APPENDIX F
THE TCP/IP PROTOCOL ARCHITECTURE

William Stallings

Supplement to
Computer Security, Third Edition
Pearson 2014
http://williamstallings.com/ComputerSecurity

TCP/IP 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 A provides a discussion of Internet standards.

F.1 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.

There is no official TCP/IP protocol model. However, based on the protocol standards that have been developed, we can organize the communication task for TCP/IP into five relatively independent layers, from bottom to top:

- 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

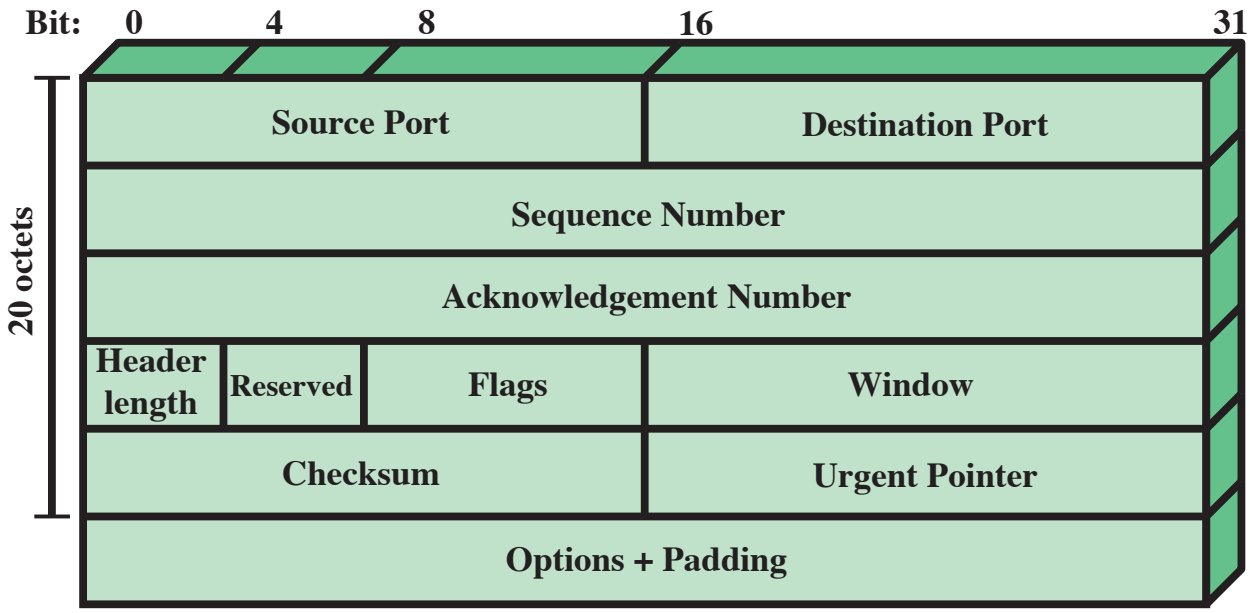
networks. This protocol is implemented not only in the end systems but also in routers. A **router** is a processor that connects two networks and whose primary function is to relay data from one network to the other on a route from the source to the destination end system.

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 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.

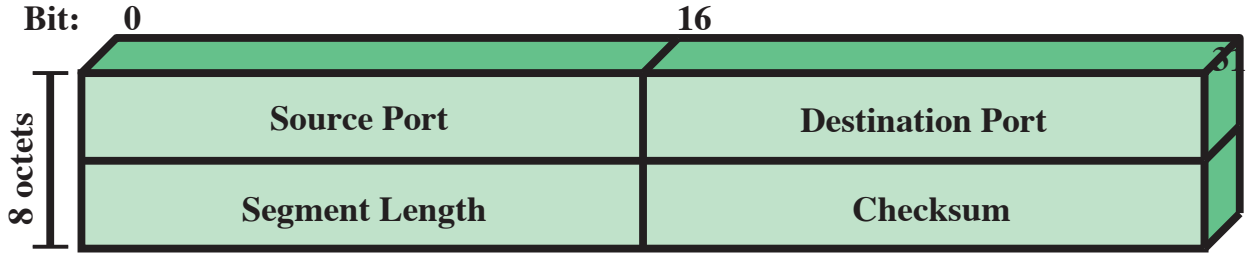
Finally, the **application layer** contains the logic needed to support the various user applications. For each different type of application, such as file transfer, a separate module is needed that is peculiar to that application.

F.2 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. For the duration of the connection each entity keeps track of segments coming and going to the other entity, in order to regulate the flow of segments and to recover from lost or damaged segments.



