*Envy*
1	*Pride*

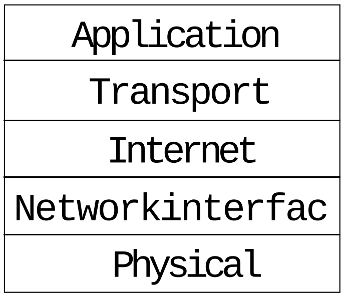

FIGURE 3.2 Internet Reference Model

But in fact, the architects of both OSI and the TCP/IP protocol suite had never seriously worried over the fact that their respective architectural models had a particular *number of layers and certainly never anticipated that the number of layers in the OSI reference model would become, one day, the only thing that many people would remember about the architecture.*

Needless to say, there is nothing special about the number 7, nor is the fact that the OSI reference model has seven layers and the Internet architecture has five deeply significant. What *is* important is the way in which the OSI and Internet architectures allocate functions among the layers and the ensuing consequences for the operation of OSI and Internet systems and the development of distributed applications.

A Quick Tour of OSI's Seven Layers

The application, presentation, and session layers of the OSI model are collectively referred to as the *upper layers* (Figure 3.3). They provide end-user services: the functions that enable applications to share and manipulate information. Not surprisingly, the remaining layers (transport, network, data link, and physical) are collectively referred to as the *lower layers*.[6] They provide an end-to-end data transport service, organizing the communication resources that exist in the real world to carry information from any source system to any destination system.

The upper layers are *application oriented*; they focus on the application processes that are the ultimate "end users" of OSI. They operate as though they were directly connected to all their peers at the transport service boundary, without regard for the way in which their communication is actually accomplished. The lower layers are *communications oriented*; they focus on the job of supporting the upper layers' complacent fiction

6. In some circles, the transport and network layers are referred to as the middle layers.

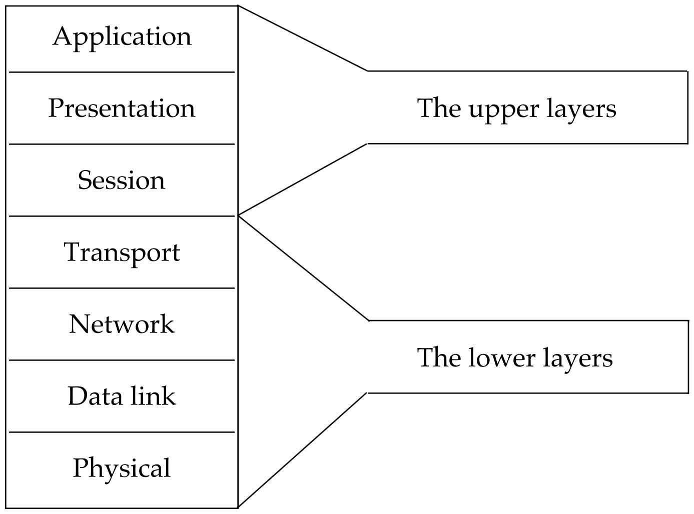

FIGURE 3.3 The Great Divide

that they are directly connected by arranging for data to be transported out of one host computer or *end system*; across the arbitrarily complex and heterogeneous real world of wires, fiber, ether, carrier networks, bridges, and routers; and into another end system.

The *application layer* provides access to OSI services. It is also (conceptually) the place in which distributed applications reside and in which they access the networking environment.

The *presentation layer* isolates applications from concerns about the representation (syntax) of the data they exchange, allowing them to deal only with their meaning (semantics). It defines a common or *canonical* form for the representation and manipulation of application information. Some computers use EBCDIC as the native character encoding, and others use ASCII, and different computers and operating systems store information in memories or disk files in different ways. Presentation layer functions allow applications to represent their data in a machine-independent fashion by providing a universal language in which to describe abstract data structures. By offering what is often called a common network programming language, the presentation layer allows applications to exchange structured information rather than raw bit strings. It also defines the way in which elements of that language are actually transmitted from one system to another. The presentation layer is thus responsible for transforming information from machine-specific data structures common to the source computer into machine-independent data structures for transmission, and from machine-independent data structures for transmission to machine-specific data structures common to the destination computer as data are received.

The *session layer* provides mechanisms for organizing and synchronizing the exchange of data between application processes. Session permits an application to mark its progress as it sends and receives data (synchronization points), provides ways for applications to control the direction of information flow (turn management), coordinates multiple independent exchanges ("activities") within the overall context of a single session, and allows applications to inform each other about the occurrence of errors and the steps to be taken to resynchronize part or all of the affected dialogue(s).

The *transport layer* provides "transparent transfer of data from a source end open system to a destination end open system" (ISO/IEC 7498: 1993). Transport is responsible for creating and maintaining the basic end-to-end connection between communicating open systems, ensuring that the bits delivered to the receiver are the same as the bits transmitted by the sender: in the same order and without modification, loss, or duplication.

The *network layer* provides "a path between transport entities, relieving the upper layers from dealing with the way in which data are transferred from one end open system to another" (ISO/IEC 7498: 1993). Network determines the path or route that the data must take from original source to final destination and forwards the data over that route. It provides a service that is independent of the underlying transmission media and includes all of the routing, relaying, and interworking functions needed to get from source to destination, regardless of the number or type of transmission resources that may be used in tandem or in parallel.

AHA *The boundary between the transport and network layers was originally conceived by early telephony-oriented OSI developers as a representation of the traditional regulatory boundary between customer premises equipment (CPE) and a public carrier network: CPE contained transport and the upper layers, and the carrier network implemented the network and other lower-layer functions. This model assumed a pervasive global carrier network to which every end system was directly attached. The popularity of LANs and other privately deployed networks based on the concept of "internetworking" had made this model obsolete even before the work on the OSI reference model was completed. The result is a network layer with an extensive internal structure, containing both internetworking functions (which are independent of* *any particular network technology) and network-specific functions (which vary depending on the type of real-world network involved). The Internet architecture captures this distinction much more clearly, by defining separate "internet" and "network-interface" layers.*

About all the OSI reference model can think of to say about the *data link layer* is that it "provides for the control of the physical layer, and detects and possibly corrects errors which may occur" (ISO/IEC 7498: 1993). The data link layer couples to a particular physical access method whatever functions are necessary to recover bit-stream errors that may be introduced during transmission (due to "noise," clock jitter, cosmic rays, and other forms of signal interference).

