weather station. More realistic devices use *duplex mode*, where all systems can send or receive with equal facility. This is often further distinguished as *half-duplex* (the system can send and receive, but not at the same time) and *full-duplex* (simultaneous sending and receiving).

The Data Link Layer

Bits are just bits. With only a physical layer, System A has no way to tell System B, “Get ready some bits,” “Here are the bits,” and “Did you get those bits okay?” The data link layer solves this problem by organizing the bit stream into a data unit called a frame.

It is important to note that frames are the data link layer PDUs, and these are not the same as the physical layer transmission frames mentioned in the previous section. For example, network engineers often speak about T1 frames or SONET frames, but these are distinct from the data link layer frames that are carried inside the T1 or SONET frames. Transmission frames have control information used to manage the physical link itself and has little to do directly with process-to-process communications. This “double-frame” arrangement might sound redundant, but many transmission frames originated with voice because digitized voice has no framing at the “data link” layer.

The data link layer moves bits across the link and can add reliability to the raw communications link. The data link layer can be very simple, or make the link appear error-free to the layer above, the network layer. The data link layer usually adds both a header and trailer to the data presented by the network layer. This is shown in Figure 1.13.

The frame header typically contains a source and destination address (known as the “physical address” since it refers to the physical communication port) and some control information. The control information is data passed from one data link layer to the
other data link layer, and not user data. The body of the frame contains the sequence of bits being transferred across the network. The trailer usually contains information used in detecting bit errors (such as cyclical redundancy check [CRC]). A maximum size is associated with the frame that cannot be exceeded because all systems must allocate memory space (buffers) for the data. In a networking context, a buffer is just special memory allocated for communications.

The data link layer performs framing, physical addressing, and error detection (error correction is another matter entirely, and can be handled in many ways, such as by resending a copy of the frame that had the errors). However, when it comes to frame error detection and correction in the real world, error detection bits are sometimes ignored and frames that defy processing due to errors are simply discarded. This does not mean that error detection and correction are not part of the data link layer standards: It means that in these cases, ignoring and discarding are the chosen methods of implementation. In discard cases, the chore of handling the error condition is “pushed up the stack” to a higher layer protocol.

This layer also performs access control (this determines whose turn it is to send over or control the link, an issue that becomes more and more interesting as the number of devices sharing the link grows). In LANs, this media access control (MAC) forms a sublayer of the data link layer and has its own addressing scheme known (not surprisingly) as the MAC layer address or MAC address. We’ll look at MAC addresses in the next chapter. For now, it is enough to note that LANs such as Ethernet do not have “real” physical layer addresses and that the MAC address performs this addressing function.

In addition, the data link layer can perform some type of flow control. Flow control makes sure senders do not overwhelm receivers: a receiver must have adequate time to process the data arriving in its buffers. At this layer, the flow control, if provided, is link-by-link. (We’ll see shortly that end-to-end—host-to-host—flow control is provided by the transport layer.) LANs do not usually provide flow control at the data link layer, although they can.

Not all destination systems are directly reachable by the sender. This means that when bits at the data link layer are sent from an originating system, the bits do not arrive at the destination system as the “next hop” along the way. Directly reachable systems are called adjacent systems, and adjacent systems are always “one hop away” from the sender. When the destination system is not directly reachable by the sender, one or more intermediate nodes are needed. Consider the network shown in Figure 1.14.

Now the sender (System A) is not directly connected to the receiver (System B). Another system, System 3, receives the frame and must forward it toward the destination. This system is usually called a switch or router (there are even other names), depending on internal architecture and network role. On a WAN (but not on a LAN), this second frame is a different frame because there is no guarantee that the second link is identical to the first. Different links need different frames. Identical frames are only delivered to systems that are directly reachable, or adjacent, to the sender, such as by an Ethernet switch on a LAN.
Networking with intermediate systems is called *hop-by-hop* delivery. A “hop” is the usual term used on the Internet or a router network to indicate the forwarding of a packet between one router or another (or between a host and router). Frames can “hop” between Layer 2 switches, but the term is most commonly used for Layer 3 router hops (which can consist of multiple switch-to-switch frame “hops”). There can be more than one intermediate system between the source and destination end systems, of course, as shown in Figure 1.15. Consider the case where End System A is sending a bit stream to End System C.
Note that the intermediate systems (routers) have two distinct physical and data link layers, reflecting the fact that the systems have two (and often more) communication links, which can differ in many ways. (The figure shows a typical WAN configuration with point-to-point links, but routers on LANs, and on some types of public data service WANs, can be deployed in more complicated ways.)