(a) TCP Header



(b) UDP Header

Figure F.1 TCP and UDP Headers

Figure F.1a 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 code based on the contents of the segment 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 does not guarantee delivery, preservation of sequence, or protection against duplication. UDP enables a process to send messages to other processes 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 F.1b.

F.3 OPERATION OF TCP/IP

Figure F.2 indicates how these protocols are configured for communications. Some sort of network access protocol, such as the Ethernet logic, is used to connect a computer to a network. This protocol enables the host to send data across the network to another host or, in the case of a host on another network, to a router. IP is implemented in all end systems and 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 being transferred to assure that all are delivered reliably to the appropriate application.

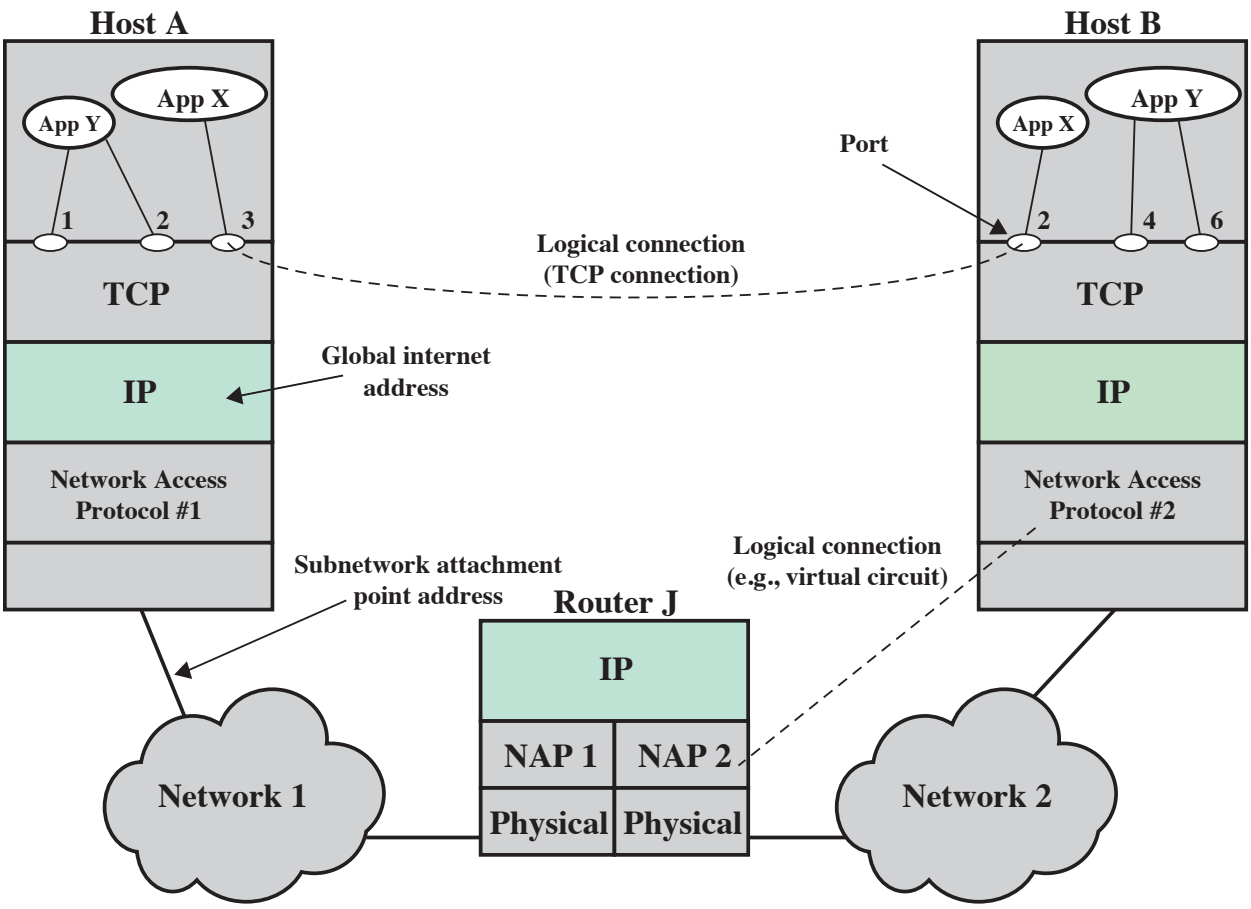


Figure F.2 TCP/IP Concepts

For successful communication, every entity in the overall system must have a unique address. In fact, two levels of addressing are needed. Each host on a network 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 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 F.3. 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 (Figure F.1a), 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.

