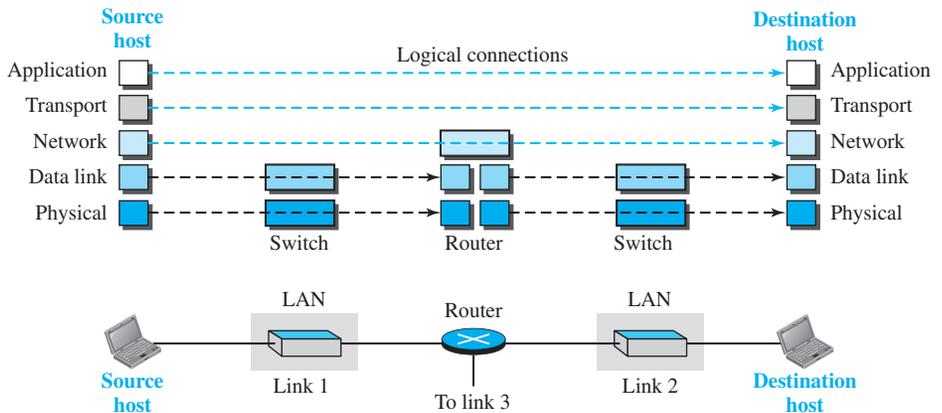


Layers in the TCP/IP Protocol Suite

After the above introduction, we briefly discuss the functions and duties of layers in the TCP/IP protocol suite. Each layer is discussed in detail in the next five parts of the book. To better understand the duties of each layer, we need to think about the logical connections between layers. Figure 2.6 shows logical connections in our simple internet.

Figure 2.6 Logical connections between layers of the TCP/IP protocol suite



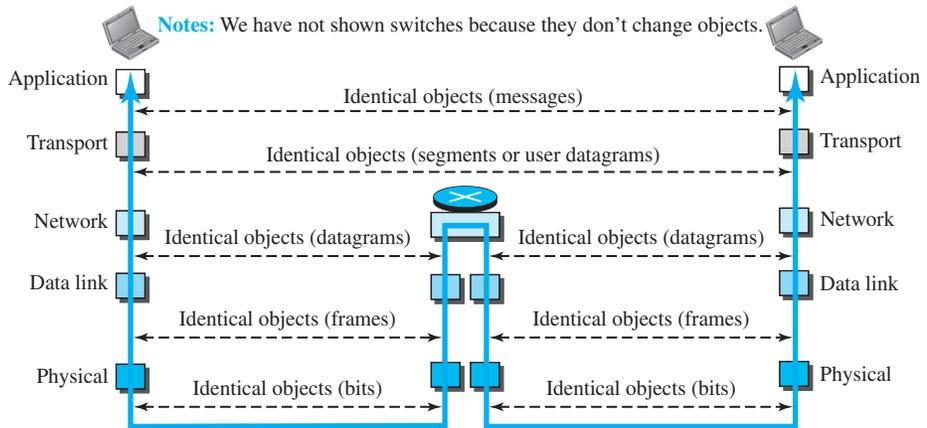
Using logical connections makes it easier for us to think about the duty of each layer. As the figure shows, the duty of the application, transport, and network layers is end-to-end. However, the duty of the data-link and physical layers is hop-to-hop, in which a hop is a host or router. In other words, the domain of duty of the top three layers is the internet, and the domain of duty of the two lower layers is the link.

Another way of thinking of the logical connections is to think about the data unit created from each layer. In the top three layers, the data unit (packets) should not be

changed by any router or link-layer switch. In the bottom two layers, the packet created by the host is changed only by the routers, not by the link-layer switches.

Figure 2.7 shows the second principle discussed previously for protocol layering. We show the identical objects below each layer related to each device.

Figure 2.7 *Identical objects in the TCP/IP protocol suite*



Note that, although the logical connection at the network layer is between the two hosts, we can only say that identical objects exist between two hops in this case because a router may fragment the packet at the network layer and send more packets than received (see fragmentation in Chapter 19). Note that the link between two hops does not change the object.