However, there is something obviously missing from this figure. There is no connection between the data link layers on the intermediate systems! How does the router know to which output port and link to forward the data in order to ultimately reach the destination? (In the figure, note that Intermediate System 1 can send data to either Intermediate System 2 or Intermediate System 3, but only through Intermediate System 3, which forwards the data, is the destination reachable.)

These forwarding decisions are made at the TCP/IP network layer.

The Network Layer

The network layer delivers data in the form of a packet from source to destination, across as many links as necessary. The biggest difference between the network layer and the data link layer is that the data link layer is in charge of data delivery between adjacent systems (directly connected systems one hop away), while the network layer delivers data to systems that are not directly connected to the source. There can be many different types of data link and physical layers on the network, depending on the variety of the link types, but the network layer is essentially the same on all systems, end systems, and intermediate systems alike.

Figure 1.16 shows the relationship between the network layer and the transport layer above and the data link layer below. A packet header is put in place at the sender and interpreted by the receiver. A router simply looks at the packet header and makes a forwarding decision based on this information. The transport layer does not play a role in the forwarding decision.

---

**FIGURE 1.16**

The network layer. These data units are packets with their own destination and source address formats.
How does the network layer know where the packet came from (so the sender can reply)? The key concept at the network layer is the network address, which provides this information. In TCP/IP, the network address is the IP address.

Every system in the network receives a network address, whether an end system or intermediate system. Systems require at least one network address (and sometimes many more). It is important to realize that this network address is different from, and independent of, the physical address used by the frames that carry the packets between adjacent systems.

Why should the systems need two addresses for the two layers? Why can’t they just both use either the data link (“physical”) address or the network address at both layers? There are actually several reasons. First, LAN addresses like those used in Ethernet come from one group (the IEEE), while those used in TCP/IP come from another group (ICANN). Also, the IP address is universally used on the Internet, while there are many types of physical addresses. Finally, there is no systematic assignment of physical addresses (and many addresses on WANs can be duplicates and so have “local significance only”). On the other hand, IP network addresses are globally administered, unique, and have a portion under which many devices are grouped. Therefore, many devices can be addressed concisely by this network portion of the IP address.

A key issue is how the network addresses “map” to physical addresses, a process known generally as address resolution. In TCP/IP, a special family of address resolution protocols takes care of this process.

The network address is a logical address. Network addresses should be organized so that devices can be grouped under a part of that address. In other words, the network address should be organized in a fashion similar to a telephone number, for example, 212-555-1212 in the North American public switched telephone network (PSTN). The sender need only look at the area code or “network” portion of this address (212) to determine if the destination is local (area codes are the same) or needs to be sent to an intermediate system to reach the 212 area code (source and destination area codes differ).

For this scheme to work effectively, however, all telephones that share the 212 area code should be grouped together. The whole telephone number beginning with 212 therefore means “this telephone in the 212 area code.” In TCP/IP, the network address is the beginning of the device’s complete IP address. A group of hosts is gathered under the network portion of the IP address. IP network addresses, like area codes, are globally administered to prevent duplication, while the rest of the IP address, like the rest of the telephone number, is locally administered, often independently.

In some cases, the packet that arrives at an intermediate system inside a frame is too large to fit inside the frame that must be sent out. This is not uncommon: different link and LAN types have different maximum frame sizes. The network layer must be able to fragment a data unit across multiple frames and reassemble the fragments at the destination. We’ll say more about fragmentation in a later chapter.
The network layer uses one or more routing tables to store information about reachable systems. The routing tables must be created, maintained, and purged of old information as the network changes due to failures, the addition or deletion of systems and links, or other configuration changes. This whole process of building tables to pass data from source to destination is called routing, and the use of these tables for packet delivery is called forwarding. The forwarding of packets inside frames always takes place hop by hop. This is shown in Figure 1.17, which adds the network layer to the data link layers already present and distinguishes between hop-by-hop forwarding and end-to-end delivery.

On the Internet, the intermediate systems that act at the packet level (Layer 3) are called routers. Devices that act on frames (Layer 2) are called switches, and some older telephony-based WAN architectures use switches as intermediate network nodes. Whether a node is called a switch or router depends on how they function internally.
In a very real sense, the network layer is at the very heart of any protocol stack, and TCP/IP is no exception. The protocol at this layer is IP, either IPv4 or IPv6 (some think that IPv6 is distinct enough to be known as TCPv6/IPv6).

The Transport Layer

Process-to-process delivery is the task of the transport layer. Getting a packet to the destination system is not quite the same thing as determining which process should receive the packet’s content. A system can be running file transfer, email, and other network processes all at the same time, and all over a single physical interface. Naturally, the destination process has to know on which process the sender originated the bits inside the packet in order to reply. Also, systems cannot simply transfer a huge multimegabit file all in one packet. Many data units exceed the maximum allowable size of a packet.

