EXHIBIT 1011 – part 2

TABLE 13.1 IEEE 802.3 10-Mbps physical layer medium alternatives.

	10BASE5	10BASE2	10BASE-T	10BROAD36	10BASE-FP
Transmission medium	Coaxial Cable (50 ohm)	Coaxial Cable (50 ohm)	Unshielded twisted pair	Coaxial Cable (75 ohm)	850-nm optical fiber pair
Signaling technique	Baseband (Manchester)	Baseband (Manchester)	Baseband (Manchester)	Broadband (DPSK)	Manchester/ On-off
Topology	Bus	Bus	Star	Bus/Tree	Star
Maximum segment length (m)	500	185	100	1800	500
Nodes per segment	100	30	—	—	33
Cable diameter (mm)	10	5	0.4–0.6	0.4–1.0	62.5/125 μm

maximum of four repeaters in the path between any two stations, thereby extending the effective length of the medium to 2.5 kilometers.

10BASE2 Medium Specification

To provide a lower-cost system than 10BASE5 for personal computer LANs, 10BASE2 was added. As with 10BASE5, this specification uses 50-ohm coaxial cable and Manchester signaling. The key difference is that 10BASE2 uses a thinner cable, which supports fewer taps over a shorter distance than the 10BASE5 cable.

Because they have the same data rate, it is possible to combine 10BASE5 and 10BASE2 segments in the same network, by using a repeater that conforms to 10BASE5 on one side and 10BASE2 on the other side. The only restriction is that a 10BASE2 segment should not be used to bridge two 10BASE5 segments, because a "backbone" segment should be as resistant to noise as the segments it connects.

10BASE-T Medium Specification

By sacrificing some distance, it is possible to develop a 10-Mbps LAN using the unshielded twisted pair medium. Such wire is often found prewired in office buildings as excess telephone cable, and can be used for LANs. Such an approach is specified in the 10BASE-T specification. The 10BASE-T specification defines a star-shaped topology. A simple system consists of a number of stations connected to a central point, referred to as a multiport repeater, via two twisted pairs. The central point accepts input on any one line and repeats it on all of the other lines.

Stations attach to the multiport repeater via a point-to-point link. Ordinarily, the link consists of two unshielded twisted pairs. Because of the high data rate and the poor transmission qualities of unshielded twisted pair, the length of a link is limited to 100 meters. As an alternative, an optical fiber link may be used. In this case, the maximum length is 500 m.

10BROAD36 Medium Specification

The 10BROAD36 specification is the only 802.3 specification for broadband. The medium employed is the standard 75-ohm CATV coaxial cable. Either a dual-cable or split-cable configuration is allowed. The maximum length of an individual segment, emanating from the headend, is 1800 meters; this results in a maximum end-to-end span of 3600 meters.

The signaling on the cable is differential phase-shift keying (DPSK). In ordinary PSK, a binary zero is represented by a carrier with a particular phase, and a binary one is represent by a carrier with the opposite phase (180-degree difference). DPSK makes use of differential encoding, in which a change of phase occurs when a zero occurs, and there is no change of phase when a one occurs. The advantage of differential encoding is that it is easier for the receiver to detect a change in phase than to determine the phase itself.

The characteristics of the modulation process are specified so that the resulting 10 Mbps signal fits into a 14 MHz bandwidth.

10BASE-F Medium Specification

The 10BASE-F specification enables users to take advantage of the distance and transmission characteristics available with the use of optical fiber. The standard actually contains three specifications:

- **10-BASE-FP (passive).** A passive-star topology for interconnecting stations and repeaters with up to 1 km per segment.
- **10-BASE-FL (link).** Defines a point-to-point link that can be used to connect stations or repeaters at up to 2 km.
- **10-BASE-FB (backbone).** Defines a point-to-point link that can be used to connect repeaters at up to 2 km.

All three of these specifications make use of a pair of optical fibers for each transmission link, one for transmission in each direction. In all cases, the signaling scheme involves the use of Manchester encoding. Each Manchester signal element is then converted to an optical signal element, with the presence of light corresponding to high and the absence of light corresponding to low. Thus, a 10-Mbps Manchester bit stream actually requires 20 Mbps on the fiber.

The 10-BASE-FP defines a passive star system that can support up to 33 stations attached to a central passive star, of the type described in Chapter 3. 10-BASE-FL and 10-BASE-FP define point-to-point connections that can be used to extend the length of a network; the key difference between the two is that 10-BASE-FP makes use of synchronous retransmission. With synchronous signaling, an optical signal coming into a repeater is retimed with a local clock and retransmitted. With conventional asynchronous signaling, used with 10-BASE-FL, no such retiming takes place, so that any timing distortions are propagated through a series of repeaters. As a result, 10BASE-FB can be used to cascade up to 15 repeaters in sequence to achieve greater length.

IEEE 802.3 100-Mbps Specifications (Fast Ethernet)

Fast Ethernet refers to a set of specifications developed by the IEEE 802.3 committee to provide a low-cost, Ethernet-compatible LAN operating at 100 Mbps. The blanket designation for these standards is 100BASE-T. The committee defined a number of alternatives to be used with different transmission media.

Figure 13.4 shows the terminology used in labeling the specifications and indicates the media used. All of the 100BASE-T options use the IEEE 802.3 MAC protocol and frame format. 100BASE-X refers to a set of options that use the physical medium specifications originally defined for Fiber Distributed Data Interface (FDDI; covered in the next section). All of the 100BASE-X schemes use two physical links between nodes: one for transmission and one for reception. 100BASE-TX makes use of shielded twisted pair (STP) or high-quality (Category 5) unshielded twisted pair (UTP). 100BASE-FX uses optical fiber.

In many buildings, each of the 100BASE-X options requires the installation of new cable. For such cases, 100BASE-T4 defines a lower-cost alternative that can use Category-3, voice grade UTP in addition to the higher-quality Category 5 UTP.[4] To achieve the 100-Mbps data rate over lower-quality cable, 100BASE-T4 dictates the use of four twisted pair lines between nodes, with the data transmission making use of three pairs in one direction at a time.

For all of the 100BASE-T options, the topology is similar to that of 10BASE-T, namely a star-wire topology.

Table 13.2 summarizes key characteristics of the 100BASE-T options.

100BASE-X

For all of the transmission media specified under 100BASE-X, a unidirectional data rate of 100 Mbps is achieved by transmitting over a single link (single twisted pair, single optical fiber). For all of these media, an efficient and effective signal

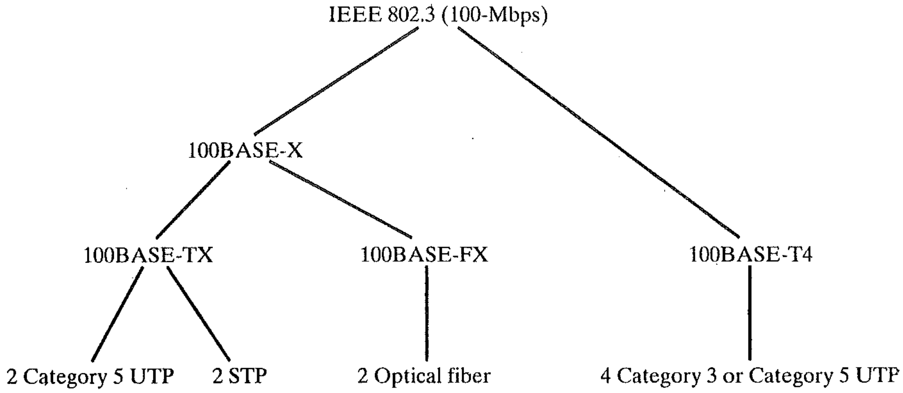


FIGURE 13.4 IEEE 802.3 100BASE-T options.

[4] See Chapter 3 for a discussion of Category 3 and Category 5 cable.

TABLE 13.2 IEEE 802.3 100BASE-T physical layer medium alternatives.

	100BASE-TX		100BASE-FX	100BASE-T4
Transmission medium	2 pair, STP	2 pair, Category 5 UTP	2 optical fibers	4 pair, Category 3, 4, or 5 UTP
Signaling technique	4B5B, NRZI	4B5B, NRZI	4B5B, NRZI	8B6T, NRZ
Data rate	100 Mbps	100 Mbps	100 Mbps	100 Mbps
Maximum segment length	100 m	100 m	100 m	100 m
Network span	200 m	200 m	400 m	200 m

encoding scheme is required. The one chosen was originally defined for FDDI, and can be referred to as 4B/5B-NRZI. See Appendix 13A for a description.

The 100BASE-X designation includes two physical-medium specifications, one for twisted pair, known as 100BASE-TX, and one for optical fiber, known as 100-BASE-FX.

100BASE-TX makes use of two pairs of twisted pair cable, one pair used for transmission and one for reception. Both STP and Category 5 UTP are allowed. The MTL-3 signaling scheme is used (described in Appendix 13A).

100BASE-FX makes use of two optical fiber cables, one for transmission and one for reception. With 100BASE-FX, a means is needed to convert the 4B/5B-NRZI code groups stream into optical signals. The technique used is known as intensity modulation. A binary 1 is represented by a burst or pulse of light; a binary 0 is represented by either the absence of a light pulse or by a light pulse at very low intensity.

100BASE-T4

100BASE-T4 is designed to produce a 100-Mbps data rate over lower-quality Category 3 cable, thus taking advantage of the large installed base of Category 3 cable in office buildings. The specification also indicates that the use of Category 5 cable is optional. 100BASE-T4 does not transmit a continuous signal between packets, which makes it useful in battery-powered applications.

For 100BASE-T4 using voice-grade Category 3 cable, it is not reasonable to expect to achieve 100 Mbps on a single twisted pair. Instead, 100BASE-T4 specifies that the data stream to be transmitted is split up into three separate data streams, each with an effective data rate of $33\frac{1}{3}$ Mbps. Four twisted pairs are used. Data are transmitted using three pairs and received using three pairs. Thus, two of the pairs must be configured for bidirectional transmission.

As with 100BASE-X, a simple NRZ encoding scheme is not used for 100BASE-T4; this would require a signaling rate of 33 Mbps on each twisted pair and does not provide synchronization. Instead, a ternary signaling scheme known as 8B6T is used (described in Appendix 13A).

13.2 TOKEN RING AND FDDI

Token ring is the most commonly used MAC protocol for ring-topology LANs. In this section, we examine two standard LANs that use token ring: IEEE 802.5 and FDDI.

IEEE 802.5 Medium Access Control

MAC Protocol

The token ring technique is based on the use of a small frame, called a *token*, that circulates when all stations are idle. A station wishing to transmit must wait until it detects a token passing by. It then seizes the token by changing one bit in the token, which transforms it from a token into a start-of-frame sequence for a data frame. The station then appends and transmits the remainder of the fields needed to construct a data frame.

When a station seizes a token and begins to transmit a data frame, there is no token on the ring, so other stations wishing to transmit must wait. The frame on the ring will make a round trip and be absorbed by the transmitting station. The transmitting station will insert a new token on the ring when both of the following conditions have been met:

- The station has completed transmission of its frame.
- The leading edge of the transmitted frame has returned (after a complete circulation of the ring) to the station.

If the bit length of the ring is less than the frame length, the first condition implies the second; if not, a station could release a free token after it has finished transmitting but before it begins to receive its own transmission. The second condition is not strictly necessary, and is relaxed under certain circumstances. The advantage of imposing the second condition is that it ensures that only one data frame at a time may be on the ring and that only one station at a time may be transmitting, thereby simplifying error-recovery procedures.

Once the new token has been inserted on the ring, the next station downstream with data to send will be able to seize the token and transmit. Figure 13.5 illustrates the technique. In the example, A sends a packet to C, which receives it and then sends its own packets to A and D.

Note that under lightly loaded conditions, there is some inefficiency with token ring because a station must wait for the token to come around before transmitting. However, under heavy loads, which is when it matters, the ring functions in a round-robin fashion, which is both efficient and fair. To see this, consider the configuration in Figure 13.5. After station A transmits, it releases a token. The first station with an opportunity to transmit is D. If D transmits, it then releases a token and C has the next opportunity, and so on.

The principal advantage of token ring is the flexible control over access that it

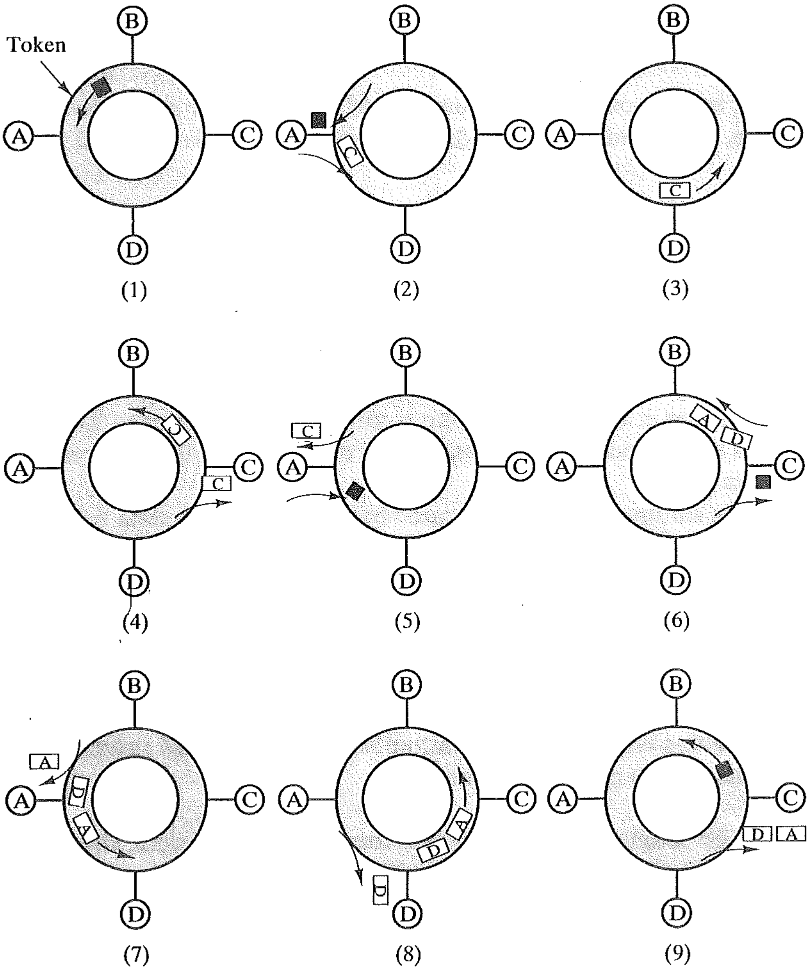


FIGURE 13.5 Token ring operation.

provides. In the simple scheme just described, the access if fair. As we shall see, schemes can be used to regulate access to provide for priority and for guaranteed bandwidth services.

The principal disadvantage of token ring is the requirement for token maintenance. Loss of the token prevents further utilization of the ring. Duplication of the token can also disrupt ring operation. One station must be selected as a monitor to ensure that exactly one token is on the ring and to ensure that a free token is reinserted, if necessary.

MAC Frame

Figure 13.6 depicts the frame format for the 802.5 protocol. It consists of the following fields:

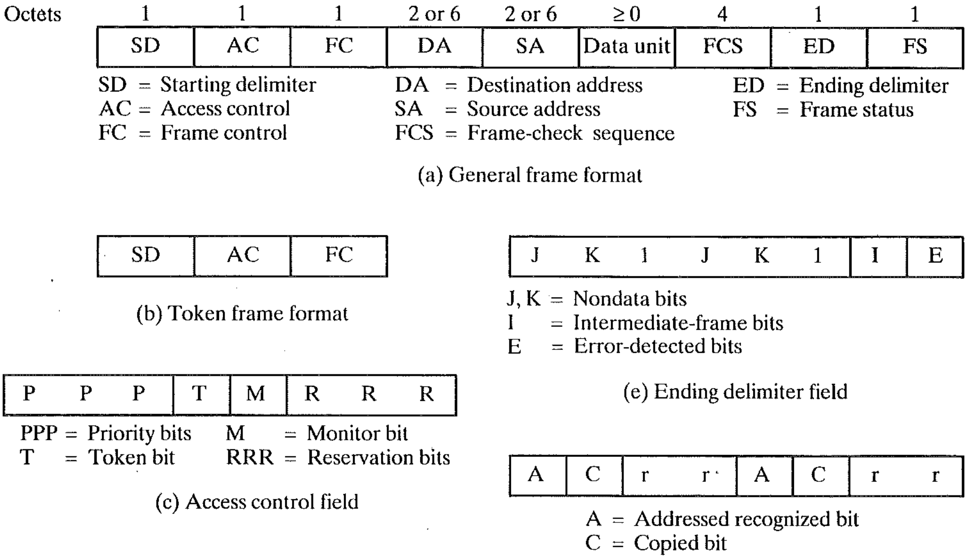


FIGURE 13.6 IEEE 802.5 frame format.

- **Starting delimiter (SD).** Indicates start of frame. The SD consists of signaling patterns that are distinguishable from data. It is coded as follows: JK0JK000, where J and K are nondata symbols. The actual form of a nondata symbol depends on the signal encoding on the medium.
- **Access control (AC).** Has the format PPPTMRRR, where PPP and RRR are 3-bit priority and reservation variables, and *M* is the monitor bit; their use is explained below. T indicates whether this is a token or data frame. In the case of a token frame, the only remaining field is ED.
- **Frame control (FC).** Indicates whether this is an LLC data frame. If not, bits in this field control operation of the token ring MAC protocol.
- **Destination address (DA).** As with 802.3.
- **Source address (SA).** As with 802.3.
- **Data unit.** Contains LLC data unit.
- **Frame check sequence (FCS).** As with 802.3.
- **End delimiter (ED).** Contains the error-detection bit (E), which is set if any repeater detects an error, and the intermediate bit (I), which is used to indicate that this is a frame other than the final one of a multiple-frame transmission.
- **Frame status (FS).** Contains the address recognized (A) and frame-copied (*C*) bits, whose use is explained below. Because the A and C bits are outside the scope of the FCS, they are duplicated to provide a redundancy check to detect erroneous settings.

We can now restate the token ring algorithm for the case when a single priority is used. In this case, the priority and reservation bits are set to 0. A station wishing to transmit waits until a token goes by, as indicated by a token bit of 0 in the AC field. The station seizes the token by setting the token bit to 1. The SD and AC fields of the received token now function as the first two fields of the outgoing frame. The station transmits one or more frames, continuing until either its supply of frames is exhausted or a token-holding timer expires. When the AC field of the last transmitted frame returns, the station sets the token bit to 0 and appends an ED field, resulting in the insertion of a new token on the ring.

Stations in the receive mode listen to the ring. Each station can check passing frames for errors and can set the *E* bit to 1 if an error is detected. If a station detects its own MAC address, it sets the *A* bit to 1; it may also copy the frame, setting the *C* bit to 1. This allows the originating station to differentiate three results of a frame transmission:

- Destination station nonexistent or not active ($A = 0, C = 0$)
- Destination station exists but frame not copied ($A = 1, C = 0$)
- Frame received ($A = 1, C = 1$)

Token Ring Priority

The 802.5 standard includes a specification for an optional priority mechanism. Eight levels of priority are supported by providing two 3-bit fields in each data frame and token: a priority field and a reservation field. To explain the algorithm, let us define the following variables:

P_f = priority of frame to be transmitted by station
P_s = service priority: priority of current token
P_r = value of P_s as contained in the last token received by this station
R_s = reservation value in current token
R_r = highest reservation value in the frames received by this station during the last token rotation

The scheme works as follows:

1. A station wishing to transmit must wait for a token with $P_s \le P_f$.
2. While waiting, a station may reserve a future token at its priority level (P_f). If a data frame goes by, and if the reservation field is less than its priority ($R_s < P_f$), then the station may set the reservation field of the frame to its priority ($R_s \leftarrow P_f$). If a token frame goes by, and if ($R_s < P_f$ AND $P_f < P_s$), then the station sets the reservation field of the frame to its priority ($R_s \leftarrow P_f$). This setting has the effect of preempting any lower-priority reservation.
3. When a station seizes a token, it sets the token bit to 1 to start a data frame, sets the reservation field of the data frame to 0, and leaves the priority field unchanged (the same as that of the incoming token frame).
4. Following transmission of one or more data frames, a station issues a new token with the priority and reservation fields set as indicated in Table 13.3.

TABLE 13.3 Actions performed by the token holder to implement the priority scheme [based on VALE92].

Conditions	Actions
Frame available AND $P_s \leq P_f$	Send frame
(Frame not available OR THT expired) AND $P_r \geq$ MAX $[R_r, P_f]$	Send token with: $P_s \leftarrow P_f$ $R_s \leftarrow$ MAX $[R_r, P_f]$
(Frame not available OR THT expired) AND $P_r <$ MAX $[R_r, P_f]$ AND $P_r > S_x$	Send token with: $P_s \leftarrow$ MAX $[R_r, P_f]$ $R_s \leftarrow 0$ Push $S_r \leftarrow P_r$ Push $S_x \leftarrow P_s$
(Frame not available OR THT expired) AND $P_r <$ MAX $[R_r, P_f]$ AND $P_r = S_x$	Send token with: $P_s \leftarrow$ MAX $[R_r, P_f]$ $R_s \leftarrow 0$ Pop S_x Push $S_x \leftarrow P_s$
(Frame not available OR (Frame available and $P_f < S_x$)) AND $P_s = S_x$ AND $R_r > S_r$	Send token with: $P_s \leftarrow R_r$ $R_s \leftarrow 0$ Pop S_x Push $S_x \leftarrow P_s$
(Frame not available OR (Frame available and $P_f < S_x$)) AND $P_s = S_x$ AND $R_r \leq S_r$	Send token with: $P_s \leftarrow R_r$ $R_s \leftarrow 0$ Pop S_r Pop S_x

The effect of the above steps is to sort the competing claims and to allow the waiting transmission of highest priority to seize the token as soon as possible. A moment's reflection reveals that, as stated, the algorithm has a ratchet effect on priority, driving it to the highest used level and keeping it there. To avoid this, a station that raises the priority (issues a token that has a higher priority than the token that it received) has the responsibility of later lowering the priority to its previous level. Therefore, a station that raises priority must remember both the old and the new priorities and must downgrade the priority of the token at the appropriate time. In essence, each station is responsible for assuring that no token circulates indefinitely because its priority is too high. By remembering the priority of earlier transmissions, a station can detect this condition and downgrade the priority to a previous, lower priority or reservation.

To implement the downgrading mechanism, two stacks are maintained by each station, one for reservations and one for priorities:

S_x = stack used to store new values of token priority
S_r = stack used to store old values of token priority

The reason that stacks rather than scalar variables are required is that the priority can be raised a number of times by one or more stations. The successive raises must be unwound in the reverse order.

To summarize, a station having a higher priority frame to transmit than the current frame can reserve the next token for its priority level as the frame passes by. When the next token is issued, it will be at the reserved priority level. Stations of lower priority cannot seize the token, so it passes to the reserving station or an intermediate station with data to send of equal or higher priority level than the reserved priority level. The station that upgraded the priority level is responsible for downgrading it to its former level when all higher-priority stations are finished. When that station sees a token at the higher priority after it has transmitted, it can assume that there is no more higher-priority traffic waiting, and it downgrades the token before passing it on.

Figure 13.7 is an example. The following events occur:

1. A is transmitting a data frame to B at priority 0. When the frame has completed a circuit of the ring and returns to A, A will issue a token frame. However, as the data frame passes D, D makes a reservation at priority 3 by setting the reservation field to 3.
2. A issues a token with the priority field set to 3.
3. If neither B nor C has data of priority 3 or greater to send, they cannot seize the token. The token circulates to D, which seizes the token and issues a data frame.
4. After D's data frame returns to D, D issues a new token at the same priority as the token that it received: priority 3.
5. A sees a token at the priority level that it used to last issue a token; it therefore seizes the token even if it has no data to send.
6. A issues a token at the previous priority level: priority 0.

Note that, after A has issued a priority 3 token, any station with data of priority 3 or greater may seize the token. Suppose that at this point station C now has priority 4 data to send. C will seize the token, transmit its data frame, and reissue a priority 3 token, which is then seized by D. By the time that a priority 3 token arrives at A, all intervening stations with data of priority 3 or greater to send will have had the opportunity. It is now appropriate, therefore, for A to downgrade the token.

Early Token Release

When a station issues a frame, if the bit length of the ring is less than that of the frame, the leading edge of the transmitted frame will return to the transmitting station before it has completed transmission; in this case, the station may issue a token as soon as it has finished frame transmission. If the frame is shorter than the bit length of the ring, then after a station has completed transmission of a frame, it must

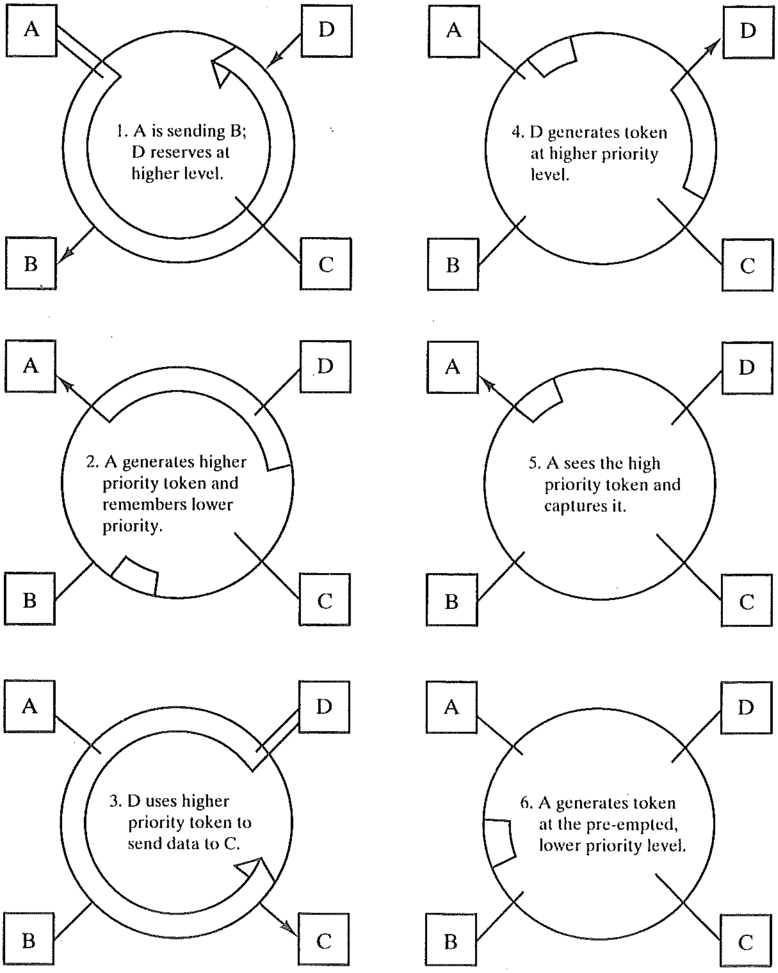

A
D
B
C
1. A is sending B; D reserves at higher level.
4. D generates token at higher priority level.
2. A generates higher priority token and remembers lower priority.
5. A sees the high priority token and captures it.
3. D uses higher priority token to send data to C.
6. A generates token at the pre-empted, lower priority level.