2.2.3 Description of Each Layer

After understanding the concept of logical communication, we are ready to briefly discuss the duty of each layer. Our discussion in this chapter will be very brief, but we come back to the duty of each layer in next five parts of the book.

Physical Layer

We can say that the physical layer is responsible for carrying individual bits in a frame across the link. Although the physical layer is the lowest level in the TCP/IP protocol suite, the communication between two devices at the physical layer is still a logical communication because there is another, hidden layer, the transmission media, under the physical layer. Two devices are connected by a transmission medium (cable or air). We need to know that the transmission medium does not carry bits; it carries electrical or optical signals. So the bits received in a frame from the data-link layer are transformed and sent through the transmission media, but we can think that the logical unit between two physical layers in two devices is a *bit*. There are several protocols that transform a bit to a signal. We discuss them in Part II when we discuss the physical layer and the transmission media.

Data-link Layer

We have seen that an internet is made up of several links (LANs and WANs) connected by routers. There may be several overlapping sets of links that a datagram can travel from the host to the destination. The routers are responsible for choosing the *best* links. However, when the next link to travel is determined by the router, the data-link layer is responsible for taking the datagram and moving it across the link. The link can be a wired LAN with a link-layer switch, a wireless LAN, a wired WAN, or a wireless WAN. We can also have different protocols used with any link type. In each case, the data-link layer is responsible for moving the packet through the link.

TCP/IP does not define any specific protocol for the data-link layer. It supports all the standard and proprietary protocols. Any protocol that can take the datagram and carry it through the link suffices for the network layer. The data-link layer takes a datagram and encapsulates it in a packet called a *frame*.

Each link-layer protocol may provide a different service. Some link-layer protocols provide complete error detection and correction, some provide only error correction. We discuss wired links in Chapters 13 and 14 and wireless links in Chapters 15 and 16.

Network Layer

The network layer is responsible for creating a connection between the source computer and the destination computer. The communication at the network layer is host-to-host. However, since there can be several routers from the source to the destination, the routers in the path are responsible for choosing the best route for each packet. We can say that the network layer is responsible for host-to-host communication and routing the packet through possible routes. Again, we may ask ourselves why we need the network layer. We could have added the routing duty to the transport layer and dropped this layer. One reason, as we said before, is the separation of different tasks between different layers. The second reason is that the routers do not need the application and transport layers. Separating the tasks allows us to use fewer protocols on the routers.

The network layer in the Internet includes the main protocol, Internet Protocol (IP), that defines the format of the packet, called a datagram at the network layer. IP also defines the format and the structure of addresses used in this layer. IP is also responsible for routing a packet from its source to its destination, which is achieved by each router forwarding the datagram to the next router in its path.

IP is a connectionless protocol that provides no flow control, no error control, and no congestion control services. This means that if any of these services is required for an application, the application should rely only on the transport-layer protocol. The network layer also includes unicast (one-to-one) and multicast (one-to-many) routing protocols. A routing protocol does not take part in routing (it is the responsibility of IP), but it creates forwarding tables for routers to help them in the routing process.

The network layer also has some auxiliary protocols that help IP in its delivery and routing tasks. The Internet Control Message Protocol (ICMP) helps IP to report some problems when routing a packet. The Internet Group Management Protocol (IGMP) is another protocol that helps IP in multitasking. The Dynamic Host Configuration Protocol (DHCP) helps IP to get the network-layer address for a host. The Address Resolution Protocol (ARP) is a protocol that helps IP to find the link-layer address of a host or

The logical connection at the transport layer is also end-to-end. The transport layer at the source host gets the message from the application layer, encapsulates it in a transport-layer packet (called a *segment* or a *user datagram* in different protocols) and sends it, through the logical (imaginary) connection, to the transport layer at the destination host. In other words, the transport layer is responsible for giving services to the application layer: to get a message from an application program running on the source host and deliver it to the corresponding application program on the destination host. We may ask why we need an end-to-end transport layer when we already have an end-to-end application layer. The reason is the separation of tasks and duties, which we discussed earlier. The transport layer should be independent of the application layer. In addition, we will see that we have more than one protocol in the transport layer, which means that each application program can use the protocol that best matches its requirement.

