PART OF THE PICTURE: The TCP/IP Communications Architecture

BY WILLIAM STALLINGS

The key to the success of distributed applications is that all the terminals and computers in the community "speak" the same language. This is the role of the underlying interconnection software. This software must ensure that all the devices transmit messages in such a way that they can be understood by the other computers and terminals in the community. With the introduction of the Systems Network Architecture (SNA) by IBM in the 1970s, this concept became a reality. However, SNA worked only with IBM equipment. Soon other vendors followed with their own proprietary communications architectures to tie together their equipment. Such an approach may be good business for the vendor, but it is bad business for the customer. Happily, that situation has changed radically with the adoption of standards for interconnection software.

TCP/IP ARCHITECTURE AND OPERATION

When communication is desired among computers from different vendors, the software development effort can be a nightmare. Different vendors use different data formats and data exchange protocols. Even within one vendor's product line, different model computers may communicate in unique ways.

As the use of computer communications and computer networking proliferates, a one-at-a-time special-purpose approach to communications software development is too costly to be acceptable. The only alternative is for computer vendors to adopt and implement a common set of conventions. For this to happen, standards are needed.

However, no single standard will suffice. Any distributed application, such as electronic mail or client/server interaction, requires a complex set of communications functions for proper operation. Many of these functions, such as reliability mechanisms, are common across many or even all applications. Thus, the communications task is best viewed as consisting of a modular architecture, in which the various elements of the architecture perform the various required functions. Hence, before one can develop standards, there should be a structure, or *protocol architecture*, that defines the communications tasks.

Two protocol architectures have served as the basis for the development of interoperable communications standards: the TCP/IP protocol suite and the OSI (Open Systems Interconnection) reference model. TCP/IP is the most widely used interoperable architecture, and has won the "protocol wars." Although some useful standards have been developed in the context of OSI, TCP/IP is now the universal interoperable protocol architecture. No product should be considered as part of a business information system that does not support TCP/IP.

TCP/IP LAYERS

The communication task using TCP/IP can be organized into five relatively independent layers: physical, network access, internet, transport, and application.

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 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., X.25), local area networks (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/IP

Figure 14-1 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, in the case of a host on another subnetwork, to a router. 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.

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. This address is used by IP for routing and delivery. Each application within 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 an application, associated with port 1 at

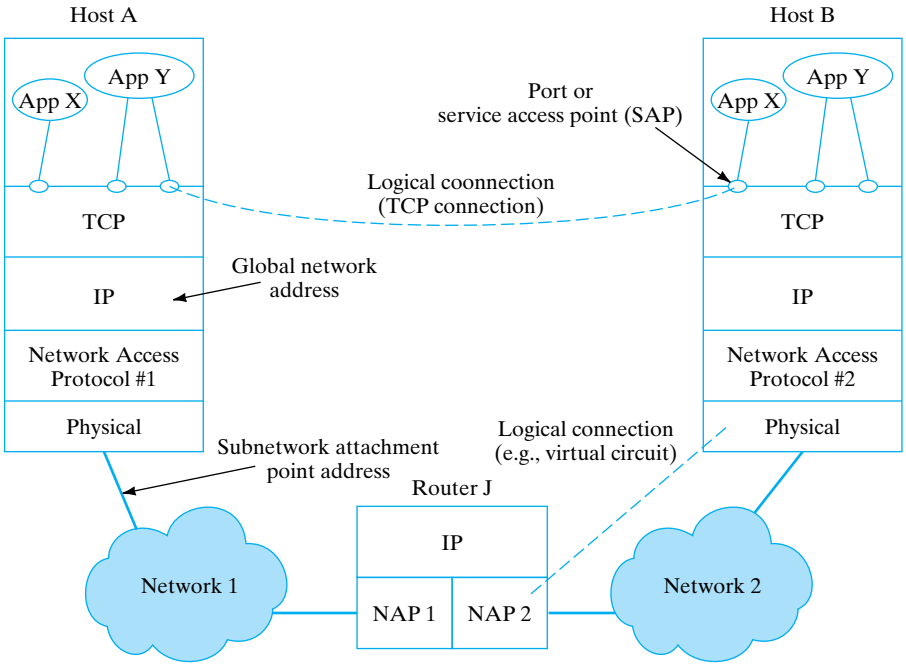


Figure 14-1 TCP/IP Concepts

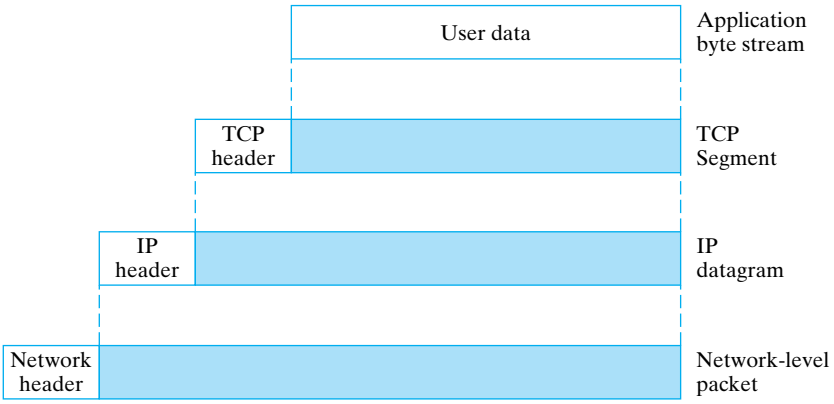


Figure 14-2 Protocol Data Units in the TCO/IP Architecture

host *A*, wishes to send a message to another application, associated with port 2 at host *B*. The application at *A* hands the message down to TCP with instructions to send it to host *B*, port 12. 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 is 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 *X* (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 14-2. 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 fields that are part of 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.