FIGURE 13.7 **IEEE token ring priority scheme.**

wait until the leading edge of the frame returns before issuing a token. In this latter case, some of the potential capacity of the ring is unused.

To allow for more efficient ring utilization, an early token release (ETR) option has been added to the 802.5 standard. ETR allows a transmitting station to release a token as soon as it completes frame transmission, whether or not the frame header has returned to the station. The priority used for a token released prior to receipt of the previous frame header is the priority of the most recently received frame.

One effect of ETR is that access delay for priority traffic may increase when the ring is heavily loaded with short frames. Because a station must issue a token before it can read the reservation bits of the frame it just transmitted, the station will not respond to reservations. Thus, the priority mechanism is at least partially disabled.

Stations that implement ETR are compatible and interoperable with those that do not complete such implementation.

IEEE 802.5 Physical Layer Specification

The 802.5 standard (Table 13.4) specifies the use of shielded twisted pair with data rates of 4 and 16 Mbps using Differential Manchester encoding. An earlier specification of a 1-Mbps system has been dropped from the most recent edition of the standard.

A recent addition to the standard is the use of unshielded twisted pair at 4 Mbps.

TABLE 13.4 IEEE 802.5 physical layer medium alternatives.

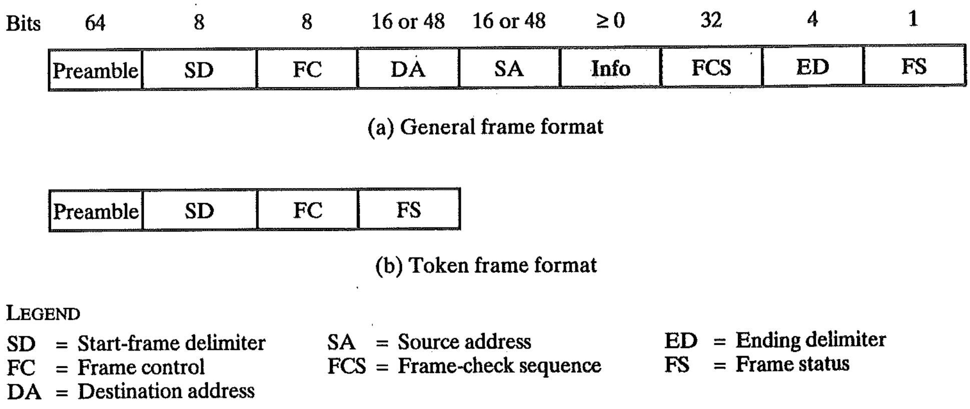
Transmission medium	Shielded twisted pair	Unshielded twisted pair
Data rate (Mbps)	4 or 16	4
Signaling technique	Differential Manchester	Differential Manchester
Maximum number of repeaters	250	72
Maximum length between repeaters	Not specified	Not specified

FDDI Medium Access Control

FDDI is a token ring scheme, similar to the IEEE 802.5 specification, that is designed for both LAN and MAN applications. There are several differences that are designed to accommodate the higher data rate (100 Mbps) of FDDI.

MAC Frame

Figure 13.8 depicts the frame format for the FDDI protocol. The standard defines the contents of this format in terms of symbols, with each data symbol corresponding to



(a) General frame format

(b) Token frame format

LEGEND

SD = Start-frame delimiter
FC = Frame control
DA = Destination address
SA = Source address
FCS = Frame-check sequence
ED = Ending delimiter
FS = Frame status

FIGURE 13.8 FDDI frame formats.

4 data bits. Symbols are used because, at the physical layer, data are encoded in 4-bit chunks. However, MAC entities, in fact, must deal with individual bits, so the discussion that follows sometimes refers to 4-bit symbols and sometime to bits. A frame other than a token frame consists of the following fields:

- **Preamble.** Synchronizes the frame with each station's clock. The originator of the frame uses a field of 16 idle symbols (64 bits); subsequent repeating stations may change the length of the field, as consistent with clocking requirements. The idle symbol is a nondata fill pattern. The actual form of a nondata symbol depends on the signal encoding on the medium.
- **Starting delimiter (SD).** Indicates start of frame. It is coded as JK, where J and K are nondata symbols.
- **Frame control (FC).** Has the bit format CLFFZZZZ, where C indicates whether this is a synchronous or asynchronous frame (explained below); L indicates the use of 16- or 48-bit addresses; FF indicates whether this is an LLC, MAC control, or reserved frame. For a control frame, the remaining 4 bits indicate the type of control frame.
- **Destination address (DA).** Specifies the station(s) for which the frame is intended. It may be a unique physical address, a multicast-group address, or a broadcast address. The ring may contain a mixture of 16- and 48-bit address lengths.
- **Source address (SA).** Specifies the station that sent the frame.
- **Information.** Contains an LLC data unit or information related to a control operation.
- **Frame check sequence (FCS).** A 32-bit cyclic redundancy check, based on the FC, DA, SA, and information fields.
- **Ending delimiter (ED).** Contains a nondata symbol (T) and marks the end of the frame, except for the FS field.
- **Frame Status (FS).** Contains the error detected (E), address recognized (A), and frame copied (F) indicators. Each indicator is represented by a symbol, which is R for "reset" or "false" and S for "set" or "true."

A token frame consists of the following fields:

- **Preamble.** As above.
- **Starting delimiter.** As above.
- **Frame control (FC).** Has the bit format 10000000 or 11000000 to indicate that this is a token.
- **Ending delimiter (ED).** Contains a pair of nondata symbols (T) that terminate the token frame.

A comparison with the 802.5 frame (Figure 13.6) shows that the two are quite similar. The FDDI frame includes a preamble to aid in clocking, which is more demanding at the higher data rate. Both 16- and 48-bit addresses are allowed in the same network with FDDI; this is more flexible than the scheme used on all the 802 standards. Finally, there are some differences in the control bits. For example, FDDI does not include priority and reservation bits; capacity allocation is handled in a different way, as described below.

MAC Protocol

The basic (without capacity allocation) FDDI MAC protocol is fundamentally the same as IEEE 802.5. There are two key differences:

1. In FDDI, a station waiting for a token seizes the token by aborting (failing to repeat) the token transmission as soon as the token frame is recognized. After the captured token is completely received, the station begins transmitting one or more data frames. The 802.5 technique of flipping a bit to convert a token to the start of a data frame was considered impractical because of the high data rate of FDDI.
2. In FDDI, a station that has been transmitting data frames releases a new token as soon as it completes data frame transmission, even if it has not begun to receive its own transmission. This is the same technique as the early token release option of 802.5. Again, because of the high data rate, it would be too inefficient to require the station to wait for its frame to return, as in normal 802.5 operation.

Figure 13.9 gives an example of ring operation. After station A has seized the token, it transmits frame F1, and immediately transmits a new token. F1 is addressed to station C, which copies it as it circulates past. The frame eventually returns to A, which absorbs it. Meanwhile, B seizes the token issued by A and transmits F2 followed by a token. This action could be repeated any number of times, so that, at any one time, there may be multiple frames circulating the ring. Each station is responsible for absorbing its own frames based on the source address field.

A further word should be said about the frame status (FS) field. Each station can check passing bits for errors and can set the E indicator if an error is detected. If a station detects its own address, it sets the A indicator; it may also copy the frame, setting the C indicator; this allows the originating station, when it absorbs a frame that it previously transmitted, to differentiate among three conditions:

- Station nonexistent/nonactive
- Station active but frame not copied
- Frame copied

When a frame is absorbed, the status indicators (E, A, C) in the FS field may be examined to determine the result of the transmission. However, if an error or failure to receive condition is discovered, the MAC protocol entity does not attempt to retransmit the frame, but reports the condition to LLC. It is the responsibility of LLC or some higher-layer protocol to take corrective action.

Capacity Allocation

The priority scheme used in 802.5 will not work in FDDI, as a station will often issue a token before its own transmitted frame returns. Hence, the use of a reservation

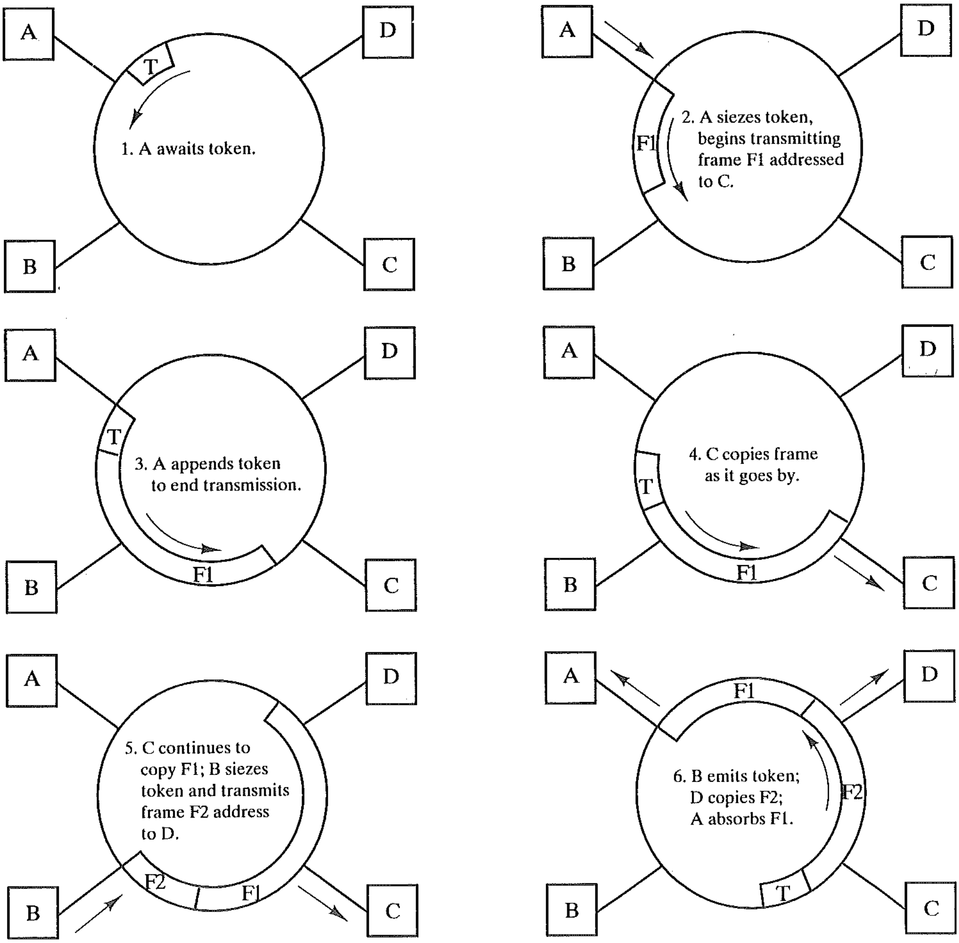

A
B
C
D
T
1. A awaits token.
2. A siezes token, begins transmitting frame F1 addressed to C.
F1
3. A appends token to end transmission.
4. C copies frame as it goes by.
5. C continues to copy F1; B siezes token and transmits frame F2 address to D.
F2
6. B emits token; D copies F2; A absorbs F1.

FIGURE 13.9 Example of FDDI token ring operation.

field is not effective. Furthermore, the FDDI standard is intended to provide for greater control over the capacity of the network than 802.5 to meet the requirements for a high-speed LAN. Specifically, the FDDI capacity-allocation scheme seeks to accommodate a mixture of stream and bursty traffic,

To accommodate this requirement, FDDI defines two types of traffic: synchronous and asynchronous. Each station is allocated a portion of the total capacity (the portion may be zero); the frames that it transmits during this time are referred to as synchronous frames. Any capacity that is not allocated or that is allocated but not used is available for the transmission of additional frames, referred to as asynchronous frames.

The scheme works as follows. A *target token-rotation time* (TTRT) is defined; each station stores the same value for TTRT. Some or all stations may be provided a *synchronous allocation* (SA_i), which may vary among stations. The allocations must be such that

$$\text{DMax} + \text{FMax} + \text{Token Time} + \sum \text{SA}_i \leq \text{TTRT}$$

where

$$\begin{aligned} \text{SA}_i &= \text{synchronous allocation for station } i \\ \text{DMax} &= \text{propagation time for one complete circuit of the ring} \\ \text{FMAX} &= \text{time required to transmit a maximum-length frame (4500 octets)} \\ \text{TokenTime} &= \text{time required to transmit a token} \end{aligned}$$

The assignment of values for SA_i is by means of a station management protocol involving the exchange of station management frames. The protocol assures that the above equation is satisfied. Initially, each station has a zero allocation, and it must request a change in the allocation. Support for synchronous allocation is optional; a station that does not support synchronous allocation may only transmit asynchronous traffic.

All stations have the same value of TTRT and a separately assigned value of SA_i. In addition, several variables that are required for the operation of the capacity-allocation algorithm are maintained at each station:

- Token-rotation timer (TRT)
- Token-holding timer (THT)
- Late counter (LC)

Each station is initialized with TRT set equal to TTRT and LC set to zero.[5] When the timer is enabled, TRT begins to count down. If a token is received before TRT expires, TRT is reset to TTRT. If TRT counts down to 0 before a token is received, then LC is incremented to 1 and TRT is reset to TTRT and again begins to count down. IF TRT expires a second time before receiving a token, LC is incremented to 2, the token is considered lost, and a Claim process (described below) is initiated. Thus, LC records the number of times, if any, that TRT has expired since the token was last received at that station. The token is considered to have arrived early if TRT has not expired since the station received the token—that is, if LC = 0.

When a station receives the token, its actions will depend on whether the token is early or late. If the token is early, the station saves the remaining time from TRT in THT, resets TRT, and enables TRT:

$$\begin{aligned} &\text{THT} \leftarrow \text{TRT} \\ &\text{TRT} \leftarrow \text{TTRT} \\ &\text{enable TRT} \end{aligned}$$

The station can then transmit according to the following rules:

[5] Note: All timer values in the standard are negative numbers with counters counting up to zero. For clarity, the discussion uses positive numbers.

1. It may transmit synchronous frames for a time SA_i.
2. After transmitting synchronous frames, or if there were no synchronous frames to transmit, THT is enabled. The station may begin transmission of asynchronous frames as long as THT > 0.

If a station receives a token and the token is late, then LC is set to zero and TRT continues to run. The station can then transmit synchronous frames for a time SA_i. The station may not transmit any asynchronous frames.

This scheme is designed to assure that the time between successive sightings of a token is on the order of TTRT or less. Of this time, a given amount is always available for synchronous traffic, and any excess capacity is available for asynchronous traffic. Because of random fluctuations in traffic, the actual token-circulation time may exceed TTRT, as demonstrated below.

Figure 13.10 provides a simplified example of a 4-station ring. The following assumptions are made:

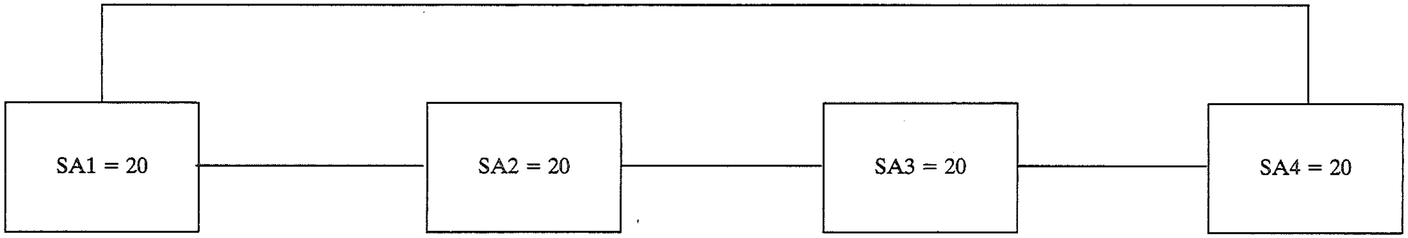
1. Traffic consists of fixed-length frames.
2. TTRT = 100 frame times.
3. $SA_i = 20$ frame times for each station.
4. Each station is always prepared to send its full synchronous allocation as many asynchronous frames as possible.
5. The total overhead during one complete token circulation is 4 frame times (one frame time per station).

One row of the table corresponds to one circulation of the token. For each station, the token arrival time is shown, followed by the value of TRT at the time of arrival, followed by the number of synchronous and asynchronous frames transmitted while the station holds the token.

The example begins after a period during which no data frames have been sent, so that the token has been circulating as rapidly as possible (4 frame times). Thus, when Station 1 receives the token at time 4, it measures a circulation time of 4 (its TRT = 96). It is therefore able to send not only its 20 synchronous frames but also 96 asynchronous frames; recall that THT is not enabled until after the station has sent its synchronous frames. Station 2 experiences a circulation time of 120 (20 frames + 96 frames + 4 overhead frames), but is nevertheless entitled to transmit its 20 synchronous frames. Note that if each station continues to transmit its maximum allowable synchronous frames, then the circulation time surges to 180 (at time 184), but soon stabilizes at approximately 100. With a total synchronous utilization of 80 and an overhead of 4 frame times, there is an average capacity of 16 frame times available for asynchronous transmission. Note that if all stations always have a full backlog of asynchronous traffic, the opportunity to transmit asynchronous frames is distributed among them.

FDDI Physical Layer Specification

The FDDI standard specifies a ring topology operating at 100 Mbps. Two media are included (Table 13.5). The optical fiber medium uses 4B/5B-NRZI encoding. Two



Arrival Time	TRT	Sync	Async	Arrival Time	TRT	Sync	Async	Arrival Time	TRT	Sync	Async	Arrival Time	TRT	Sync	Async
0	100	0	0	1	100	0	0	2	100	0	0	3	100	0	0
4	96	20	96	121	80*	20	0	142	60*	20	0	163	40*	20	0
184	20*	20	0	205	96*	20	0	226	76*	20	0	247	56*	20	0
268	36*	20	0	289	12	20	12	322	80*	20	0	343	60*	20	0
364	40*	20	0	385	4	20	4	410	92*	20	0	431	72*	20	0
452	52*	20	0	473	16	20	16	510	92*	20	0	531	72*	20	0
552	52*	20	0	573	0	20	0	594	8	20	8	623	80*	20	0
644	60*	20	0	665	8	20	8	694	0	20	0	715	88*	20	0
736	68*	20	0	757	8	20	8	786	8	20	8	815	88*	20	0
836	68*	20	0	857	0	20	0	878	8	20	8	907	96*	20	0
928	76*	20	0	949	8	20	8	978	0	20	0	999	4	20	4
1024	80*	20	0	1045	4	20	4	1070	8	20	8	1099	0	20	0

*LC = 1; otherwise LC = 0

FIGURE 13.10 Operation of FDDI capaciaty allocation scheme.

TABLE 13.5 FDDI physical layer medium alternatives.

Transmission medium	Optical fiber	Twisted pair
Data rate (Mbps)	100	100
Signaling technique	4B/5B/NRZI	MLT-3
Maximum number of repeaters	100	100
Maximum length between repeaters	2 km	100 m

twisted pair media are specified: 100-ohm Category 5 unshielded twisted pair[6] and 150-ohm shielded twisted pair. For both twisted pair media, MLT-3 encoding is used. See Appendix 13A for a discussion of these encoding schemes.

13.3 100VG-ANYLAN

Like 100BASE-T, 100VG-AnyLAN[7] is intended to be a 100-Mbps extension to the 10-Mbps Ethernet and to support IEEE 802.3 frame types. It also provides compatibility with IEEE 802.5 token ring frames. 100VG-AnyLAN uses a new MAC scheme known as *demand priority* to determine the order in which nodes share the network. Because this specification does not use CSMA/CD, it has been standardized under a new working group, IEEE 802.12, rather than allowed to remain in the 802.3 working group.

Topology

The topology for a 100VG-AnyLAN network is hierarchical star. The simplest configuration consists of a single central hub and a number of attached devices. More complex arrangements are possible, in which there is a single root hub, with one or more subordinate level-2 hubs; a level-2 hub can have additional subordinate hubs at level 3, and so on to an arbitrary depth.

Medium Access Control

The MAC algorithm for 802.12 is a round-robin scheme with two priority levels. We first describe the algorithm for a single-hub network and then discuss the general case.

Single-Hub Network

When a station wishes to transmit a frame, it first issues a request to the central hub and then awaits permission from the hub to transmit. A station must designate each request as normal-priority or high-priority.

[6] See Chapter 3 for a discussion of Category 5 unshielded twisted pair.

[7] VG = voice grade. AnyLAN = support for multiple LAN frame types.

The central hub continually scans all of its ports for a request in round-robin fashion. Thus, an *n*-port hub looks for a request first on port 1, then on port 2, and so on up to port *n*. The scanning process then begins again at port 1. The hub maintains two pointers: a high-priority pointer and a normal-priority pointer. During one complete cycle, the hub grants each high-priority request in the order in which the requests are encountered. If at any time there are no pending high-priority requests, the hub will grant any normal-priority requests that it encounters.

Figure 13.11 gives an example. The sequence of events is as follows:

1. The hub sets both pointers to port 1 and begins scanning. The first request encountered is a low-priority request from port 2. The hub grants this request and updates the low-priority pointer to port 3.
2. Port 2 transmits a low-priority frame. The hub receives this frame and retransmits it. During this period, two high-priority requests are generated.
3. Once the frame from port 2 is transmitted, the hub begins granting high-priority requests in round-robin order, beginning with port 1 and followed by port 5. The high-priority pointer is set to port 6.
4. After the high-priority frame from port 5 completes, there are no outstanding high-priority requests and the hub turns to the normal-priority requests. Four requests have arrived since the last low-priority frame was transmitted: from ports 2, 7, 3, and 6. Because the normal-priority pointer is set to port 3, these requests will be granted in the order 3, 6, 7, and 2 if no other requests intervene.

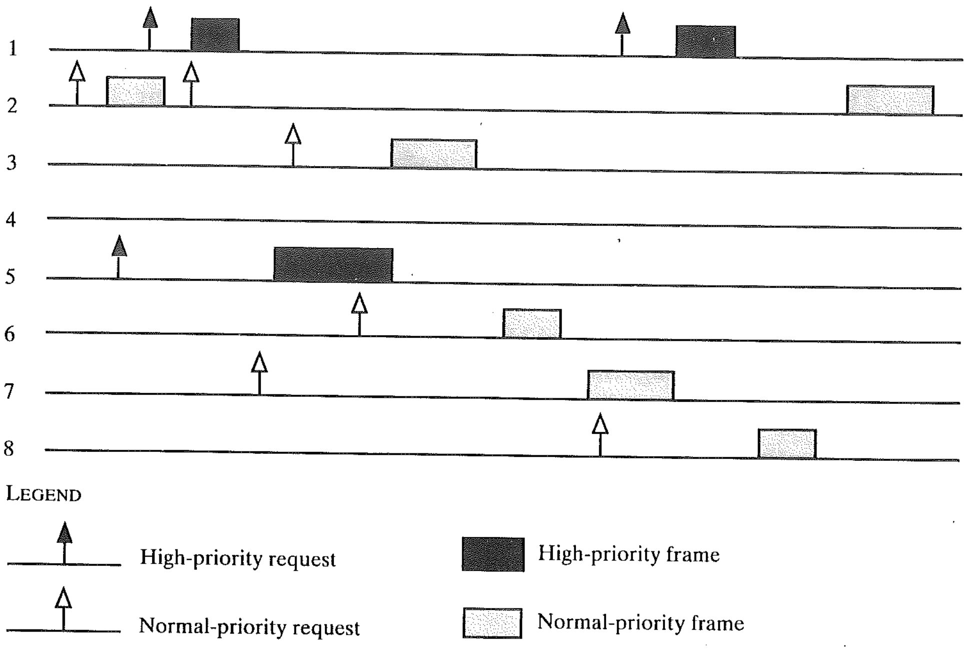


FIGURE 13.11 Example frame sequence in a single-repeater network.

5. The frames from ports 3, 6, and 7 are transmitted in turn. During the transmission of frame 7, a high-priority request arrives from port 1 and a normal priority request arrives from port 8. The hub sets the normal-priority pointer to port 8.
6. Because high-priority requests take precedence, port 1 is granted access next.
7. After the frame from port 1 is transmitted, the hub has two outstanding normal-priority requests. The request from port 2 has been waiting the longest; however, port 8 is next in round-robin order to be satisfied and so its request is granted, followed by that of port 2.

Hierarchical Network

In a hierarchical network, all of the end-system ports on all hubs are treated as a single set of ports for purposes of the round-robin algorithm. The hubs are configured to cooperate in scanning the ports in the proper order. Put another way, the set of hubs is treated logically as a single hub.