As we said, there are a few transport-layer protocols in the Internet, each designed for some specific task. The main protocol, Transmission Control Protocol (TCP), is a connection-oriented protocol that first establishes a logical connection between transport layers at two hosts before transferring data. It creates a logical pipe between two TCPs for transferring a stream of bytes. TCP provides flow control (matching the sending data rate of the source host with the receiving data rate of the destination host to prevent overwhelming the destination), error control (to guarantee that the segments arrive at the destination without error and resending the corrupted ones), and congestion control to reduce the loss of segments due to congestion in the network. The other common protocol, User Datagram Protocol (UDP), is a connectionless protocol that transmits user datagrams without first creating a logical connection. In UDP, each user datagram is an independent entity without being related to the previous or the next one (the meaning of the term *connectionless*). UDP is a simple protocol that does not provide flow, error, or congestion control. Its simplicity, which means small overhead, is attractive to an application program that needs to send short messages and cannot afford the retransmission of the packets involved in TCP, when a packet is corrupted or lost. A new protocol, Stream Control Transmission Protocol (SCTP) is designed to respond to new applications that are emerging in the multimedia. We will discuss UDP, TCP, and SCTP in Chapter 24.

Application Layer

As Figure 2.6 shows, the logical connection between the two application layers is end-to-end. The two application layers exchange *messages* between each other as though there were a bridge between the two layers. However, we should know that the communication is done through all the layers.

Communication at the application layer is between two *processes* (two programs running at this layer). To communicate, a process sends a request to the other process and receives a response. Process-to-process communication is the duty of the application layer. The application layer in the Internet includes many predefined protocols.

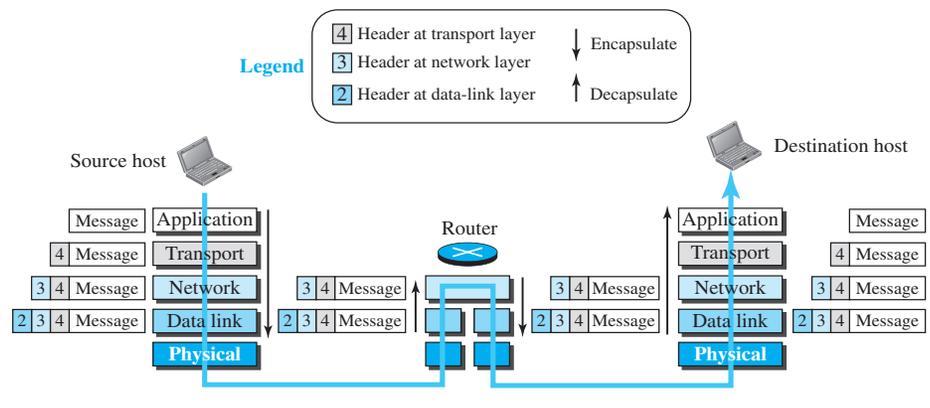
a user can also create a pair of processes to be run at the two hosts. In Chapter 25, we explore this situation.

The Hypertext Transfer Protocol (HTTP) is a vehicle for accessing the World Wide Web (WWW). The Simple Mail Transfer Protocol (SMTP) is the main protocol used in electronic mail (e-mail) service. The File Transfer Protocol (FTP) is used for transferring files from one host to another. The Terminal Network (TELNET) and Secure Shell (SSH) are used for accessing a site remotely. The Simple Network Management Protocol (SNMP) is used by an administrator to manage the Internet at global and local levels. The Domain Name System (DNS) is used by other protocols to find the network-layer address of a computer. The Internet Group Management Protocol (IGMP) is used to collect membership in a group. We discuss most of these protocols in Chapter 26 and some in other chapters.

2.2.4 Encapsulation and Decapsulation

One of the important concepts in protocol layering in the Internet is encapsulation/decapsulation. Figure 2.8 shows this concept for the small internet in Figure 2.5.

Figure 2.8 Encapsulation/Decapsulation



We have not shown the layers for the link-layer switches because no encapsulation/decapsulation occurs in this device. In Figure 2.8, we show the encapsulation in the source host, decapsulation in the destination host, and encapsulation and decapsulation in the router.

