CHAPTER 2

PROTOCOLS AND ARCHITECTURE

When computers, terminals, and/or other data processing devices exchange data, typical tasks to be performed are as follows:

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.

The exchange of information between computers for the purpose of cooperative action is referred to as computer communications. Similarly, when two or more computers are interconnected via a comm network, the set of computer stations is referred to as computer network.

In discussing computer communications and computer networks, we deal with two conceptes:

- Protocols
- Computer-communications architecture, or protocol architecture

A **protocol** is used for communication between entities in different entities in different systems.

Entities: An **entity** is anything capable of sending or receiving information, e.g. user application programs, file transfer packages, data-base management systems, electronic mail facilities, and terminals.
Systems: A **system** is a physically distinct object that contains one or more entities, e.g. computers, terminals and remote sensors.

The key elements of a protocol are:
• **Syntax**: Concerns the format of the data blocks
• **Semantics**: Includes control information for coordination and error handling
• **Timing**: Includes speed matching and sequencing

• 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.

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).

In general terms, communications can be said to involve three agents: applications, computers, and networks. The communication task is handled by 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 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.

Host-to-host layer, or **transport layer:** 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.

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.

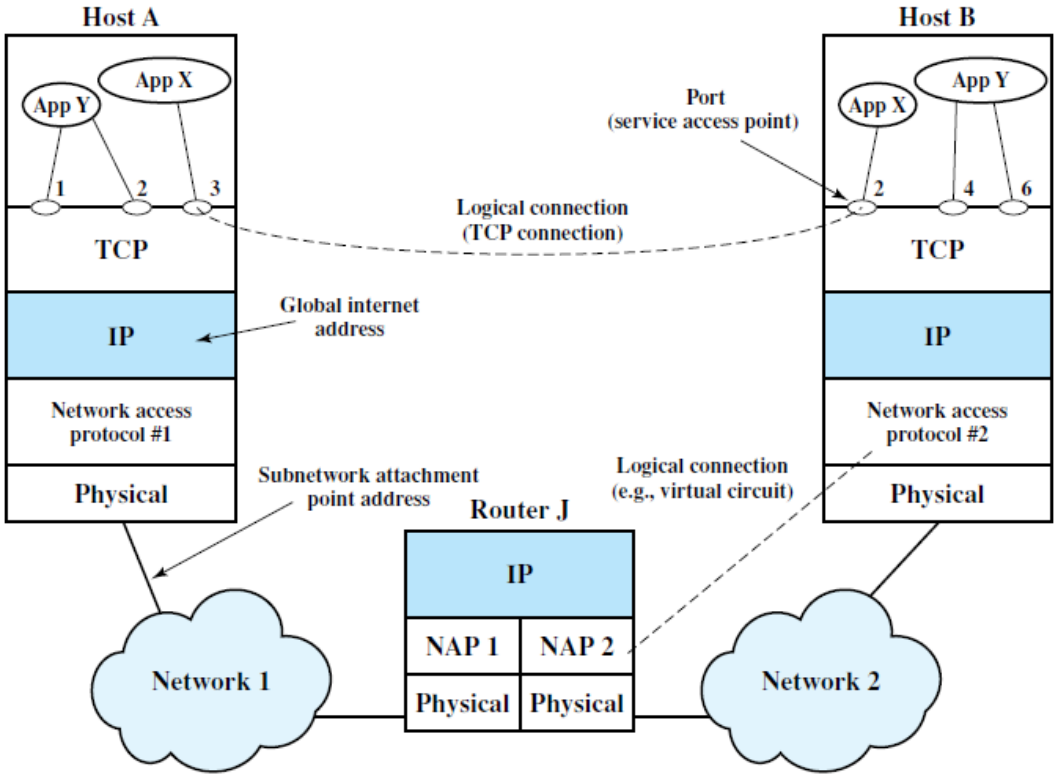


Figure 2.1 TCP/IP Concepts

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.
• **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.