The *physical layer* provides "mechanical, electrical, functional, and procedural means to activate a physical connection for bit transmission . . ." (ISO/IEC 7498: 1993), which is OSI's attempt to dress up the unvarying role of the physical layer in any network architecture: to transform bits in a computer system into electromagnetic (or equivalent) signals for a particular transmission medium (wire, fiber, ether, etc.).

AHA *The data link and physical layers appear in the OSI reference model primarily for the sake of completeness (during the deliberations over the text of the reference model, it often seemed to the authors that somewhere, someone with considerable authority had declared, "Let there be nothing in the world of communication for which OSI has no layer"). Since these two layers deal with functions that are so inherently specific to each individual networking technology, the layering principle of grouping together related functions is largely irrelevant. This has not, of course, prevented endless arguments about whether there is or is not an addressing function in the data link layer or whether medium access control for an IEEE 802 local area network is a data link layer function or a physical layer function.*

For most real network technologies, it is both impractical and unnecessary to determine where the boundary between these two layers lies or even whether to describe the functions of the real network as "data link layer functions" or "physical layer functions." Since many real networks also include functions that are, from the technical standpoint of the OSI architecture, "in the network layer," the TCP/IP model of real networks as simply individual network services is much better. OSI recognized this after the fact by introducing what amounts to a codicil to its reference model[7]—namely, the “subnetwork” concept, which collects everything below the OSI internetworking protocol into a single abstraction, forgoing formal discrimination of network layer functions, data link layer functions, and physical layer functions.

A Quick Tour of Internet’s Five Layers

How does the OSI allocation of functions among seven layers compare to the layering applied in the Internet architecture? In *Internetworking with TCP/IP,* Douglas Comer (1991) characterizes the TCP/IP architecture as comprising application-level and network-level internet services. This distinction between upper layers (actually, in the Internet architecture, a single upper layer) and lower layers follows the same logic for TCP/IP as for OSI (see Figure 3.4).

The application-level internet services are a set of application programs that operate across a TCP/IP-based internet. All of the end-user functions, which are divided among three OSI layers, are incorporated in the TCP/IP architecture into each application program individually. TCP/IP applications, therefore, are designed to operate directly over a raw transport interface.

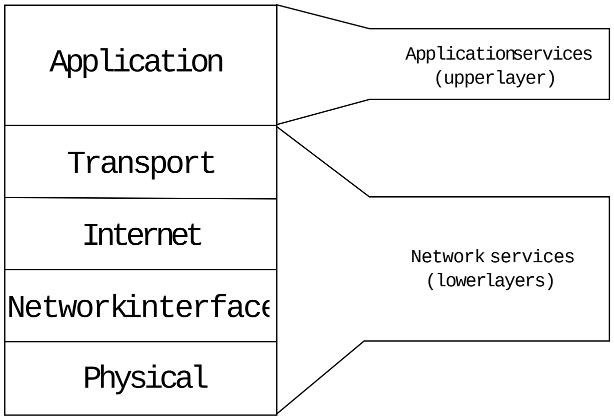
Application
Transport
Internet
Networkinterface
Physical
Applicationservices (upper layer)
Network services (lowerlayers)

FIGURE 3.4 Upper Layer/Lower Layers Distinction in TCP/IP

7. The codicil is the *Internal Organization of the Network Layer* (ISO/IEC 8648: 1988), which was published five years after the reference model as a way of “burying the hatchet” in the war between two factions, each of which painted the other in extreme terms. Faction 1, the network-centrics, saw the world as composed of powerful, pervasive public data networks that graciously suffered the attachment, for a fee, of relatively insignificant pieces of end-user equipment; faction 2, the host-centrics, saw the world as composed of a global fraternity of vigorous, autonomous computer systems and LANs, filled with important end-user applications, which occasionally, when it could not be avoided, permitted the public networks to carry some of their communication.

The network-level internet services correspond more or less directly to the services provided by OSI's lower layers. The OSI transport layer provides much the same end-to-end communications service as the TCP/IP transport layer. OSI offers both virtual circuit (connection) and datagram network layer services (see "Connections and Connectionless," later in the chapter); TCP/IP offers only a datagram internetworking service. The network interface layer in the TCP/IP architecture corresponds to a combination of the OSI data link layer and the network-specific functions of the OSI network layer. The physical layer, of course, does what the physical layer must do.

The monolithic upper layer in the Internet architecture reflects the deliberate involvement of the TCP/IP architecture in the way in which distributed applications are organized. In this respect, OSI provides a more comprehensive model. On the other hand, the correspondence between the lower layers of the Internet architecture and real-world internetworks is much clearer and more accurate than it is in the OSI case. It has been suggested that OSI is the "better" model of distributed applications and that TCP/IP is the "better" model for the networks that support their communication.

Terminology

The principles applied to the development of the OSI reference model are similar to those of the TCP/IP architecture; unfortunately, the terminology is not. The OSI architects were convinced that none of the familiar terms of network engineering, freighted as they were with preexisting real-world connotations, would suffice for the highly formal and precise descriptions they imagined for their reference model. The elusive and mystifying world of "OSI-speak" was created to insulate the ethereal and pure concept space of OSI from contamination by the existing networks of mortals. The resulting terminology reads more like German existentialism than Tanenbaum's *Computer Networks* (1988).

OSI is unquestionably encumbered with too many obscure terms with confusing definitions. A definition is supposed to be "a brief and precise description of a thing by its properties" (Thatcher and McQueen 1977). OSI's definitions are far from brief and are often imprecise; in many cases, brevity and clarity are sacrificed for the sake of either precision or political compromise. The language of TCP/IP ("Internet-ese"), on the other hand, is disarmingly accessible and bears a striking resemblance to terms that one might use to describe real networks.