Figure 13.12 indicates port ordering in a hierarchical network. The order is generated by traversing a tree representation of the network, in which the branches under each node in the tree are arranged in increasing order from left to right. With this convention, the port order is generated by traversing the tree in what is referred to as preorder traversal, which is defined recursively as follows:

1. Visit the root.
2. Traverse the subtrees from left to right.

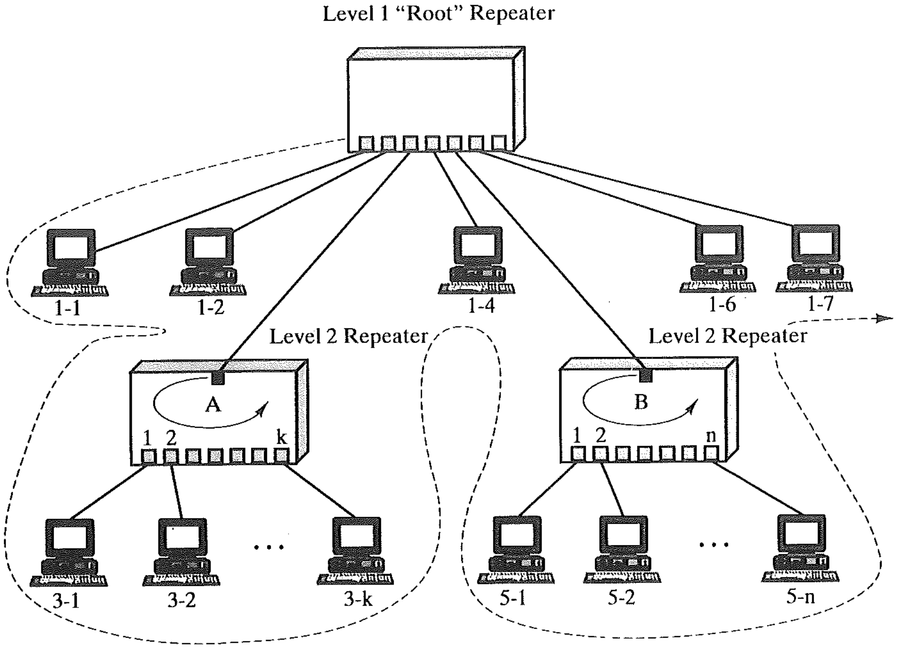


FIGURE 13.12 Port ordering in a two-level IEEE 802.12 network.

This method of traversal is also known as a depth-first search of the tree.

Let us now consider the mechanics of medium access and frame transmission in a hierarchical network. There are a number of contingencies to consider. First, consider the behavior of the root hub. This hub performs the high-priority and normal-priority round-robin algorithms for all directly attached devices. Thus, if there are one or more pending high-priority requests, the hub grants these requests in round-robin fashion. If there are no pending high-priority requests, the hub grants any normal-priority requests in round-robin fashion. When a request is granted by the root hub to a directly-attached end system, that system may immediately transmit a frame. When a request is granted by the root hub to a directly-attached level-2 hub, then control passes to the level-2 hub, which then proceeds to execute its own round-robin algorithms.

Any end system that is ready to transmit sends a request signal to the hub to which it attaches. If the end system is attached directly to the root hub, then the request is conveyed directly to the root hub. If the end system is attached to a lower-level hub, then the request is transmitted directly to that hub. If that hub does not currently have control of the round-robin algorithm, then it passes the request up to the next higher-level hub. Eventually, all requests that are not granted at a lower level are passed up to the root hub.

The scheme described so far does enforce a round-robin discipline among all attached stations, but two refinements are needed. First, a preemption mechanism is needed. This is best explained by an example. Consider the following sequence of events:

1. Suppose that the root hub (R) in Figure 13.12 is in control and that there are no high-priority requests pending anywhere in the network. However, stations 5-1, 5-2, and 5-3 have all issued normal-priority requests, causing hub B to issue a normal-priority request to R.
2. R will eventually grant this request, passing control to B.
3. B then proceeds to honor its outstanding requests one at a time.
4. While B is honoring its first normal-priority request, station 1-6 issues a high-priority request.
5. In response to the request from 1-6, R issues a preempt signal to B; this tells B to relinquish control after the completion of the current transmission.
6. R grants the request of 1-6 and then continues its round-robin algorithm.

The second refinement is a mechanism to prevent a nonroot hub from retaining control indefinitely. To see the problem, suppose that B in Figure 13.12 has a high-priority request pending from 5-1. After receiving control from R, B grants the request to 5-1. Meanwhile, other stations subordinate to B issue high-priority requests. B could continue in round-robin fashion to honor all of these requests. If additional requests arrive from other subordinates of B during these other transmissions, then B would be able to continue granting requests indefinitely, even though there are other high-priority requests pending elsewhere in the network. To prevent this kind of lockup, a subordinate hub may only retain control for a signal round-robin cycle through all of its ports.

The IEEE 802.12 MAC algorithm is quite effective. When multiple stations offer high loads, the protocol behaves much like a token ring protocol, with network access rotating among all high-priority requesters, followed by low-priority requesters when there are no outstanding high-priority requests. At low load, the protocol behaves in a similar fashion to CSMA/CD under low load: A single requester gains medium access almost immediately.

100VG-AnyLANPhysical Layer Specification

The current version of IEEE 801.12 calls for the use of 4-pair unshielded twisted pair (UTP) using Category 3, 4, or 5 cable. Future versions will also support 2-pair Category-5 UTP, shielded twisted pair, and fiber optic cabling. In all cases, the data rate is 100 Mbps.

Signal Encoding

A key objective of the 100VG-AnyLAN effort is to be able to achieve 100 Mbps over short distances using ordinary voice-grade (Category 3) cabling. The advantage of this is that in many existing buildings, there is an abundance of voice-grade cabling and very little else. Thus, if this cabling can be used, installation costs are minimized.

With present technology, a data rate of 100 Mbps over one or two Category 3 pairs is impractical. To meet the objective, 100VG-AnyLAN specifies a novel encoding scheme that involves using four pair to transmit data in a half-duplex mode. Thus, to achieve a data rate of 100 Mbps, a data rate of only 25 Mbps is needed on each channel. An encoding scheme known as 5B6B is used. (See Appendix 13A for a description.)

Data from the MAC layer can be viewed as a stream of bits. The bits from this stream are taken five at a time to form a stream of quintets that are then passed down to the four transmission channels in round-robin fashion. Next, each quintet passes through a simple scrambling algorithm to increase the number of transitions between 0 and 1 and to improve the signal spectrum. At this point, it might be possible to simply transmit the data using NRZ. However, even with the scrambling, the further step of 5B6B encoding is used to ensure synchronization and also to maintain dc balance.

Because the MAC frame is being divided among four channels, the beginning and ending of a MAC frame must be delimited on each of the channels, which is the purpose of the delimiter generators. Finally, NRZ transmission is used on each channel.

13.4 ATM LANs

A document on customer premises networks jointly prepared by Apple, Bellcore, Sun, and Xerox [ABSX92] identifies three generations of LANs:

- **First Generation.** Typified by the CSMA/CD and Token Ring LANs. The first generation provided terminal-to-host connectivity and supported client/server architectures at moderate data rates.

- **Second Generation.** Typified by FDDI. The second generation responds to the need for backbone LANs and for support of high-performance workstations.
- **Third Generation.** Typified by ATM LANs. The third generation is designed to provide the aggregate throughputs and real-time transport guarantees that are needed for multimedia applications.

Typical requirements for a third generation LAN include the following:

1. Support multiple, guaranteed classes of service. A live video application, for example, may require a guaranteed 2-Mbps connection for acceptable performance, while a file transfer program can utilize a *background* class of service.
2. Provide scalable throughput that is capable of growing in both per-host capacity (to enable applications that require large volumes of data in and out of a single host) and in aggregate capacity (to enable installations to grow from a few to several hundred high-performance hosts).
3. Facilitate the interworking between LAN and WAN technology.

ATM is ideally suited to these requirements. Using virtual paths and virtual channels, multiple classes of service are easily accommodated, either in a preconfigured fashion (permanent connections) or on demand (switched connections). ATM is easily scalable by adding more ATM switching nodes and using higher (or lower) data rates for attached devices. Finally, with the increasing acceptance of cell-based transport for wide-area networking, the use of ATM for a premises network enables seamless integration of LANs and WANs.

The term ATM LAN has been used by vendors and researchers to apply to a variety of configurations. At the very least, an ATM LAN implies the use of ATM as a data transport protocol somewhere within the local premises. Among the possible types of ATM LANs:

- **Gateway to ATM WAN.** An ATM switch acts as a router and traffic concentrator for linking a premises network complex to an ATM WAN.
- **Backbone ATM switch.** Either a single ATM switch or a local network of ATM switches interconnect other LANs.
- **Workgroup ATM.** High-performance multimedia workstations and other end systems connect directly to an ATM switch.

These are all "pure" configurations. In practice, a mixture of two or all three of these types of networks is used to create an ATM LAN.

Figure 13.13 shows an example of a backbone ATM LAN that includes links to the outside world. In this example, the local ATM network consists of four switches interconnected with high-speed, point-to-point links running at the standardized ATM rates of 155 and 622 Mbps. On the premises, there are three other LANs, each of which has a direct connection to one of the ATM switches. The data rate from an ATM switch to an attached LAN conforms to the native data rate of that LAN. For example, the connection to the FDDI network is at 100 Mbps. Thus,

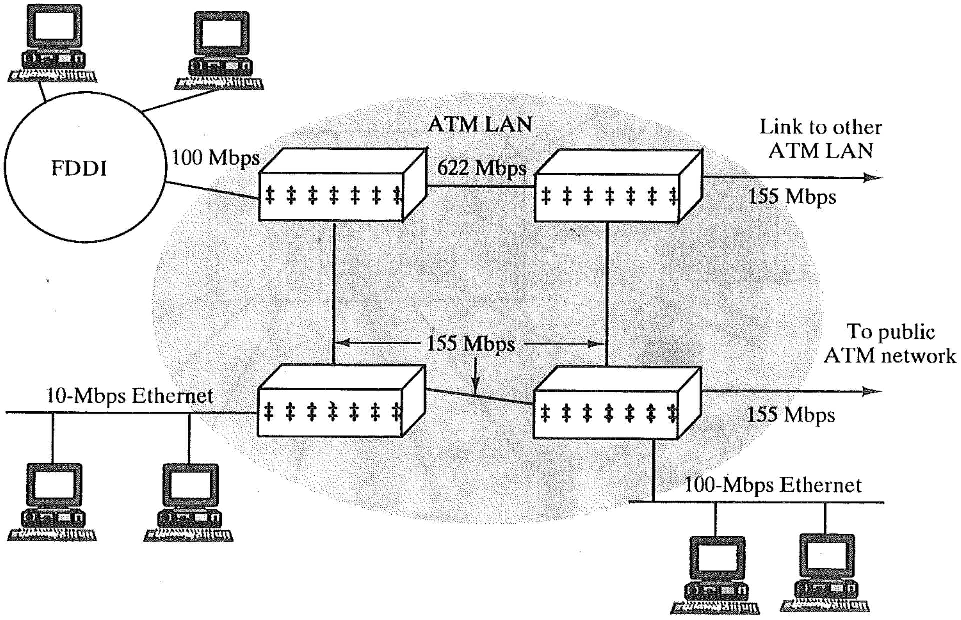


FIGURE 13.13 Example ATM LAN configuration

the switch must include some buffering and speed conversion capability to map the data rate from the attached LAN to an ATM data rate. The ATM switch must also perform some sort of protocol conversion from the MAC protocol used on the attached LAN to the ATM cell stream used on the ATM network. A simple approach is for each ATM switch that attaches to a LAN to function as a bridge or router.[8]

An ATM LAN configuration such as that shown in Figure 13.13 provides a relatively painless method for inserting a high-speed backbone into a local environment. As the on-site demand rises, it is a simple matter to increase the capacity of the backbone by adding more switches, increasing the throughput of each switch, and increasing the data rate of the trunks between switches. With this strategy, the load on individual LANs within the premises can be increased, and the number of LANs can grow.

However, this simple backbone ATM LAN does not address all of the needs for local communications. In particular, in the simple backbone configuration, the end systems (workstations, servers, etc.) remain attached to shared-media LANs with the limitations on data rate imposed by the shared medium.

A more advanced, and more powerful approach, is to use ATM technology in a hub. Figure 13.14 suggests the capabilities that can be provided with this approach. Each ATM hub includes a number of ports that operate at different data rates and that use different protocols. Typically, such a hub consists of a number of rack-mounted modules, with each module containing ports of a given data rate and protocol.

[8] The functionality of bridges and routers is examined in depth in Chapters 14 and 16.

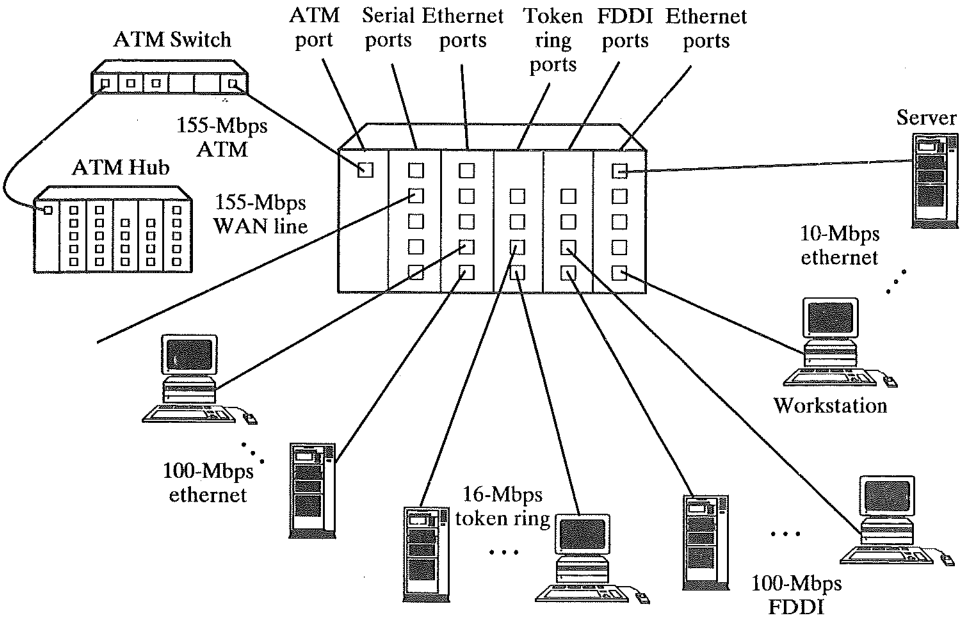


FIGURE 13.14 ATM LAN hub configuration.

The key difference between the ATM hub shown in Figure 13.14 and the ATM nodes depicted in Figure 13.13 is the way in which individual end systems are handled. Notice that in the ATM hub, each end system has a dedicated point-to-point link to the hub. Each end system includes the communications hardware and software to interface to a particular type of LAN, but in each case, the LAN contains only two devices: the end system and the hub! For example, each device attached to a 10-Mbps Ethernet port operates using the CSMA/CD protocol at 10 Mbps. However, because each end system has its own dedicated line, the effect is that each system has its own dedicated 10-Mbps Ethernet. Therefore, each end system can operate at close to the maximum 10-Mbps data rate.

The use of a configuration such as that of either Figure 13.13 or 13.14 has the advantage that existing LAN installations and LAN hardware—so-called legacy LANs—can continue to be used while ATM technology is introduced. The disadvantage is that the use of such a mixed-protocol environment requires the implementation of some sort of protocol conversion capability, a topic that is explored in Section 10.3. A simpler approach, but one that requires that end systems be equipped with ATM capability, is to implement a "pure" ATM LAN.

One issue that was not addressed in our discussion so far has to do with the interoperability of end systems on a variety of interconnected LANs. End systems attached directly to one of the legacy LANs implement the MAC layer appropriate to that type of LAN. End systems attached directly to an ATM network implement the ATM and AAL protocols. As a result, there are three areas of compatibility to consider:

1. Interaction between an end system on an ATM network and an end system on a legacy LAN.
2. Interaction between an end system on a legacy LAN and an end system on another legacy LAN of the same type (e.g., two IEEE 802.3 networks).
3. Interaction between an end system on a legacy LAN and an end system on another legacy LAN of a different type (e.g., an IEEE 802.3 network and an IEEE 802.5 network).

A discussion of approaches to satisfying these requirement involves consideration of bridge logic. Accordingly, we defer this discussion until Chapter 14.

13.5 FIBRE CHANNEL

As the speed and memory capacity of personal computers, workstations, and servers have grown, and as applications have become ever more complex with greater reliance on graphics and video, the requirement for greater speed in delivering data to the processor has grown. This requirement affects two methods of data communications with the processor: I/O channel and network communications.

An I/O channel is a direct point-to-point or multipoint communications link, predominantly hardware-based and designed for high speed over very short distances. The I/O channel transfers data between a buffer at the source device and a buffer at the destination device, moving only the user contents from one device to another, without regard for the format or meaning of the data. The logic associated with the channel typically provides the minimum control necessary to manage the transfer plus hardware error detection. I/O channels typically manage transfers between processors and peripheral devices, such as disks, graphics equipment, CD-ROMs, and video I/O devices.

A network is a collection of interconnected access points with a software protocol structure that enables communication. The network typically allows many different types of data transfer, using software to implement the networking protocols and to provide flow control, error detection, and error recovery. As we have discussed in this book, networks typically manage transfers between end systems over local, metropolitan, or wide-area distances.

Fibre Channel is designed to combine the best features of both technologies—the simplicity and speed of channel communications with the flexibility and interconnectivity that characterize protocol-based network communications. This fusion of approaches allows system designers to combine traditional peripheral connection, host-to-host internetworking, loosely-coupled processor clustering, and multimedia applications in a single multi-protocol interface. The types of channel-oriented facilities incorporated into the Fibre Channel protocol architecture include

- Data-type qualifiers for routing frame payload into particular interface buffers

- Link-level constructs associated with individual I/O operations
- Protocol interface specifications to allow support of existing I/O channel architectures, such as the Small Computer System Interface (SCSI)

The types of network-oriented facilities incorporated into the Fibre Channel protocol architecture include

- Full multiplexing of traffic between multiple destinations
- Peer-to-peer connectivity between any pair of ports on a Fiber Channel network
- Capabilities for internetworking to other connection technologies

Depending on the needs of the application, either channel or networking approaches can be used for any data transfer. The Fibre Channel Association, which is the industry consortium promoting Fibre Channel, lists the following ambitious requirements that Fibre Channel is intended to satisfy [FCA94]:

- Full duplex links with two fibers per link
- Performance from 100 Mbps to 800 Mbps on a single link (200 Mbps to 1600 Mbps per link)
- Support for distances up to 10 km
- Small connectors
- High-capacity utilization with distance insensitivity
- Greater connectivity than existing multidrop channels
- Broad availability (i.e., standard components)
- Support for multiple cost/performance levels, from small systems to supercomputers
- Ability to carry multiple existing interface command sets for existing channel and network protocols

The solution was to develop a simple generic transport mechanism based on point-to-point links and a switching network. This underlying infrastructure supports a simple encoding and framing scheme that in turn supports a variety of channel and network protocols.

Fibre Channel Elements

The key elements of a Fibre Channel network are the end systems, called *nodes*, and the network itself, which consists of one or more switching elements. The collection of switching elements is referred to as a *fabric*. These elements are interconnected by point-to-point links between ports on the individual nodes and switches. Communication consists of the transmission of frames across the point-to-point links.

Figure 13.15 illustrates these basic elements. Each node includes three or more ports, called N_ports, for interconnection. Similarly, each fabric-switching element includes one or more ports, called F_ports. Interconnection is by means of

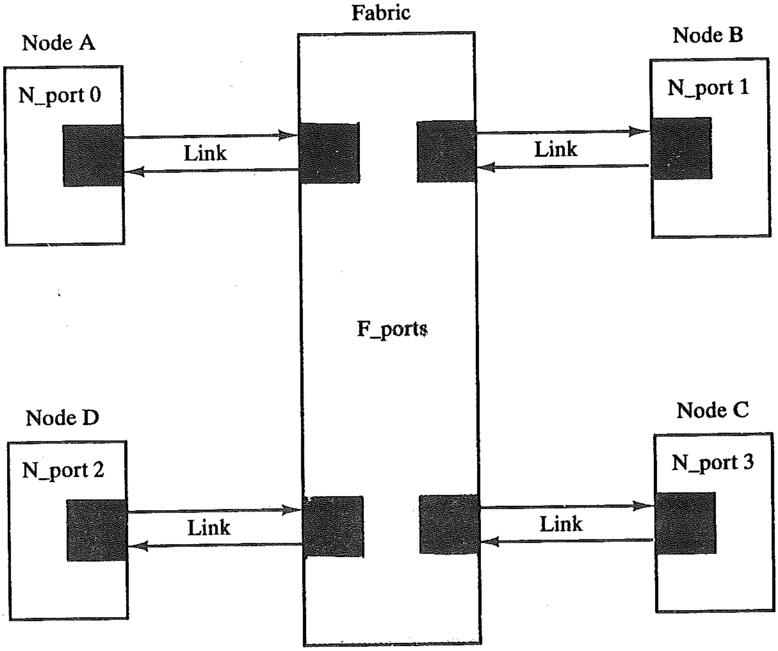


FIGURE 13.15 Fibre channel port types.

bidirectional links between ports. Any node can communicate with any other node connected to the same fabric using the services of the fabric. All routing of frames between N_ports is done by the fabric. Frames may be buffered within the fabric, making it possible for different nodes to connect to the fabric at different data rates.

A fabric can be implemented as a single fabric element, as depicted in Figure 13.15, or as a more general network of fabric elements, as shown in Figure 13.16. In either case, the fabric is responsible for buffering and for routing frames between source and destination nodes.

The Fibre Channel network is quite different from the other LANs that we have examined so far. Fibre Channel is more like a traditional circuit-switched or packet-switched network, in contrast to the typical shared-medium LAN. Thus, Fibre Channel need not be concerned with medium access control issues. Because it is based on a switching network, the Fibre Channel scales easily in terms of N_ports, data rate, and distance covered. This approach provides great flexibility. Fibre Channel can readily accommodate new transmission media and data rates by adding new switches and F_ports to an existing fabric. Thus, an existing investment is not lost with an upgrade to new technologies and equipment. Further, as we shall see, the layered protocol architecture accommodates existing I/O interface and networking protocols, preserving the pre-existing investment.

Fibre Channel Protocol Architecture

The Fibre Channel standard is organized into five levels. These are illustrated in Figure 13.17, with brief definitions in Table 13.6. Each level defines a function or set

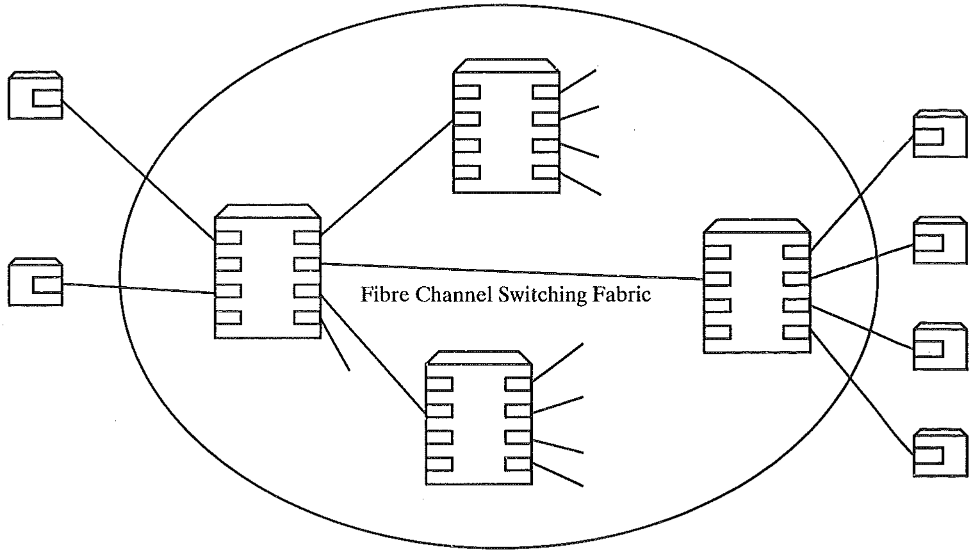
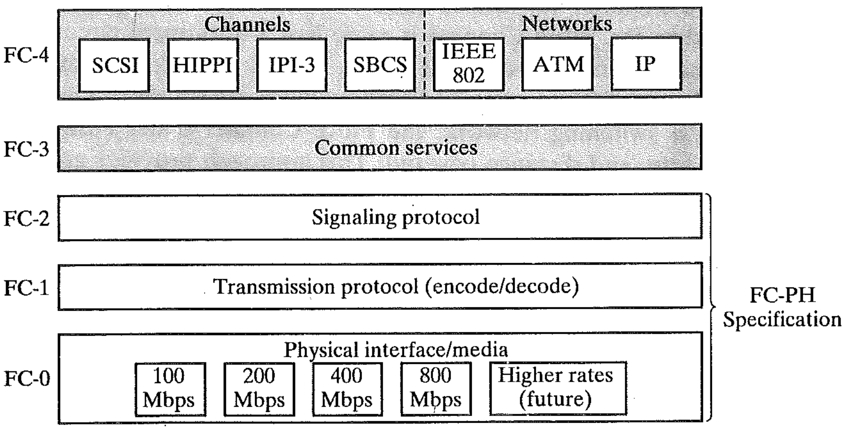


FIGURE 13.16 Fibre channel network.

of related functions. The standard does not dictate a correspondence between levels and actual implementations, with a specific interface between adjacent levels. Rather, the standard refers to the level as a "document artifice" used to group related functions.

Levels FC-0 through FC-2 of the Fibre Channel hierarchy are currently defined in a standard referred to as Fiber Channel Physical and Signaling Interface (FC-PH). Currently, there is no final standard for FC-3. At level FC-4, individual standards have been produced for mapping a variety of channel and network protocols onto lower levels.

We briefly examine each of these levels in turn in the remainder of this section.



FC-4: Channels — SCSI, HIPPI, IPI-3, SBCS; Networks — IEEE 802, ATM, IP

FC-3: Common services

FC-2: Signaling protocol

FC-1: Transmission protocol (encode/decode)

FC-0: Physical interface/media — 100 Mbps, 200 Mbps, 400 Mbps, 800 Mbps, Higher rates (future)

FC-PH Specification

FIGURE 13.17 Fibre channel levels.

TABLE 13.6 Fibre channel levels.