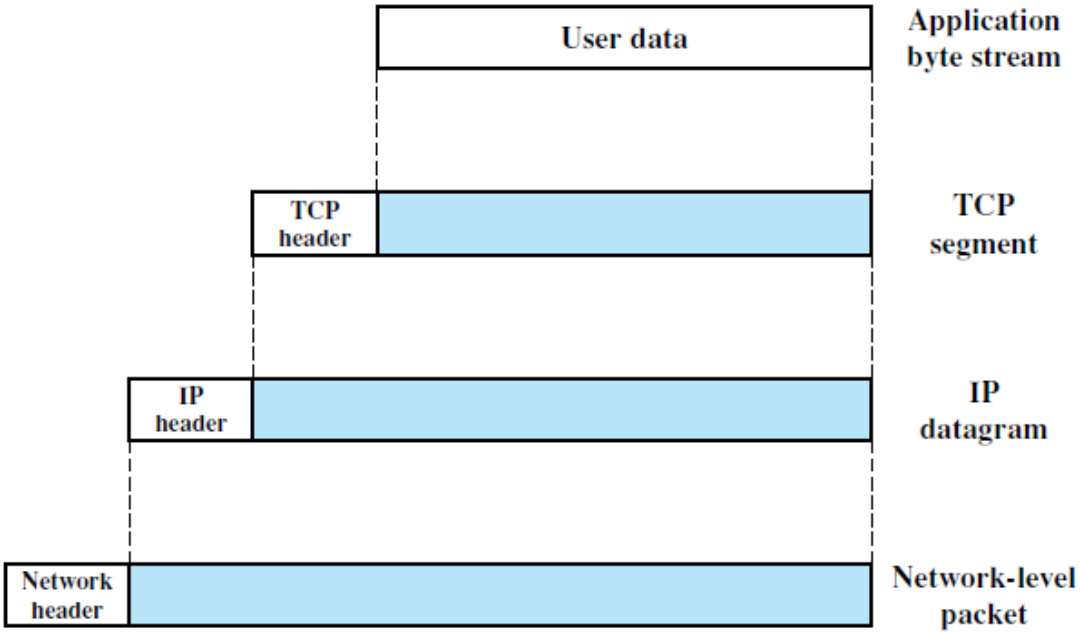


Figure 2.2 Protocol Data Units (PDUs) in the TCP/IP Architecture

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. Based on 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.

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 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.**

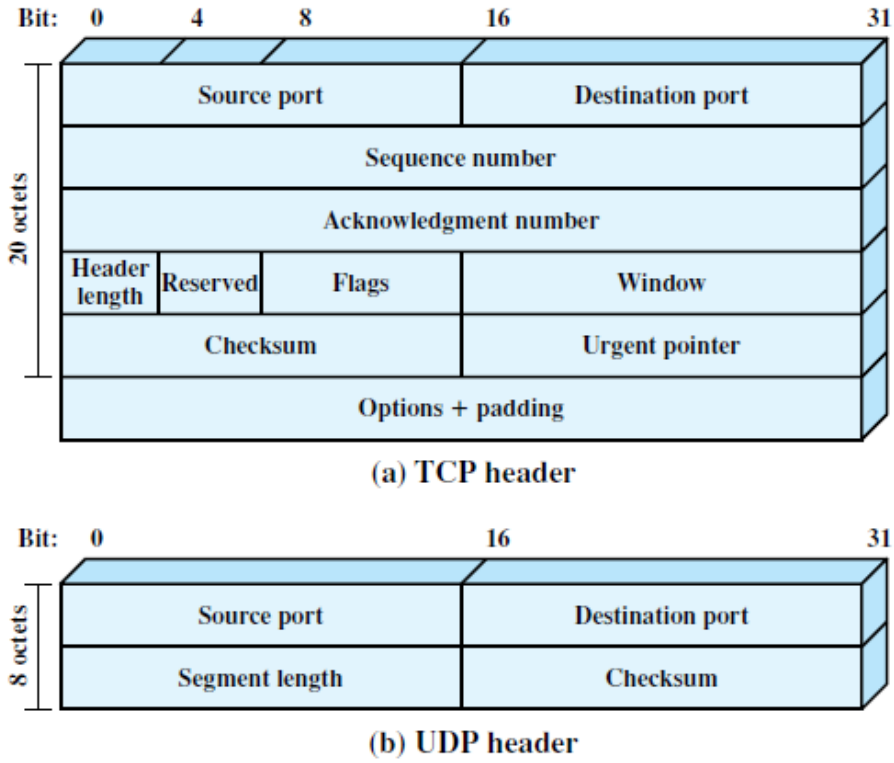


Figure 2.3 TCP and UDP Headers

TCP/IP Application: Simple Mail Transfer Protocol (SMTP)
File Transfer Protocol (FTP)
TELNET