Nevertheless, it is difficult to understand either OSI or TCP/IP without first becoming familiar with some of the most frequently used terminology. In some cases, readers need to be familiar with a term because it expresses an important concept; in other cases, they need to be familiar with a term because they will encounter it in the OSI standards and will otherwise be inappropriately intimidated by it. Since most of the difficult terminology belongs to OSI, most of what follows applies to that architecture; except where noted, the terminology of TCP/IP is intuitive.

The situation regarding terminology is similar to that in which Owl finds himself in the House at Pooh Corner:

Owl explained about the Necessary Dorsal Muscles. He had explained this to Pooh and Christopher Robin once before, and had been waiting ever since for a chance to do it again, because it is a thing which you can easily explain twice before anybody knows what you are talking about (Milne 1954).

Entities

In the OSI architecture, the service provided by a layer is, conceptually, the result of the collective activity of all the computer systems that participate in OSI. Each open system contains components (vertical slices) of each layer that OSI calls *subsystems*. A subsystem represents the functionality of a single layer that is actually present in an individual open system. An open system therefore contains seven subsystems, each one corresponding to one of the seven layers identified in Figure 3.1. Collectively, subsystems of the same rank form a layer.

All the functions present in a given subsystem may not be active at the same time. For example, if a layer offers both a datagram and a virtual-circuit type of service, but only the datagram service is being used, then only the datagram-oriented functions of that layer are active. The active elements within a subsystem are called *entities*.

In effect, an entity is the "stuff" inside a layer. Informally, the terms *layer entity*, *entity*, and *layer* are used interchangeably, with the understanding that the formal meaning is "an active element within a hierarchical division of an open system" (ISO/IEC 7498: 1993).

Given the multinational composition of the standards organizations responsible for OSI, it is not surprising that a great deal of energy is expended in the selection of universally acceptable names for the *"things" to which open systems networking standards must refer. Early in the OSI work, it became clear that an architecturally neutral term for the "thing" that sits in an OSI layer and represents the activities that take place there would be needed.* Process, module, *and other terms borrowed from the realms of programming languages and operating systems all carried the implication of implementation, and the OSI architects were particularly concerned to avoid the impression that every "thing" in the OSI architecture necessarily finds its counterpart in an implementation. They came up with the wonderfully metaphysical term* entity, *which the* Oxford English Dictionary *defines as "Being, existence, as opposed to non-existence; the existence, as distinguished from the qualities or relations, of anything." Perfect! Then, returning from the meeting in Berlin at which the agreement on this term was reached, Lyman noticed the following posted on the underside of the toilet cover in the lavatory of the airplane, which added an entirely unique perspective to OSI terminology.*

DO NOT THROW

☐	cups	towels	☐
☐	bottles	napkins	☐
☐	razor blades	diapers	☐

entities

IN TOILET

Notation

The notations *(N)*, *(N+1)*, and *(N–1)* are used to identify an arbitrary layer entity and the layer entities hierarchically adjacent to it (Figure 3.5). Typically, the value of (N) is an integer (1 through 7; the physical layer is numbered 1, and the application layer is numbered 7). In many OSI standards, the first letter of the name of a layer is used for (N) rather than an integer. Thus, the terms *layer-4-entity*, *(4)-entity*, *transport entity*, and *T-entity* all refer to the same thing.

Services

The relationship between (N)-entities in adjacent layers is expressed in OSI by the following concepts and terms:

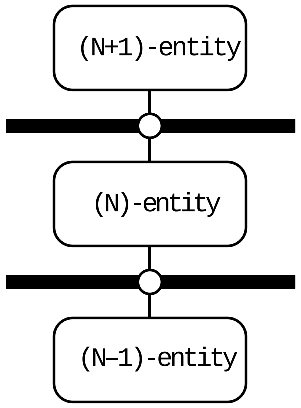

FIGURE 3.5 Entity Notation

- Each (N)-entity performs *(N)-functions*.
- The (N+1)-entities assume that a well-defined set of *(N)-facilities* is provided by (N)-entities.
- The set of (N)-facilities that (N)-entities provide to (N+1)-entities is called the *(N)-service*.

Thus, the transport layer performs transport functions and provides a set of transport facilities that constitute a transport service to the session layer. Similar terminology applies to the TCP/IP architecture; for example, the TCP/IP internet layer provides a datagram service to the transport layer.

OSI takes great pains to formally define services, and a service-definition standard is provided for each layer. The purpose of the service definition is to formally identify the functions to be performed (and the facilities to be provided) by a layer, so that a protocol can be developed to provide a well-defined and manageable set of functions. The existence of a formal (N)-layer service definition also assists in the design of layer (N+1), since it can be assumed that certain functions are already performed in layer (N) and (in theory) should not be duplicated at any layer above (N). For example, if the transport service provides end-to-end reliable delivery, the session layer should not.

Services provide a formal way to express the relationship that exists between an entity in one layer and an entity in a layer immediately above or below it. The OSI service model contains the following elements:

- A user of the service provided by layer (N) resides at layer (N+1), and is called an *(N)-service-user*.

- The elements involved in the provision of the (N)-service are called (N)-entities (as described in the preceding section).
- The (N)-entities that actively participate in providing the (N)-service are collectively referred to as the *(N)-service-provider*.
- The (conceptual) point at which an (N)-service is provided to an (N+1)-entity by the (N)-service provider is called an *(N)-service-access-point*, or (N)-SAP.
- The information exchanged at an (N)-SAP is called *(N)-service-data*, and individual units of that data are called *(N)-service-data-units*, or (N)-SDUs.

These relationships are illustrated in Figure 3.6.

