CHAPTER 2

PROTOCOL ARCHITECTURE, TCP/IP, AND INTERNET-BASED APPLICATIONS

To destroy communication completely, there must be no rules in common between transmitter and receiver—neither of alphabet nor of syntax.

—On Human Communication, Colin Cherry

KEY POINTS

- A protocol architecture is the layered structure of hardware and software that supports the exchange of data between systems and supports distributed applications, such as electronic mail and file transfer.
- At each layer of a protocol architecture, one or more common protocols are implemented in communicating systems. Each protocol provides a set of rules for the exchange of data between systems.
- The most widely used protocol architecture is the TCP/IP protocol suite, which consists of the following layers: physical, network access, internet, transport, and application.
- Another important protocol architecture is the seven-layer OSI model.

This chapter provides a context for the detailed material that follows. It shows how the concepts of Parts Two through Five fit into the broader area of computer networks and computer communications. This chapter may be read in its proper sequence or it may be deferred until the beginning of Part Three, Four, or Five.[1]

We begin this chapter by introducing the concept of a layered protocol architecture. We then examine the most important such architecture, the TCP/IP protocol suite. TCP/IP is an Internet-based concept and is the framework for developing a complete range of computer communications standards. Virtually all computer vendors now provide support for this architecture. Another well-known architecture is the Open Systems Interconnection (OSI) reference model. OSI is a standardized architecture that is often used to describe communications functions but that is now rarely implemented. OSI is briefly introduced in this chapter and examined in more detail in Appendix H.

2.1 THE NEED FOR A PROTOCOL ARCHITECTURE

When computers, terminals, and/or other data processing devices exchange data, the procedures involved can be quite complex. Consider, for example, the transfer of a file between two computers. There must be a data path between the two computers,

[1]The reader may find it helpful just to skim this chapter on a first reading and then reread it more carefully just before embarking on Part Five.

either directly or via a communication network. But more is needed. Typical tasks to be performed are as follow:

1. The source system must either activate the direct data communication path or inform the communication network of the identity of the desired destination system.
2. The source system must ascertain that the destination system is prepared to receive data.
3. The file transfer application on the source system must ascertain that the file management program on the destination system is prepared to accept and store the file for this particular user.
4. If the file formats used on the two systems are different, one or the other system must perform a format translation function.

It is clear that there must be a high degree of cooperation between the two computer systems. Instead of implementing the logic for this as a single module, the task is broken up into subtasks, each of which is implemented separately. In a protocol architecture, the modules are arranged in a vertical stack. Each layer in the stack performs a related subset of the functions required to communicate with another system. It relies on the next lower layer to perform more primitive functions and to conceal the details of those functions. It provides services to the next higher layer. Ideally, layers should be defined so that changes in one layer do not require changes in other layers.

Of course, it takes two to communicate, so the same set of layered functions must exist in two systems. Communication is achieved by having the corresponding, or **peer**, layers in two systems communicate. The peer layers communicate by means of formatted blocks of data that obey a set of rules or conventions known as a **protocol**. The key features of a protocol are as follows:

- **Syntax:** Concerns the format of the data blocks
- **Semantics:** Includes control information for coordination and error handling
- **Timing:** Includes speed matching and sequencing

Appendix 2A provides a specific example of a protocol, the Internet standard Trivial File Transfer Protocol (TFTP).

2.2 THE TCP/IP PROTOCOL ARCHITECTURE

The TCP/IP protocol architecture is a result of protocol research and development conducted on the experimental packet-switched network, ARPANET, funded by the Defense Advanced Research Projects Agency (DARPA), and is generally referred to as the TCP/IP protocol suite. This protocol suite consists of a large collection of protocols that have been issued as Internet standards by the Internet Activities Board (IAB). Appendix D provides a discussion of Internet standards.

The TCP/IP Layers

In general terms, communications can be said to involve three agents: applications, computers, and networks. Examples of applications include file transfer and

electronic mail. The applications that we are concerned with here are distributed applications that involve the exchange of data between two computer systems. These applications, and others, execute on computers that can often support multiple simultaneous applications. Computers are connected to networks, and the data to be exchanged are transferred by the network from one computer to another. Thus, the transfer of data from one application to another involves first getting the data to the computer in which the application resides and then getting the data to the intended application within the computer. With these concepts in mind, we can organize the communication task into five relatively independent layers.

- Physical layer
- Network access layer
- Internet layer
- Host-to-host, or transport layer
- Application layer

The **physical layer** covers the physical interface between a data transmission device (e.g., workstation, computer) and a transmission medium or network. This layer is concerned with specifying the characteristics of the transmission medium, the nature of the signals, the data rate, and related matters.

The **network access layer** is concerned with the exchange of data between an end system (server, workstation, etc.) and the network to which it is attached. The sending computer must provide the network with the address of the destination computer, so that the network may route the data to the appropriate destination. The sending computer may wish to invoke certain services, such as priority, that might be provided by the network. The specific software used at this layer depends on the type of network to be used; different standards have been developed for circuit switching, packet switching (e.g., frame relay), LANs (e.g., Ethernet), and others. Thus it makes sense to separate those functions having to do with network access into a separate layer. By doing this, the remainder of the communications software, above the network access layer, need not be concerned about the specifics of the network to be used. The same higher-layer software should function properly regardless of the particular network to which the computer is attached.

The network access layer is concerned with access to and routing data across a network for two end systems attached to the same network. In those cases where two devices are attached to different networks, procedures are needed to allow data to traverse multiple interconnected networks. This is the function of the **internet layer**. The Internet Protocol (IP) is used at this layer to provide the routing function across multiple networks. This protocol is implemented not only in the end systems but also in routers. A router is a processor that connects two networks and whose primary function is to relay data from one network to the other on its route from the source to the destination end system.

Regardless of the nature of the applications that are exchanging data, there is usually a requirement that data be exchanged reliably. That is, we would like to be assured that all of the data arrive at the destination application and that the data arrive in the same order in which they were sent. As we shall see, the mechanisms

for providing reliability are essentially independent of the nature of the applications. Thus, it makes sense to collect those mechanisms in a common layer shared by all applications; this is referred to as the **host-to-host layer**, or **transport layer**. The Transmission Control Protocol (TCP) is the most commonly used protocol to provide this functionality.

Finally, the **application layer** contains the logic needed to support the various user applications. For each different type of application, such as file transfer, a separate module is needed that is peculiar to that application.

Operation of TCP and IP

Figure 2.1 indicates how these protocols are configured for communications. To make clear that the total communications facility may consist of multiple networks, the constituent networks are usually referred to as **subnetworks**. Some sort of network access protocol, such as the Ethernet logic, is used to connect a computer to a subnetwork. This protocol enables the host to send data across the subnetwork to another host or, if the target host is on another subnetwork, to a router that will forward the data. IP is implemented in all of the end systems and the routers. It acts as a relay to move a block of data from one host, through one or more routers, to another host. TCP is implemented only in the end systems; it keeps track of the blocks of data to assure that all are delivered reliably to the appropriate application.

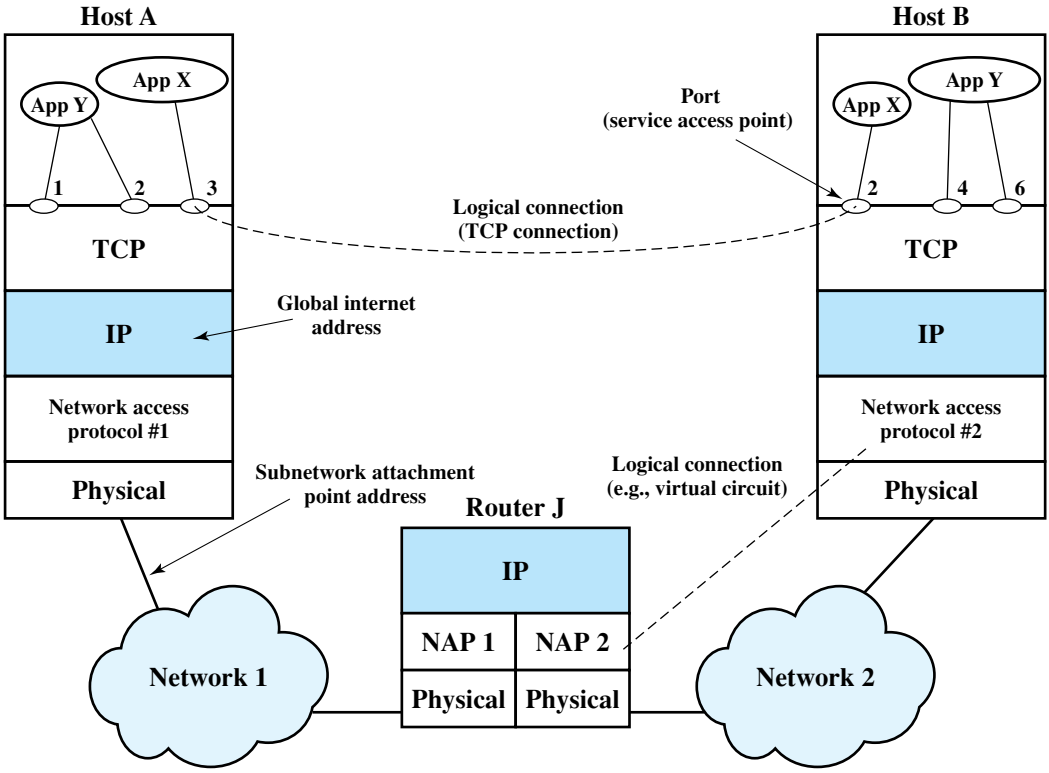


Figure 2.1 TCP/IP Concepts

For successful communication, every entity in the overall system must have a unique address. Actually, two levels of addressing are needed. Each host on a subnetwork must have a unique global internet address; this allows the data to be delivered to the proper host. Each process with a host must have an address that is unique within the host; this allows the host-to-host protocol (TCP) to deliver data to the proper process. These latter addresses are known as **ports**.