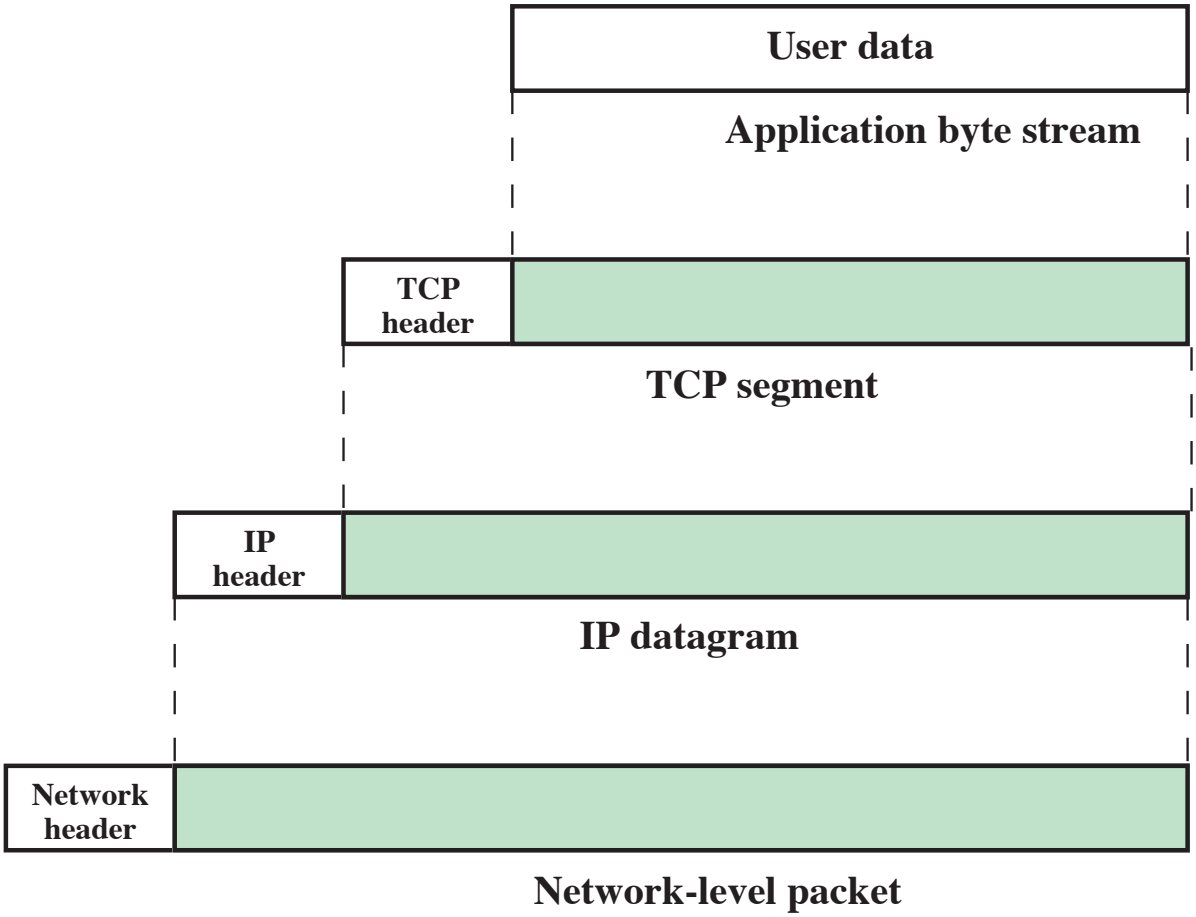


Figure F.3 Protocol Data Units (PDUs) in the TCP/IP Architecture

Next, TCP hands each segment over to IP, with instructions to transmit it to B. These segments must be transmitted across one or more networks 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 (Figure F.3) 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 network 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 network to router J. The packet header contains the information that the network needs in order to transfer the data across the network. Examples of items that may be contained in this header include:

- **Destination network address:** The network must know to which attached device the packet is to be delivered, in this case router J.
- **Facilities requests:** The network access protocol might request the use of certain network 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 network 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.

F.4 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. 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.

SSH (Secure Shell) provides a secure 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. SSH also supports file transfer between the local host and a remote server. SSH enables the user and the remote server to authenticate each other; it also encrypts all traffic in both directions. SSH traffic is carried on a TCP connection.