Applying these principles to the specific example of the interaction between the session and transport layers, we have the following:

- A session entity, a user of the service provided by layer 4, the transport layer, resides at layer 5 and is called a transport service user.
- The elements involved in the provision of the transport service are called transport entities.
- The transport entities that actively participate in providing the transport service are collectively referred to as the transport service provider.
- The (conceptual) point at which the transport service is provided to a session entity by the transport service provider is called a transport service access point, or TSAP.
- The information exchanged at a trasport service access point is called transport service data, and individual units of that data are formally called transport service data units, or TSDUs.

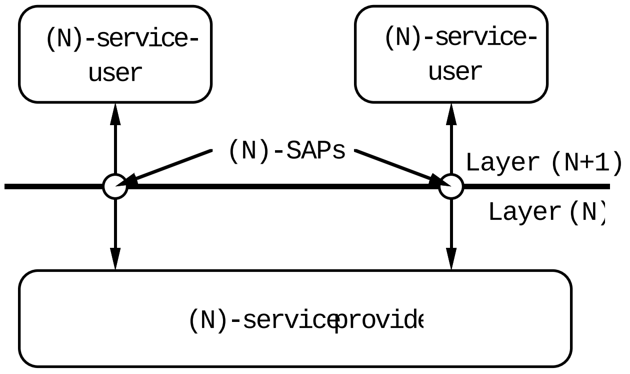

FIGURE 3.6 Generic Service Model

The Queue Model

The OSI service model provides a convention for describing the interactions between adjacent entities. The exchanges across a service boundary are modeled as a pair of queues, where each exchange represents an atomic (simultaneous) interaction at two (N)-SAPs (see Figure 3.7). Objects, called *service primitives*, are placed in or removed from a queue by the service users and by the service provider. The service primitives indicate some action that must be (or has been) performed by one of the other participants in the service interaction.

As an example, imagine that (N)-service-user A wishes to establish an (N)-connection to (N)-service-user B. An object, the (N)-CONNECT service primitive, is submitted by (N)-service-user A to the (N)-service-provider in the form of a *request* primitive; i.e., it is placed in an imaginary queue that exists between service users and the service provider. This is step 1 in Figure 3.8. The queue is accessible via A's (N)-service-

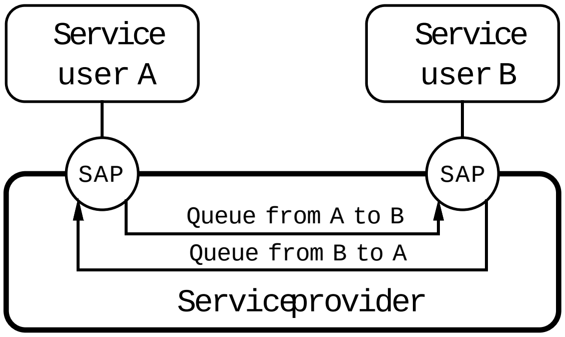

FIGURE 3.7 The Queue Model

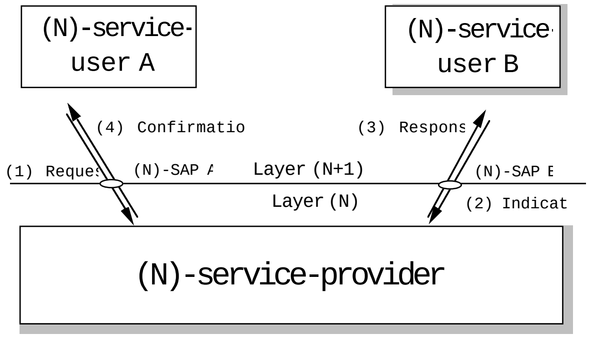

FIGURE 3.8 Confirmed Service

access-point. The (N)-service-provider removes the request primitive from the queue and informs (N)-service-user B of the request by inserting an (N)-CONNECT *indication* primitive in the queue (step 2). (N)-service-user B accesses the queue via its own (N)-service-access-point. B removes the indication primitive from the queue and accepts the connection request by placing a positive (N)-CONNECT *response* primitive in the queue (step 3). The (N)-service-provider removes the response primitive from the queue and places a *confirmation* primitive in A's queue (step 4). OSI calls this a *confirmed service*.

There is also an *unconfirmed service*. Here, only the request primitive is available to (N)-service-users, and only the indication primitive is available to the (N)-service-provider. (N)-service-user A places the request primitive in a queue when it wishes to send (N)-service-data to (N)-service-user B. The (N)-service-provider places the indication primitive in a queue to:

- Notify (N)-service-user B of a request from (N)-service-user A.
- Notify one or more (N)-service-users of an event or action instigated by the (N)-service-provider (hence, the term *provider-initiated*).

The unconfirmed services are illustrated in Figures 3.9 and 3.10.

Connections and Connectionless

One of the most basic concepts of network architecture is the distinction between the *connection* and *connectionless* models of communication. The connection model is based on the establishment and maintenance of "state information" that is held in common by the communicating par-