This process of dividing message content into packets is known as segmentation. The network layer forwards each and every packet independently, and does not recognize any relationship between the packets. (Is this a file transfer or email packet? The network layer does not care.) The transport layer, in contrast, can make sure the whole message, often strung out in a sequence of packets, arrives in order (packets can be delivered out of sequence) and intact (there are no errors in the entire message). This function of the transport layer involves some method of flow control and error control (error detection and error correction) at the transport layer, functions which are absent at the network layer. The transport-layer protocol that performs all of these functions is TCP.

The transport-layer protocol does not have to do any of this, of course. In many cases, the content of the packet forms a complete unit all by itself, called a datagram. (The term “datagram” is often used to refer to the whole IP packet, but not in this book.) Self-contained datagrams are not concerned with sequencing or flow control, and these functions are absent in the User Datagram Protocol (UDP) at the transport layer.

So there are two very popular protocol packages at the transport layer:

- **TCP**—This is a connection-oriented, “reliable” service that provides ordered delivery of packet contents.
- **UDP**—This is a connectionless, “unreliable” service that does not provide ordered delivery of packet contents.

In addition to UDP and TCP, there are other transport-layer protocols that can be used in TCP/IP, all of which differ in terms of how they handle transport-layer tasks. Developers are not limited to the standard choices for applications. If neither TCP nor UDP nor any other defined transport-layer service is appropriate for your application, you can write your own transport-layer protocols and get others to adapt it (or use your application package exclusively).
In TCP/IP, it is often said that the network layer (IP itself) offers an “unreliable” or “best effort” service, while the transport layer adds “reliability” in the form of flow and error control. Later in this book, we’ll see why these terms are unfortunate and what they really mean.

The network layer gets a single packet to the right system, and the transport layer gets the entire message to the right process. Figure 1.18 shows the transport layer breaking up a message at the sender into three pieces (each labeled “TL data” for transport-layer data and “TH” for transport-layer header). The figure then shows the transport layer reassembling the message at the receiver from the various segments that make up a message. In TCP/IP, there are also data units known as datagrams, which are always handled as self-contained units. There are profound differences between how the transport layer treats segments and datagrams, but this figure is just a general illustration of segment handling.

The functions that the transport layer, which in some protocols is called the end-to-end layer, might have to include follow:

Process addressing and multiplexing—Also known as “service-point addressing,” the transport layer has to decide which process originated the message and to which process the message must be delivered. These are also known as port addresses in TCP/IP. Port addresses are an important portion of the application socket in TCP/IP.

Segment handling—In cases where each message is divided into segments, each segment has a sequence number used to put the message back together at the destination. When datagrams are used, each data unit is handled independently and sequencing is not necessary.
Connection control—The transport layer can be connectionless or connection-oriented (in fact, several layers can operate in either one of these ways). Connectionless (CL) layers treat every data unit as a self-contained, independent unit. Connection-oriented (CO) layers go through a three-phase process every time there is data to send to a destination after an idle period (connection durations can vary). First, some control messages establish the connection, then the data are sent (and exchanged if replies are necessary), and finally the connection is closed. Many times, a comparison is made between a telephone conversation (“dial, talk, hang up”) with connections and an intercom (“push and talk any time”) for connectionless communications, but this is not precise. Generally, segments are connection-oriented data units, and datagrams are connectionless data units.

Flow control—Just as with the data link layer, the transport layer can include flow control mechanisms to prevent senders from overwhelming receivers. In this case, however, the flow control is end-to-end rather than link-by-link. Datagrams do not require this service.

Error control—This is another function that can be performed at the data link layer, but again end-to-end at the transport layer rather than link-by-link. Communications links are not the only source of errors, which can occur inside a system as well. Again, datagrams do not require this service.

Figure 1.19 shows the relationship between the network layer and transport layer more clearly. The network layer operates from network interface to network interface, while the transport layer is more specific and operates from process to process.
The Application Layer

It might seem that once data are transferred from end-system process to end-system process, the networking task is pretty much complete. There is a lot that still needs to be done at the application level itself. In models of protocol stacks, it is common to place another layer between the transport layer and the user, the application layer. However, the TCP/IP protocol stack really stops at the transport layer (where TCP and UDP are). It is up to the application programmer to decide what should happen at the client and server level at that point, although there are individual RFCs for guidance, such as for FTP.