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 application.

A SIMPLE EXAMPLE

Figure 14-3 puts all of these concepts together, showing the interaction between modules to transfer one block of data. For simplicity, the example shows two systems connected to the same network, so that no router is involved. Let us say that the file transfer module in computer X is transferring a file one record at a time to computer Y. At X, each record is handed over to TCP. We can picture this action as being in the form of a command or procedure call. The arguments of this procedure call include the destination computer address, the destination port, and the record. TCP appends the destination port and other control information to the record to create a TCP segment. This is then handed down to IP by another procedure call. In this case, the arguments for the command are the destination computer address and the TCP segment. The resulting IP datagram is handed down to the network access layer, which constructs a network-level packet.

The network accepts the packet from X and delivers it to Y. The network access module

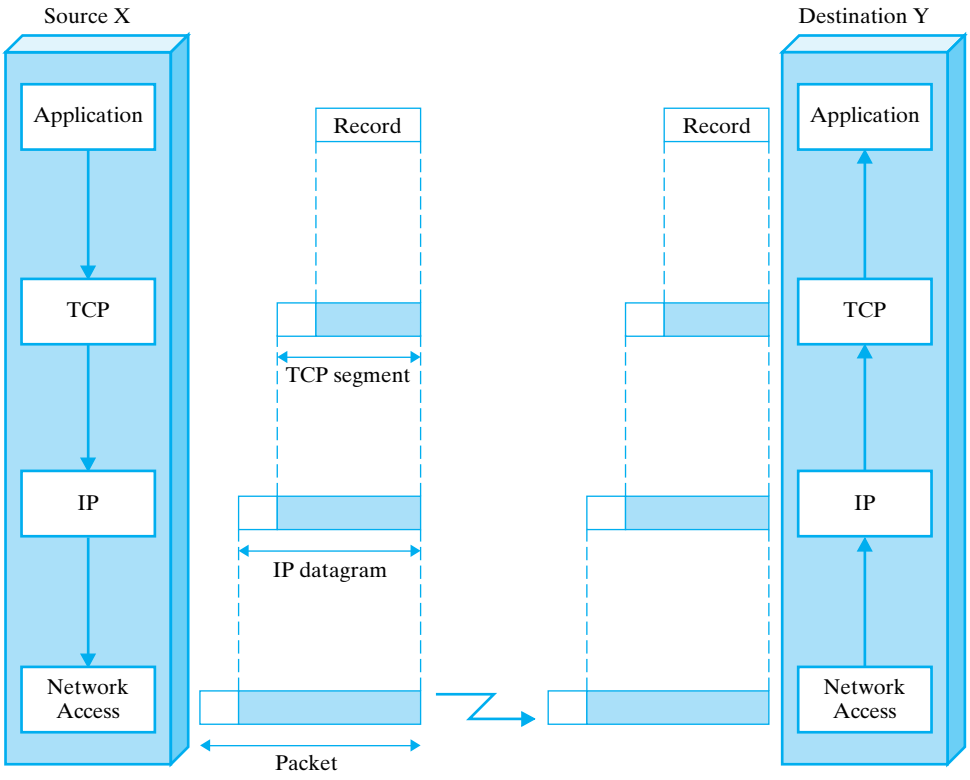


Figure 14-3 Operation of TCP/IP

in Y receives the packet, strips off the header, and transfers the enclosed transport PDU to Y's IP module, which strips off the IP header and passes the resulting TCP segment to TCP. TCP examines the segment header and, on the basis of the destination port field in the header, delivers the enclosed record to the appropriate application, in this case the file transfer module in Y.

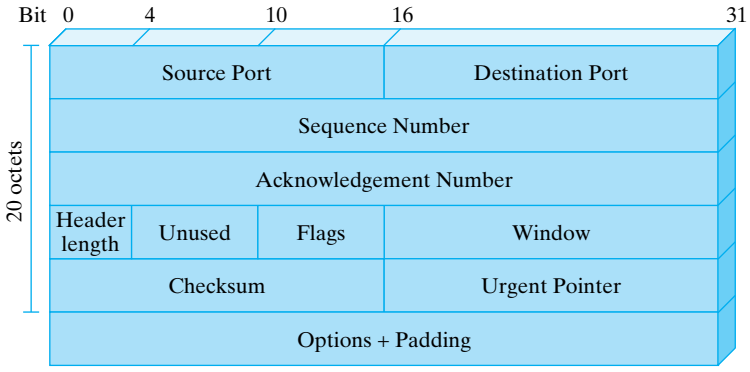
TCP AND UDP

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.