Let us trace a simple operation. Suppose that a process, associated with port 3 at host A, wishes to send a message to another process, associated with port 2 at host B. The process at A hands the message down to TCP with instructions to send it to host B, port 2. TCP hands the message down to IP with instructions to send it to host B. Note that IP need not be told the identity of the destination port. All it needs to know is that the data are intended for host B. Next, IP hands the message down to the network access layer (e.g., Ethernet logic) with instructions to send it to router J (the first hop on the way to B).

To control this operation, control information as well as user data must be transmitted, as suggested in Figure 2.2. Let us say that the sending process generates a block of data and passes this to TCP. TCP may break this block into smaller pieces to make it more manageable. To each of these pieces, TCP appends control information known as the TCP header, forming a **TCP segment**. The control information is to be used by the peer TCP protocol entity at host B. Examples of items in this header include:

- **Destination port:** When the TCP entity at B receives the segment, it must know to whom the data are to be delivered.
- **Sequence number:** TCP numbers the segments that it sends to a particular destination port sequentially, so that if they arrive out of order, the TCP entity at B can reorder them.

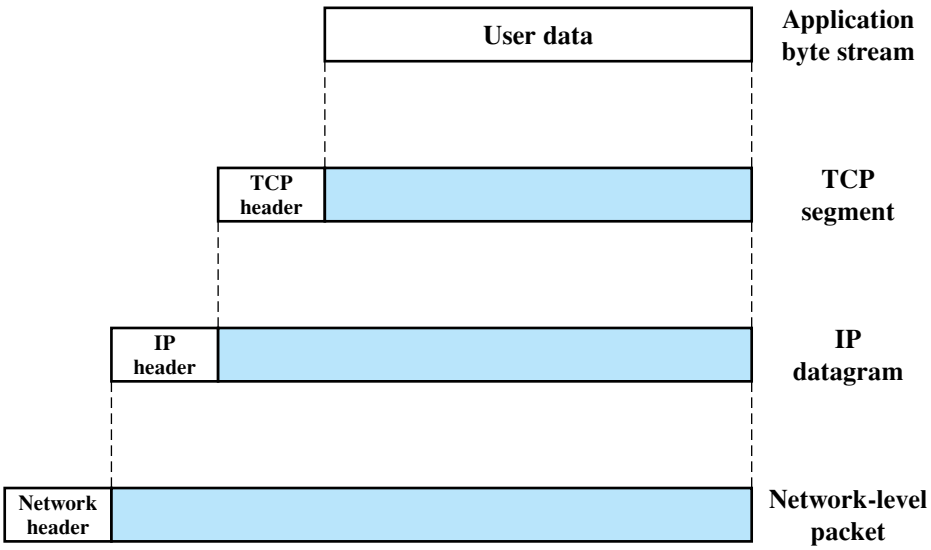


Figure 2.2 Protocol Data Units (PDUs) in the TCP/IP Architecture

- **Checksum:** The sending TCP includes a code that is a function of the contents of the remainder of the segment. The receiving TCP performs the same calculation and compares the result with the incoming code. A discrepancy results if there has been some error in transmission.

Next, TCP hands each segment over to IP, with instructions to transmit it to B. These segments must be transmitted across one or more subnetworks and relayed through one or more intermediate routers. This operation, too, requires the use of control information. Thus IP appends a header of control information to each segment to form an **IP datagram**. An example of an item stored in the IP header is the destination host address (in this example, B).

Finally, each IP datagram is presented to the network access layer for transmission across the first subnetwork in its journey to the destination. The network access layer appends its own header, creating a packet, or frame. The packet is transmitted across the subnetwork to router J. The packet header contains the information that the subnetwork needs to transfer the data across the subnetwork. Examples of items that may be contained in this header include:

- **Destination subnetwork address:** The subnetwork must know to which attached device the packet is to be delivered.
- **Facilities requests:** The network access protocol might request the use of certain subnetwork facilities, such as priority.

At router J, the packet header is stripped off and the IP header examined. On the basis of the destination address information in the IP header, the IP module in the router directs the datagram out across subnetwork 2 to B. To do this, the datagram is again augmented with a network access header.

When the data are received at B, the reverse process occurs. At each layer, the corresponding header is removed, and the remainder is passed on to the next higher layer, until the original user data are delivered to the destination process.

TCP and UDP

For most applications running as part of the TCP/IP protocol architecture, the transport layer protocol is TCP. TCP provides a reliable connection for the transfer of data between applications. A connection is simply a temporary logical association between two entities in different systems. A logical connection refers to a given pair of port values. For the duration of the connection each entity keeps track of TCP segments coming and going to the other entity, in order to regulate the flow of segments and to recover from lost or damaged segments.

Figure 2.3a shows the header format for TCP, which is a minimum of 20 octets, or 160 bits. The Source Port and Destination Port fields identify the applications at the source and destination systems that are using this connection. The Sequence Number, Acknowledgment Number, and Window fields provide flow control and error control. The checksum is a 16-bit frame check sequence used to detect errors in the TCP segment. Chapter 20 provides more details.

In addition to TCP, there is one other transport-level protocol that is in common use as part of the TCP/IP protocol suite: the User Datagram Protocol (UDP). UDP does not guarantee delivery, preservation of sequence, or protection against