FC-O physical media • Optical cable with laser or LED transmitters for long distance transmissions • Copper coaxial cable for highest speeds over short distances • Shielded twisted pair for lower speeds over short distances
FC-1 byte synchronization and encoding • 8B/10B encoding/decoding scheme provides balance, is simple to implement, and provides useful error-detection capability • Special code character maintains byte and word alignment
FC-2 actual transport mechanism • Framing protocol and flow control between N_ports • Three classes of service between ports
FC-3 common services layer • Port-related services • Services across two or more ports in a node
FC-4 upper layer protocols • Supports a variety of channel and network protocols

Physical Interface and Media

Fibre Channel level FC-0 allows a variety of physical media and data rates; this is one of the strengths of the specification. Currently, data rates ranging from 100 Mbps to 800 Mbps per fiber are defined. The physical media are optical fiber, coaxial cable, and shielded twisted pair. Depending on the data rate and medium involved, maximum distances for individual point-to-point links range from 50 meters to 10 km.

Transmission Protocol

FC-1, the transmission protocol level, defines the signal encoding technique used for transmission and for synchronization across the point-to-point link. The encoding scheme used is 8B/10B, in which each 8 bits of data from level FC-2 is converted into 10 bits for transmission. See Appendix 13A for a description.

Framing Protocol

Level FC-2, referred to as the Framing Protocol level, deals with the transmission of data between N_ports in the form of frames. Among the concepts defined at this level are

- Node and N_port and their identifiers
- Topologies
- Classes of service provided by the fabric
- Segmentation of data into frames and reassembly
- Grouping of frames into logical entities called sequences and exchanges
- Sequencing, flow control, and error control

Common Services

FC-3 provides a set of services that are common across multiple N_Ports of a node. The functions so-far defined in the draft FC-3 documents include

- **Striping.** Makes use of multiple N_Ports in parallel to transmit a single information unit across multiple links simultaneously; this achieves higher aggregate throughput. A likely use is for transferring large data sets in real time, as in video-imaging applications.
- **Hunt Groups.** A hunt group is a set of associated N_Ports at a single node. This set is assigned an alias identifier that allows any frame sent to this alias to be routed to any available N_Port within the set. This may decrease latency by decreasing the chance of waiting for a busy N_Port.
- **Multicast.** Delivers a transmission to multiple destinations. This includes sending to all N_Ports on a fabric (broadcast) or to a subset of the N_Ports on a fabric.

Mapping

FC-4 defines the mapping of various channel and network protocols to FC-PH. I/O channel interfaces include

- **Small Computer System Interface (SCSI).** A widely used high-speed interface typically implemented on personal computers, workstations, and servers.[9] SCSI is used to support high-capacity and high-data-rate devices, such as disks and graphics and video equipment.
- **High-Performance Parallel Interface (HIPPI).** A high-speed channel standard primarily used for mainframe/supercomputer environments. At one time, HIPPI and extensions to HIPPI were viewed as a possible general-purpose high-speed LAN solution, but HIPPI has been superseded by Fibre Channel.

Network interfaces include

- **IEEE 802.** IEEE 802 MAC frames map onto Fibre Channel frames.
- **Asynchronous Transfer Mode**
- **Internet Protocol (IP).** This protocol is described in Chapter 16.

The FC-4 mapping protocols make use of the FC-PH capabilities to transfer upper-layer protocol (ULP) information. Each FC-4 specification defines the formats and procedures for ULP.

Fibre Channel Physical Media and Topologies

One of the major strengths of the Fibre Channel standard is that it provides a range of options for the physical medium, the data rate on that medium, and the topology of the network.

[9] See [STAL96] for a detailed discussion of SCSI.

Transmission Media

Table 13.7 summarizes the options that are available under Fibre Channel for physical transmission medium and data rate. Each entry specifies the maximum point-to-point link distance (between ports) that is defined for a given transmission medium at a given data rate. These media may be mixed in an overall configuration. For example, a single-mode optical link could be used to connect switches in different buildings, with multimode optical links used for vertical distribution inside, and shielded twisted pair or coaxial cable links to individual workstations.

Topologies

The most general topology supported by Fibre Channel is referred to as a fabric or switched topology. This is an arbitrary topology that includes at least one switch to interconnect a number of N_ports, as shown in Figure 13.18a. The fabric topology may also consist of a number of switches forming a switched network, with some or all of these switches also supporting end nodes (Figure 13.16).

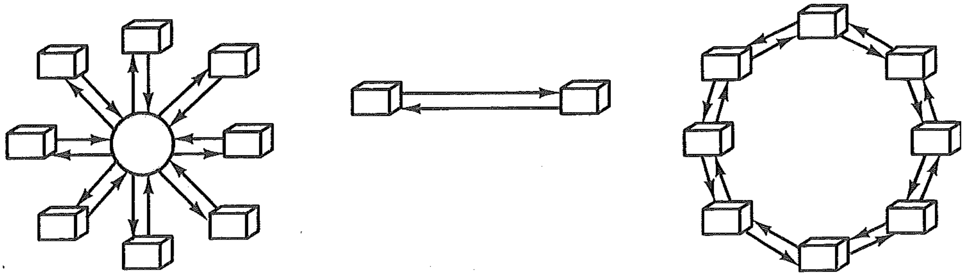
Routing in the fabric topology is transparent to the nodes. Each port in the configuration has a unique address. When data from a node are transmitted into the fabric, the edge switch to which the node is attached uses the destination port address in the incoming data frame to determine the destination port location. The switch then either delivers the frame to another node attached to the same switch or transfers the frame to an adjacent switch to begin the routing of the frame to a remote destination.

The fabric topology provides scalability of capacity: As additional ports are added, the aggregate capacity of the network increases, thus minimizing congestion and contention, and increasing throughput. The fabric is protocol-independent and largely distance-insensitive. The technology of the switch itself and of the transmission links connecting the switch to nodes may be changed without affecting the overall configuration. Another advantage of the fabric topology is that the burden on nodes is minimized. An individual Fibre Channel node (end systems) is only responsible for managing a simple point-to-point connection between itself and the fabric; the fabric is responsible for routing between N_ports and error detection.

In addition to the fabric topology, the Fibre Channel standard defines two other topologies. With the point-to-point topology (Figure 13.18b) there are only

TABLE 13.7 Maximum distance for Fibre Channel media types.

	800 Mbps	400 Mbps	200 Mbps	100 Mbps
Single mode fiber	10 km	10 km	10 km	—
50-μm multimode fiber	0.5 km	1 km	2 km	10 km
62.5-μm multimode fiber	175 m	350 m	1500 m	1500 m
Video coaxial cable	25 m	50 m	75 m	100 m
Miniature coaxial cable	10 m	15 m	25 m	35 m
Shielded twisted pair	—	—	50 m	100 m



(a) Fabric (switched)

(b) Point to point

(c) Arbitrated loop

FIGURE 13.18 Basic Fibre Channel topologies.

two N_ports, and these are directly connected, with no intervening fabric switches. In this case, there is no routing.

Finally, the arbitrated loop topology (Figure 13.18c) is a simple, low-cost topology for connecting up to 126 nodes in a loop. The ports on an arbitrated loop must contain the functions of both N_ports and F_ports; these are called NL_ports. The arbitrated loop operates in a manner roughly equivalent to the token ring protocols that we have seen. Each port sees all frames and passes and ignores those not addressed to itself. There is a token acquisition protocol to control access to the loop.

The fabric and arbitrated loop topologies may be combined in one configuration to optimize the cost of the configuration. In this case, one of the nodes on the arbitrated loop must be a fabric-loop (FL_port) node so that it participates in routing with the other switches in the fabric configuration.

The type of topology need not be configured manually by a network manager. Rather, the type of topology is discovered early in the link initialization process.

13.6 WIRELESS LANS

A set of wireless LAN standards has been developed by the IEEE 802.11 committee. The terminology and some of the specific features of 802.11 are unique to this standard and are not reflected in all commercial products. However, it is useful to be familiar with the standard as its features are representative of required wireless LAN capabilities.

Figure 13.19 indicates the model developed by the 802.11 working group. The smallest building block of a wireless LAN is a basic service set (BSS), which consists of some number of stations executing the same MAC protocol and competing for access to the same shared medium. A basic service set may be isolated, or it may connect to a backbone distribution system through an access point. The access point functions as a bridge. The MAC protocol may be fully distributed or controlled by a central coordination function housed in the access point. The basic service set generally corresponds to what is referred to as a cell in the literature.

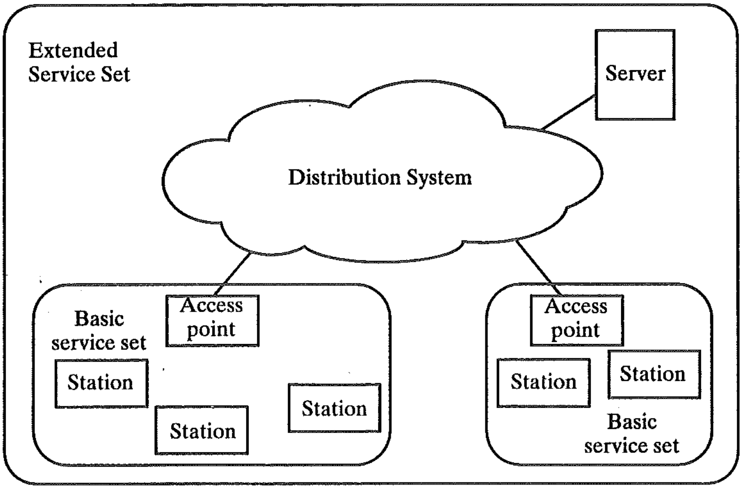


FIGURE 13.19 IEEE 802.11 architecture.

An extended service set (ESS) consists of two or more basic service sets interconnected by a distribution system. Typically, the distribution system is a wired backbone LAN. The extended service set appears as a single logical LAN to the logical link control (LLC) level.

The standard defines three types of stations, based on mobility:

- **No-transition.** A station of this type is either stationary or moves only within the direct communication range of the communicating stations of a single BSS.
- **BSS-transition.** This is defined as a station movement from one BSS to another BSS within the same ESS. In this case, delivery of data to the station requires that the addressing capability be able to recognize the new location of the station.
- **ESS-transition.** This is defined as a station movement from a BSS in one ESS to a BSS within another ESS. This case is supported only in the sense that the station can move. Maintenance of upper-layer connections supported by 802.11 cannot be guaranteed. In fact, disruption of service is likely to occur.

Physical Medium Specification

Three physical media are defined in the current 802.11 standard:

- Infrared at 1 Mbps and 2 Mbps operating at a wavelength between 850 and 950 nm.
- Direct-sequence spread spectrum operating in the 2.4-GHz ISM band. Up to 7 channels, each with a data rate of 1 Mbps or 2 Mbps, can be used.
- Frequency-hopping spread spectrum operating in the 2.4-GHz ISM band. The details of this option are for further study.

Medium Access Control

The 802.11 working group considered two types of proposals for a MAC algorithm: distributed-access protocols which, like CSMA/CD, distributed the decision to transmit over all the nodes using a carrier-sense mechanism; and centralized access protocols, which involve regulation of transmission by a centralized decision maker. A distributed access protocol makes sense of an ad hoc network of peer workstations and may also be attractive in other wireless LAN configurations that consist primarily of bursty traffic. A centralized access protocol is natural for configurations in which a number of wireless stations are interconnected with each other and with some sort of base station that attaches to a backbone wired LAN; it is especially useful if some of the data is time-sensitive or high priority.

The end result of the 802.11 is a MAC algorithm called DFWMAC (distributed foundation wireless MAC) that provides a distributed access-control mechanism with an optional centralized control built on top of that. Figure 13.20 illustrates the architecture. The lower sublayer of the MAC layer is the distributed coordination function (DCF). DCF uses a contention algorithm to provide access to all traffic. Ordinary asynchronous traffic directly uses DCF. The point coordination function (PCF) is a centralized MAC algorithm used to provide contention-free service. PCF is built on top of DCF and exploits features of DCF to assure access for its users. Let us consider these two sublayers in turn.

Distributed Coordination Function

The DCF sublayer makes use of a simple CSMA algorithm. If a station has a MAC frame to transmit, it listens to the medium. If the medium is idle, the station may transmit; otherwise, the station must wait until the current transmission is complete

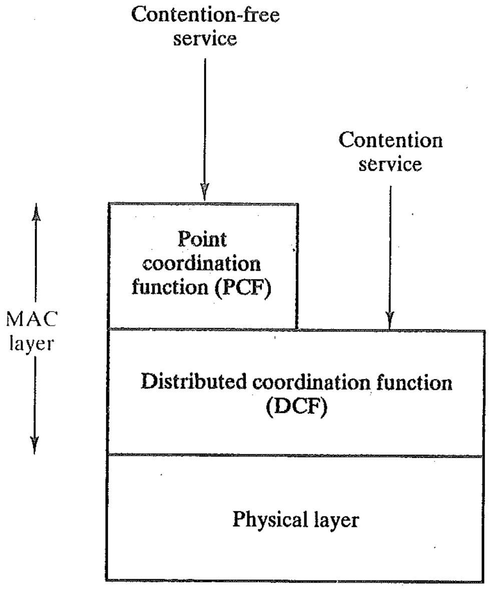

Contention-free service
Contention service
Point coordination function (PCF)
MAC layer
Distributed coordination function (DCF)
Physical layer

FIGURE 13.20 IEEE 802.11 protocol architecture.

before transmitting. The DCF does not include a collision-detection function (i.e., CSMA/CD) because collision detection is not practical on a wireless network. The dynamic range of the signals on the medium is very large, so that a transmitting station cannot effectively distinguish incoming weak signals from noise and the effects of its own transmission.

To ensure the smooth and fair functioning of this algorithm, DCF includes a set of delays that amounts to a priority scheme. Let us start by considering a single delay known as an interframe space (IFS). In fact, there are three different IFS values, but the algorithm is best explained by initially ignoring this detail. Using an IFS, the rules for CSMA access are as follows:

1. A station with a frame to transmit senses the medium. If the medium is idle, the station waits to see if the medium remains idle for a time equal to IFS, and, if this is so, the station may immediately transmit.
2. If the medium is busy (either because the station initially finds the medium busy or because the medium becomes busy during the IFS idle time), the station defers transmission and continues to monitor the medium until the current transmission is over.
3. Once the current transmission is over, the station delays another IFS. If the medium remains idle for this period, then the station backs off using a binary exponential backoff scheme and again senses the medium. If the medium is still idle, the station may transmit.

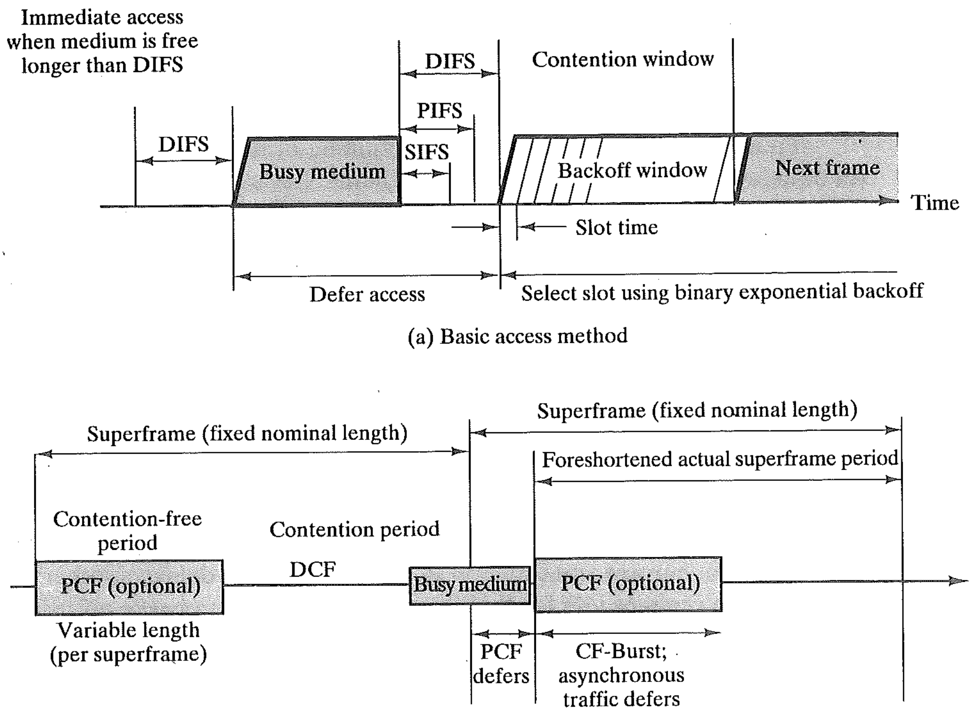
As with Ethernet, the binary exponential backoff provides a means of handling a heavy load. If a station attempts to transmit and finds the medium busy, it backs off a certain amount and tries again. Repeated failed attempts to transmit result in longer and longer backoff times.

The above scheme is refined for DCF to provide priority-based access by the simple expedient of using three values for IFS:

- **SIFS (short IFS).** The shortest IFS, used for all immediate response actions, as explained below.
- **PIFS (point coordination function IFS).** A mid-length IFS, used by the centralized controller in the PCF scheme when issuing polls.
- **DIFS (distributed coordination function IFS).** The longest IFS, used as a minimum delay for asynchronous frames contending for access.

Figure 13.21a illustrates the use of these time values. Consider first the SIFS. Any station using SIFS to determine transmission opportunity has, in effect, the highest priority, because it will always gain access in preference to a station waiting an amount of time equal to PIFS or DIFS. The SIFS is used in the following circumstances:

- **Acknowledgment (ACK).** When a station receives a frame addressed only to itself (not multicast or broadcast) it responds with an ACK frame after waiting only for an SIFS gap; this has two desirable effects. First, because collision detection is not used, the likelihood of collisions is greater than with


Immediate access when medium is free longer than DIFS
DIFS
Busy medium
DIFS
PIFS
SIFS
Contention window
Backoff window
Next frame
Time
Slot time
Defer access
Select slot using binary exponential backoff
Superframe (fixed nominal length)
Superframe (fixed nominal length)
Foreshortened actual superframe period
Contention-free period
Contention period
DCF
PCF (optional)
Busy medium
PCF (optional)
Variable length (per superframe)
PCF defers
CF-Burst; asynchronous traffic defers

(a) Basic access method

(b) PCF superframe construction

FIGURE 13.21 IEEE 802.11 MAC timing.

CSMA/CD, and the MAC-level ACK provides for efficient collision recovery. Second, the SIFS can be used to provide efficient delivery of an LLC protocol data unit (PDU) that requires multiple MAC frames. In this case, the following scenario occurs. A station with a multiframe LLC PDU to transmit sends out the MAC frames one at a time. Each frame is acknowledged by the recipient after SIFS. When the source receives an ACK, it immediately (after SIFS) sends the next frame in the sequence. The result is that once a station has contended for the channel, it will maintain control of the channel until it has sent all of the fragments of an LLC PDU.

- **Clear to Send (CTS).** A station can ensure that its data frame will get through by first issuing a small Request to Send (RTS) frame. The station to which this frame is addressed should immediately respond with a CTS frame if it is ready to receive. All other stations receive the RTS and defer using the medium until they see a corresponding CTS, or until a timeout occurs.
- **Poll response.** This is explained in the discussion of PCF, below.

The next longest IFS interval is the PIFS; this is used by the centralized controller in issuing polls and takes precedence over normal-contention traffic. However, those frames transmitted using SIFS have precedence over a PCF poll.

Finally, the DIFS interval is used for all ordinary asynchronous traffic.

Point Coordination Function

PCF is an alternative access method implemented on top of the DCF. The operation consists of polling with the centralized polling master (point coordinator). The point coordinator makes use of PIFS when issuing polls. Because PIFS is smaller than DIFS, the point coordinator can seize the medium and lock out all asynchronous traffic while it issues polls and receives responses.

As an extreme, consider the following possible scenario. A wireless network is configured so that a number of stations with time-sensitive traffic are controlled by the point coordinator while remaining traffic, using CSMA, contends for access. The point coordinator could issue polls in a round-robin fashion to all stations configured for polling. When a poll is issued, the polled station may respond using SIFS. If the point coordinator receives a response, it issues another poll using PIFS. If no response is received during the expected turnaround time, the coordinator issues a poll.

If the discipline of the preceding paragraph were implemented, the point coordinator would lock out all asynchronous traffic by repeatedly issuing polls. To prevent this situation, an interval known as the superframe is defined. During the first part of this interval, the point coordinator issues polls in a round-robin fashion to all stations configured for polling. The point coordinator then idles for the remainder of the superframe, allowing a contention period for asynchronous access.

Figure 13.21b illustrates the use of the superframe. At the beginning of a superframe, the point coordinator may optionally seize control and issue polls for a give period of time. This interval varies because of the variable frame size issued by responding stations. The remainder of the superframe is available for contention-based access. At the end of the superframe interval, the point coordinator contends for access to the medium using PIFS. If the medium is idle, the point coordinator gains immediate access, and a full superframe period follows. However, the medium may be busy at the end of a superframe. In this case, the point coordinator must wait until the medium is idle to gain access; this results in a foreshortened superframe period for the next cycle.

13.7 RECOMMENDED READING

[STAL97] covers, in greater detail, all of the LAN systems discussed in this chapter

[HEGE93] and [BIRD94] cover CSMA/CD and token ring LANs, respectively, in some depth. Two detailed accounts of FDDI are [MILL95] and [SHAH94]; the former provides more detail on physical-layer issues, while the latter has more coverage of the MAC protocol. [SPUR95] contains a concise summary of the specifications for 100BASE-T, including configuration guidelines for a single segment of each media type, as well as guidelines for building multi-segment Ethernets using a variety of media types. [WATS95] provides good technical summaries of 100VG-AnyLAN. [KAVA95] and [NEWM94] are good survey articles on LAN ATM architecture and configurations. The most comprehensive description of Fiber Channel is [STEP95]. This book provides a detailed technical treatment of each layer of the Fibre Channel architecture. A shorter but worthwhile treatment is [FCA94], which is a 50-page book from the Fiber Channel Association, an industry consortium formed to promote the Fiber Channel.

Books with rigorous treatments of LAN/MAN performance include [STUC85], [HAMM86], [SPRA91], and [BERT92].

BERT92 Bertsekas, D. and Gallager, R. *Data Networks.* Englewood Cliffs, NJ: Prentice Hall, 1992.

BIRD94 Bird, D. *Token Ring Network Design.* Reading, MA: Addison-Wesley, 1994.

FCA94 Fibre Channel Association. *Fibre Channel: Connection to the Future.* Austin, TX: Fibre Channel Association, 1994.

HAMM86 Hammond, J. and O'Reilly, P. *Performance Analysis of Local Computer Networks.* Reading, MA: Addison-Wesley, 1986.

HEGE93 Hegering, H. and Lapple, A. *Ethernet: Building a Communications Infrastructure.* Reading, MA: Addison-Wesley, 1993.

KAVA95 Kavak, N. "Data Communication in ATM Networks." *IEEE Network*, May/June 1995.

MILL95 Mills, A. *Understanding FDDI.* Englewood Cliffs, NJ: Prentice-Hall, 1995.

NEWM94 Newman, P. "ATM Local Area Networks." *IEEE Communications Magazine*, March 1994.

SHAH94 Shah, A. and Ramakrishnan, G. FDDI: *A High-Speed Network.* Englewood Cliffs, NJ: Prentice Hall, 1994.

SPRA91 Apragins, J. Hammond, J. and Pawlikowski, K. *Telecommunications Protocols and Design.* Reading, MA: Addison-Wesley, 1991.

SPUR95 Spurgeon, C. *Quick Reference Guide to Ethernet.* Austin, TX: Harris Park Press, 1995.

STAL97 Stallings, W. *Local and Metropolitan Area Networks*, Fifth Edition. Upper Saddle River, NJ: 1997.

STEP95 Stephens, G. and Dedek, J. *Fiber Channel.* Menlo Park, CA: Ancot Corporation, 1995.

STUC85 Stuck, B. and Arthurs, E. A *Computer Communications Network Performance Analysis Primer.* Englewood Cliffs, NJ: Prentice Hall, 1985.

WATS95 Watson, G., et al. "The Demand Priority MAC Protocol." *IEEE Network*, January/February 1995.

Recommended Web Sites

- http://wwwhost.ots.utexas.edu/ethernet: Provides general Ethernet information, technical specifications, an Ethernet reading list, and an image of inventor Robert Metcalf's original 1976 Ethernet drawing.
- http://www.astral.org: Site of the Alliance for Strategic Token Ring Advancement and Leadership, a vendor organization.
- http://www.amdahl.com/ext/CARP/FCA/FCA.html: Web site of the Fibre Channel Association.
- http://www.iol.unh.edu: University of New Hampshire (equipment testing for ATM, FDDI, Fast Ethernet, FDSE, Ethernet, OSPF, Network Management (SNMP), Token Ring, and VG-AnyLAN).

13.8 PROBLEMS

13.1 A disadvantage of the contention approach for LANs is the capacity wasted due to multiple stations attempting to access the channel at the same time. Suppose that time is divided into discrete slots with each of N stations attempting to transmit with probability p during each slot. What fraction of slots is wasted due to multiple simultaneous transmission attempts?

13.2 A simple medium access control protocol would be to use a fixed assignment time-division multiplexing (TDM) scheme, as described in section 2.1. Each station is assigned one time slot per cycle for transmission. For the bus and tree, the length of each slot is the time to transmit 100 bits plus the end-to-end propagation delay. For the ring, assume a delay of one bit time per station, and assume that a round-robin assignment is used. Stations monitor all time slots for reception. Assume a propagation time of 2×10^8 m/sec. What are the limitations, in terms of number of stations and throughput per station, for

a. A 1-km, 10-Mbps baseband bus?

b. A 1-km (headend to farthest point), 10-Mbps broadband bus?

c. A 10-Mbps broadband tree consisting of a 0.5-km trunk emanating from the headend and five 0.1-km branches from the trunk at the following points: 0.05 km, 0.15 km, 0.25 km, 0.35 km, 0.45 km?

d. A 10-Mbps ring with a total length of 1 km?

e. A 10-Mbps ring with a length of 0.1 km between repeaters?

f. Compute throughput for all of the above for 10 and 100 stations.

13.3 Consider two stations on a baseband bus at a distance of 1 km from each other. Let the data rate be 1 Mbps, the frame length be 100 bits, and the propagation velocity be 2×10^8 m/s. Assume that each station generates frames at an average rate of 1000 frames per second. For the ALOHA protocol, if one station begins to transmit a frame at time t, what is the probability of collision? Repeat for slotted ALOHA. Repeat for ALOHA and slotted-ALOHA at 10 Mbps.