Although it is common to gather these TCP/IP applications into their own layer, there really is no such thing in TCP/IP as an application layer to act as some kind of “glue” between the application’s user and the network.

In nearly all TCP/IP stacks, the application layer is part of the application process. In spite of the lack of a defined layer, a TCP/IP application might still have a lot to do, and in some ways the application layer is the most complex “layer” of all.

There are two major tasks that the application often needs to accomplish: session support and conversion of internal representation. Not all applications need both, of course, and some applications might not need either, but this overview includes both major functions.

Session Support

A session is a type of dialog controller between two processes that establishes, maintains, and synchronizes (controls) the interaction (dialog). A session decides if the communication can be half-duplex (both ends take turns sending) or full-duplex (both ends can send whenever they want). It also keeps a kind of “history” of the interaction between endpoints, so that when things go wrong or when the two communicate again, some information does not have to be resent.

In practical terms, the session consists of all “state variables” necessary to construct the history of the connection between the two devices. It is more difficult, but not impossible, to implement sessions in a connectionless environment because there is no easy way to associate the variables with a convenient label.

Internal Representation Conversion

The role of internal representation conversion is to make sure that the data exchange over the network is useful to the receivers. If the internal representation of data differs on the two systems (integer size, bit order in memory, etc.), the application layer translates between the formats so the application program does not have to. This layer can also provide encryption and compression functions, although it is more common to implement these last two functions separately from the network.

Standard protocol specifications can use the Abstract Syntax Notation 1 (ASN.1) definitions for translation purposes. ASN.1 can be used in programming, network
management, and other places. ASN.1 defines various things such as which bit is “first on the wire” regardless of how it is stored internally, how many bits are to be sent for the numbers 0 through 255 (8), and so on. Everything can be translated into ASN.1, sent across the network, and translated back to whatever internal format is required at the destination.

The role of internal representation conversion is shown in Figure 1.20. The figure shows four sequential memory locations, each storing the letter “a” followed by the integer 259. Note that not only are there differences between the amount of memory addressed at once, but also in the order of the bits for numerics.

In some protocol stacks, the application program can rely on the services of a fully functional conversion for internal representation to perform these services. However, in TCP/IP, every network application program must do these things for itself.

### Applications in TCP/IP

TCP/IP does not provide session or presentation services directly to an application. Programmers are on their own, but this does not mean they have to create everything from scratch. For example, applications can use a character-based presentation service called the Network Virtual Terminal (NVT), part of the Internet’s telnet remote access specification. Other applications can use Sun’s External Data Representation (XDR) or IBM’s (and Microsoft’s) NetBIOS programming libraries for presentation services. In this respect, there are many presentation layer services that TCP/IP can use, but there is no formal presentation service standard in TCP/IP that all applications must use.

Host TCP/IP implementations typically provide a range of applications that provide users with access to the data handled by the transport-layer protocols. These applications use a number of protocols that are not part of TCP/IP proper, but are used with TCP/IP. These protocols include the Hyper-Text Transfer Protocol (HTTP) used by Web browsers, the Simple Message Transfer Protocol (SMTP) used for email, and many others.
In TCP/IP, the application protocol, the application service, and the user application itself often share the same name. The file transfer protocol in TCP/IP, called FTP, is at once an application protocol, an application service, and an application run by a user. It can sometimes be confusing as to just which aspect of FTP is under discussion.

The role of TCP/IP applications is shown in Figure 1.21. Note that this “layer” sits on top of the TCP/IP protocol stack and interfaces with programs or users directly. Some protocols provide separate layers for sessions, internal representation conversion, and application services. In practice, these are seldom implemented independently. It just makes more sense to bundle them together by major application, as in TCP/IP.

**THE TCP/IP PROTOCOL SUITE**

To sum up, the five layers of TCP/IP are physical, data link, network, transport, and application. The TCP/IP stack is a hierarchical model made up of interactive modules. Each module provides a specific function. In TCP/IP, the layers contain relatively independent protocols that can be “mixed and matched” depending on the needs of the system to provide whatever function is desired. TCP/IP is hierarchical in the sense that each higher layer protocol is supported by one or more lower layer protocols.

Figure 1.22 maps some of the protocols used in TCP/IP to the various layers of TCP/IP. Every protocol in the figure will be discussed in this book, most in chapters all their own.
FIGURE 1.22
TCP/IP protocols and layers. Note the position of some protocols between layers.

With few exceptions, the TCP/IP protocol suite does not really define any low-level protocols below the network layer. TCP/IP usually specifies how to put IP packets into frames and how to get them out again. Many RFCs define IP mapping into these lower-layer protocols. We'll talk more about this mapping process in Chapter 2.