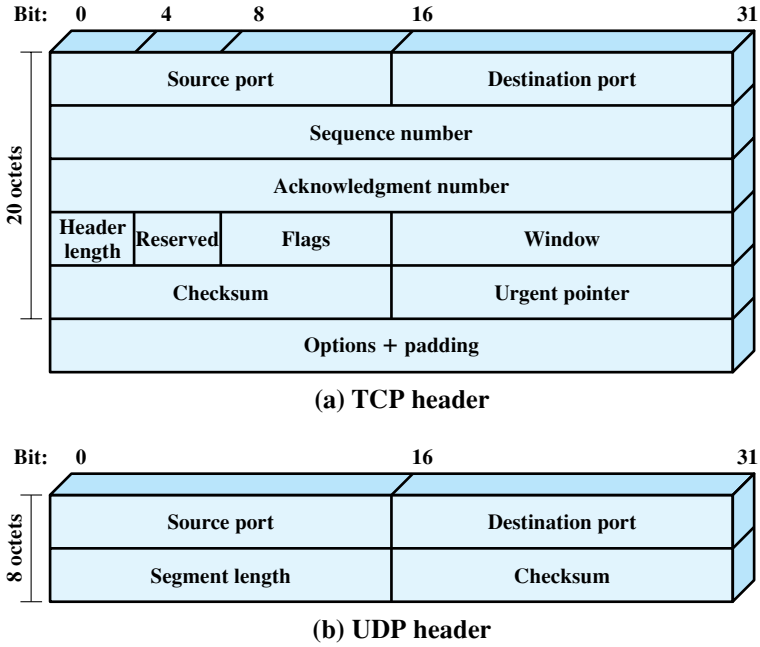


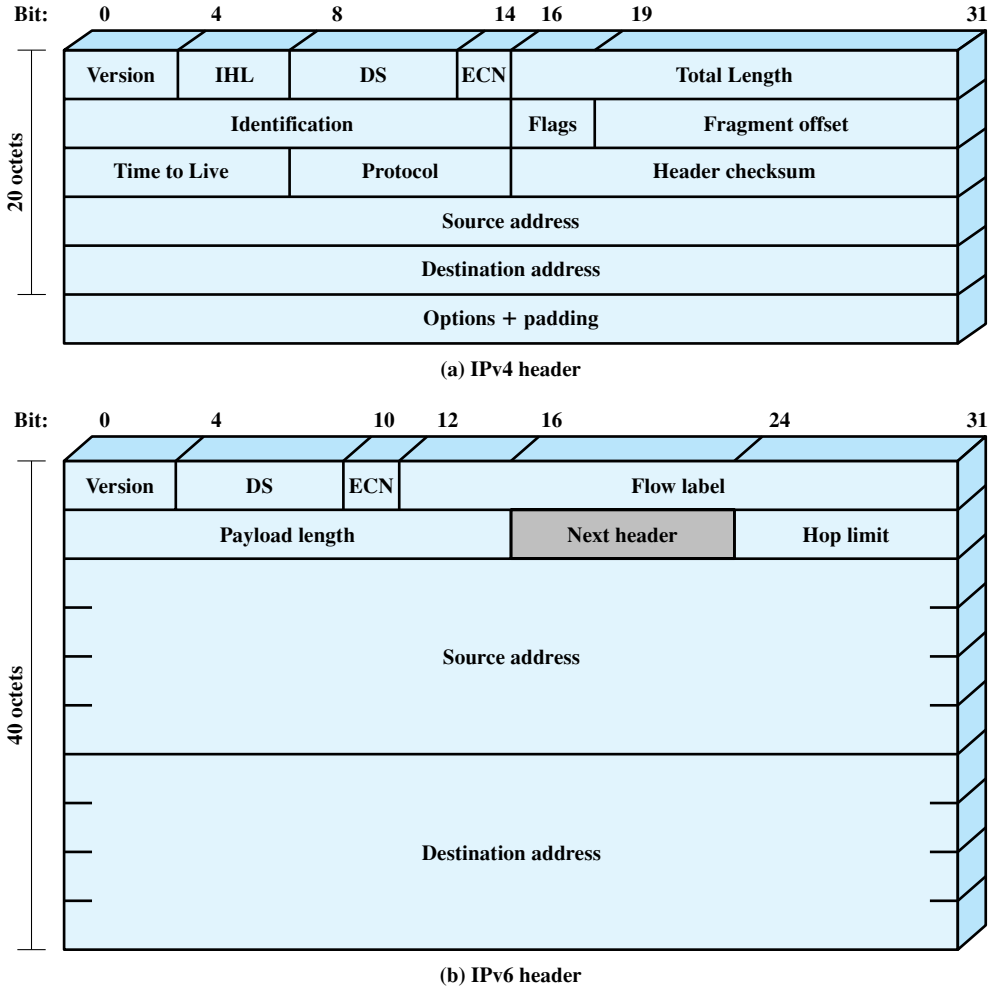
Figure 2.3 TCP and UDP Headers

duplication. UDP enables a procedure to send messages to other procedures with a minimum of protocol mechanism. Some transaction-oriented applications make use of UDP; one example is SNMP (Simple Network Management Protocol), the standard network management protocol for TCP/IP networks. Because it is connectionless, UDP has very little to do. Essentially, it adds a port addressing capability to IP. This is best seen by examining the UDP header, shown in Figure 2.3b. UDP also includes a checksum to verify that no error occurs in the data; the use of the checksum is optional.

IP and IPv6

For decades, the keystone of the TCP/IP protocol architecture has been IP. Figure 2.4a shows the IP header format, which is a minimum of 20 octets, or 160 bits. The header, together with the segment from the transport layer, forms an IP-level PDU referred to as an IP datagram or an IP packet. The header includes 32-bit source and destination addresses. The Header Checksum field is used to detect errors in the header to avoid misdelivery. The Protocol field indicates which higher-layer protocol is using IP. The ID, Flags, and Fragment Offset fields are used in the fragmentation and reassembly process. Chapter 18 provides more detail.

In 1995, the Internet Engineering Task Force (IETF), which develops protocol standards for the Internet, issued a specification for a next-generation IP, known then as IPng. This specification was turned into a standard in 1996 known as IPv6. IPv6 provides a number of functional enhancements over the existing IP, designed



DS = Differentiated services field
ECN = Explicit congestion notification field

Note: The 8-bit DS/ECN fields were formerly known as the Type of Service field in the IPv4 header and the Traffic Class field in the IPv6 header.

Figure 2.4 IP Headers

to accommodate the higher speeds of today's networks and the mix of data streams, including graphic and video, that are becoming more prevalent. But the driving force behind the development of the new protocol was the need for more addresses. The current IP uses a 32-bit address to specify a source or destination. With the explosive growth of the Internet and of private networks attached to the Internet, this address length became insufficient to accommodate all systems needing addresses. As Figure 2.4b shows, IPv6 includes 128-bit source and destination address fields.

Ultimately, all installations using TCP/IP are expected to migrate from the current IP to IPv6, but this process will take many years, if not decades.

TCP/IP Applications

A number of applications have been standardized to operate on top of TCP. We mention three of the most common here.

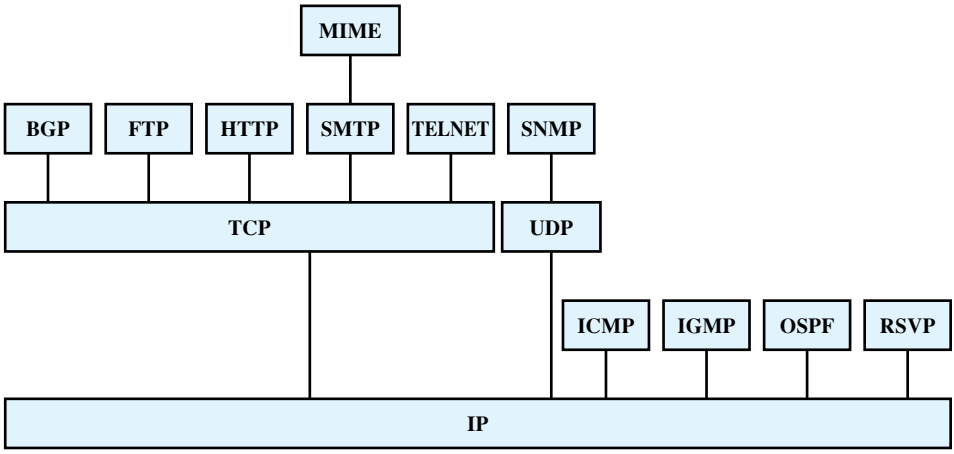
The **Simple Mail Transfer Protocol (SMTP)** provides a basic electronic mail transport facility. It provides a mechanism for transferring messages among separate hosts. Features of SMTP include mailing lists, return receipts, and forwarding. The SMTP protocol does not specify the way in which messages are to be created; some local editing or native electronic mail facility is required. Once a message is created, SMTP accepts the message and makes use of TCP to send it to an SMTP module on another host. The target SMTP module will make use of a local electronic mail package to store the incoming message in a user's mailbox.

The **File Transfer Protocol (FTP)** is used to send files from one system to another under user command. Both text and binary files are accommodated, and the protocol provides features for controlling user access. When a user wishes to engage in file transfer, FTP sets up a TCP connection to the target system for the exchange of control messages. This connection allows user ID and password to be transmitted and allows the user to specify the file and file actions desired. Once a file transfer is approved, a second TCP connection is set up for the data transfer. The file is transferred over the data connection, without the overhead of any headers or control information at the application level. When the transfer is complete, the control connection is used to signal the completion and to accept new file transfer commands.

TELNET provides a remote logon capability, which enables a user at a terminal or personal computer to logon to a remote computer and function as if directly connected to that computer. The protocol was designed to work with simple scroll-mode terminals. TELNET is actually implemented in two modules: User TELNET interacts with the terminal I/O module to communicate with a local terminal. It converts the characteristics of real terminals to the network standard and vice versa. Server TELNET interacts with an application, acting as a surrogate terminal handler so that remote terminals appear as local to the application. Terminal traffic between User and Server TELNET is carried on a TCP connection.

Protocol Interfaces Each layer in the TCP/IP protocol suite interacts with its immediate adjacent layers. At the source, the application layer makes use of the services of the end-to-end layer and provides data down to that layer. A similar relationship exists at the interface of the end-to-end and internet layers and at the interface of the internet and network access layers. At the destination, each layer delivers data up to the next higher layer.

This use of each individual layer is not required by the architecture. As Figure 2.5 suggests, it is possible to develop applications that directly invoke the services of any one of the layers. Most applications require a reliable end-to-end protocol and thus make use of TCP. Some special-purpose applications do not need the services of TCP. Some of these applications, such as the Simple Network Management Protocol (SNMP), use an alternative end-to-end protocol known as the User Datagram Protocol (UDP); others may make use of IP directly. Applications that do not involve internetworking and that do not need TCP have been developed to invoke the network access layer directly.



BGP = Border Gateway Protocol
FTP = File Transfer Protocol
HTTP = Hypertext Transfer Protocol
ICMP = Internet Control Message Protocol
IGMP = Internet Group Management Protocol
IP = Internet Protocol
MIME = Multipurpose Internet Mail Extension
OSPF = Open Shortest Path First
RSVP = Resource ReSerVation Protocol
SMTP = Simple Mail Transfer Protocol
SNMP = Simple Network Management Protocol
TCP = Transmission Control Protocol
UDP = User Datagram Protocol

Figure 2.5 Some Protocols in the TCP/IP Protocol Suite

2.3 THE OSI MODEL

The Open Systems Interconnection (OSI) reference model was developed by the International Organization for Standardization (ISO)[2] as a model for a computer protocol architecture and as a framework for developing protocol standards. The OSI model consists of seven layers:

- Application
- Presentation
- Session
- Transport
- Network
- Data link
- Physical

Figure 2.6 illustrates the OSI model and provides a brief definition of the functions performed at each layer. The intent of the OSI model is that protocols be developed to perform the functions of each layer.

[2]ISO is not an acronym (in which case it would be IOS), but a word, derived from the Greek *isos*, meaning *equal*.