13.4 Repeat Problem 13.3 for a broadband bus. Assume that the two stations are 1 km apart and that one is very near the headend.

13.5 The binary exponential backoff algorithm is defined by IEEE 802 as follows:

> The delay is an integral multiple of slot time. The number of slot times to delay before the nth retransmission attempt is chosen as a uniformly distributed random integer r in the range $0 < r < 2^K$, where $K = \min(n, 10)$.

Slot time is, roughly, twice the round-trip propagation delay. Assume that two stations always have a frame to send. After a collision, what is the mean number of retransmission attempts before one station successfully retransmits? What is the answer if three stations always have frames to send?

13.6 Consider the 100VG-AnyLAN network shown in Figure 13.12 and suppose that requests are issued in the following order:

1-2(N), 3-8(N), 1-1(H), 1-2(N), 5-1(N), 3-6(H), 1-4(N), 5-2(N), 1-1(H)

Generate a timing diagram similar to that of Figure 13.11 that shows the transmission sequence of the above requests. Assume all frames are of equal length and that requests arrive at a uniform rate of one per 0.5 frame times.

13.7 For a token ring LAN, suppose that the destination station removes the data frame and immediately sends a short acknowledgment frame to the sender, rather than letting the original frame return to sender. How will this procedure affect performance?

13.8 Another medium access control technique for rings is the slotted ring. A number of fixed-length slots circulate continuously on the ring. Each slot contains a leading bit to designate the slot as empty or full. A station wishing to transmit waits until an empty slot arrives, marks the slot full, and inserts a frame of data as the slot goes by. The full slot makes a complete round trip, to be marked empty again by the station that marked it full. In what sense are the slotted ring and token ring protocols the complement of each other?

13.9 Consider a slotted ring of length 10 km with a data rate of 10 Mbps and 500 repeaters, each of which introduces a 1-bit delay. Each slot contains room for one source-address byte, one destination-address byte, two data bytes, and five control bits for a total length of 37 bits. How many slots are on the ring?

13.10 Compare the capacity allocation schemes for IEEE 802.5 token ring and FDDI. What are the relative pros and cons?

13.11 Rework the example of Figure 13.10 using a TTRT of 12 frames and assume that no station ever has more than 8 asynchronous frames to send.

13.12 With 8B6T coding, the effective data rate on a single channel is 33 Mbps with a signaling rate of 25 Mbaud. If a pure ternary scheme were used, what would be the effective data rate for a signaling rate of 25 Mbaud?

13.13 With 8B6T coding, the DC algorithm sometimes negates all of the ternary symbols in a code group. How does the receiver recognize this condition? How does the receiver discriminate between a negated code group and one that has not been negated? For example, the code group for data byte 00 is +–00+– and the code group for data byte 38 is the negation of that, namely –+00–+.

13.14 Draw the MLT decoder state diagram that corresponds to the encoder state diagram of Figure 13.2.

13.15 For the bit stream 0101110, sketch the waveforms for NRZ, NRZI, Manchester, and Differential Manchester, and MLT-3.

13.16 Fill in all the values for the 5B6B decoding table, whose outline is shown below.

Received Sextet	Mode 2 Output Quintet	Mode 4 Output Quintet
000000		
• • •		
111111		

13A APPENDIX

Digital Signal Encoding for LANs

In Chapter 4, we looked at some of the common techniques for encoding digital data for transmission, including Manchester and Differential Manchester, which are used in some of the LAN standards. In this appendix, we examine some additional encoding schemes referred to in this chapter.

4B/5B-NRZI

This scheme, which is actually a combination of two encoding algorithms, is used both for 100BASE-X and FDDI. To understand the significance of this choice, first consider the simple alternative of an NRZ (non-return to zero) coding scheme. With NRZ, one signal state represents binary one, and one signal state represents binary zero. The disadvantage of this approach is its lack of synchronization. Because transitions on the medium are unpredictable, there is no way for the receiver to synchronize its clock to the transmitter. A solution to this problem is to encode the binary data to guarantee the presence of transitions. For example, the data could first be encoded using Manchester encoding. The disadvantage of this approach is that the efficiency is only 50%. That is, because there can be as many as two transitions per bit time, a signaling rate of 200 million signal elements per second (200 Mbaud) is needed to achieve a data rate of 100 Mbps; this represents an unnecessary cost and technical burden.

Greater efficiency can be achieved using the 4B/5B code. In this scheme, encoding is done four bits at a time; each four bits of data are encoded into a symbol with five *code bits*, such that each code bit contains a single signal element. The block of five code bits is called a code group. In effect, each set of 4 bits is encoded as 5 bits. The efficiency is thus raised to 80%; 100 Mbps is achieved with 125 Mbaud.

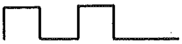
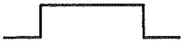
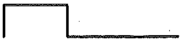
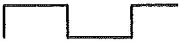
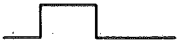
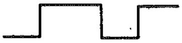
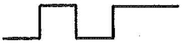
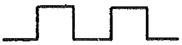
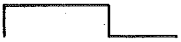
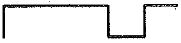
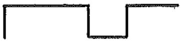
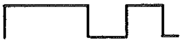
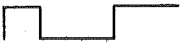
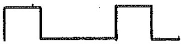
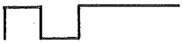
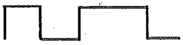
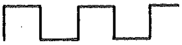
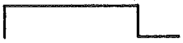
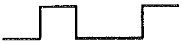
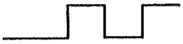
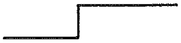
To ensure synchronization, there is a second stage of encoding: each code bit of the 4B/5B stream is treated as a binary value and encoded using Nonreturn to Zero Inverted (NRZI) (see Figure 4.2). In this code, a binary 1 is represented with a transition at the beginning of the bit interval, and a binary 0 is represented with no transition at the beginning of the bit interval; there are no other transitions. The advantage of NRZI is that it employs differential encoding. Recall from Chapter 4 that in differential encoding, the signal is decoded by comparing the polarity of adjacent signal elements rather than the absolute value of a signal element. A benefit of this scheme is that it is generally more reliable in detecting a transition in the presence of noise and distortion than in comparing a value to a threshold.

Now we are in a position to describe the 4B/5B code and to understand the selections that were made. Table 13.8 shows the symbol encoding. Each 5-bit code group pattern is shown, together with its NRZI realization. Because we are encoding four bits with a 5-bit pattern, only 16 of the 32 possible patterns are needed for data encoding. The codes selected to represent the 16 4-bit data blocks are such that a transition is present at least twice for each 5-code group code. No more than three zeros in a row are allowed across one or more code groups.

The encoding scheme can be summarized as follows:

1. A simple NRZ encoding is rejected because it does not provide synchronization; a string of 1s or 0s will have no transitions.
2. The data to be transmitted must first be encoded to assure transitions. The 4B/5B code is chosen over Manchester because it is more efficient.
3. The 4B/5B code is further encoded using NRZI so that the resulting differential signal will improve reception reliability.
4. The specific 5-bit patterns for the encoding of the 16 4-bit data patterns are chosen to guarantee no more than three zeros in a row, to provide for adequate synchronization.

TABLE 13.8 4B/5B code groups.

Data input (4 bits)	Code group (5 bits)	NRZI pattern	Interpretation
0000	11110		Data 0
0001	01001		Data 1
0010	10100		Data 2
0011	10101		Data 3
0100	01010		Data 4
0101	01011		Data 5
0110	01110		Data 6
0111	01111		Data 7
1000	10010		Data 8
1001	10011		Data 9
1010	10110		Data A
1011	10111		Data B
1100	11010		Data C
1101	11011		Data D
1110	11100		Data E
1111	11101		Data F
	11111		Idle
	11000		Start of stream delimiter, part 1
	10001		Start of stream delimiter, part 2
	01101		End of stream delimiter, part 1
	00111		End of stream delimiter, part 2
	00100		Transmit error
	other		Invalid codes

Those code groups not used to represent data are either declared invalid or are assigned special meaning as control symbols. These assignments are listed in Table 13.8. The nondata symbols fall into the following categories:

- **Idle.** The idle code group is transmitted between data transmission sequences. It consists of a constant flow of binary ones, which in NRZI comes out as a continuous alternation between the two signal levels. This continuous fill pattern establishes and maintains synchronization and is used in the CSMA/CD protocol to indicate that the shared medium is idle.
- **Start-of-stream delimiter.** Used to delineate the starting boundary of a data transmission sequence; it consists of two different code groups.
- **End-of-stream delimiter.** Used to terminate normal data transmission sequences; it consists of two different code groups.
- **Transmit error.** This code group is interpreted as a signaling error. The normal use of this indicator is for repeaters to propagate received errors.

MLT-3

Although 4B/5B-NRZI is effective over optical fiber, it is not suitable, as is, for use over twisted pair. The reason is that the signal energy is concentrated in such a way as to produce undesirable radiated emissions from the wire. MLT-3, which is used on both 100BASE-TX and the twisted pair version of FDDI, is designed to overcome this problem.

The following steps are involved:

1. *NRZI to NRZ conversion*: The 4B/5B NRZI signal of the basic 100BASE-X is converted back to NRZ.
2. *Scrambling*: The bit stream is scrambled to produce a more uniform spectrum distribution for the next stage.
3. *Encoder*: The scrambled bit stream is encoded using a scheme known as MLT-3.
4. *Driver*: The resulting encoding is transmitted.

The effect of the MLT-3 scheme is to concentrate most of the energy in the transmitted signal below 30 MHz, which reduces radiated emissions; this, in turn, reduces problems due to interference.

The MLT-3 encoding produces an output that has a transition for every binary one that uses three levels: a positive voltage (+V), a negative voltage (–V) and no voltage (0). The encoding rules are best explained with reference to the encoder state diagram shown in Figure 13.22:

1. If the next input bit is zero, then the next output value is the same as the preceding value.
2. If the next input bit is one, then the next output value involves a transition:
   a. If the preceding output value was either +V or –V, then the next output value is 0.
   b. If the preceding output value was 0, then the next output value is nonzero, and that output is of the opposite sign to the last nonzero output.

Figure 13.23 provides an example. Every time there is an input of 1, there is a transition. The occurrences of +V and –V alternate.

8B6T

The 8B6T encoding algorithm uses ternary signaling. With this method, each signal element can take on one of three values (positive voltage, negative voltage, zero voltage). A pure ternary code is one in which the full information-carrying capacity of the ternary signal is exploited. However, pure ternary is not attractive for the same reasons that a pure binary

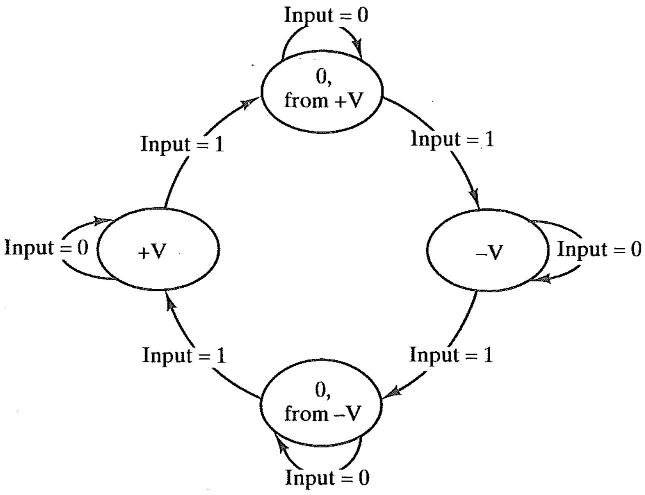


FIGURE 13.22 MLT-3 encoder state diagram.

(NRZ) code is rejected: lack of synchronization. However, there are schemes referred to as **block-coding methods** that approach the efficiency of ternary and overcome this disadvantage. A new block coding scheme known as 8B6T is used for 100BASE-T4.

With 8B6T, the data to be transmitted are handled in 8-bit blocks. Each block of 8 bits is mapped into a code group of six ternary symbols. The stream of code groups is then transmitted in round-robin fashion across the three output channels (Figure 13.24). Thus, the ternary transmission rate on each output channel is

$$\frac{6}{8} \times 33\frac{1}{3} = 25\text{Mbaud}$$

Table 13.9 shows a portion of the 8B6T code table; the full table maps all possible 8-bit patterns into a unique code group of six ternary symbols. The mapping was chosen with two requirements in mind: synchronization and DC balance. For synchronization, the codes were chosen so as to maximize the average number of transitions per code group. The second requirement is to maintain DC balance, so that the average voltage on the line is zero; for this purpose, all of the selected code groups have either an equal number of positive and negative symbols or an excess of one positive symbol. To maintain balance, a DC balancing algorithm is used. In essence, this algorithm monitors the cumulative weight of all code groups transmitted on a single pair. Each code group has a weight of 0 or 1. To maintain balance, the algorithm may negate a transmitted code group (change all + symbols to − symbols

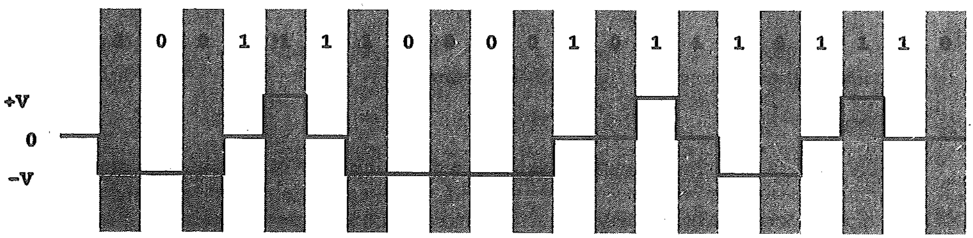


FIGURE 13.23 Example of MLT-3 encoding.

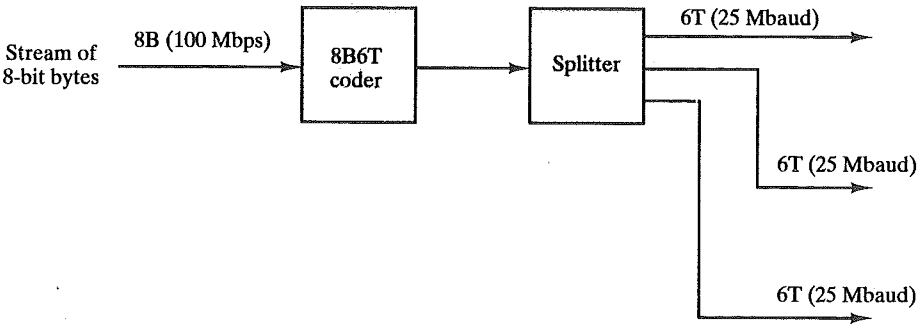


FIGURE 13.24 8B6T transmission scheme.

and all − symbols to + symbols), so that the cumulative weight at the conclusion of each code group is always either 0 or 1.

5B6B

The 5B6B encoding is used in the 100VG-AnyLAN specification in the following way: A MAC frame is divided into 5-bit chunks (quintets), and each successive chunk is transmitted over a different channel in round-robin fashion. Thus, to achieve a data rate of 100 Mbps, a data rate of only 25 Mbps is needed on each channel.

To ensure adequate transitions on each line for synchronization, an encoding scheme known as 5B6B is used. The 5B6B scheme is based on the same strategy as the 4B/5B scheme described earlier. In this case, each group of 5 input bits is mapped into a set of 6 output bits. Thus, for an effective data rate of 25 Mbps, a signaling rate of 30 Mbaud is required.

With the 5B6B scheme, there are 32 possible 5-bit inputs. Ideally, we would like to assign to each 5-bit input a 6-bit code that has an equal number of ones and zeros; this would maintain a dc balance of zero. However, there are only twenty 6-bit code words that have three ones and zeros. These codes are assigned to 20 of the input patterns. For the remaining 12 input patterns, two code words are assigned, one with four zeros and two ones (mode

TABLE 13.9 Portion of 8B6T code table.

Data octet	6T code group	Data octet	6T code group	Data octet	6T code group	Data octet	6T code group
00	+−00+−	10	+0+−−0	20	00−++−	30	+−00−+
01	0+−+−0	11	++0−0−	21	−−+00+	31	0+−−+0
02	+−0+−0	12	+0+−0−	22	++−0+−	32	+−0−+0
03	−0++−0	13	0++−0−	23	++−0−+	33	−0+−+0
04	−0+0+−	14	0++−−0	24	00+0−+	34	−0+0−+
05	0+−−0+	15	++00−−	25	00+0+−	35	0+−+0−
06	+−0−0+	16	+0+0−−	26	00−00+	36	+−0+0−
07	−0+−0+	17	0++0−−	27	−−+++−	37	−0++0−
08	−+00+−	18	0+−0+−	28	−0−++0	38	−+00−+
09	0−++−0	19	0+−0−+	29	−−0+0+	39	0−+−+0
0A	−+0+−0	1A	0+−++−	2A	−0−+0+	3A	−+0−+0
0B	+0−+−0	1B	0+−00+	2B	0−−+0+	3B	+0−−+0
0C	+0−0+−	1C	0−+00+	2C	0−−++0	3C	+0−0−+
0D	0−+−0+	1D	0−+++−	2D	−−00++	3D	0−++0−
0E	−+0−0+	1E	0−+0−+	2E	−0−0++	3E	−+0+0−
0F	+0−−0+	1F	0−+0+−	2F	0−−0++	3F	+0−+0−

2) and one with two zeros and four ones (mode 4). Successive instances of any of these 24 unbalanced code words must alternate between mode 2 and mode 4 output to maintain balance. If, during reception, a station or repeater receives two of the same type of unbalanced words in a row (with any number of intervening balanced words), the receiver knows that a transmission error has occurred and will ask for a retransmission of the data.

Table 13.10 shows the complete 5B6B encoding scheme. There is a unique output code word for 12 of the input patterns; for the rest, the transmitter keeps track of whether the last unbalanced transmitted word was mode 2 or mode 4 and transmits the appropriate output code word to maintain balance.

8B/10B

The encoding scheme used for Fibre Channel is 8B/10B, in which each 8 bits of data is converted into 10 bits for transmission. This scheme has a similar philosophy to the 4B/5B scheme used for FDDI, as discussed earlier. The 8B/10B scheme was developed and patented by IBM for use in its 200-megabaud ESCON interconnect system [WIDM83]. The 8B/10B scheme is more powerful than 4B/5B in terms of transmission characteristics and error detection capability.

The developers of this code list the following advantages:

- It can be implemented with relatively simple and reliable transceivers at low cost.
- It is well-balanced, with minimal deviation from the occurrence of an equal number of 1 and 0 bits across any sequence.
- It provides good transition density for easier clock recovery.
- It provides useful error-detection capability.

The 8B/10B code is an example of the more general *m*B*n*B code, in which *m* binary source bits are mapped into *n* binary bits for transmission. Redundancy is built into the code to provide the desired transmission features by making $n > m$.

Figure 13.25 illustrates the operation of this code. The code actually combines two other codes, a 5B/6B code and a 3B/4B code. The use of these two codes is simply an artifact that simplifies the definition of the mapping and the implementation; the mapping could have been defined directly as an 8B/10B code. In any case, a mapping is defined that maps

TABLE 13.10 5B6B encoding table.

Input Quintet	Mode 2 Output	Mode 4 Output	Input Quintet	Mode 2 Output	Mode 4 Output
00000	001100	110011	10000	000101	111010
00001	101100	101100	10001	100101	100101
00010	100010	101110	10010	001001	110110
00011	001101	001101	10011	010110	010110
00100	001010	110101	10100	111000	111000
00101	010101	010101	10101	011000	100111
00110	001110	001110	10110	011001	011001
00111	001011	001011	10111	100001	011110
01000	000111	000111	11000	110001	110001
01001	100011	100011	11001	101010	101010
01010	100110	100110	11010	010100	101011
01011	000110	111001	11011	110100	110100
01100	101000	010111	11100	011100	011100
01101	011010	011010	11101	010011	010011
01110	100100	100100	11110	010010	101101
01111	101001	101001	11111	110010	110010

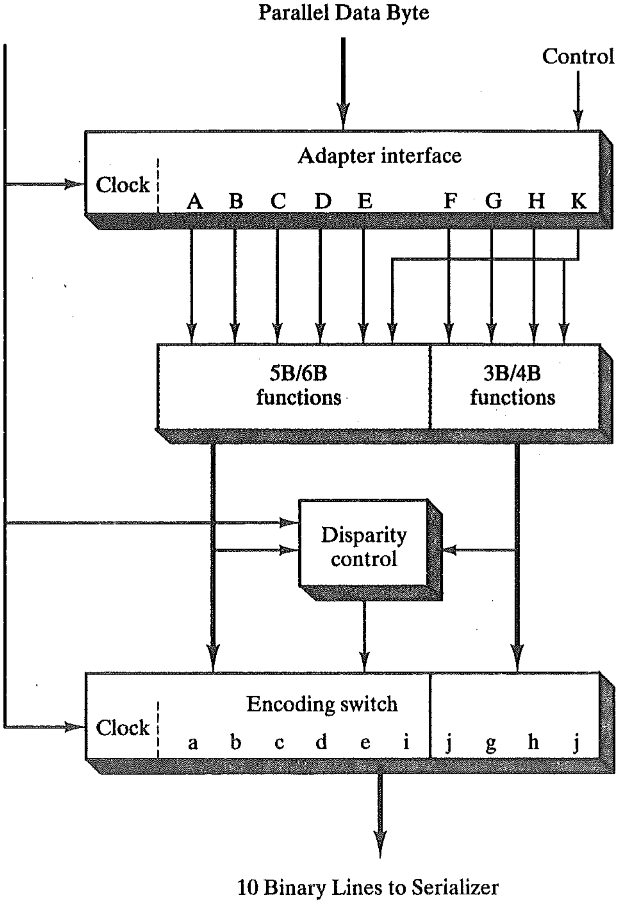


FIGURE 13.25 8B/10B encoding.

each of the possible 8-bit source blocks into a 10-bit code block. There is also a function called *disparity control.* In essence, this function keeps track of the excess of zeros over ones or ones over zeros. An excess in either direction is referred to as a disparity. If there is a disparity, and if the current code block would add to that disparity, then the disparity control block complements the 10-bit code block. This has the effect of either eliminating the disparity or at least moving it in the opposite direction of the current disparity.

The encoding mechanism also includes a control line input, K, which indicates whether the lines A through H are data or control bits. In the latter case, a special nondata 10-bit block is generated. A total of 12 of these nondata blocks is defined as valid in the standard; these are used for synchronization and for other control purposes.

13B APPENDIX

PERFORMANCE ISSUES

THE CHOICE OF a LAN or MAN architecture is based on many factors, but one of the most important is performance. Of particular concern is the behavior (throughput, response time) of the network under heavy load. Here, we provide an introduction to this topic. A more detailed discussion can be found in [STAL97].

The Effect of Propagation Delay and Transmission Rate

In Chapter 6, we introduced the parameter *a*, defined as

$$a = \frac{\text{Propagation time}}{\text{Transmission time}}$$

In that context, we were concerned with a point-to-point link, with a given propagation time between the two endpoints and a transmission time for either a fixed or an average frame size. It was shown that *a* could be expressed as

$$a = \frac{\text{Length of data link in bits}}{\text{Length of frame in bits}}$$

This parameter is also important in the context of LANs and MANs, and, in fact, determines an upper bound on utilization. Consider a perfectly efficient access mechanism that allows only one transmission at a time. As soon as one transmission is over, another station begins transmitting. Furthermore, the transmission is pure data; there are no overhead bits. What is the maximum possible utilization of the network? It can be expressed as the ratio of total throughput of the network to its capacity:

$$U = \frac{\text{Throughput}}{\text{Capacity}} \tag{13.1}$$

Now define, as in Chapter 6,

R = data rate of the channel
d = maximum distance between any two stations
V = velocity of signal propagation
L = average or fixed frame length

The throughput is just the number of bits transmitted per unit time. A frame contains L bits, and the amount of time devoted to that frame is the actual transmission time (L/R) plus the propagation delay (d/V). Thus,

$$\text{Throughput} = \frac{L}{d/V + L/R} \tag{13.2}$$

But by our definition of *a*, above

$$a = \frac{d/V}{L/R} = \frac{Rd}{LV} \tag{13.3}$$

Substituting (13.2) and (13.3) into (13.1)

$$U = \frac{1}{1 + a} \tag{13.4}$$

Note that the above differs from Equation (6.2), because the latter assumed a half-duplex protocol (no piggybacked acknowledgments).

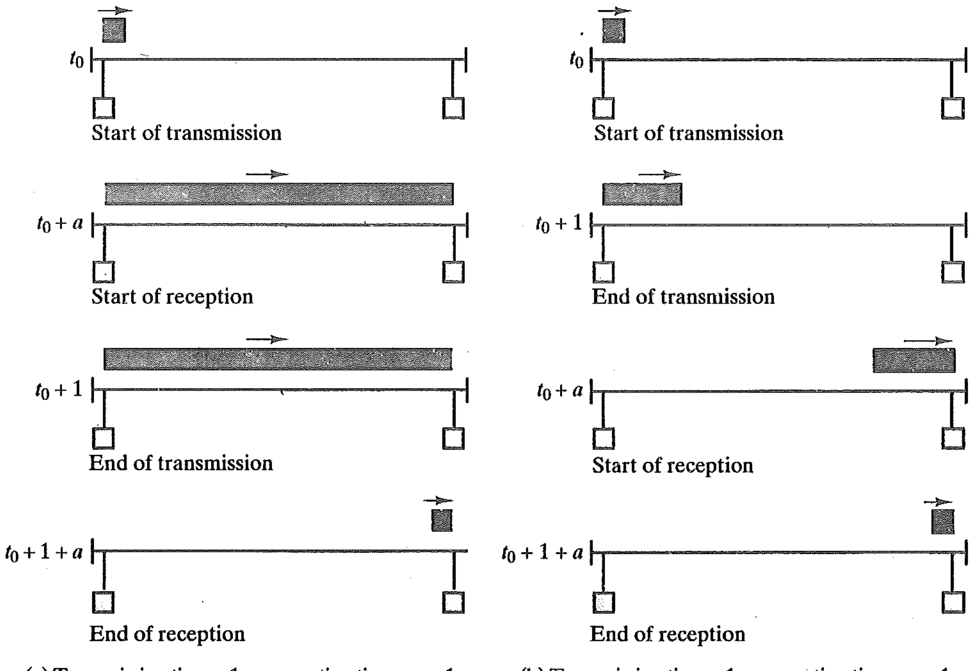
So, utilization varies with a; this can be grasped intuitively by studying Figure 13.26, which shows a baseband bus with two stations as far apart as possible (worst case) that take turns sending frames. If we normalize time such that frame transmission time = 1, then the propagation time = a. For $a < 1$, the sequence of events is as follows:

1. A station begins transmitting at t_0.
2. Reception begins at $t_0 + a$.
3. Transmission is completed at $t_0 + 1$.
4. Reception ends at $t_0 + 1 + a$.
5. The other station begins transmitting.