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

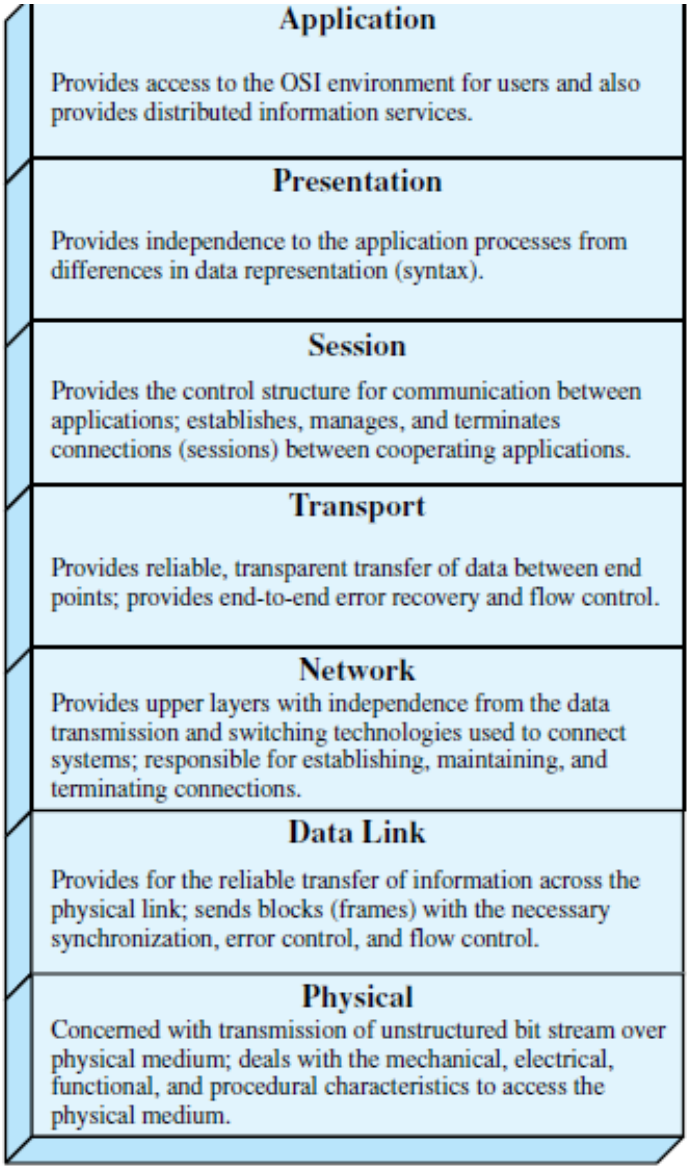


Figure 2.6 The OSI Layers

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. Instead, the TCP/IP architecture has come to dominate. There are a number of reasons for this outcome:

1. TCP/IP protocols were mature and well tested at a time when similar OSI protocols were in the development stage.
2. When businesses began to recognize the need for interoperability across networks, only TCP/IP was available and ready to go.
3. Another reason is that the OSI model is unnecessarily complex, with seven layers to accomplish what TCP/IP does with fewer layers.

COMPARISON OF OSI and TCP/IP PROTOCOL ARCHITECTURES

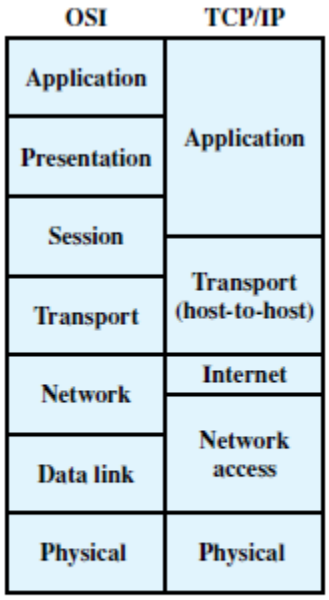


Figure 2.7 A Comparison of the OSI and TCP/IP Protocol Architectures

STANDARDIZATION WITHIN A PROTOCOL ARCHITECTURE

- Because the functions of each layer are well defined in OSI, 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.

The overall function is broken up into a number of modules, making the interfaces between modules as simple as possible. The design principle of information hiding is used: Lower layers are concerned with greater levels of 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.

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.

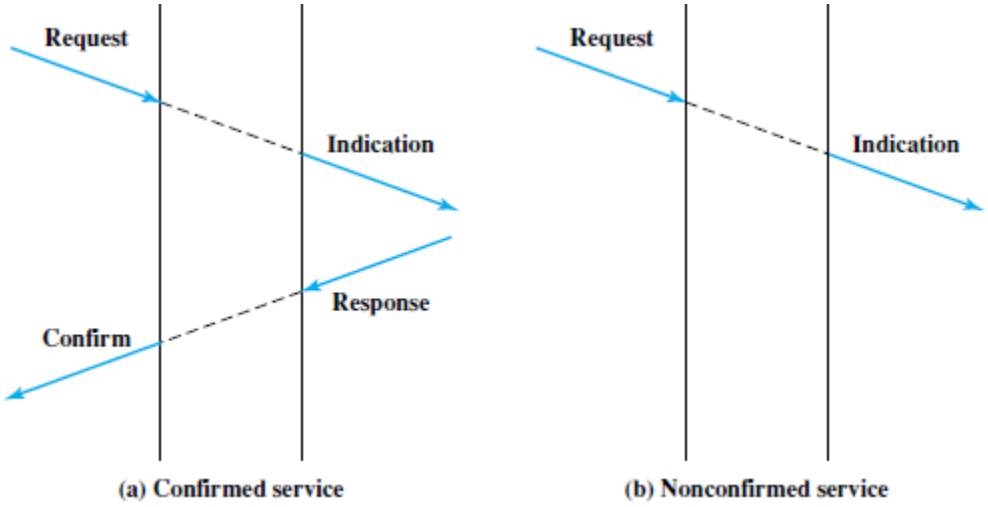
Four types of primitives are used in standards to define the interaction between adjacent layers in the architecture

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

Example: Consider 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.



(a) Confirmed service (b) Nonconfirmed service