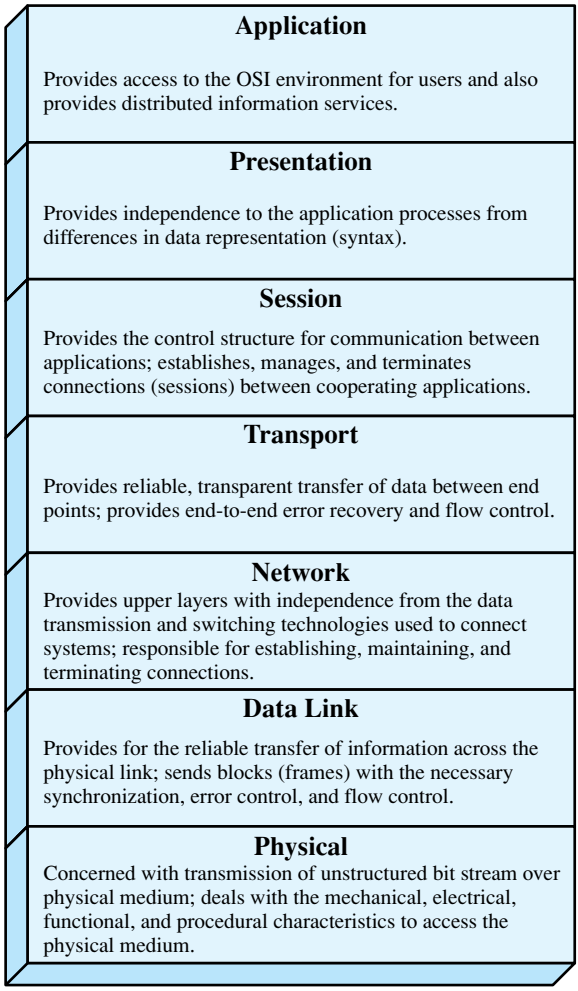


Figure 2.6 The OSI Layers

The designers of OSI assumed that this model and the protocols developed within this model would come to dominate computer communications, eventually replacing proprietary protocol implementations and rival multivendor models such as TCP/IP. This has not happened. Although many useful protocols have been developed in the context of OSI, the overall seven-layer model has not flourished. Instead, the TCP/IP architecture has come to dominate. There are a number of reasons for this outcome. Perhaps the most important is that the key TCP/IP protocols were mature and well tested at a time when similar OSI protocols were in the development stage. When businesses began to recognize the need for interoperability across networks, only TCP/IP was available and ready to go. Another reason is that the OSI model is unnecessarily complex, with seven layers to accomplish what TCP/IP does with fewer layers.

Figure 2.7 illustrates the layers of the TCP/IP and OSI architectures, showing roughly the correspondence in functionality between the two.

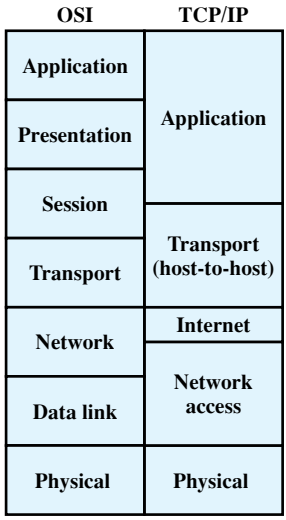


Figure 2.7 A Comparison of the OSI and TCP/IP Protocol Architectures

2.4 STANDARDIZATION WITHIN A PROTOCOL ARCHITECURE

Standardization within the OSI Framework[3]

The principal motivation for the development of the OSI model was to provide a framework for standardization. Within the model, one or more protocol standards can be developed at each layer. The model defines in general terms the functions to be performed at that layer and facilitates the standards-making process in two ways:

- Because the functions of each layer are well defined, standards can be developed independently and simultaneously for each layer. This speeds up the standards-making process.
- Because the boundaries between layers are well defined, changes in standards in one layer need not affect already existing software in another layer. This makes it easier to introduce new standards.

Figure 2.8 illustrates the use of the OSI model as such a framework. The overall communications function is decomposed into seven distinct layers. That is, the overall function is broken up into a number of modules, making the interfaces between modules as simple as possible. In addition, the design principle of information hiding is used: Lower layers are concerned with greater levels of

[3]The concepts introduced in this subsection apply as well to the TCP/IP architecture.

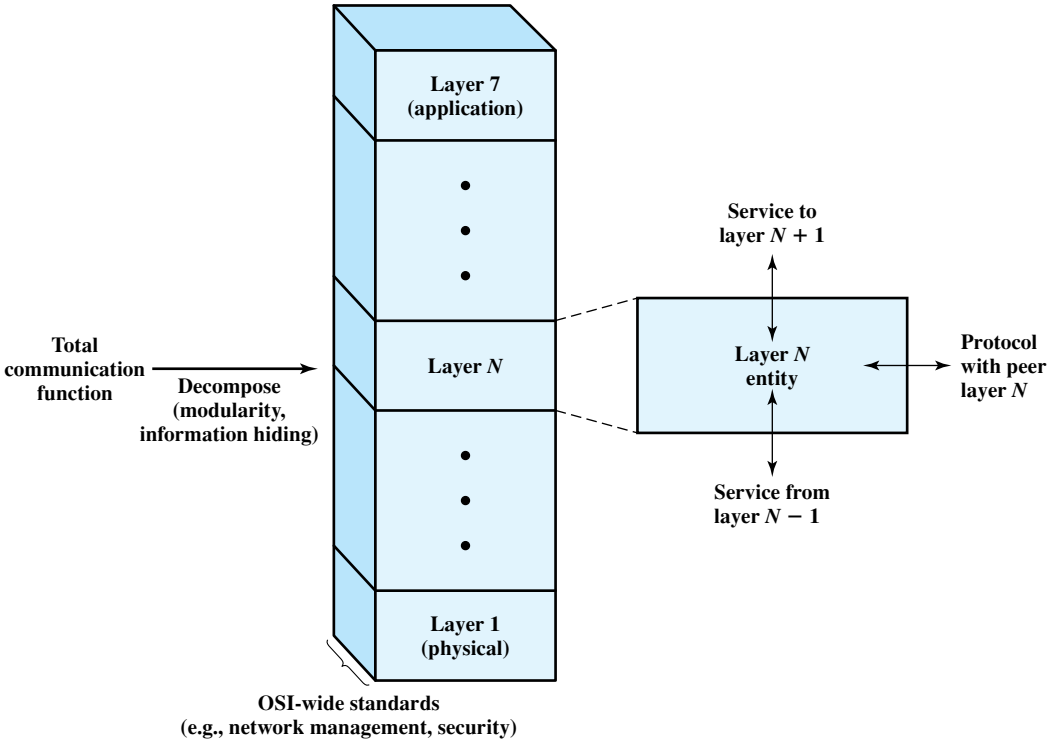


Figure 2.8 The OSI Architecture as a Framework for Standardization

detail; upper layers are independent of these details. Each layer provides services to the next higher layer and implements a protocol to the peer layer in other systems.

Figure 2.9 shows more specifically the nature of the standardization required at each layer. Three elements are key:

- **Protocol specification:** Two entities at the same layer in different systems cooperate and interact by means of a protocol. Because two different open systems are involved, the protocol must be specified precisely. This includes the format of the protocol data units exchanged, the semantics of all fields, and the allowable sequence of PDUs.
- **Service definition:** In addition to the protocol or protocols that operate at a given layer, standards are needed for the services that each layer provides to the next higher layer. Typically, the definition of services is equivalent to a functional description that defines what services are provided, but not how the services are to be provided.
- **Addressing:** Each layer provides services to entities at the next higher layer. These entities are referenced by means of a service access point (SAP). Thus, a network service access point (NSAP) indicates a transport entity that is a user of the network service.

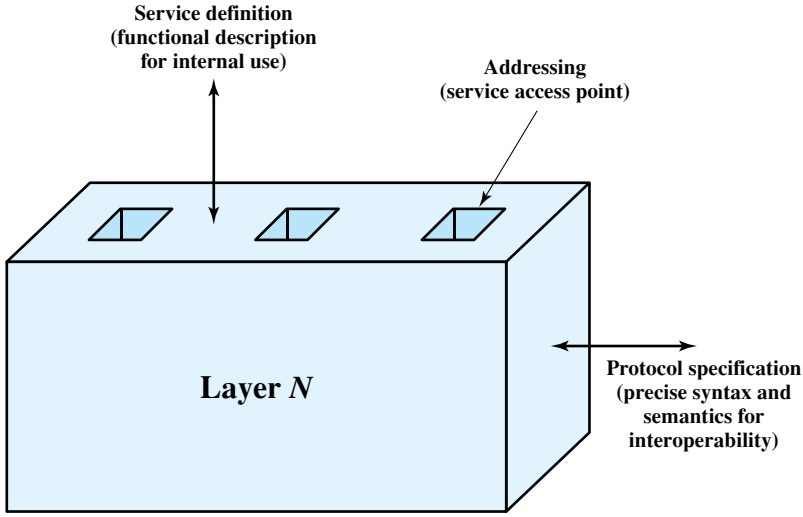


Figure 2.9 Layer-Specific Standards

The need to provide a precise protocol specification for open systems is self-evident. The other two items listed warrant further comment. With respect to service definitions, the motivation for providing only a functional definition is as follows. First, the interaction between two adjacent layers takes place within the confines of a single open system and is not the concern of any other open system. Thus, as long as peer layers in different systems provide the same services to their next higher layers, the details of how the services are provided may differ from one system to another without loss of interoperability. Second, it will usually be the case that adjacent layers are implemented on the same processor. In that case, we would like to leave the system programmer free to exploit the hardware and operating system to provide an interface that is as efficient as possible.

With respect to addressing, the use of an address mechanism at each layer, implemented as a service access point, allows each layer to multiplex multiple users from the next higher layer. Multiplexing may not occur at each layer, but the model allows for that possibility.

Service Primitives and Parameters

The services between adjacent layers in the OSI architecture are expressed in terms of primitives and parameters. A primitive specifies the function to be performed, and the parameters are used to pass data and control information. The actual form of a primitive is implementation dependent. An example is a procedure call.