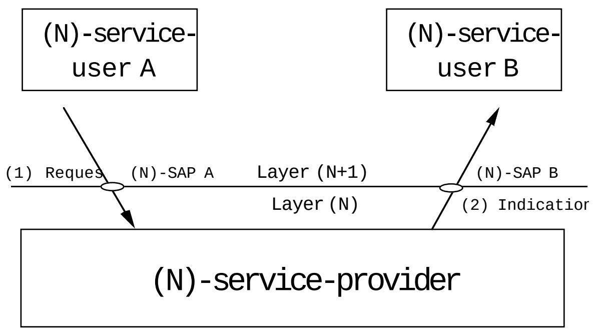

FIGURE 3.9 User-initiated Unconfirmed Service

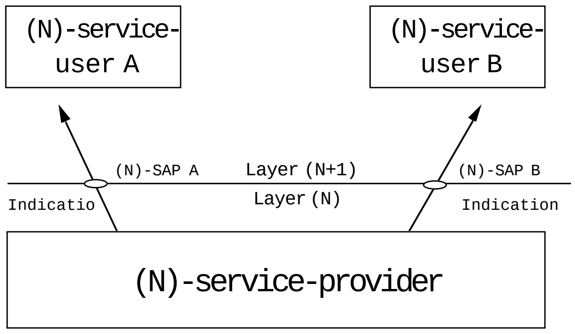

FIGURE 3.10 Provider-initiated Unconfirmed Service

ties and the underlying service provider; the state information establishes a context within which the parties interact with the service provider and communicate with each other. The connectionless model is based on individually self-contained units of communication (often called "datagrams"), which are exchanged independently without reference to any shared state (that is, there is no "connection" between the communicating parties). In the connection model, each individual unit of communication is interpreted by reference to the shared state of the connection (which captures information such as the identity of the communicating parties, the current status of flow-control variables, the way in which data have been encoded for transmission, and the sequence numbers of data units that have not yet been mutually acknowledged). In the connectionless model, each unit of communication carries within it all the information that each party needs to interpret it, since there is no shared state to refer to.

AHA *The OSI reference model terms* connection-oriented *and* connectionless, *describing virtual circuit and datagram modes of operation, were coined by Lyman Chapin and John Gurzick during the production of the first draft of the connectionless addendum to the OSI reference model on the roof of The Pointe resort in Phoenix, Arizona, in 1981. Sometime between 1983 and 1987, the connection-oriented "X.25 crowd," who were not about to hyphenate the noun* connection *(to create an adjective) without attaching a similarly demeaning shackle to the rival* connectionless, *succeeded in changing the "official" term to* connectionless-mode*—an injury to English grammar that at least had the dubious virtue of leaving everyone equally dissatisfied. A few reminders of the original terms persist; the standard acronym for* *the "connection-mode network service," for example, is* CONS, *not* CMNS, *and the title of ISO/IEC 8073 is "Connection-oriented Transport Protocol."*

A common mistake is to assume that either the connection model or the connectionless model must be used uniformly throughout a network architecture; that is, if one layer is defined using the connection model, then all the other layers must also use the connection model. In fact, the two models are complementary: it is appropriate to use the connection model to define a protocol in one layer (e.g., the transport layer) and the connectionless model to define a protocol in a different layer (e.g., the network layer), the combination of which can be used to provide a connection-oriented (transport) service to a higher layer.

The TCP/IP and OSI architectures employ both models in all layers, with one important exception: in TCP/IP, only the connectionless model is used to define the services and protocols of the internet layer. The Internet architecture refers to the two models as simply "connections" and "datagrams." The OSI reference model, with its penchant for "precise" terminology, uses the terms *connection-mode* and *connection-oriented* for the connection model and the term *connectionless-mode* for the connectionless model.

AHA *In the earliest work on OSI, communication between peer entities was modeled exclusively in terms of connection-based interactions, which were* de rigueur *among the telephony-oriented people*[8] *who dominated early OSI standardization activity. Consequently, the assumption that a connection is a basic prerequisite for communication in OSI permeated early drafts of the reference model, and came to be perceived as a dominant and prerequisite feature of the OSI architecture. This widely held perception caused many people who were familiar with the use of the connectionless model for internetworking in TCP/IP and other architectures to dismiss OSI as applicable only to X.25 and other connection-oriented networks. The pejorative association of OSI with X.25 has been hard to shake, despite the fact that the connectionless internetworking model has been fully incorporated into the OSI architecture,*

8. The temptation to attach a descriptive collective label to people with perspectives or beliefs different from one's own is usually irresistible, albeit deplorable; wherefore, those who came to OSI from traditional telephony backgrounds have been dubbed "wire stringers." Wire stringers believe in connections (there being no such thing as "connectionless telephony") and are skeptical of what they call "lossy datagrams"; internetworks built on the connectionless model were therefore dismissed as "academic toys." Not surprisingly, computer nerds are as likely as normal people to succumb to petty variations on the Lee Atwater syndrome.

and a complete set of protocols and services to support it has been defined and standardized.

Communication using a connection proceeds through three distinct phases:

1. *Connection establishment*, during which the parties that intend to communicate negotiate and agree on the terms of their interaction and perform any necessary "setup" functions (such as the allocation of buffers, the establishment of underlying communication links, and the initialization of state variables).
2. *Data transfer*, during which information is exchanged according to the rules established during connection establishment.
3. *Connection release*, during which the context established for communication is dismantled (buffers freed, underlying links torn down, state data structures deallocated).

Connection-mode operation is based on the familiar model of a telephone conversation:

1. Dial the phone.
2. Talk to the party at the other end.
3. Hang up.

In contrast, connectionless communication has just one phase of operation: transmission of a single, self-contained unit of data in a package that contains all relevant information. It is based on the equally familiar model of the basic postal mail service: put all necessary information (address, return address, postage, "airmail" label, etc.) on the envelope and drop it in the mailbox slot.

Connectionless data transmission has been described disparingly as "send and pray"; but is more accurately described as "best-effort delivery." A service provider, be it a datagram network or a postal authority, wouldn't last long if its users truly believed that packet or mail forwarding and delivery could only be accomplished through divine intervention.

What about Protocols?

A protocol is a well-defined set of rules for what amounts to a "conversation" between computer systems. The OSI architecture defines an [N]-*protocol* as "a set of rules and formats (semantic and syntactic) which determines the communication behavior of (N)-entities in the performance of (N)-functions" (ISO/IEC 7498: 1993).

Imagine how a typical telephone call is structured. Fred dials Wilma's telephone number. Wilma answers with a "Hello?" Fred says, "Hello, Wilma, this is Fred." Wilma replies (and implicitly acknowledges that Fred has indeed reached Wilma) with "Oh, hello, Fred." This is more or less the equivalent of a connection establishment. Fred and Wilma exchange pleasantries, Fred tells Wilma he'll be late for dinner, etc.; i.e., they transfer data. They exchange good-byes and hang up, the equivalent of connection release. In general, the caller intuits a great deal about the nature of the phone call before actually dialing; most callers anticipate that they will share a common language with the party called, it is considered rude or suspicious if one does not identify oneself or say hello, and most understand that an "uh-huh" or a "yep" is an explicit acknowledgment that the listening party has heard and understood what the speaking party has said. Although these semantics of a conversation are (thankfully) not written in ISO standards, they do constitute an implicit set of rules that people generally adhere to when calling one another, at least in the United States. The same is true for the mail system. There is a convention applied to identify the sender and intended recipient of a letter. (Conventions certainly exist to simplify the processing of mail, but the degrees of latitude that the U.S. Postal Service and PTTs extend to postal patrons are often nothing shy of remarkable.)