For $a > 1$, events 2 and 3 are interchanged. In both cases, the total time for one "turn" is $1 + a$, but the transmission time is only 1 for a utilization of $1/(1 + a)$.

The same effect as above can be seen to apply to a ring network in Figure 13.27. Here we assume that one station transmits and then waits to receive its own transmission before any other station transmits. The identical sequence of events outlined above applies.

Typical values of a range from about 0.01 to 0.1 for LANs and 0.1 to well over 1.0 for MANs. Table 13.11 gives some representative values for a bus topology. As can be seen, for larger and/or higher-speed networks, utilization suffers. For this reason, the restriction of only one frame at a time is lifted for LANs such as FDDI.



(a) Transmission time = 1; propagation time = $a < 1$

(b) Transmission time = 1; propagation time = $a > 1$

FIGURE 13.26 The effect of a on utilization for baseband bus.

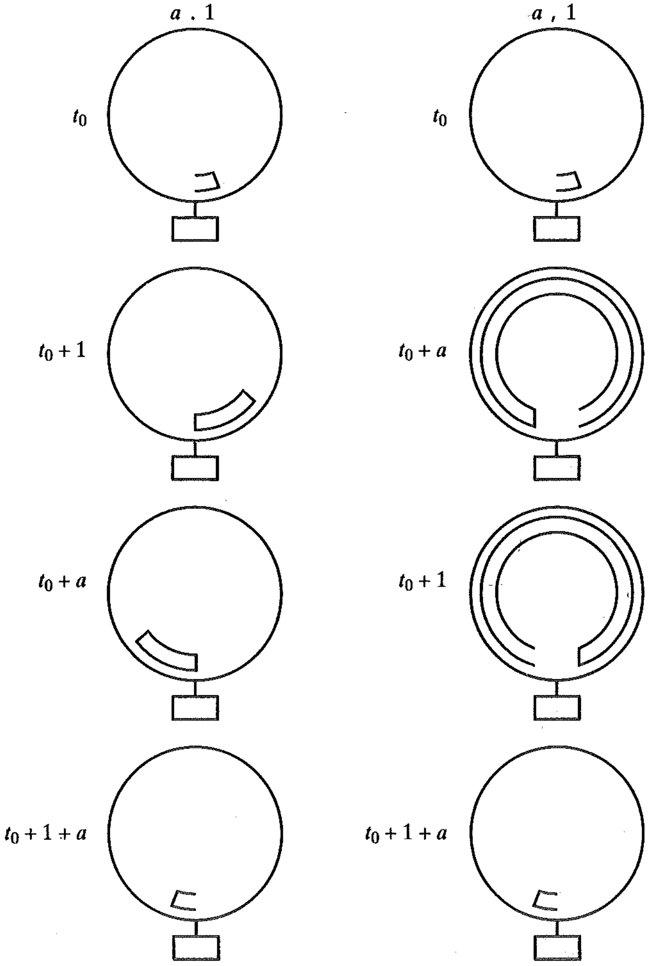


FIGURE 13.27 The effect of *a* on utilization: ring.

Finally, the analysis above assumes a "perfect" protocol, for which a new frame can be transmitted as soon as an old frame is received. In practice, the MAC protocol adds overhead that makes utilization worse. This is demonstrated in the next subsection for token-passing and for CSMA/CD.

Simple Performance Models of Token Passing and CSMA/CD

The purpose of this section is to give the reader some insight into the relative performance of the most important LAN protocols—CSMA/CD, token bus, and token ring—by developing two simple performance models. It is hoped that this exercise will aid in delineating the results of more rigorous analyses.

For these models, we assume a local network with *N* active stations, and a maximum normalized propagation delay of *a*. To simplify the analysis, we assume that each station is always prepared to transmit a frame; this allows us to develop an expression for maximum achievable utilization (*U*). Although this should not be construed to be the sole figure of

TABLE 13.11 Representative values of *a*.

Data rate (Mbps)	Frame size (bits)	Network length (km)	a	$\frac{1}{1+a}$
1	100	1	0.05	0.95
1	1,000	10	0.05	0.95
1	100	10	0.5	0.67
10	100	1	0.5	0.67
10	1,000	1	0.05	0.95
10	1,000	10	0.5	0.67
10	10,000	10	0.05	0.95
100	35,000	200	2.8	0.26
100	1,000	50	25	0.04

merit for a local network, it is the single most analyzed figure of merit, and does permit useful performance comparisons.

First, let us consider token ring. Time on the ring will alternate between data frame transmission and token passing. Refer to a single instance of a data frame followed by a token as a cycle and define the following:

C = average time for one cycle
T_1 = average time to transmit a data frame
T_2 = average time to pass a token

It should be clear that the average cycle rate is just $1/C = 1/(T_1 + T_2)$. Intuitively,

$$U = \frac{T_1}{T_1 + T_2} \tag{13.5}$$

That is, the throughput, normalized to system capacity, is just the fraction of time that is spent transmitting data.

Refer now to Figure 13.26; time is normalized such that frame transmission time equals 1 and propagation time equals a. Note that the propagation time must include repeater delays. For the case of $a < 1$, a station transmits a frame at time t_0, receives the leading edge of its own frame at $t_0 + a$, and completes transmission at $t_0 + 1$. The station then emits a token, which takes an average time a/N to reach the next station. Thus, one cycle takes $1 + a/N$ and the transmission time is 1. So $U = 1/(1 + a/N)$.

For $a > 1$, the reasoning is slightly different. A station transmits at t_0, completes transmission at $t_0 + 1$, and receives the leading edge of its frame at $t_0 + a$. At that point, it is free to emit a token, which takes an average time a/N to reach the next station. The cycle time is, therefore, $a + a/N$ and $U = 1/(a\,(1 + 1/N))$.

Summarizing,

Token:
$$U = \begin{cases} \dfrac{1}{1 + a/N} & a < 1 \\[2ex] \dfrac{1}{a\,(a + 1/N)} & a > 1 \end{cases} \tag{13.6}$$

The reasoning above applies equally well to token bus, where we assume that the logical ordering is the same as the physical ordering and that token-passing time is, therefore, a/N.

For CSMA/CD, consider time on the medium to be organized into slots whose length is twice the end-to-end propagation delay. This is a convenient way to view the activity on the medium; the slot time is the maximum time, from the start of transmission, required to detect a collision. Again, assume that there are N active stations. Clearly, if each station always has a frame to transmit, and does so, there will be nothing but collisions on the line. As a result, we assume that each station restrains itself to transmitting during an available slot with probability P.

Time on the medium consists of two types of intervals: first is a transmission interval, which lasts $1/2a$ slots; second is a contention interval, which is a sequence of slots with either a collision or no transmission in each slot. The throughput is just the amount of time spent in transmission intervals (similar to the reasoning for Equation (13.5)).

To determine the average length of a contention interval, we begin by computing A, the probability that exactly one station attempts a transmission in a slot and, therefore, acquires the medium. This is just the binomial probability that any one station attempts to transmit and the others do not:

$$\begin{aligned} A &= \binom{N}{1}p^1(1-P)^{N-1} \\ &= NP(1-P)^{N-1} \end{aligned}$$

This function takes on a maximum over P when $P = 1/N$:

$$A = (1 - 1/N)^{N-1}$$

We are interested in the maximum because we want to calculate the maximum throughput of the medium; it should be clear that maximum throughput will be achieved if we maximize the probability of successful seizure of the medium. Therefore, the following rule should be enforced: During periods of heavy usage, a station should restrain its offered load to $1/N$. (This assumes that each station knows the value of N. In order to derive an expression for maximum possible throughput, we live with this assumption.) On the other hand, during periods of light usage, maximum utilization cannot be achieved because the load is too low; this region is not of interest here.

Now we can estimate the mean length of a contention interval, w, in slots:

$$\begin{aligned} \mathrm{E}[w] &= \sum_{i=1}^{\infty} i \times \Pr\begin{bmatrix} i \text{ slots in a row with a collision or no} \\ \text{transmission followed by a slot with one} \\ \text{transmission} \end{bmatrix} \\ &= \sum_{i=1}^{\infty} i(1 - A)^i A \end{aligned}$$

The summation converges to

$$\mathrm{E}[w] = \frac{1 - A}{A}$$

We can now determine the maximum utilization, which is just the length of a transmission interval as a proportion of a cycle consisting of a transmission and a contention interval:

$$\text{CSMA / CD:} \qquad U = \frac{1/2a}{1/2a + (1 - A)/A} = \frac{1}{1 + 2a(1 - A)/A} \tag{13.7}$$

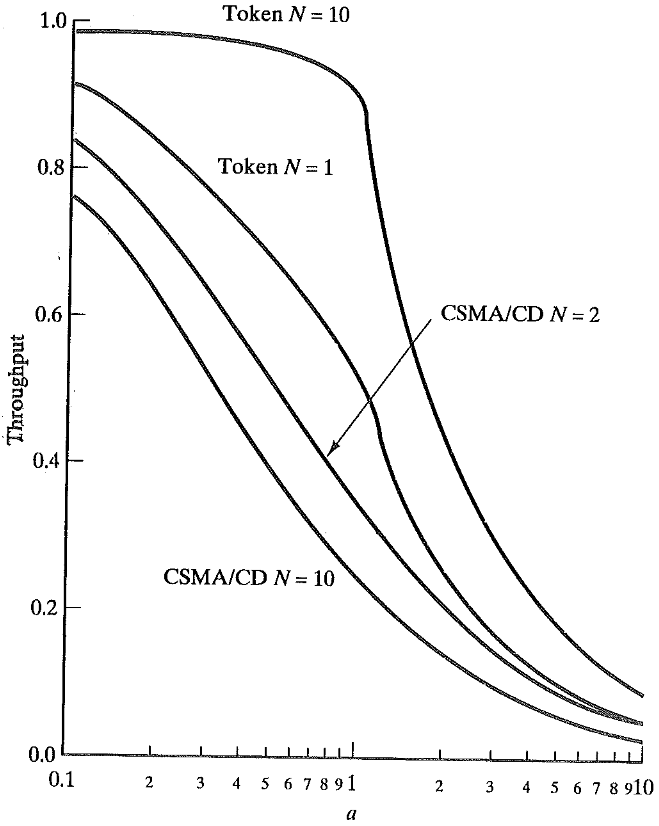


FIGURE 13.28 Throughput as a function of a for token passing and CSMA/CD.

Figure 13.28 shows normalized throughput as a function of various values of N and for both token passing and CSMA/CD. For both protocols, throughput declines as a increases; this is to be expected. The dramatic difference, though, between the two protocols is seen in Figure 13.29, which shows throughput as a function of N. Token-passing performance actually improves as a function of N, because less time is spent in token passing. Conversely, the performance of CSMA/CD decreases because of the increased likelihood of collision or no transmission.

It is interesting to note the asymptotic value of U as N increases. For token passing,

$$\text{Token:} \qquad \lim_{N \to \infty} U = \begin{cases} 1 & a < 1 \\ {}^{1}/_{a} & a < 1 \end{cases} \tag{13.8}$$

For CSMA/CD, we need to know that $\lim_{N \to \infty} (1 - 1/N)^{N-1} = 1/e$. Then we have

$$\text{CSMA/CD:} \qquad \lim_{N \to \infty} U = \frac{1}{1 + 3.44a} \tag{13.9}$$

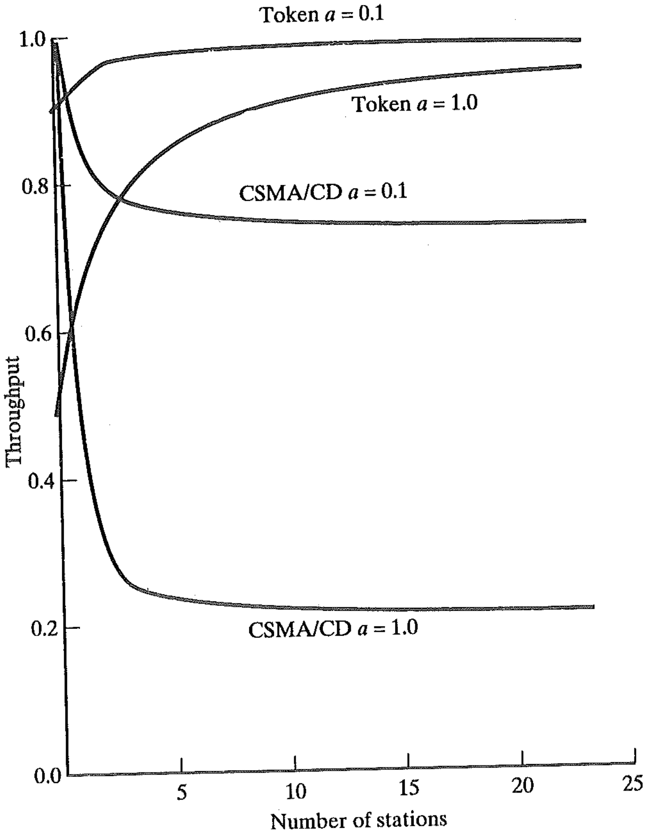
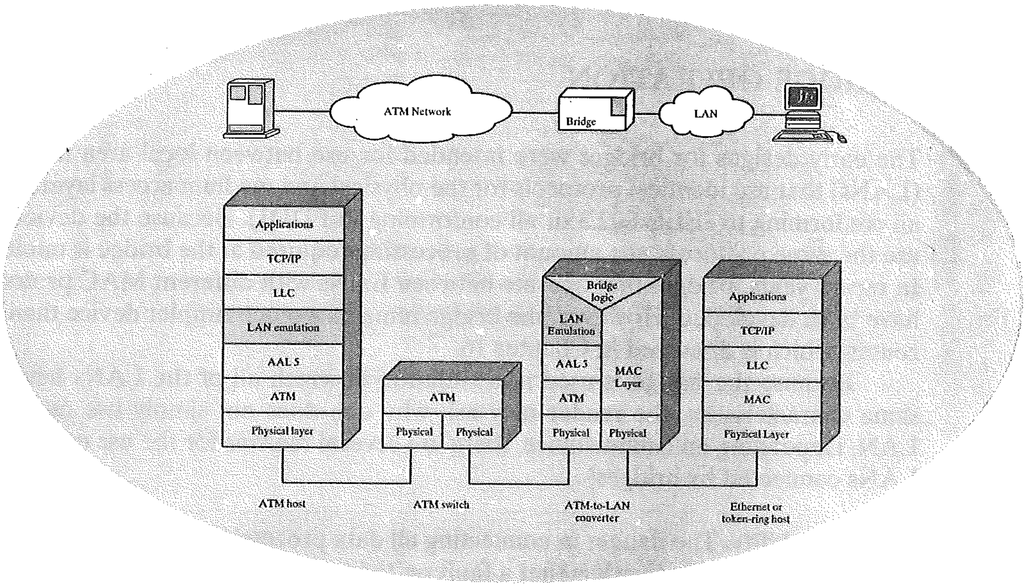


FIGURE 13.29 Throughput as a function of N for token pasing and CSMA/CD.

CHAPTER 14

BRIDGES



In most cases, a LAN or MAN is not an isolated entity. An organization may have more than one type of LAN at a given site to satisfy a spectrum of needs. An organization may have multiple LANs of the same type at a given site to accommodate performance or security requirements. Furthermore, an organization may have LANs and possibly MANs at various sites and may need them to be interconnected for central control of distributed information exchange.

The simplest approach to extending the range of LAN coverage is the use of bridges to interconnect a number of individual LANs. A more general solution, which allows the interconnection of local and wide area networks, is the use of an internetworking protocol and routers. This latter area is discussed in Chapter 16. Here, we concentrate on the bridge.

The chapter begins with a discussion of the basic operation of bridges. Then, we look at the most complex design issues associated with bridges, which is routing. Finally, we will return to the topic of ATM LANs, introduced in Chapter 13, and examine the concept of ATM LAN emulation.

14.1 BRIDGE OPERATION

The early designs for bridges were intended for use between local area networks (LANs) that use identical protocols for the physical and medium access layers (e.g., all conforming to IEEE 802.3 or all conforming to FDDI). Because the devices all use the same protocols, the amount of processing required at the bridge is minimal. In recent years, bridges that operate between LANs with different MAC protocols have been developed. However, the bridge remains a much simpler device than the router, which is discussed in Chapter 16.

Because the bridge is used in a situation in which all of the LANs have the same characteristics, the reader may ask why one does not simply use one large LAN. Depending on circumstance, there are several reasons for the use of multiple LANs connected by bridges:

- **Reliability.** The danger in connecting all data processing devices in an organization to one network is that a fault on the network may disable communication for all devices. By using bridges, the network can be partitioned into self-contained units.
- **Performance.** In general, performance on a LAN or MAN declines with an increase in the number of devices or with the length of the medium. A number of smaller LANs will often give improved performance if devices can be clustered so that *intra*-network traffic significantly exceeds *inter*-network traffic.
- **Security.** The establishment of multiple LANs may improve security of communications. It is desirable to keep different types of traffic (e.g., accounting, personnel, strategic planning) that have different security needs on physically separate media. At the same time, the different types of users with different levels of security need to communicate through controlled and monitored mechanisms.

- **Geography.** Clearly, two separate LANs are needed to support devices clustered in two geographically distant locations. Even in the case of two buildings separated by a highway, it may be far easier to use a microwave bridge link than to attempt to string coaxial cable between the two buildings. In the case of widely separated networks, two *half bridges* are needed (see Figure 14.3, below).

Functions of a Bridge

Figure 14.1 illustrates the operation of a bridge between two LANs, A and B. The bridge performs the following functions:

- Reads all frames transmitted on A, and accepts those addressed to stations on B.
- Using the medium access control protocol for B, retransmits the frames onto B.
- Does the same for B-to-A traffic.

Several design aspects of a bridge are worth highlighting:

1. The bridge makes no modification to the content or format of the frames it receives, nor does it encapsulate them with an additional header. Each frame to be transferred is simply copied from one LAN and repeated with exactly the same bit pattern as the other LAN. Because the two LANs use the same LAN protocols, it is permissible to do this.

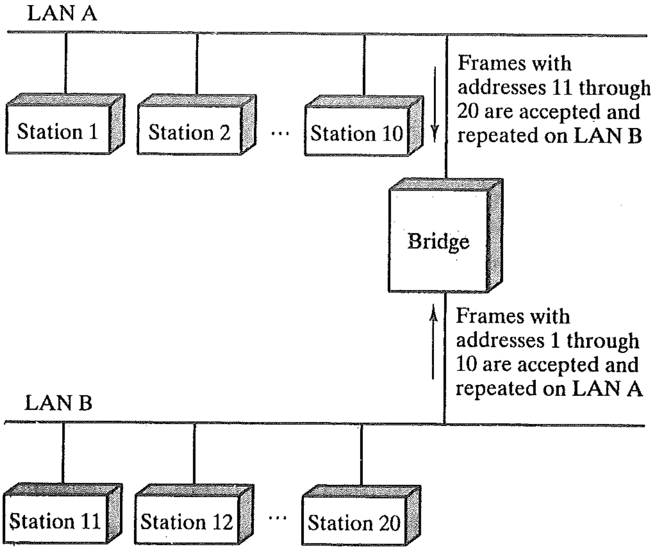

LAN A
Station 1
Station 2
Station 10
Frames with addresses 11 through 20 are accepted and repeated on LAN B
Bridge
Frames with addresses 1 through 10 are accepted and repeated on LAN A
LAN B
Station 11
Station 12
Station 20

FIGURE 14.1 Bridge operation.

2. The bridge should contain enough buffer space to meet peak demands. Over a short period of time, frames may arrive faster than they can be retransmitted.
3. The bridge must contain addressing and routing intelligence. At a minimum, the bridge must know which addresses are on each network in order to know which frames to pass. Further, there may be more than two LANs interconnected by a number of bridges. In that case, a frame may have to be routed through several bridges in its journey from source to destination.
4. A bridge may connect more than two LANs.

The bridge provides an extension to the LAN that requires no modification to the communications software in the stations attached to the LANs. It appears to all stations on the two (or more) LANs that there is a single LAN on which each station has a unique address. The station uses that unique address and need not explicitly discriminate between stations on the same LAN and stations on other LANs; the bridge takes care of that.

The description above has applied to the simplest sort of bridge. More sophisticated bridges can be used in more complex collections of LANs. These constructions would include additional functions, such as,

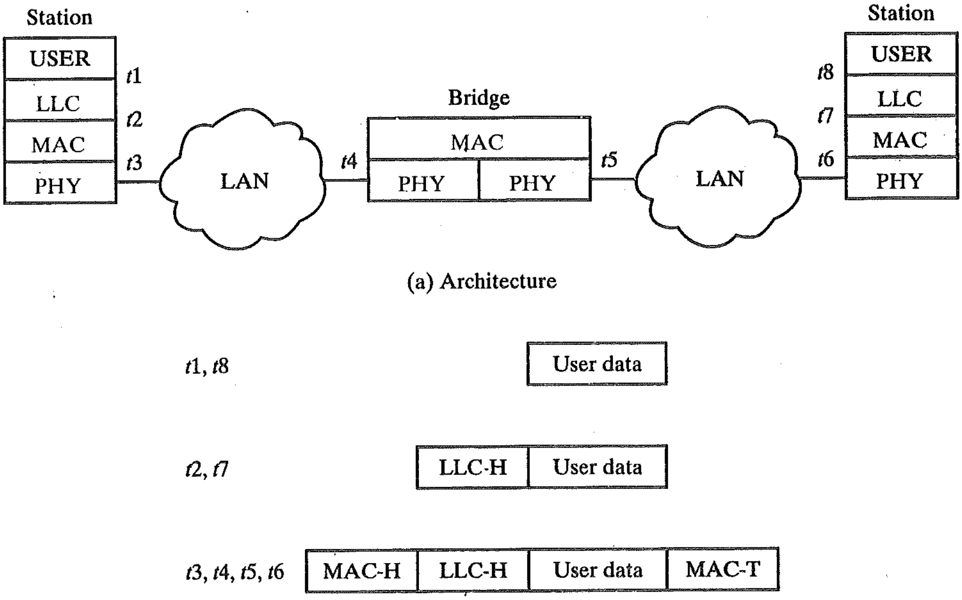
- Each bridge can maintain status information on other bridges, together with the cost and number of bridge-to-bridge hops required to reach each network. This information may be updated by periodic exchanges of information among bridges; this allows the bridges to perform a dynamic routing function.
- A control mechanism can manage frame buffers in each bridge to overcome congestion. Under saturation conditions, the bridge can give precedence to en route packets over new packets just entering the internet from an attached LAN, thus preserving the investment in line bandwidth and processing time already made in the en route frame.

Bridge Protocol Architecture

The IEEE 802.1D specification defines the protocol architecture for MAC bridges. In addition, the standard suggests formats for a globally administered set of MAC station addresses across multiple homogeneous LANs. In this subsection, we examine the protocol architecture of these bridges.

Within the 802 architecture, the endpoint or station address is designated at the MAC level. Thus, it is at the MAC level that a bridge can function. Figure 14.2 shows the simplest case, which consists of two LANs connected by a single bridge. The LANs employ the same MAC and LLC protocols. The bridge operates as previously described. A MAC frame whose destination is not on the immediate LAN is captured by the bridge, buffered briefly, and then transmitted on the other LAN. As far as the LLC layer is concerned, there is a dialogue between peer LLC entities in the two endpoint stations. The bridge need not contain an LLC layer, as it is merely serving to relay the MAC frames.

Figure 14.2b indicates the way in which data is encapsulated using a bridge. Data are provided by some user to LLC. The LLC entity appends a header and



(a) Architecture

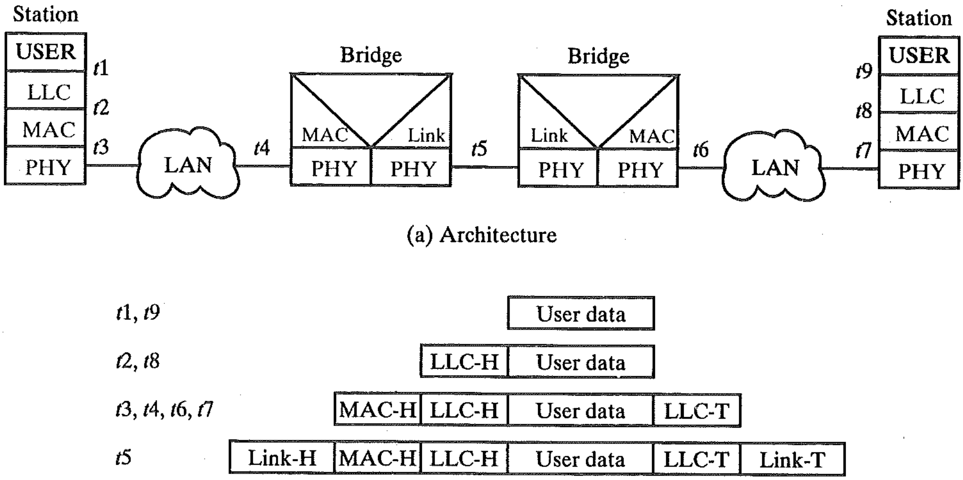
(b) Operation

FIGURE 14.2 Connection of two LANs by a bridge.

passes the resulting data unit to the MAC entity, which appends a header and a trailer to form a MAC frame. On the basis of the destination MAC address in the frame, it is captured by the bridge. The bridge does not strip off the MAC fields; its function is to relay the MAC frame intact to the destination LAN. Thus, the frame is deposited on the destination LAN and captured by the destination station.