Four types of primitives are used in standards to define the interaction between adjacent layers in the architecture. These are defined in Table 2.1. The layout of Figure 2.10a suggests the time ordering of these events. For example, consider

Table 2.1 Service Primitive Types

Request	A primitive issued by a service user to invoke some service and to pass the parameters needed to specify fully the requested service
Indication	A primitive issued by a service provider either to 1. indicate that a procedure has been invoked by the peer service user on the connection and to provide the associated parameters, or 2. notify the service user of a provider-initiated action
Response	A primitive issued by a service user to acknowledge or complete some procedure previously invoked by an indication to that user
Confirm	A primitive issued by a service provider to acknowledge or complete some procedure previously invoked by a request by the service user

the transfer of data from an (*N*) entity to a peer (*N*) entity in another system. The following steps occur:

1. The source (*N*) entity invokes its (*N* – 1) entity with a *request* primitive. Associated with the primitive are the parameters needed, such as the data to be transmitted and the destination address.
2. The source (*N* – 1) entity prepares an (*N* – 1) PDU to be sent to its peer (*N* – 1) entity.
3. The destination (*N* – 1) entity delivers the data to the appropriate destination (*N*) entity via an *indication* primitive, which includes the data and source address as parameters.
4. If an acknowledgment is called for, the destination (*N*) entity issues a *response* primitive to its (*N* – 1) entity.
5. The (*N* – 1) entity conveys the acknowledgment in an (*N* – 1) PDU.
6. The acknowledgment is delivered to the (*N*) entity as a *confirm* primitive.

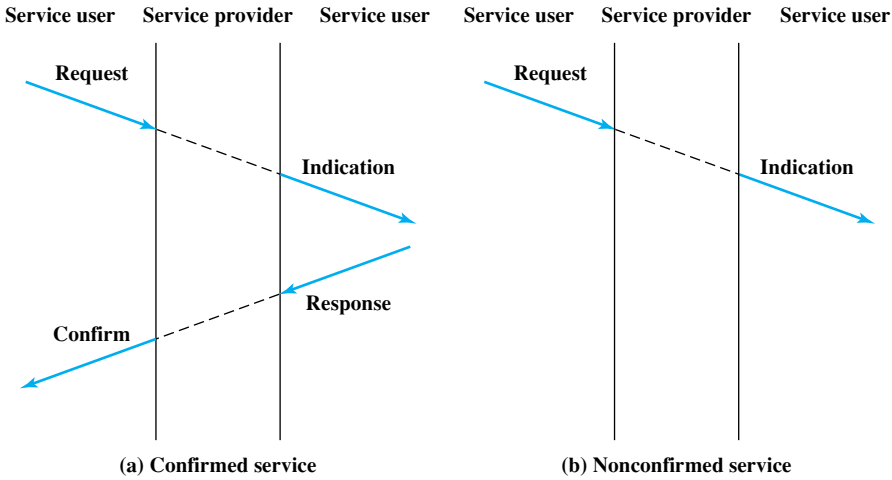


Figure 2.10 Time Sequence Diagrams for Service Primitives

This sequence of events is referred to as a **confirmed service**, as the initiator receives confirmation that the requested service has had the desired effect at the other end. If only request and indication primitives are involved (corresponding to steps 1 through 3), then the service dialogue is a **nonconfirmed service**; the initiator receives no confirmation that the requested action has taken place (Figure 2.10b).

2.5 TRADITIONAL INTERNET-BASED APPLICATIONS

A number of applications have been standardized to operate on top of TCP. We mention three of the most common here.

The **Simple Mail Transfer Protocol (SMTP)** provides a basic electronic mail transport facility. It provides a mechanism for transferring messages among separate hosts. Features of SMTP include mailing lists, return receipts, and forwarding. The SMTP protocol does not specify the way in which messages are to be created; some local editing or native electronic mail facility is required. Once a message is created, SMTP accepts the message and makes use of TCP to send it to an SMTP module on another host. The target SMTP module will make use of a local electronic mail package to store the incoming message in a user's mailbox.

The **File Transfer Protocol (FTP)** is used to send files from one system to another under user command. Both text and binary files are accommodated, and the protocol provides features for controlling user access. When a user wishes to engage in file transfer, FTP sets up a TCP connection to the target system for the exchange of control messages. This connection allows user ID and password to be transmitted and allows the user to specify the file and file actions desired. Once a file transfer is approved, a second TCP connection is set up for the data transfer. The file is transferred over the data connection, without the overhead of any headers or control information at the application level. When the transfer is complete, the control connection is used to signal the completion and to accept new file transfer commands.

TELNET provides a remote logon capability, which enables a user at a terminal or personal computer to logon to a remote computer and function as if directly connected to that computer. The protocol was designed to work with simple scroll-mode terminals. TELNET is actually implemented in two modules: User TELNET interacts with the terminal I/O module to communicate with a local terminal. It converts the characteristics of real terminals to the network standard and vice versa. Server TELNET interacts with an application, acting as a surrogate terminal handler so that remote terminals appear as local to the application. Terminal traffic between User and Server TELNET is carried on a TCP connection.

2.6 MULTIMEDA

With the increasing availability of broadband access to the Internet has come an increased interest in Web-based and Internet-based multimedia applications. The terms *multimedia* and *multimedia applications* are used rather loosely in the literature and in commercial publications, and no single definition of the term *multimedia* has been agreed (e.g., [JAIN94], [GRIM91], [PURC98], [PACK99]). For our purposes, the definitions in Table 2.2 provide a starting point.

Table 2.2 Multimedia Terminology

Media
Refers to the form of information and includes text, still images, audio, and video.
Multimedia
Human-computer interaction involving text, graphics, voice and video. Multimedia also refers to storage devices that are used to store multimedia content.
Streaming media
Refers to multimedia files, such as video clips and audio, that begin playing immediately or within seconds after it is received by a computer from the Internet or Web. Thus, the media content is consumed as it is delivered from the server rather than waiting until an entire file is downloaded.

One way to organize the concepts associated with multimedia is to look at a taxonomy that captures a number of dimensions of this field. Figure 2.11 looks at multimedia from the perspective of three different dimensions: type of media, applications, and the technology required to support the applications.

Media Types

Typically, the term *multimedia* refers to four distinct types of media: text, audio, graphics, and video.

From a communications perspective, the term **text** is self-explanatory, referring to information that can be entered via a keyboard and is directly readable and printable. Text messaging, instant messaging, and text (non-html) e-mail are common examples, as

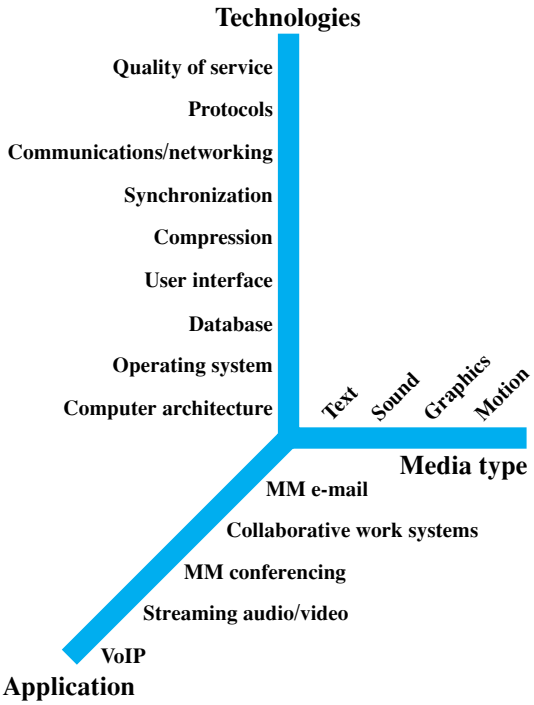


Figure 2.11 A Multimedia Taxonomy

are chat rooms and message boards. However, the term often is used in the broader sense of data that can be stored in files and databases and that does not fit into the other three categories. For example, an organization's database my contain files of numerical data, in which the data are stored in a more compact form than printable characters.

The term **audio** generally encompasses two different ranges of sound. Voice, or speech, refers to sounds that are produced by the human speech mechanism. Generally, a modest bandwidth (under 4 kHz) is required to transmit voice. Telephony and related applications (e.g., voice mail, audio teleconferencing, telemarketing) are the most common traditional applications of voice communications technology. A broader frequency spectrum is needed to support music applications, including the download of music files.

The **image** service supports the communication of individual pictures, charts, or drawings. Image-based applications include facsimile, computer-aided design (CAD), publishing, and medical imaging. Images can be represented in a vector graphics format, such as is used in drawing programs and PDF files. In a raster graphics format, an image is represented as a two-dimensional array of spots, called pixels.[4] The compressed JPG format is derived from a raster graphics format.

The **video** service carries sequences of pictures in time. In essence, video makes use of a sequence of raster-scan images.

Multimedia Applications

The Internet, until recently, has been dominated by information retrieval applications, e-mail, and file transfer, plus Web interfaces that emphasized text and images. Increasingly, the Internet is being used for multimedia applications that involve massive amounts of data for visualization and support of real-time interactivity. Streaming audio and video are perhaps the best known of such applications. An example of an interactive application is a virtual training environment involving distributed simulations and real-time user interaction [VIN98]. Some other examples are shown in Table 2.3.