Computers establish connections and send datagrams in much the same manner, but the semantics and syntactical elements are more rigorously defined. The normal flow of a computer conversation is a highly structured sequence of actions. Possible exceptions to the normal flow must be considered and accommodated by introducing some action to be taken in response to the exception. And of course, the words exchanged must be understood by the systems that exchange them. The set of actions that define an (N)-protocol defines the *state machine* for the protocol. The words exchanged between communicating (N)-entities are called *(N)-protocol data units* or *(N)-PDUs*. Since computers must be able to distinguish between words that convey actions to the bit level of detail, the structure of each (N)-PDU exchanged is defined for each (N)-protocol. The bits that (N)-protocols interpret to determine what actions to take are collectively called *(N)-protocol-control-information,* or *(N)-PCI*. The data that an (N)-service-provider moves from one (N)-service-user to another are called *(N)-user-data*.

A generic illustration of the relationships among many of these terms is provided in Figure 3.11, and a concrete example is provided in Figure 3.12. Note that the terminology is different in the Internet architecture; in fact, it is different for each layer. Figure 3.13 illustrates that the same principles can be applied.

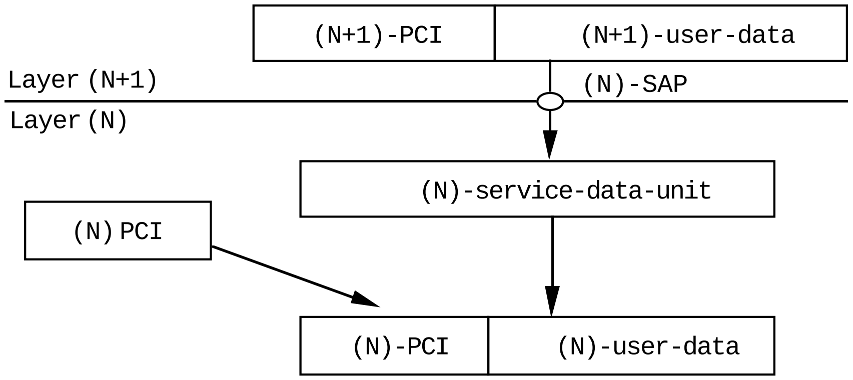

FIGURE 3.11 Relationships among Protocol and Service Terminology

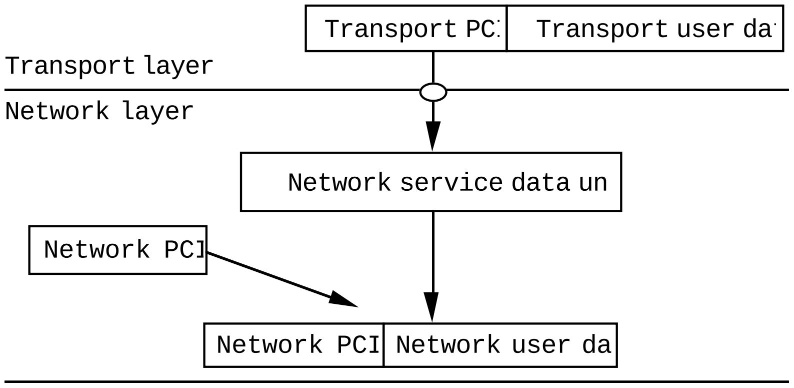

FIGURE 3.12 Protocol and Service Terminology: A Concrete Example

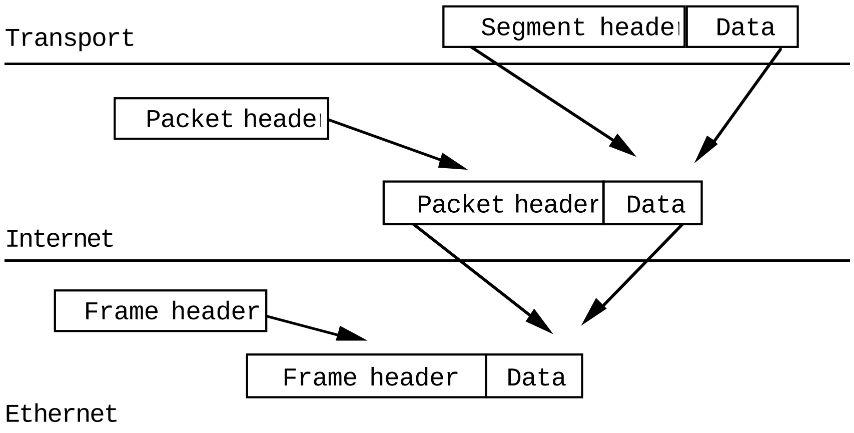

FIGURE 3.13 Terminology of the Internet Architecture

Protocol Headers and User Data

A more familiar term for (N)-protocol-control-information is *protocol header*. As one progresses iteratively down the layered architecture, the formal "rules of thumb" for the association of protocol headers with user data are:

- The (N)-user-data to be transmitted/transferred are prepended with (N)-PCI, forming an (N)-PDU.
- One or more (N)-PDUs are passed across an (N)-SAP as one of a set of service parameters that comprise an (N)-SDU, called (N)-service-user-data.
- The (N)-service-user-data themselves are prepended with (N–1)-PCI, forming one or more (N–1)-PDUs.