The concept of a MAC relay bridge is not limited to the use of a single bridge to connect two nearby LANs. If the LANs are some distance apart, then they can be connected by two bridges that are in turn connected by a communications facility. For example, Figure 14.3 shows the case of two bridges connected by a point-to-point link. In this case, when a bridge captures a MAC frame, it appends a link layer (e.g., HDLC) header and trailer to transmit the MAC frame across the link to the other bridge. The target bridge strips off these link fields and transmits the original, unmodified MAC frame to the destination station.

The intervening communications facility can be a network, such as a wide-area–packet-switching network, as illustrated in Figure 14.4. In this case, the bridge is somewhat more complicated, although it performs the same function of relaying MAC frames. The connection between bridges is via an X.25 virtual circuit. Again, the two LLC entities in the end systems have a direct logical relationship with no intervening LLC entities. Thus, in this situation, the X.25 packet layer is operating below an 802 LLC layer. As before, a MAC frame is passed intact between the endpoints. When the bridge on the source LAN receives the frame, it appends an X.25 packet-layer header and an X.25 link-layer header and trailer and sends the data to the DCE (packet-switching node) to which it attaches. The DCE strips off the link-layer fields and sends the X.25 packet through the network to another DCE. The



(a) Architecture

FIGURE 14.3 Bridge over a point-to-point link.

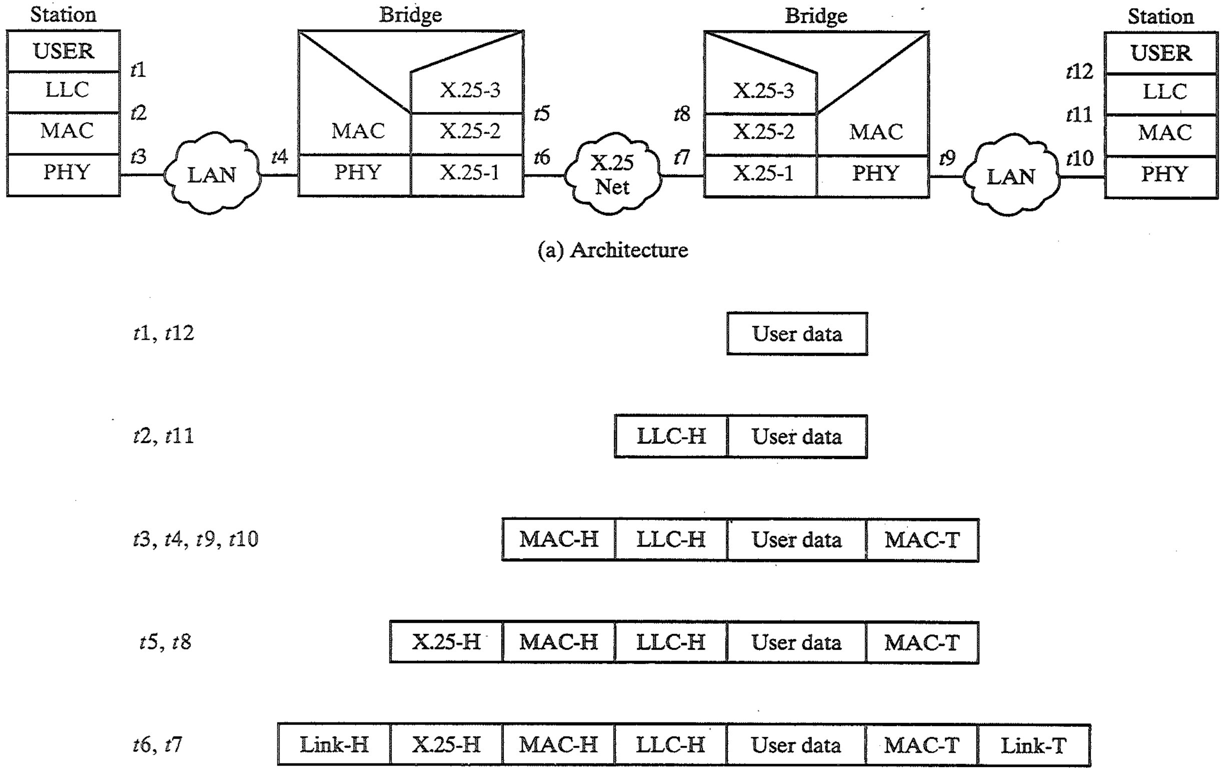
target DCE appends the link-layer field and sends this to the target bridge. The target bridge strips off all the X.25 fields and transmits the original unmodified MAC frame to the destination endpoint.

14.2 ROUTING WITH BRIDGES

In the configuration of Figure 14.1, the bridge makes the decision to relay a frame on the basis of destination MAC address. In a more complex configuration, the bridge must also make a routing decision. Consider the configuration of Figure 14.5. Suppose that station 1 transmits a frame on LAN A intended for station 5. The frame will be read by both bridge 101 and bridge 102. For each bridge, the addressed station is not on a LAN to which the bridge is attached. Therefore, each bridge must make a decision of whether or not to retransmit the frame on its other LAN, in order to move it closer to its intended destination. In this case, bridge 101 should repeat the frame on LAN B, whereas bridge 102 should refrain from retransmitting the frame. Once the frame has been transmitted on LAN B, it will be picked up by both bridges 103 and 104. Again, each must decide whether or not to forward the frame. In this case, bridge 104 should retransmit the frame on LAN E, where it will be received by the destination, station 5.

Thus, we see that, in the general case, the bridge must be equipped with a routing capability. When a bridge receives a frame, it must decide whether or not to forward it. If the bridge is attached to two or more networks, then it must decide whether or not to forward the frame and, if so, on which LAN the frame should be transmitted.

The routing decision may not always be a simple one. In Figure 14.6, bridge 107 is added to the previous configuration, directly linking LAN A and LAN E. Such an addition may be made to provide for higher overall internet availability. In this case, if Station 1 transmits a frame on LAN A intended for station 5 on LAN



(a) Architecture

(b) Operation

FIGURE 14.4 Bridge over an X.25 network.

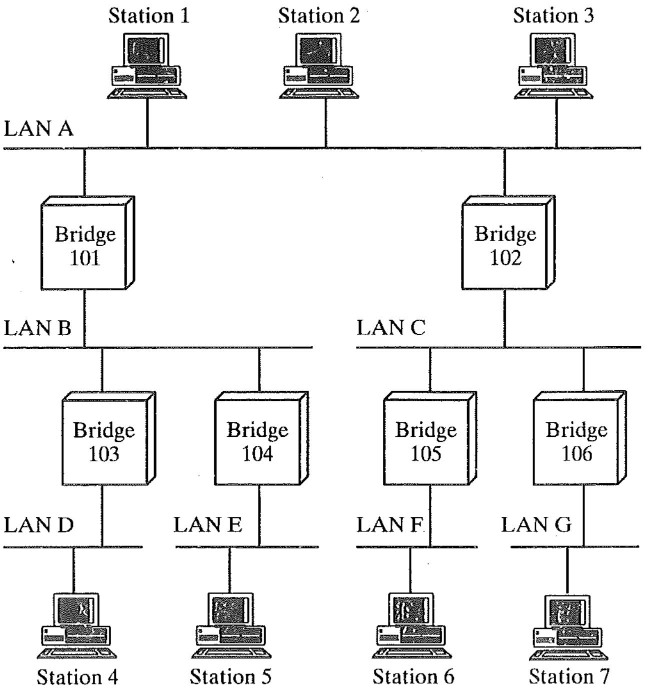


FIGURE 14.5 Configuration of bridges and LANs.

E, then either bridge 101 or bridge 107 could forward the frame. It would appear preferable for bridge 107 to forward the frame, as it will involve only one *hop*, whereas if the frame travels through bridge 101, it must suffer two hops. Another consideration is that there may be changes in the configuration. For example, bridge 107 may fail, in which case subsequent frames from station 1 to station 5 should go through bridge 101. We can say, then, that the routing capability must take into account the topology of the internet configuration and may need to be dynamically altered.

Figure 14.6 suggests that a bridge knows the identity of each station on each LAN. In a large configuration, such an arrangement is unwieldy. Furthermore, as stations are added to and dropped from LANs, all directories of station location must be updated. It would facilitate the development of a routing capability if all MAC-level addresses were in the form of a network part and a station part. For example, the IEEE 802.5 standard suggests that 16-bit MAC addresses consist of a 7-bit LAN number and an 8-bit station number, and that 48-bit addresses consist of a 14-bit LAN number and a 32-bit station number.[1] In the remainder of this discussion, we assume that all MAC addresses include a LAN number and that routing is based on the use of that portion of the address only.

[1] The remaining bit in the 16-bit format is used to indicate whether this is a group or individual address. Of the two remaining bits in the 48-bit format, one is used to indicate whether this is a group or individual address, and the other is used to indicate whether this is a locally administered or globally administered address.

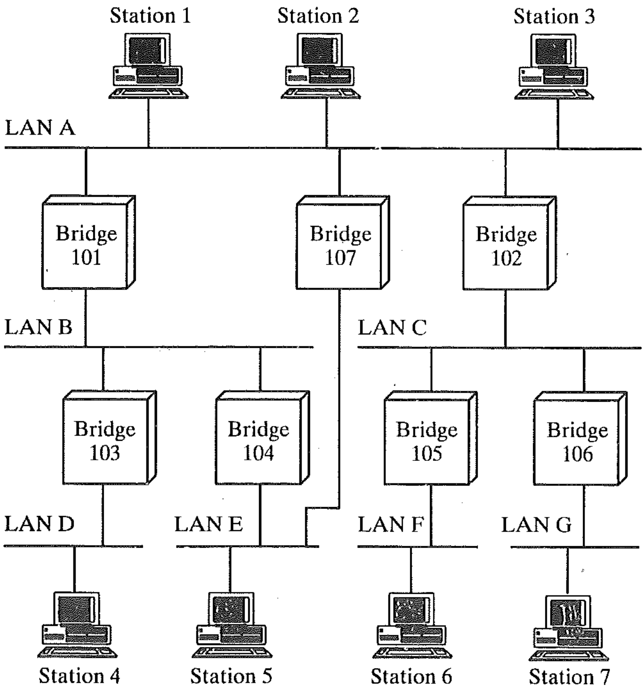


FIGURE 14.6 Configuration of bridges and LANs, with alternate routes.

A variety of routing strategies have been proposed and implemented in recent years. The simplest and most common strategy is *fixed routing.* This strategy is suitable for small LAN collections and for interconnections that are relatively stable. More recently, two groups within the IEEE 802 committee have developed specifications for routing strategies. The IEEE 802.1 group has issued a standard for routing based on the use of a *spanning tree* algorithm. The token ring committee, IEEE 802.5, has issued its own specification, referred to as *source routing.* We examine these three strategies in turn.

Fixed Routing

Fixed routing was introduced in our discussion of routing for packet-switching networks. For fixed routing with bridges, a route is selected for each source-destination pair of LANs in the internet. If alternate routes are available between two LANs, then typically the route with the least number of hops is selected. The routes are fixed, or at least only change when there is a change in the topology of the internet.

Figure 14.7 shows a fixed-routing design for the configuration of Figure 14.6. A central routing matrix shows, for each source-destination pair of LANs, the identity of the first bridge on the route. So, for example, the route from LAN E to LAN F begins by going through bridge 107 to LAN A. Again, consulting the matrix, the route from LAN A to LAN F goes through bridge 102 to LAN C. Finally, the route from LAN C to LAN F is directly through bridge 105. Thus, the

CENTRAL ROUTING DIRECTORY

		Source LAN						
		A	B	C	D	E	F	G
Destination LAN	A	—	101	102	103	107	105	106
	B	101	—	102	103	104	105	106
	C	102	101	—	103	107	105	106
	D	101	103	102	—	104	105	106
	E	107	104	102	103	—	105	106
	F	102	101	105	103	107	—	106
	G	102	101	106	103	107	105	—

Bridge 101 Table

from LAN A		from LAN B	
Dest	Next	Dest	Next
B	B	A	A
C	—	C	A
D	B	D	—
E	—	E	—
F	—	F	A
G	—	G	A

Bridge 102 Table

from LAN A		from LAN C	
Dest	Next	Dest	Next
B	—	A	A
C	C	B	A
D	—	D	A
E	—	E	A
F	C	F	—
G	C	G	—

Bridge 103 Table

from LAN B		from LAN D	
Dest	Next	Dest	Next
A	—	A	B
C	—	B	B
D	D	C	B
E	—	E	B
F	—	F	B
G	—	G	B

Bridge 104 Table

from LAN B		from LAN E	
Dest	Next	Dest	Next
A	—	A	—
C	—	B	B
D	—	C	—
E	E	D	B
F	—	F	—
G	—	G	—

Bridge 105 Table

from LAN C		from LAN F	
Dest	Next	Dest	Next
A	—	A	C
B	—	B	C
D	—	C	C
E	—	D	C
F	F	E	C
G	—	G	C

Bridge 106 Table

from LAN C		from LAN G	
Dest	Next	Dest	Next
A	—	A	C
B	—	B	C
D	—	C	C
E	—	D	C
F	—	E	C
G	G	F	C

Bridge 107 Table

from LAN A		from LAN E	
Dest	Next	Dest	Next
B	—	A	A
C	—	B	—
D	—	C	A
E	E	D	—
F	—	F	A
G	—	G	A

FIGURE 14.7 **Fixed routing (using Figure 14.6).**

complete route from LAN E to LAN F is bridge 107, LAN A, bridge 102, LAN C, bridge 105.

Only one column of this matrix is needed in each bridge for each LAN to which it attaches. For example, bridge 105 has two tables, one for frames arriving from LAN C and one for frames arriving from LAN F. The table shows, for each possible destination MAC address, the identity of the LAN to which the bridge should forward the frame. The table labeled "From LAN C" is derived from the column labeled C in the routing matrix. Every entry in that column that contains bridge number 105 results in an entry in the corresponding table in bridge 105.

Once the directories have been established, routing is a simple matter. A bridge copies each incoming frame on each of its LANs. If the destination MAC address corresponds to an entry in its routing table, the frame is retransmitted on the appropriate LAN.

The fixed routing strategy is widely used in commercially available products; it has the advantage of simplicity and minimal processing requirements. However, in a complex internet, in which bridges may be dynamically added and in which failures must be allowed for, this strategy is too limited. In the next two subsections, we cover more powerful alternatives.

Spanning Tree Routing

The spanning tree approach is a mechanism in which bridges automatically develop a routing table and update that table in response to changing topology. The algorithm consists of three mechanisms: frame forwarding, address learning, and loop resolution.

Frame Forwarding

In this scheme, a bridge maintains a *filtering database*, which is based on MAC address. Each entry consists of a MAC individual or group address, a port number, and an aging time (described below); we can interpret this in the following fashion. A station is listed with a given port number if it is on the *same side* of the bridge as the port. For example, for bridge 102 of Figure 14.5, stations on LANs C, F, and G are on the same side of the bridge as the LAN A port; and stations on LANs A, B, D, and E are on the same side of the bridge as the LAN C port. When a frame is received on any port, the bridge must decide whether that frame is to be forwarded through the bridge and out through one of the bridge's other ports. Suppose that a bridge receives a MAC frame on port *x*. The following rules are applied (Figure 14.8):

1. Search the forwarding database to determine if the MAC address is listed for any port except port *x*.
2. If the destination MAC address is not found, flood the frame by sending it out on all ports except the port by which it arrived.
3. If the destination address is in the forwarding database for some port $y \neq x$, then determine whether port *y* is in a blocking or a forwarding state. For reasons explained below, a port may sometimes be blocked, which prevents it from receiving or transmitting frames.

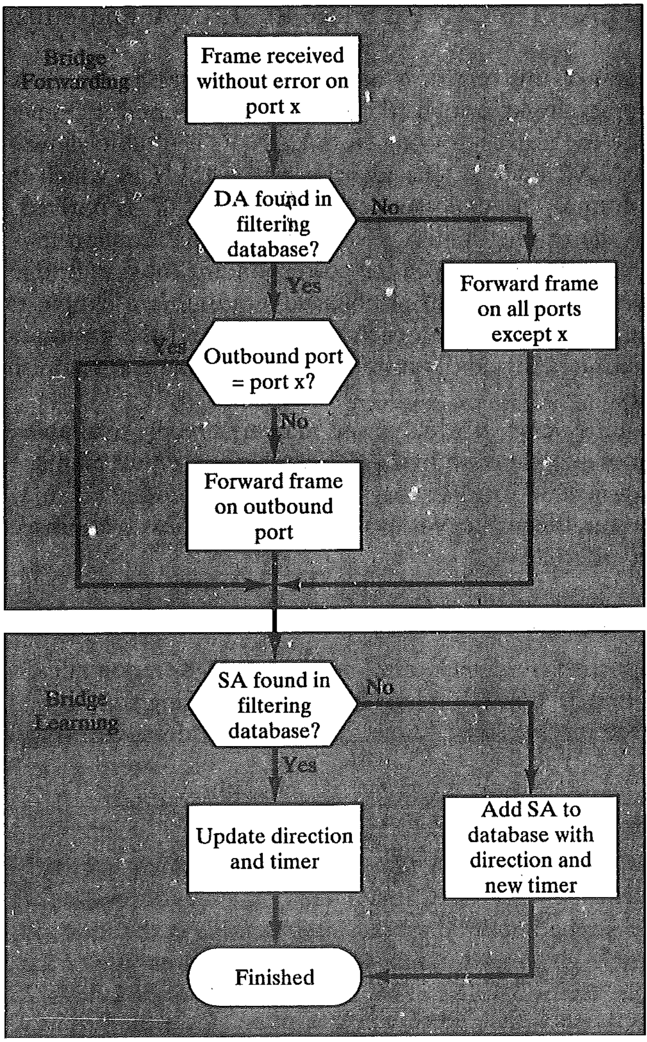

Bridge Forwarding
Frame received without error on port x
DA found in filtering database?
No
Forward frame on all ports except x
Yes
Outbound port = port x?
Yes
No
Forward frame on outbound port
Bridge Learning
SA found in filtering database?
No
Yes
Update direction and timer
Add SA to database with direction and new timer
Finished

FIGURE 14.8 Bridge forwarding and learning.

4. If port *y* is not blocked, transmit the frame through port *y* onto the LAN to which that port attaches.

Rule (2) is needed because of the dynamic nature of the filtering database. When a bridge is initialized, the database is empty. Because the bridge does not know where to send the frame, it floods the frame onto all of its LANs except the LAN on which the frame arrives. As the bridge gains information, the flooding activity subsides.

Address Learning

The above scheme is based on the use of a filtering database that indicates the direction, from the bridge, of each destination station. This information can be preloaded into the bridge, as in static routing. However, an effective automatic mechanism for learning the direction of each station is desirable. A simple scheme for acquiring this information is based on the use of the source address field in each MAC frame (Figure 14.8).

When a frame arrives on a particular port, it clearly has come from the direction of the incoming LAN. The source address field of the frame indicates the source station. Thus, a bridge can update its filtering database for that MAC address. To allow for changes in topology, each entry in the database is equipped with an aging timer. When a new entry is added to the database, its timer is set; the recommended default value is 300 seconds. If the timer expires, then the entry is eliminated from the database, as the corresponding direction information may no longer be valid. Each time a frame is received, its source address is checked against the database. If the entry is already in the database, the entry is updated (the direction may have changed) and the timer is reset. If the entry is not in the database, a new entry is created, with its own timer.

The above discussion indicated that the individual entries in the database are station addresses. If a two-level address structure (LAN number, station number) is used, then only LAN addresses need to be entered in the database. Both schemes work the same. The only difference is that the use of station addresses requires a much larger database than the use of LAN addresses.

Note from Figure 14.8 that the bridge learning process is applied to all frames, not just those that are forwarded.

Spanning Tree Algorithm

The address learning mechanism described above is effective if the topology of the internet is a tree; that is, if there are no alternate routes in the network. The existence of alternate routes means that there is a closed loop. For example in Figure 14.6, the following is a closed loop: LAN A, bridge 101, LAN B, bridge 104, LAN E, bridge 107, LAN A.

To see the problem created by a closed loop, consider Figure 14.9. At time t_0, station A transmits a frame addressed to station B. The frame is captured by both bridges. Each bridge updates its database to indicate that station A is in the direction of LAN X, and retransmits the frame on LAN Y. Say that bridge α retransmits at time t_1 and bridge β a short time later, t_2. Thus, B will receive two copies of the frame. Furthermore, each bridge will receive the other's transmission on LAN Y. Note that each transmission is a MAC frame with a source address of A and a destination address of B. Each bridge, then, will update its database to indicate that station A is in the direction of LAN Y. Neither bridge is capable now of forwarding a frame addressed to station A.

But the problem is potentially more serious. Assume that the two bridges do not yet know of the existence of station B. In this case, we have the following scenario. A transmits a frame addressed to B. Each bridge captures the frame. Then, each bridge, because it does not have information about B, automatically retransmits a copy of the frame on LAN Y. The frame transmitted by bridge α is captured by station B *and* by bridge β. Because bridge β does not know where B is, it takes this frame and retransmits it on LAN X. Similarly, bridge a receives bridge β's transmission on LAN Y and retransmits the frame on LAN X. There are now two

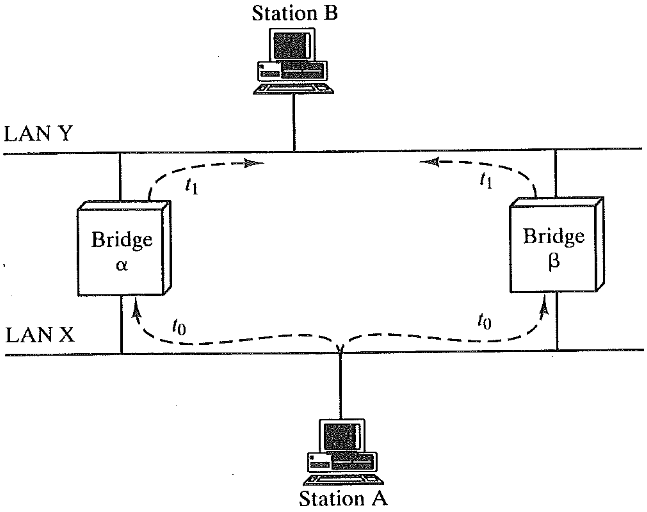


FIGURE 14.9 Loop of bridges.

frames on LAN X that will be picked up for retransmission on LAN Y. This process repeats indefinitely.

To overcome the above problem, a simple result from graph theory is used: For any connected graph, consisting of nodes and edges connecting pairs of nodes, there is a spanning tree of edges that maintains the connectivity of the graph but contains no closed loops. In terms of internets, each LAN corresponds to a graph node, and each bridge corresponds to a graph edge. Thus, in Figure 14.6, the removal of one (and only one) of bridges 107, 101, or 104, results in a spanning tree. What is desired is to develop a simple algorithm by which the bridges of the internet can exchange sufficient information to automatically (without user intervention) derive a spanning tree. The algorithm must be dynamic. That is, when a topology change occurs, the bridges must be able to discover this fact and automatically derive a new spanning tree.

The algorithm is based on the use of the following:

1. Each bridge is assigned a unique identifier; in essence, the identifier consists of a MAC address for the bridge plus a priority level.
2. There is a special group MAC address that means "all bridges on this LAN." When a MAC frame is transmitted with the group address in the destination address field, all of the bridges on the LAN will capture that frame and interpret it as a frame addressed to itself.
3. Each port of a bridge is uniquely identified within the bridge, with a *port identifier.*

With this information established, the bridges are able to exchange routing information in order to determine a spanning tree of the internet. We will explain the operation of the algorithm using Figures 14.10 and 14.11 as an example. The following concepts are needed in the creation of the spanning tree:

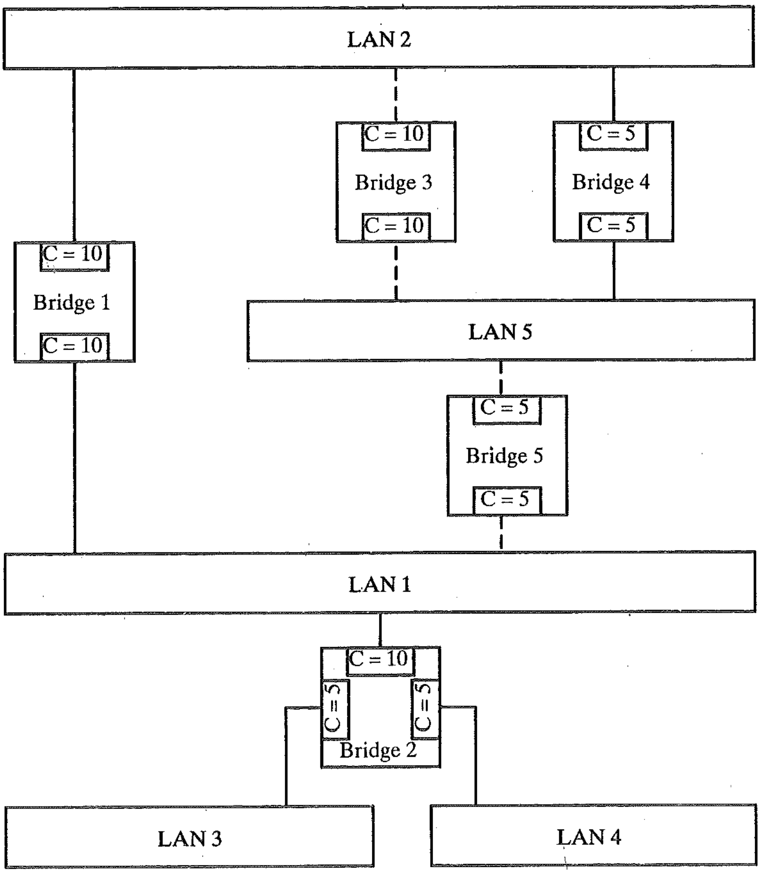


FIGURE 14.10 Example configuration for spanning tree algorithm.