[GONZ00] lists the following multimedia application domains:

- **Multimedia information systems:** Databases, information kiosks, hypertexts, electronic books, and multimedia expert systems
- **Multimedia communication systems:** Computer-supported collaborative work, videoconferencing, streaming media, and multimedia teleservices
- **Multimedia entertainment systems:** 3D computer games, multiplayer network games, infotainment, and interactive audiovisual productions
- **Multimedia business systems:** Immersive electronic commerce, marketing, multimedia presentations, video brochures, virtual shopping, and so on.
- **Multimedia educational systems:** Electronic books, flexible teaching materials, simulation systems, automatic testing, distance learning, and so on.

One point worth noting is highlighted in Figure 2.11. Although traditionally the term *multimedia* has connoted the simultaneous use of multiple media types (e.g., video annotation of a text document), the term has also come to refer to applications that require real-time processing or communication of video or audio

[4]A pixel, or picture element, is the smallest element of a digital image that can be assigned a gray level. Equivalently, a pixel is an individual dot in a dot-matrix representation of a picture.

Table 2.3 Domains of Multimedia Systems and Example Applications

Domain	Example Application
Information management	Hypermedia, multimedia-capable databases, content-based retrieval
Entertainment	Computer games, digital video, audio (MP3)
Telecommunication	Videoconferencing, shared workspaces, virtual communities
Information publishing/delivery	Online training, electronic books, streaming media

alone. Thus, voice over IP (VoIP), streaming audio, and streaming video are considered multimedia applications even though each involves a single media type.

Elastic and Inelastic Traffic

Before discussing multimedia technologies, it will be useful to look at a key consideration, namely the type of network traffic generated by various media and applications.

Traffic on a network or internet can be divided into two broad categories: elastic and inelastic. A consideration of their differing requirements clarifies the need for an enhanced internet architecture.

Elastic traffic can adjust, over wide ranges, to changes in delay and throughput across an internet and still meet the needs of its applications. This is the traditional type of traffic supported on TCP/IP-based internets and is the type of traffic for which internets were designed. With TCP, traffic on individual connections adjusts to congestion by reducing the rate at which data are presented to the network.

Elastic applications include common Internet-based applications, such as file transfer, electronic mail, remote logon, network management, and Web access. But there are differences among the requirements of these applications. For example,

- E-mail is generally quite insensitive to changes in delay.
- When file transfer is done online, as it frequently is, the user expects the delay to be proportional to the file size and so is sensitive to changes in throughput.
- With network management, delay is generally not a serious concern. However, if failures in an internet are the cause of congestion, then the need for network management messages to get through with minimum delay increases with increased congestion.
- Interactive applications, such as remote logon and Web access, are quite sensitive to delay.

So, even if we confine our attention to elastic traffic, an Internet service that can allocate resources to traffic streams based on need, rather than just providing equal allocation, is useful.

Inelastic traffic does not easily adapt, if at all, to changes in delay and throughput across an internet. The prime example is real-time traffic, such as voice and video. The requirements for inelastic traffic may include the following:

- **Throughput:** A minimum throughput value may be required. Unlike most elastic traffic, which can continue to deliver data with perhaps degraded service, many inelastic applications require a firm minimum throughput.

- **Delay:** An example of a delay-sensitive application is stock trading; someone who consistently receives later service will consistently act later, and with greater disadvantage.
- **Delay variation:** The larger the allowable delay, the longer the real delay in delivering the data and the greater the size of the delay buffer required at receivers. Real-time interactive applications, such as teleconferencing, may require a reasonable upper bound on delay variation.
- **Packet loss:** Real-time applications vary in the amount of packet loss, if any, that they can sustain.

These requirements are difficult to meet in an environment with variable queuing delays and congestion losses. Accordingly, inelastic traffic introduces two new requirements into the internet architecture. First, some means is needed to give preferential treatment to applications with more demanding requirements. Applications need to be able to state their requirements, either ahead of time in some sort of service request function, or on the fly, by means of fields in the IP packet header. A second requirement in supporting inelastic traffic in an internet architecture is that elastic traffic must still be supported.

Multimedia Technologies

Figure 2.11 lists some of the technologies that are relevant to the support of multimedia applications. As can be seen, a wide range of technologies is involved. The lowest four items on the list are beyond the scope of this book. The other items represent only a partial list of communications and networking technologies for multimedia. These technologies and others are explored throughout the book. Here, we give a brief comment on each area.

- **Compression:** Digitized video, and to a much lesser extent audio, can generate an enormous amount of traffic on a network. A streaming application, which is delivered to many users, magnifies the traffic. Accordingly, standards have been developed for producing significant savings through compression. The most notable such standards are JPG for still images and MPG for video. Compression is examined in Part Six.
- **Communications/networking:** This broad category refers to the transmission and networking technologies (e.g., SONET, ATM) that can support high-volume multimedia traffic.
- **Protocols:** A number of protocols are instrumental in supporting multimedia traffic. One example is the Real-time Transport Protocol (RTP), which is designed to support inelastic traffic. RTP uses buffering and discarding strategies to assure that real-time traffic is received by the end user in a smooth continuous stream. Another example is the Session Initiation Protocol (SIP), an application-level control protocol for setting up, modifying, and terminating real-time sessions between participants over an IP data network.
- **Quality of service (QoS):** The Internet and its underlying local area and wide area networks must include a QoS capability to provide differing levels of service

to different types of application traffic. A QoS capability can deal with priority, delay constraints, delay variability constraints, and other similar requirements.

All of these matters are explored subsequently in this text.

2.7 RECOMMENDED READING AND WEB SITES

For the reader interested in greater detail on TCP/IP, there are two three-volume works that are more than adequate. The works by Comer and Stevens have become classics and are considered definitive [COME06, COME99, COME01]. The works by Stevens and Wright are equally worthwhile and more detailed with respect to protocol operation [STEV94, STEV96, WRIG95]. A more compact and very useful reference work is [RODR02], which covers the spectrum of TCP/IP-related protocols in a technically concise but thorough fashion, including coverage of some protocols not found in the other two works.

[GREE80] is a good tutorial overview of the concept of a layered protocol architecture. Two early papers that provide good discussions of the design philosophy of the TCP/IP protocol suite are [LEIN85] and [CLAR88].

Although somewhat dated, [FURH94] remains a good overview of multimedia topics. [VOGE95] is a good introduction to QoS considerations for multimedia. [HELL01] is a lengthy and worthwhile theoretical treatment of multimedia.

CLAR88 Clark, D. "The Design Philosophy of the DARPA Internet Protocols." *ACM SIGCOMM Computer Communications Review,* August 1988.

COME99 Comer, D., and Stevens, D. *Internetworking with TCP/IP, Volume II: Design Implementation, and Internals*. Upper Saddle River, NJ: Prentice Hall, 1994.

COME01 Comer, D., and Stevens, D. *Internetworking with TCP/IP, Volume III: Client-Server Programming and Applications*. Upper Saddle River, NJ: Prentice Hall, 2001.

COME06 Comer, D. *Internetworking with TCP/IP, Volume I: Principles, Protocols, and Architecture*. Upper Saddle River, NJ: Prentice Hall, 2006.

FURH94 Furht, B. "Multimedia Systems: An Overview." *IEEE Multimedia,* Spring 1994.

GREE80 Green, P. "An Introduction to Network Architecture and Protocols." *IEEE Transactions on Communications,* April 1980.

HELL01 Heller, R., et al. "Using a Theoretical Multimedia Taxonomy Framework." *ACM Journal of Educational Resources in Computing,* Spring 2001.

LEIN85 Leiner, B.; Cole, R.; Postel, J.; and Mills, D. "The DARPA Internet Protocol Suite." *IEEE Communications Magazine,* March 1985.

RODR02 Rodriguez, A., et al. *TCP/IP Tutorial and Technical Overview*. Upper Saddle River: NJ: Prentice Hall, 2002.

STEV94 Stevens, W. *TCP/IP Illustrated, Volume 1: The Protocols.* Reading, MA: Addison-Wesley, 1994.

STEV96 Stevens, W. *TCP/IP Illustrated, Volume 3: TCP for Transactions, HTTP, NNTP, and the UNIX(R) Domain Protocol.* Reading, MA: Addison-Wesley, 1996.

VOGE95 Vogel, A., et al. "Distributed Multimedia and QoS: A Survey." *IEEE Multimedia,* Summer 1995.

WRIG95 Wright, G., and Stevens, W. *TCP/IP Illustrated, Volume 2: The Implementation*. Reading, MA: Addison-Wesley, 1995.

Recommended Web sites:[5]

- **TCP/IP Resources List**: A useful collection of FAQs, tutorials, guides, Web sites, and books about TCP/IP.
- **Networking Links**: Excellent collection of links related to TCP/IP.
- **Bongo Project**: Running IP over bongo drums. An excellent demonstration of the flexibility of a layered protocol architecture and a source of ideas for projects.

2.8 KEY TERMS, REVIEW QUESTIONS, AND PROBLEMS

Key Terms

application layer	network layer	quality of service (QoS)
checksum	Open Systems Interconnection (OSI)	router
data link layer	peer layer	service access point (SAP)
elastic traffic	physical layer	session layer
header	port	subnetwork
inelastic traffic	presentation layer	Transmission Control Protocol (TCP)
Internet	protocol	transport layer
Internet Protocol (IP)	protocol architecture	User Datagram Protocol (UDP)
Internetworking	protocol data unit (PDU)	
multimedia		

Review Questions

2.1. What is the major function of the network access layer?
2.2. What tasks are performed by the transport layer?
2.3. What is a protocol?
2.4. What is a protocol data unit (PDU)?
2.5. What is a protocol architecture?
2.6. What is TCP/IP?
2.7. What are some advantages to layering as seen in the TCP/IP architecture?
2.8. What is a router?
2.9. Which version of IP is the most prevalent today?
2.10. Does all traffic running on the Internet use TCP?
2.11. Compare the address space between IPv4 and IPv6. How many bits are used in each?