All of which basically says that at each layer, in order to exchange user data, a protocol is operated. To convey the rules of the protocol from sender to receiver, PCI or header information is attached to the user data to describe them, distinguish them from other data being exchanged, and tell the receiver what to do with them. The header information is meaningful only to the peers of a given layer, so when the combination of header and data is passed down to an adjacent, lower layer, it is treated like one lump of user data. Entities at the lower layer also have a job to do and rules to follow, so the process continues until you get to the lowest layer, where electrons follow nature's course (see Figure 3.14).

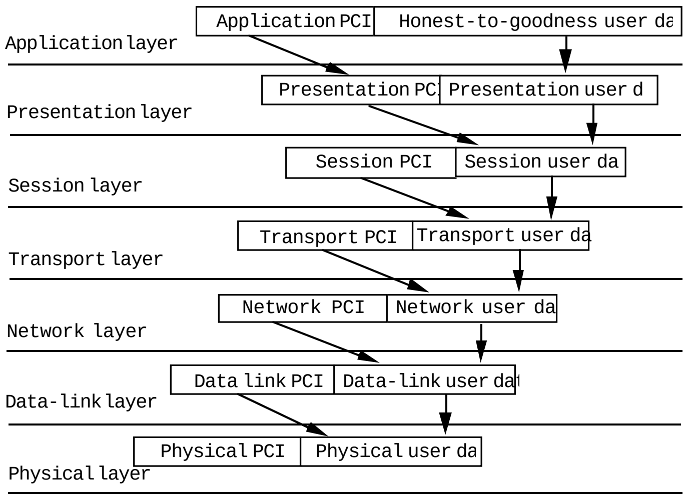

FIGURE 3.14 Separating PCI from User Data

There are certainly those who would conclude that the developers of OSI were a bit too obsessed with protocol control information, and wry comments made during OSI standards meetings, such as "Remember, we're in the header business, not the data business . . . ," provide anecdotal substantiation of this claim. OSI is by no means fit and trim. The expectation is that feature for feature, OSI brings enough that is new and helpful toward the goal of open distributed processing to encourage its development and use.

Relating Service to Protocol

The relationship between a service and a protocol is straightforward: for every primitive action, there is a related set of protocol exchanges that enable the service provider to accomplish what it has been directed to do by its service users (or to notify its service users of exception situations that arise during the provision of service). This relationship is illustrated in Figure 3.15.

Note that for certain service primitives, there may be no explicit exchange of (N)-PDUs between the (N)-entities that comprise the (N)-service-provider. In this situation, the service primitive exchange models a local action taken by a service user or the service provider; e.g., passing locally significant parameters. For other service primitives, several exchanges of (N)-PDUs might follow the issuance of one primitive request before the associated indication is generated. In the data transfer phase of the transport layer, for example, a transport PDU containing transport user data is sent by the transport entity serving A to the trans-

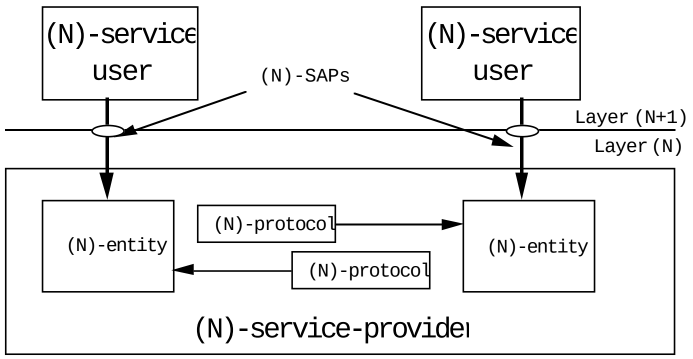

FIGURE 3.15 Relating Service to Protocol

port entity serving B as a result of the issuance of a T DATA.request primitive by transport user A. A transport PDU containing an explicit acknowledgment of receipt of those data is returned by the transport entity serving user B prior to the issuance of the T-DATA.indication primitive to user B.

Time-Sequence Diagrams

A second model applied nearly as often in OSI standards as the queue model is the time-sequence diagram. The time-sequence diagram again attempts to represent the interaction of service users and a service provider but adds the dimension of time. Compare the time-sequence diagram in Figure 3.16 to the queue model example in Figure 3.7.

The time-sequence diagram is actually more powerful than the queue model when extended to illustrate both protocol exchange and service primitive interaction, as shown in Figure 3.17.

This extension does not appear in (m)any OSI standards but is used extensively in this book because it is effective in demonstrating the order in which things occur.