Encapsulation at the Source Host

At the source, we have only encapsulation.

1. At the application layer, the data to be exchanged is referred to as a *message*. A message normally does not contain any header or trailer, but if it does, we refer to the whole as the message. The message is passed to the transport layer.
2. The transport layer takes the message as the payload, the load that the transport layer should take care of. It adds the transport layer header to the payload, which contains the identifiers of the source and destination application programs that

want to communicate plus some more information that is needed for the end-to-end delivery of the message, such as information needed for flow, error control, or congestion control. The result is the transport-layer packet, which is called the *segment* (in TCP) and the *user datagram* (in UDP). The transport layer then passes the packet to the network layer.

3. The network layer takes the transport-layer packet as data or payload and adds its own header to the payload. The header contains the addresses of the source and destination hosts and some more information used for error checking of the header, fragmentation information, and so on. The result is the network-layer packet, called a *datagram*. The network layer then passes the packet to the data-link layer.
4. The data-link layer takes the network-layer packet as data or payload and adds its own header, which contains the link-layer addresses of the host or the next hop (the router). The result is the link-layer packet, which is called a *frame*. The frame is passed to the physical layer for transmission.

Decapsulation and Encapsulation at the Router

At the router, we have both decapsulation and encapsulation because the router is connected to two or more links.

1. After the set of bits are delivered to the data-link layer, this layer decapsulates the datagram from the frame and passes it to the network layer.
2. The network layer only inspects the source and destination addresses in the datagram header and consults its forwarding table to find the next hop to which the datagram is to be delivered. The contents of the datagram should not be changed by the network layer in the router unless there is a need to fragment the datagram if it is too big to be passed through the next link. The datagram is then passed to the data-link layer of the next link.
3. The data-link layer of the next link encapsulates the datagram in a frame and passes it to the physical layer for transmission.

Decapsulation at the Destination Host

At the destination host, each layer only decapsulates the packet received, removes the payload, and delivers the payload to the next-higher layer protocol until the message reaches the application layer. It is necessary to say that decapsulation in the host involves error checking.

2.2.5 Addressing

It is worth mentioning another concept related to protocol layering in the Internet, *addressing*. As we discussed before, we have logical communication between pairs of layers in this model. Any communication that involves two parties needs two addresses: source address and destination address. Although it looks as if we need five pairs of addresses, one pair per layer, we normally have only four because the physical layer does not need addresses; the unit of data exchange at the physical layer is a bit, which definitely cannot have an address. Figure 2.9 shows the addressing at each layer.

As the figure shows, there is a relationship between the layer, the address used in that layer, and the packet name at that layer. At the application layer, we normally use names to define the site that provides services, such as *someorg.com*, or the e-mail

Figure 2.9 Addressing in the TCP/IP protocol suite

Packet names	Layers	Addresses
Message	Application layer	Names
Segment / User datagram	Transport layer	Port numbers
Datagram	Network layer	Logical addresses
Frame	Data-link layer	Link-layer addresses
Bits	Physical layer	

address, such as *somebody@coldmail.com*. At the transport layer, addresses are called port numbers, and these define the application-layer programs at the source and destination. Port numbers are local addresses that distinguish between several programs running at the same time. At the network-layer, the addresses are global, with the whole Internet as the scope. A network-layer address uniquely defines the connection of a device to the Internet. The link-layer addresses, sometimes called MAC addresses, are locally defined addresses, each of which defines a specific host or router in a network (LAN or WAN). We will come back to these addresses in future chapters.

2.2.6 Multiplexing and Demultiplexing

Since the TCP/IP protocol suite uses several protocols at some layers, we can say that we have multiplexing at the source and demultiplexing at the destination. Multiplexing in this case means that a protocol at a layer can encapsulate a packet from several next-higher layer protocols (one at a time); demultiplexing means that a protocol can decapsulate and deliver a packet to several next-higher layer protocols (one at a time). Figure 2.10 shows the concept of multiplexing and demultiplexing at the three upper layers.

Figure 2.10 Multiplexing and demultiplexing