[5]Because URLs sometimes change, they are not included. For all of the Web sites listed in this and subsequent chapters, the appropriate link is at this book's Web site at williamstallings.com/DCC/DCC8e.html.

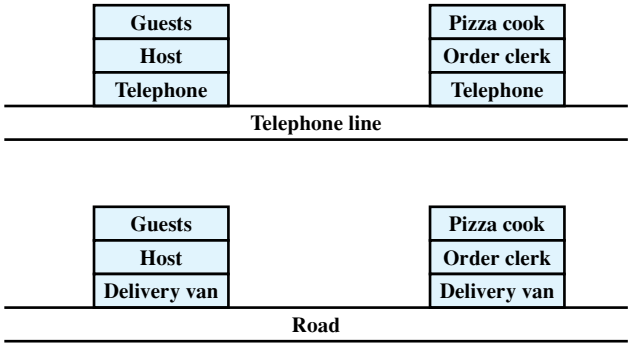


Figure 2.12 Architecture for Problem 2.1

Problems

2.1 Using the layer models in Figure 2.12, describe the ordering and delivery of a pizza, indicating the interactions at each level.

2.2 **a.** The French and Chinese prime ministers need to come to an agreement by telephone, but neither speaks the other's language. Further, neither has on hand a translator that can translate to the language of the other. However, both prime ministers have English translators on their staffs. Draw a diagram similar to Figure 2.12 to depict the situation, and describe the interaction and each level.

b. Now suppose that the Chinese prime minister's translator can translate only into Japanese and that the French prime minister has a German translator available. A translator between German and Japanese is available in Germany. Draw a new diagram that reflects this arrangement and describe the hypothetical phone conversation.

2.3 List the major disadvantages with the layered approach to protocols.

2.4 Two blue armies are each poised on opposite hills preparing to attack a single red army in the valley. The red army can defeat either of the blue armies separately but will fail to defeat both blue armies if they attack simultaneously. The blue armies communicate via an unreliable communications system (a foot soldier). The commander with one of the blue armies would like to attack at noon. His problem is this: If he sends a message to the other blue army, ordering the attack, he cannot be sure it will get through. He could ask for acknowledgment, but that might not get through. Is there a protocol that the two blue armies can use to avoid defeat?

2.5 A broadcast network is one in which a transmission from any one attached station is received by all other attached stations over a shared medium. Examples are a bus-topology local area network, such as Ethernet, and a wireless radio network. Discuss the need or lack of need for a network layer (OSI layer 3) in a broadcast network.

2.6 In Figure 2.2, exactly one protocol data unit (PDU) in layer N is encapsulated in a PDU at layer $(N - 1)$. It is also possible to break one N-level PDU into multiple $(N - 1)$-level PDUs (segmentation) or to group multiple N-level PDUs into one $(N - 1)$-level PDU (blocking).

a. In the case of segmentation, is it necessary that each $(N - 1)$-level segment contain a copy of the N-level header?

b. In the case of blocking, is it necessary that each N-level PDU retain its own header, or can the data be consolidated into a single N-level PDU with a single N-level header?

2.7 A TCP segment consisting of 1500 bits of data and 160 bits of header is sent to the IP layer, which appends another 160 bits of header. This is then transmitted through two networks, each of which uses a 24-bit packet header. The destination network has a maximum packet size of 800 bits. How many bits, including headers, are delivered to the network layer protocol at the destination?

2.8 Why is UDP needed? Why can't a user program directly access IP?

2.9 IP, TCP, and UDP all discard a packet that arrives with a checksum error and do not attempt to notify the source. Why?

2.10 Why does the TCP header have a header length field while the UDP header does not?

2.11 The previous version of the TFTP specification, RFC 783, included the following statement:

> All packets other than those used for termination are acknowledged individually unless a timeout occurs.

The RFC 1350 specification revises this to say:

> All packets other than duplicate ACK's and those used for termination are acknowledged unless a timeout occurs.

The change was made to fix a problem referred to as the "Sorcerer's Apprentice." Deduce and explain the problem.

2.12 What is the limiting factor in the time required to transfer a file using TFTP?

2.13 A user on a UNIX host wants to transfer a 4000-byte text file to a Microsoft Windows host. In order to do this, he transfers the file by means of TFTP, using the netascii transfer mode. Even though the transfer was reported as being performed successfully, the Windows host reports the resulting file size is 4050 bytes, rather than the original 4000 bytes. Does this difference in the file sizes imply an error in the data transfer? Why or why not?

2.14 The TFTP specification (RFC 1350) states that the transfer identifiers (TIDs) chosen for a connection should be randomly chosen, so that the probability that the same number is chosen twice in immediate succession is very low. What would be the problem of using the same TIDs twice in immediate succession?

2.15 In order to be able retransmit lost packets, TFTP must keep a copy of the data it sends. How many packets of data must TFTP keep at a time to implement this retransmission mechanism?

2.16 TFTP, like most protocols, will never send an error packet in response to an error packet it receives. Why?

2.17 We have seen that in order to deal with lost packets, TFTP implements a timeout-and-retransmit scheme, by setting a retransmission timer when it transmits a packet to the remote host. Most TFTP implementations set this timer to a fixed value of about 5 seconds. Discuss the advantages and the disadvantages of using a fixed value for the retransmission timer.

2.18 TFTP's timeout-and-retransmission scheme implies that all data packets will eventually be received by the destination host. Will these data also be received uncorrupted? Why or why not?

2.19 This chapter mentions the use of Frame Relay as a specific protocol or system used to connect to a wide area network. Each organization will have a certain collection of services available (like Frame Relay) but this is dependent upon provider provisioning, cost and customer premises equipment. What are some of the services available to you in your area?

Note: The following problem concern materials in Appendix H.

2.20 Based on the principles enunciated in Table H.1,

a. Design an architecture with eight layers and make a case for it.

b. Design one with six layers and make a case for that.

APPENDIX 2A THE TRIVIAL FILE TRANSFER PROTOCOL

This appendix provides an overview of the Internet standard Trivial File Transfer Protocol (TFTP), defined in RFC 1350. Our purpose is to give the reader some flavor for the elements of a protocol. TFTP is simple enough to provide a concise example, but includes most of the significant elements found in other, more complex, protocols.

Introduction to TFTP

TFTP is far simpler than the Internet standard FTP (RFC 959). There are no provisions for access control or user identification, so TFTP is only suitable for public access file directories. Because of its simplicity, TFTP is easily and compactly implemented. For example, some diskless devices use TFTP to download their firmware at boot time.

TFTP runs on top of UDP. The TFTP entity that initiates the transfer does so by sending a read or write request in a UDP segment with a destination port of 69 to the target system. This port is recognized by the target UDP module as the identifier of the TFTP module. For the duration of the transfer, each side uses a transfer identifier (TID) as its port number.

TFTP Packets

TFTP entities exchange commands, responses, and file data in the form of packets, each of which is carried in the body of a UDP segment. TFTP supports five types of packets (Figure 2.13); the first two bytes contains an opcode that identifies the packet type:

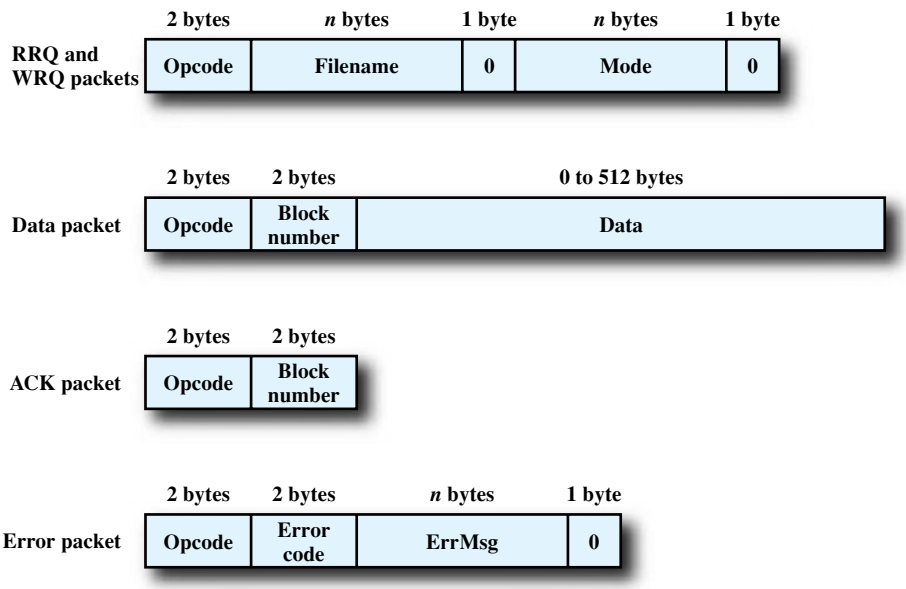


Figure 2.13 TFTP Packet Formats

Table 2.4 TFTP Error Codes

Value	Meaning
0	Not defined, see error message (if any)
1	File not found
2	Access violation
3	Disk full or allocation exceeded
4	Illegal TFTP operation
5	Unknown transfer ID
6	File already exists
7	No such user

- **RRQ:** The read request packet requests permission to transfer a file from the other system. The packet includes a file name, which is a sequence of ASCII[6] bytes terminated by a zero byte. The zero byte is the means by which the receiving TFTP entity knows when the file name is terminated. The packet also includes a mode field, which indicates whether the data file is to be interpreted as a string of ASCII bytes (netascii mode) or as raw 8-bit bytes (octet mode) of data. In netascii mode, the file is transferred as lines of characters, each terminated by a carriage return, line feed. Each system must translate between its own format for character files and the TFTP format.
- **WRQ:** The write request packet requests permission to transfer a file to the other system.
- **Data:** The block numbers on data packets begin with one and increase by one for each new block of data. This convention enables the program to use a single number to discriminate between new packets and duplicates. The data field is from zero to 512 bytes long. If it is 512 bytes long, the block is not the last block of data; if it is from zero to 511 bytes long, it signals the end of the transfer.
- **ACK:** This packet is used to acknowledge receipt of a data packet or a WRQ packet. An ACK of a data packet contains the block number of the data packet being acknowledged. An ACK of a WRQ contains a block number of zero.
- **Error:** An error packet can be the acknowledgment of any other type of packet. The error code is an integer indicating the nature of the error (Table 2.4). The error message is intended for human consumption and should be in ASCII. Like all other strings, it is terminated with a zero byte.