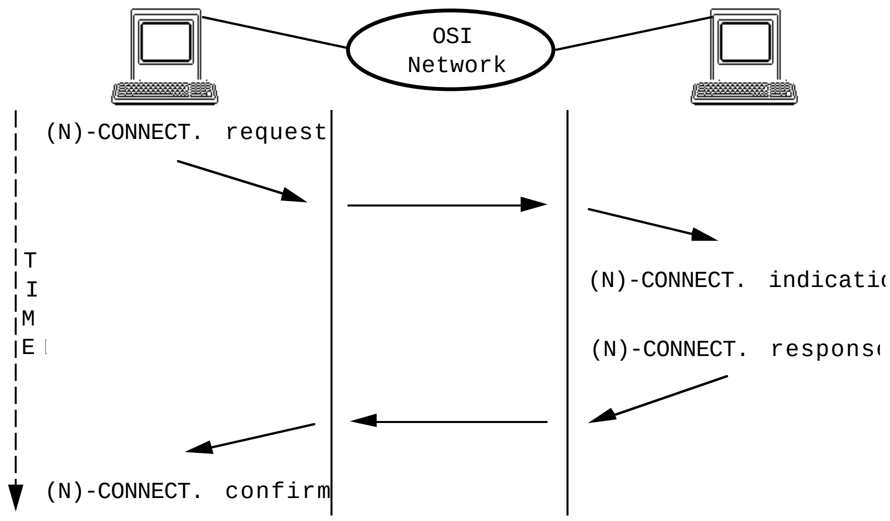

FIGURE 3.16 Time-Sequence Diagram

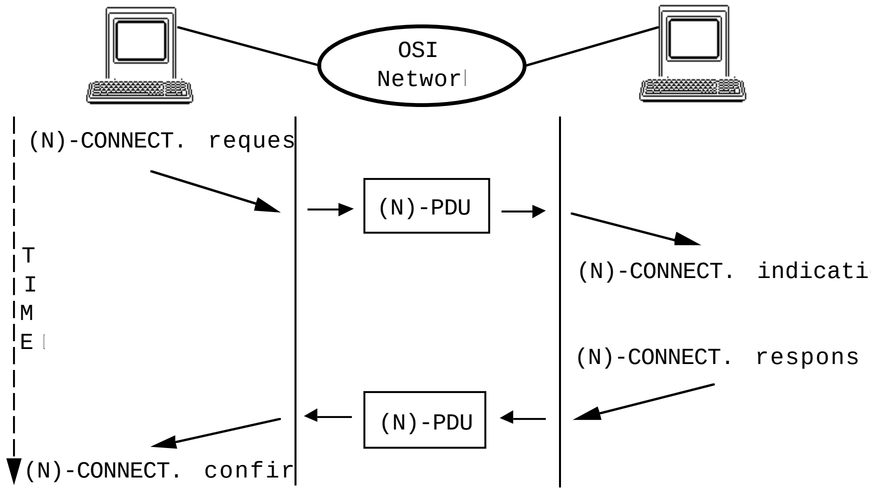

FIGURE 3.17 Extended Time-Sequence Diagram

A Final Fling with OSI Fundamentals

Applications typically deal with arbitrary-length octet streams, and information exchanged between applications ranges from single keystrokes to multimegabyte images. Transmission media used in networking today vary widely in the maximum transmission unit size that can be accommodated with tolerable and detectable loss, from several hundred to several thousand octets. Computer bus technologies have similar physical limitations but can handle tens of thousands, even millions of octets. Thus, when the octet streams exchanged between applications are physically larger than an offered maximum transmission unit size, functions must be present to chop up the stream into small pieces and then put the pieces back together again. *Segmentation* is the process of breaking an (N)-service-data-unit into several (N)-protocol-data-units; *reassembly* is the process of recombining the (N)-PDUs into an (N)-SDU (see Figure 3.18).

Sometimes, it is useful to clump together small pieces of informa-

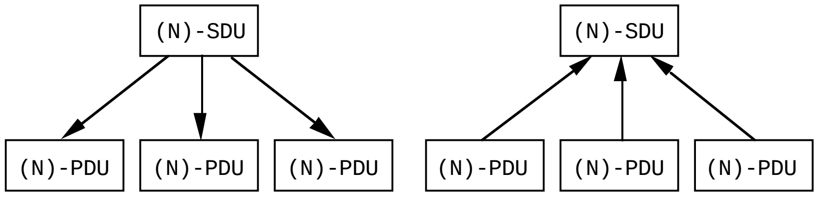

FIGURE 3.18 Segmentation and Reassembly

tion and transmit them in one swell foop. Two forms of this processing exist in OSI. If the combining process occurs across two adjacent layers, the terms *concatenation* and *separation* apply. Concatenation is the process of combining several (N)-PDUs into one (N–1)-SDU; separation is the reverse function of concatenation (see Figure 3.19). If the combining process occurs within a layer, the terms *blocking* and *unblocking* apply. Blocking is the process of combining several (N)-SDUs into a single (N)-PDU; unblocking is the reverse function (see Figure 3.20).

A similar formal terminology is defined for connections. *Multiplexing* is the process of supporting several (N)-connections using a single (N–1) connection or (N–1)-association; *demultiplexing* is the reverse function. Correspondingly, *splitting* is the process of using several (N–1)-connections to support a single (N)-connection, with *recombining* being the reverse function. These are much harder to draw, so this is left as an exercise for readers.

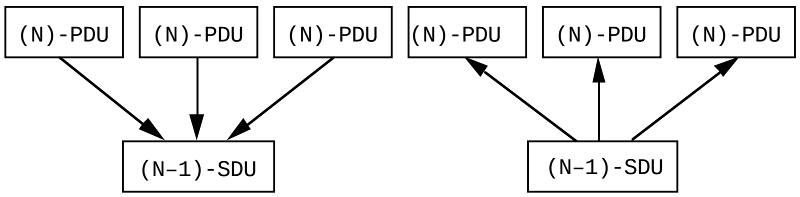

FIGURE 3.19 Concatenation and Separation

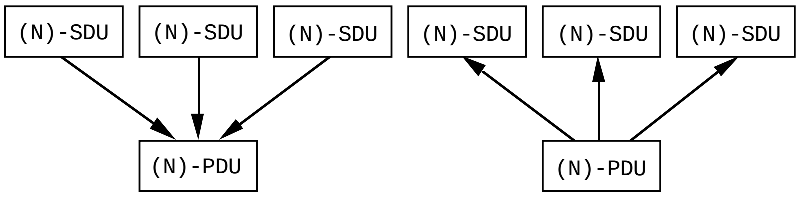

FIGURE 3.20 Blocking and Unblocking

Conclusion

This chapter has provided an overview of the formal methodology used to describe OSI, and has compared this to the accumulated folklore that describes the Internet architecture. The authors observed that many architectural "fundamentals" are common to both the OSI and Internet architectures and that it is in the application of these fundamentals that the architectures often diverge. This chapter has also identified the terminology readers will most frequently encounter in ISO/IEC, CCITT, and Internet standards, first defining terms in the context of their native architectures and then relating the formal and often impenetrable OSI terms to their more commonly encountered Internet counterparts.

The ISO reference model provides a nearly complete description of OSI architecture and terminology. To obtain a complete description of Internet architecture and terminology, readers must “trawl” the RFC directories. Two noteworthy sources for information on this subject are *Perspective on the ARPANET Reference Model* (RFC 871) and *Hitchhikers Guide to the Internet* (RFC 1118).