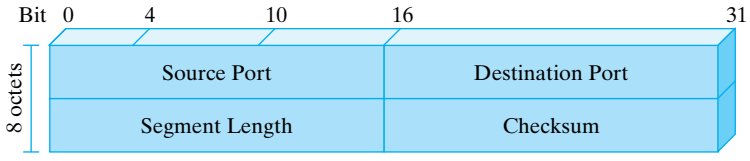
Figure 14-4a shows the header format for TCP, which is a minimum of 20 octets, or 160 bits. The Source Port and Destination Port fields identify the applications at the source and destination systems that are using this connection. The Sequence Number, Acknowledgment Number, and Window fields provide flow control and error control. In essence, each segment is sequentially numbered and must be

acknowledged by the receiver so that the sender knows that the segment was successfully received. The Windows field is passed from one side to the other to indicate how many data the other side may send before receiving additional permission. Finally, the checksum is a 16-bit frame check sequence used to detect errors in the TCP segment.

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 provides a connectionless service for application-level procedures. UDP does not guarantee delivery, preservation of sequence, or protection against duplication. UDP enables procedures 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 14-4b.



(a) TCP Header



(b) UDP Header

Figure 14-4 TCP and UDP Headers

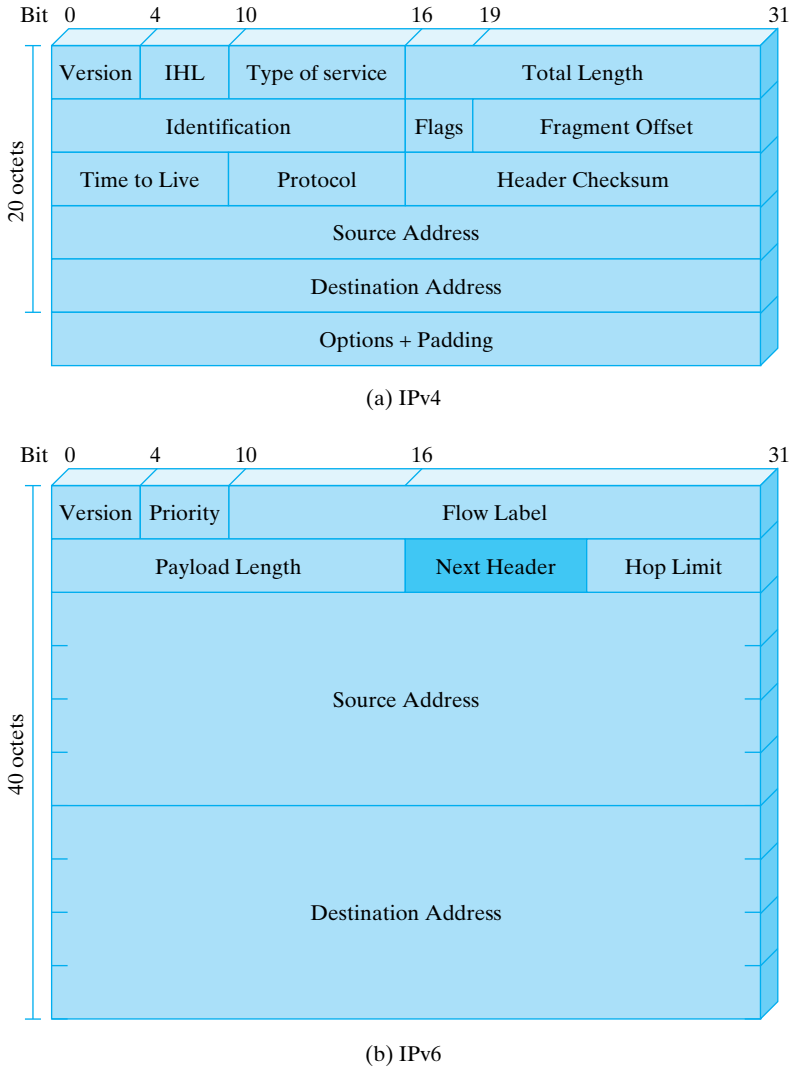


Figure 14-5 IP Heasers

IP AND IPV6

For decades, the keystone of the TCP/IP protocol architecture has been the Internet Protocol (IP). Figure 14-5a shows the IP header format, which is a minimum of 20 octets, or 160 bits. 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 whether TCP, UDP, or some other higher-layer protocol is

using IP. The Flags and Fragment Offset fields are used in the fragmentation and reassembly process.

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 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 of the systems needing addresses. As Figure 14-5b shows, IPv6 includes 128-bit source and destination address fields.

Ultimately, all of the 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 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. These allow user ID and password to be transmitted, and allow 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 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.

TO PROBE FURTHER

The topics in this section are covered in detail in *Data and Computer Communications, Fifth Edition*, by William Stallings (Prentice Hall, 1997). Links to web sites with further information can be found at http://www.shore.net/,ws/DCC5e.