All packets other than duplicate ACKs (explained subsequently) and those used for termination are to be acknowledged. Any packet can be acknowledged by an error packet. If there are no errors, then the following conventions apply. A WRQ or a data packet is acknowledged by an ACK packet. When a RRQ is sent, the other side responds (in the absence of error) by beginning to transfer the file; thus, the first data block serves as an acknowledgment of the RRQ packet. Unless a file transfer is complete, each ACK packet from one side is followed by a data packet from the other, so that the data packet functions as an acknowledgment. An error packet can be acknowledged by any other kind of packet, depending on the circumstance.

[6]ASCII is the American Standard Code for Information Interchange, a standard of the American National Standards Institute. It designates a unique 7-bit pattern for each letter, with an eighth bit used for parity. ASCII is equivalent to the International Reference Alphabet (IRA), defined in ITU-T Recommendation T.50. See Appendix E for a discussion.

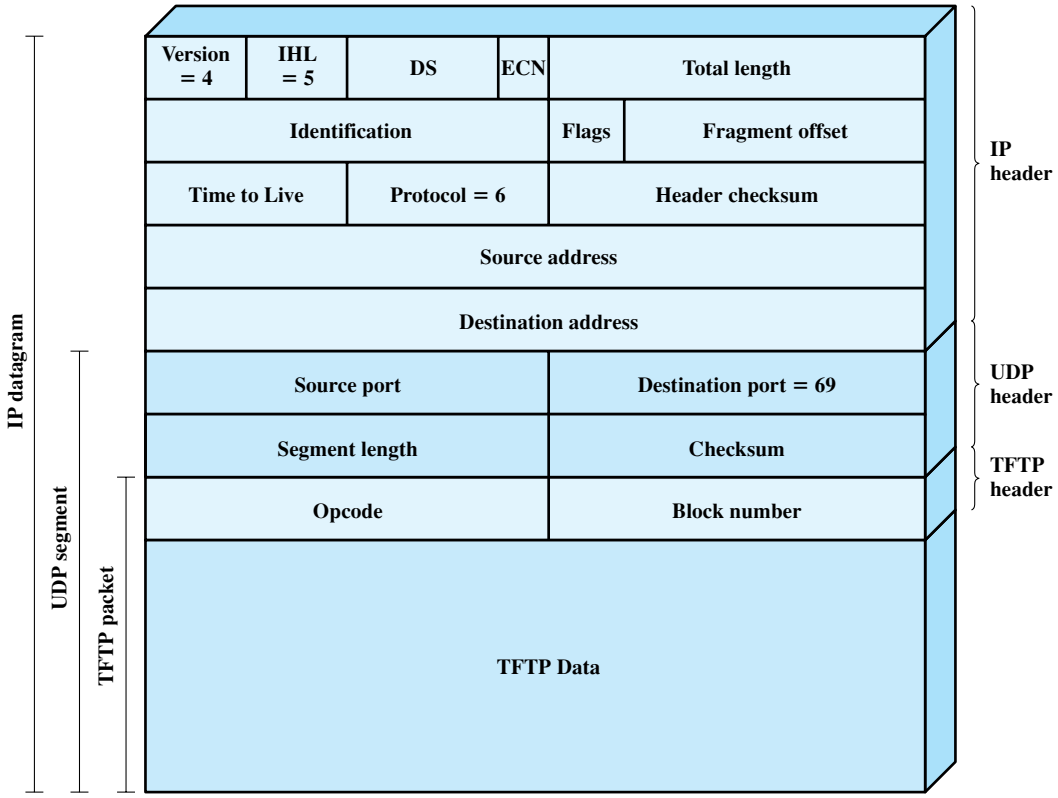


Figure 2.14 A TFTP Packet in Context

Figure 2.14 shows a TFTP data packet in context. When such a packet is handed down to UDP, UDP adds a header to form a UDP segment. This is then passed to IP, which adds an IP header to form an IP datagram.

Overview of a Transfer

The example illustrated in Figure 2.15 is of a simple file transfer operation from A to B. No errors occur and the details of the option specification are not explored.

The operation begins when the TFTP module in system A sends a write request (WRQ) to the TFTP module in system B. The WRQ packet is carried as the body of a UDP segment. The write request includes the name of the file (in this case, XXX) and a mode of octet, or raw data. In the UDP header, the destination port number is 69, which alerts the receiving UDP entity that this message is intended for the TFTP application. The source port number is a TID selected by A, in this case 1511. System B is prepared to accept the file and so responds with an ACK with a block number of 0. In the UDP header, the destination port is 1511, which enables the UDP entity at A to route the incoming packet to the TFTP module, which can match this TID with the TID in the WRQ. The source port is a TID selected by B for this file transfer, in this case 1660.

Following this initial exchange, the file transfer proceeds. The transfer consists of one or more data packets from A, each of which is acknowledged by B. The final data packet contains less than 512 bytes of data, which signals the end of the transfer.

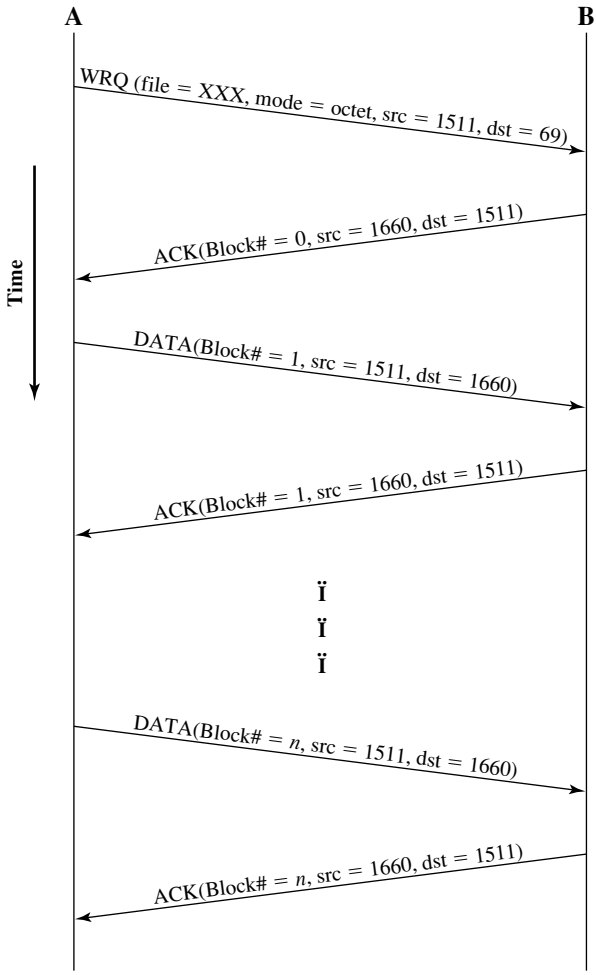


Figure 2.15 Example TFTP Operation

Errors and Delays

If TFTP operates over a network or internet (as opposed to a direct data link), it is possible for packets to be lost. Because TFTP operates over UDP, which does not provide a reliable delivery service, there needs to be some mechanism in TFTP to deal with lost packets. TFTP uses the common technique of a timeout mechanism. Suppose that A sends a packet to B that requires an acknowledgment (i.e., any packet other than duplicate ACKs and those used for termination). When A has transmitted the packet, it starts a timer. If the timer expires before the acknowledgment is received from B, A retransmits the same packet. If in fact the original packet was lost, then the retransmission will be the first copy of this packet received by B. If the original packet was not lost but the acknowledgment from B was lost, then B will receive two copies of the same packet from A and simply acknowledges both copies. Because of the use of block numbers, this causes no confusion. The only exception to this rule is for duplicate ACK packets. The second ACK is ignored.

Syntax, Semantics, and Timing

In Section 2.1, it was mentioned that the key features of a protocol can be classified as syntax, semantics, and timing. These categories are easily seen in TFTP. The formats of the various TFTP packets form the **syntax** of the protocol. The **semantics** of the protocol are shown in the definitions of each of the packet types and the error codes. Finally, the sequence in which packets are exchanged, the use of block numbers, and the use of timers are all aspects of the **timing** of TFTP.