- **Root bridge.** The bridge with the lowest value of bridge identifier is chosen to be the root of the spanning tree.
- **Path cost.** Associated with each port on each bridge is a path cost, which is the cost of transmitting a frame onto a LAN through that port. A path between two stations will pass through 0 or more bridges. At each bridge, the cost of transmission is added to give a total cost for a particular path. In the simplest case, all path costs would be assigned a value of 1; thus, the cost of a path would simply be a count of the number of bridges along the path. Alternatively, costs could be assigned in inverse proportion to the data rate of the corresponding LAN, or any other criterion chosen by the network manager.
- **Root port.** Each bridge discovers the first hop on the minimum-cost path to the root bridge. The port used for that hop is labeled the root port. When the cost is equal for two ports, the lower port number is selected so that a unique spanning tree is constructed.

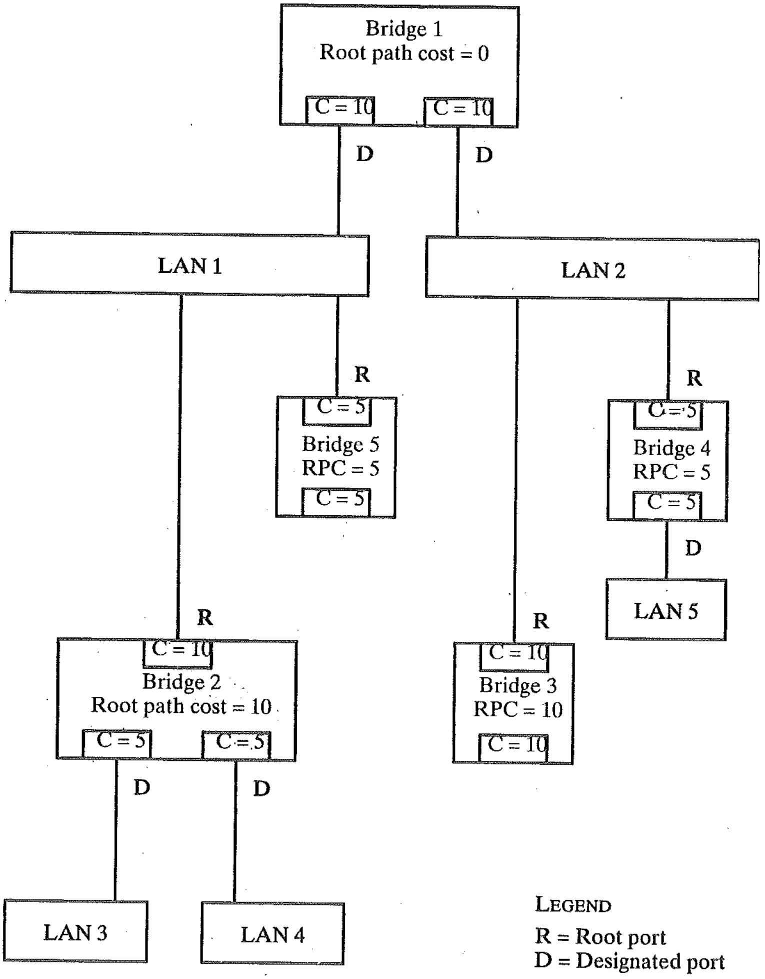


FIGURE 14.11 Spanning tree for configuration of Figure 14.13.

- **Root path cost.** For each bridge, the cost of the path to the root bridge with minimum cost (the path that starts at the root port) is the root path cost for that bridge.
- **Designated bridge, designated port.** On each LAN, one bridge is chosen to be the designated bridge. This is the bridge on that LAN that provides the minimum cost path to the root bridge. This is the only bridge allowed to forward frames to and from the LAN for which it is the designated bridge. The port of the designated bridge that attaches the bridge to the LAN is the designated port. For all LANs to which the root bridge is attached, the root bridge is the designated bridge. All internet traffic to and from the LAN passes through the designated port.

In general terms, the spanning tree is constructed in the following fashion:

1. Determine the root bridge.
2. Determine the root port on all other bridges.
3. Determine the designated port on each LAN. This will be the port with the minimum root path cost. In the case of two or more bridges with the same root path cost, the highest-priority bridge is chosen as the designated bridge. If the designated bridge has two or more ports attached to this LAN, then the port with the lowest value of port identifier is chosen.

By this process, when two LANs are directly connected by more than one bridge, all of the bridges but one are eliminated. This cuts any loops that involve two LANs. It can be demonstrated that this process also eliminates all loops involving more than two LANs and that connectivity is preserved. Thus, this process discovers a spanning tree for the given internet. In our example, the solid lines indicate the bridge ports that participate in the spanning tree.

The steps outlined above require that the bridges exchange information. The information is exchanged in the form of bridge protocol data units (BPDUs). A BPDU transmitted by one bridge is addressed to and received by all of the other bridges on the same LAN. Each BPDU contains the following information:

- The identifier of this bridge and the port on this bridge
- The identifier of the bridge that this bridge considers to be the root
- The root path cost for this bridge

To begin, all bridges consider themselves to be the root bridge. Each bridge will broadcast a BPDU on each of its LANs that asserts this fact. On any given LAN, only one claimant will have the lowest-valued identifier and will maintain its belief. Over time, as BPDUs propagate, the identity of the lowest-valued bridge identifier throughout the internet will be known to all bridges. The root bridge will regularly broadcast the fact that it is the root bridge on all of the LANs to which it is attached; this allows the bridges on those LANs to determine their root port and the fact that they are directly connected to the root bridge. Each of these bridges in turn broadcasts a BPDU on the other LANs to which it is attached (all LANs except the one on its root port), indicating that it is one hop away from the root bridge. This activity is propagated throughout the internet. Every time that a bridge receives a BPDU, it transmits BPDUs, indicating the identity of the root bridge and the number of hops to reach the root bridge. On any LAN, the bridge claiming to be the one that is closest to the root becomes the designated bridge.

We can trace some of this activity with the configuration in Figure 14.10. At startup time, bridges 1, 3, and 4 all transmit BPDUs on LAN 2, each claiming to be the root bridge. When bridge 3 receives the transmission from bridge 1, it recognizes a superior claimant and defers. Bridge 3 has also received a claiming BPDU from bridge 5 via LAN 5. Bridge 3 recognizes that bridge 1 has a superior claim to be the root bridge; it therefore assigns its LAN 2 port to be its root port, and sets the root path cost to 10. By similar actions, bridge 4 ends up with a root path cost of 5 via LAN 2; bridge 5 has a root path cost of 5 via LAN 1; and bridge 2 has a root path cost of 10 via LAN 1.

Now consider the assignment of designated bridges. On LAN 5, all three bridges transmit BPDUs, attempting to assert a claim to be designated bridge. Bridge 3 defers because it receives BPDUs from the other bridges that have a lower root path cost. Bridges 4 and 5 have the same root path cost, but bridge 4 has the higher priority and therefore becomes the designated bridge.

The results of all this activity are shown in Figure 14.11. Only the designated bridge on each LAN is allowed to forward frames. All of the ports on all of the other bridges are placed in a blocking state. After the spanning tree is established, bridges continue to periodically exchange BPDUs to be able to react to any change in topology, cost assignments, or priority assignment. Anytime that a bridge receives a BPDU on a port, it makes two assessments:

1. If the BPDU arrives on a port that is considered the designated port, does the transmitting port have a better claim to be the designated port?
2. Should this port be my root port?

Source Routing

The IEEE 802.5 committee has developed a bridge routing approach referred to as source routing. With this approach, the sending station determines the route that the frame will follow and includes the routing information with the frame; bridges read the routing information to determine if they should forward the frame.

Basic Operation

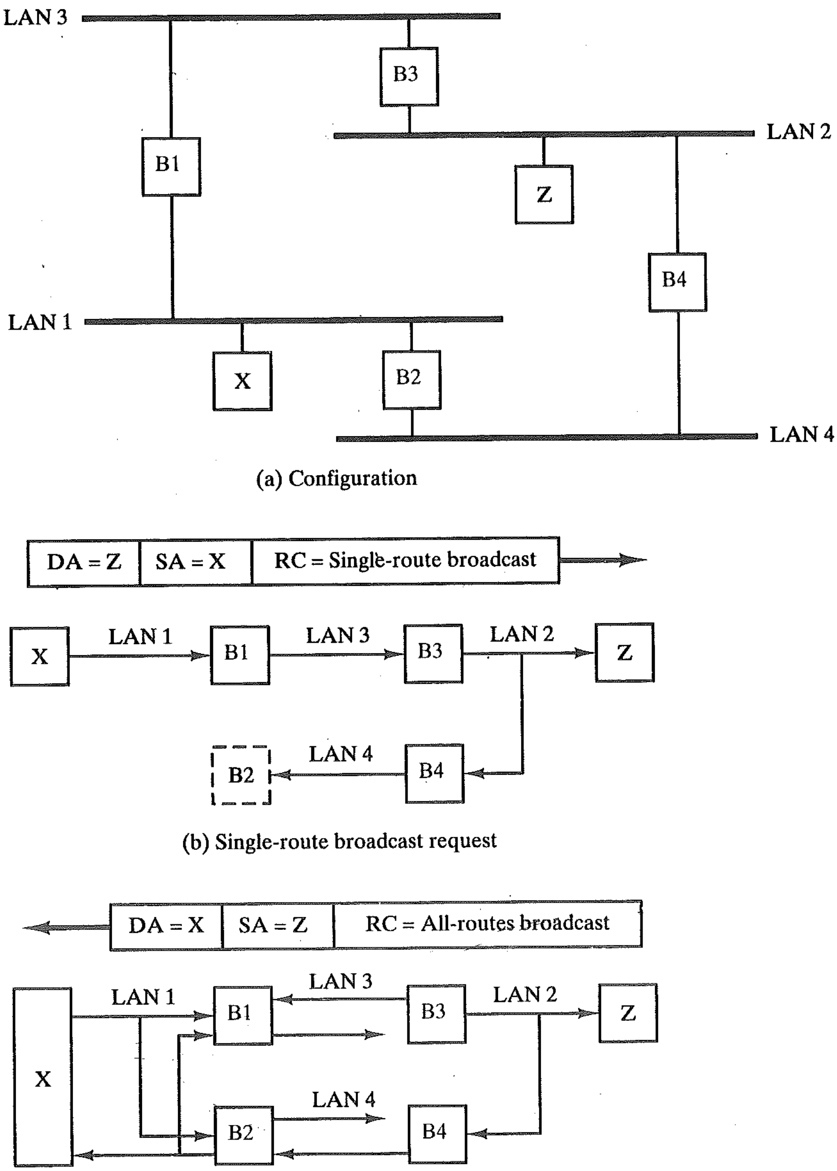
The basic operation of the algorithm can be described by making reference to the configuration in Figure 14.12. A frame from station X can reach station Z by either of the following routes:

- LAN 1, bridge B1, LAN 3, bridge B3, LAN 2
- LAN 1, bridge B2, LAN 4, bridge B4, LAN 2

Station X may choose one of these two routes and place the information, in the form of a sequence of LAN and bridge identifiers, in the frame to be transmitted. When a bridge receives a frame, it will forward that frame if the bridge is on the designated route; all other frames are discarded. In this case, if the first route above is specified, bridges B1 and B3 will forward the frame; if the second route is specified, bridges B2 and B4 will forward the frame.

Note that with this scheme, bridges need not maintain routing tables. The bridge makes the decision whether or not to forward a frame solely on the basis of the routing information contained in that frame. All that is required is that the bridge know its own unique identifier and the identifier of each LAN to which it is attached. The responsibility for designing the route falls to the source station.

For this scheme to work, there must be a mechanism by which a station can determine a route to any destination station. Before addressing this issue, we need to discuss various types of routing directives.



(a) Configuration

(b) Single-route broadcast request

(c) All-routes broadcast response

FIGURE 14.12 Route discovery example.

Routing Directives and Addressing Modes

The source routing scheme developed by the IEEE 802.5 committee includes four different types of routing directives. Each frame that is transmitted includes an indicator of the type of routing desired. The four directive types are

- **Null.** No routing is desired. In this case, the frame can only be delivered to stations on the same LAN as the source station.
- **Nonbroadcast.** The frame includes a route, consisting of a sequence of LAN numbers and bridge numbers, that defines a unique route from the source station to the destination station. Only bridges on that route forward the frame, and only a single copy of the frame is delivered to the destination station.
- **All-routes broadcast.** The frame will reach each LAN of the internet by all possible routes. Thus, each bridge will forward each frame once to each of its ports in a direction away from the source node, and multiple copies of the frame may appear on a LAN. The destination station will receive one copy of the frame for each possible route through the network.
- **Single-route broadcast.** Regardless of the destination address of the frame, the frame will appear once, and only once, on each LAN in the internet. For this effect to be achieved, the frame is forwarded by all bridges that are on a spanning tree (with the source node as the root) of the internet. The destination station receives a single copy of the frame.

Let us first examine the potential application of each of these four types of routing, and then examine the mechanisms that may be employed to achieve these procedures. First, consider null routing. In this case the bridges that share the LAN with the source station are told not to forward the frame; this will be done if the intended destination is on the same LAN as the source station. Nonbroadcast routing is used when the two stations are not on the same LAN and the source station knows a route that can be used to reach the destination station. Only the bridges on that route will forward the frame.

The remaining two types of routing can be used by the source to discover a route to the destination. For example, the source station can use all-routes broadcasting to send a request frame to the intended destination. The destination returns a response frame on each of the routes, using nonbroadcast routing, followed by the incoming request frame. The source station can pick one of these routes and send future frames on that route. Alternatively, the source station could use single-route broadcasting to send a single request frame to the destination station. The destination station could send its response frame via all-routes broadcasting. The incoming frames would reveal all of the possible routes to the destination station, and the source station could pick one of these for future transmissions. Finally, single-route broadcasting could be used for group addressing, as discussed below.

Now consider the mechanisms for implementing these various routing directives. Each frame must include an indicator of which of the four types of routing is required. For null routing, the frame is ignored by the bridge. For nonbroadcast routing, the frame includes an ordered list of LAN numbers and bridge numbers. When a bridge receives a nonbroadcast frame, it forwards the frame only if the routing information contains the sequence LAN *i*, Bridge *x*, LAN *j*, where

LAN i = LAN from which the frame arrived
Bridge x = this bridge
LAN j = another LAN to which this bridge is attached

For all-routes broadcasting, the source station marks the frame, but includes no routing information. Each bridge that forwards the frame will add its bridge number and the outgoing LAN number to the frame's routing information field. Thus, when the frame reaches its destination, it will include a sequenced list of all LANs and bridges visited. To prevent the endless repetition and looping of frames, a bridge obeys the following rule. When an all-routes broadcast frame is received, the bridge examines the routing information field. If the field contains the number of a LAN to which the bridge is attached, the bridge will refrain from forwarding the frame on that LAN. Put the other way, the bridge will only forward the frame to a LAN that the frame has not already visited.

Finally, for single-route broadcasting, a spanning tree of the internet must be developed. This can either be done automatically, as in the 802.1 specification, or manually. In either case, as with the 802.1 strategy, one bridge on each LAN is the designated bridge for that LAN, and is the only one that forwards single-route frames.

It is worth noting the relationship between addressing mode and routing directive. There are three types of MAC addresses:

- **Individual.** The address specifies a unique destination station.
- **Group.** The address specifies a group of destination addresses; this is also referred to as *multicast.*
- **All-stations.** The address specifies all stations that are capable of receiving this frame; this is also referred to as *broadcast.* We will refrain from using this latter term as it is also used in the source routing terminology.

In the case of a single, isolated LAN, group and all-stations addresses refer to stations on the same LAN as the source station. In an internet, it may be desirable to transmit a frame to multiple stations on multiple LANs. Indeed, because a set of LANs interconnected by bridges should appear to the user as a single LAN, the ability to do group and all-stations addressing across the entire internet is mandatory.

Table 14.1 summarizes the relationship between routing specification and addressing mode. If no routing is specified, then all addresses refer only to the immediate LAN. If nonbroadcast routing is specified, then addresses may refer to any station on any LAN visited on the nonbroadcast route. From an addressing point of view, this combination is not generally useful for group and all-stations addressing. If either the all-routes or single-route specification is included in a frame, then all stations on the internet can be addressed. Thus, the total internet acts as a single network from the point of view of MAC addresses. Because less traffic is generated by the single-route specification, this single-network characteristic is to be preferred for group and all-stations addressing. Note also that the single-route mechanism in source routing is equivalent to the 802.1 spanning tree

TABLE 14.1 Effects of various combinations of addressing and source routing.

Addressing mode	Routing specification			
	No routing	Nonbroadcast	All-routes	Single-route
Indi idual	Received by station if it is on the same LAN	Received by station if it is on one of the LANs on the route	Received by station if it is on any LAN	Received by station if it is on any LAN
Group	Received by all group members on the same LAN	Received by all group members on all LANs visited on this route	Received by all group members on all LANs	Received by all group members on all LANs
All-Stations	Received by all stations on the same LAN	Received by all stations on all LANs visited on this route	Received by all stations on all LANs	Received by all stations on all LANs

approach. Thus, the spanning tree approach supports both group and all-stations addressing.

Route Discovery and Selection

With source routing, bridges are relieved of the burden of storing and using routing information. Thus, the burden falls on the stations that wish to transmit frames. Clearly, some mechanism is needed by which the source stations can know the route by which frames are to be sent. Three strategies suggest themselves:

1. Manually load the information into each station. This is simple and effective but has several drawbacks. First, anytime that the configuration is changed, the routing information at all stations must be updated. Secondly, this approach does not provide for automatic adjustment in the face of the failure of a bridge or LAN.
2. One station on a LAN can query other stations on the same LAN for routing information about distant stations. This approach may reduce the overall amount of routing messages that must be transmitted, compared to options 3 or 4, below. However, at least one station on each LAN must have the needed routing information, so this is not a complete solution.
3. When a station needs to learn the route to a destination station, it engages in a dynamic route discovery procedure.

Option 3 is the most flexible and the one that is specified by IEEE 802.5; as was mentioned earlier, two approaches are possible. The source station can transmit an all-routes request frame to the destination. Thus, all possible routes to the destination are discovered. The destination station can then send back a nonbroadcast response on each of the discovered routes, allowing the source to choose which route to follow in subsequently transmitting the frame. This approach generates quite a bit of both forward and backward traffic, and requires the destination station to receive and transmit a number of frames. An alternative is for the source station to transmit a single-route request frame. Only one copy of this frame will reach the destination. The destination responds with an all-routes response frame, which generates all possible routes back to the source. Again, the source can choose among these alternative routes.

Figure 14.12 illustrates the latter approach. Assume that the spanning tree that has been chosen for this internet consists of bridges B1, B3, and B4. In this example, station X wishes to discover a route to station Z. Station X issues a single-route request frame. Bridge B2 is not on the spanning tree and so does not forward the frame. The other bridges do forward the frame, and it reaches station Z. Note that bridge B4 forwards the frame to LAN 4, although this is not necessary; it is simply an effect of the spanning tree mechanism. When Z receives this frame, it responds with an all-routes frame. Two messages reach X: one on the path LAN 2, B3, LAN 3, B1, LAN 1, and the other on the path LAN 2, B4, LAN 4, B2, LAN 1. Note that a frame that arrived by the latter route is received by bridge B1 and forwarded onto LAN 3. However, when bridge B3 receives this frame, it sees in the routing information field that the frame has already visited LAN 2; therefore, it does not forward the frame. A similar fate occurs for the frame that follows the first route and is forwarded by bridge B2.

Once a collection of routes has been discovered, the source station needs to select one of the routes. The obvious criterion would be to select the minimum-hop route. Alternatively, a minimum-cost route could be selected, where the cost of a network is inversely proportional to its data rate. In either case, if two or more routes are equivalent by the chosen criterion, then there are two alternatives:

1. Choose the route corresponding to the response message that arrives first. One may assume that that particular route is less congested than the others as the frame on that route arrived earliest.
2. Choose randomly. This should have the effect, over time, of leveling the load among the various bridges.

Another point to consider is how often to update a route. Routes should certainly be changed in response to network failures and should perhaps be changed in response to network congestion. If connection-oriented logical link control is used (see Chapter 5), then one possibility is to rediscover the route with each new connection. Another alternative, which works with either connection-oriented or connectionless service, is to associate a timer with each selected route, and to rediscover the route when its time expires.

14.3 ATM LAN EMULATION

One issue that was not addressed in our discussion of ATM LANs in Section 13.4 has to do with the interoperability of end systems on a variety of interconnected LANs. End systems attached directly to one of the legacy LANs implement the

MAC layer appropriate to that type of LAN. End systems attached directly to an ATM network implement the ATM and AAL protocols. As a result, there are three areas of compatibility to consider:

1. Interaction between an end system on an ATM network and an end system on a legacy LAN.
2. Interaction between an end system on a legacy LAN and an end system on another legacy LAN of the same type (e.g., two IEEE 802.3 networks).
3. Interaction between an end system on a legacy LAN and an end system on another legacy LAN of a different type (e.g., an IEEE 802.3 network and an IEEE 802.5 network).

The most general solution to this problem is the router, which is explored in Chapter 16. In essence, the router operates at the level of the Internet Protocol (IP). All of the end systems implement IP, and all networks are interconnected with routers. If data are to travel beyond the scope of an individual LAN, they are directed to the local router. There, the LLC and MAC layers are stripped off, and the IP PDU is routed across one or more other networks to the destination LAN, where the appropriate LLC and MAC layers are invoked. Similarly, if one or both of the end systems are directly attached to an ATM network, the AAL and ATM layers are stripped off or added to an IP PDU.

While this approach is effective, it introduces a certain amount of processing overhead and delay at each router. In very large internetworks, these delays can become substantial. Networks of 1000 routers are increasingly common, and networks of as many as 10,000 routers have been installed [LANG95]. A technique that exploits the efficiency of ATM and that reduces the number of required routers is desired.

Another way to approach the problem is to convert all end systems such that they operate directly on ATM. In this case, there is a seamless technology used throughout any network, including local and wide area components. However, with millions of Ethernet and token ring nodes installed on today's shared-media LANs, most organizations simply can't afford a one-shot upgrade of all systems to ATM. In addition, although the cost of ATM interface cards is dropping, Ethernet and token ring interfaces remain cheaper for the time being.

In response to this need, the ATM Forum[2] has created a specification for the coexistence of legacy LANs and ATM LANs, known as ATM LAN emulation [ATM95]. The objective on ATM LAN emulation is to enable existing shared-

[2] The ATM Forum is a nonprofit international industry consortium, which is playing a crucial role in the development of ATM standards. In the ITU and the constituent member bodies from the participating countries, the process of developing standards is characterized by wide participation on the part of government, users, and industry representatives, and by consensus decision-making. This process can be quite time-consuming. While ITU-T has streamlined its efforts, the delays involved in developing standards is particularly significant in the area of ATM technology. Because of the strong level of interest in ATM, the ATM Forum was created with the goal of accelerating the development of ATM standards. The ATM Forum has seen more active participation from computing vendors than has been the case in ITU-T. Because the forum works on the basis of majority rule rather than consensus, it has been able to move rapidly to define some of the needed details for the implementation of ATM. This effort, in turn, has fed into the ITU-T standardization effort.

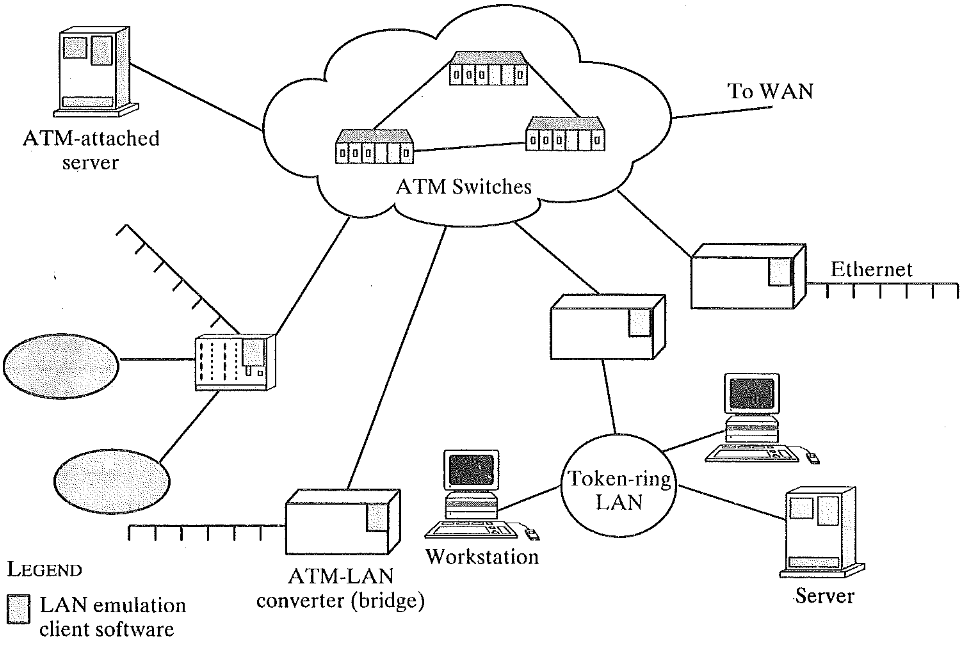


FIGURE 14.13 Example ATM LAN emulation configuration.

media LAN nodes to interoperate across an ATM network and to interoperate with devices that connect directly to ATM switches.

Figure 14.13 illustrates the type of configuration that can be constructed using ATM LAN emulation. In its present form, the ATM specification satisfies two of the three requirements listed earlier; namely, ATM LAN emulation defines the following:

1. The way in which end systems on two separate LANs of the same type (same MAC layer) can exchange MAC frames across the ATM network.
2. The way in which an end system on a LAN can interoperate with an end system emulating the same LAN type and attached directly to an ATM switch.

The specification does not as yet address interoperability between end systems on different LANs with different MAC protocols.

Protocol Architecture

Figure 14.14 indicates the protocol architecture involved in ATM LAN emulation. In this case, we are looking at the interaction of an ATM-attached system with an end system attached to a legacy LAN. Note that the end system attached to a legacy LAN is unaffected; it is able to use the ordinary repertoire of protocols, including the MAC protocol specific to this LAN and LLC running on top of the MAC. Thus, the end system runs TCP/IP over LLC, and various application-level protocols on top of that; the various application-level protocols are unaware that there is an

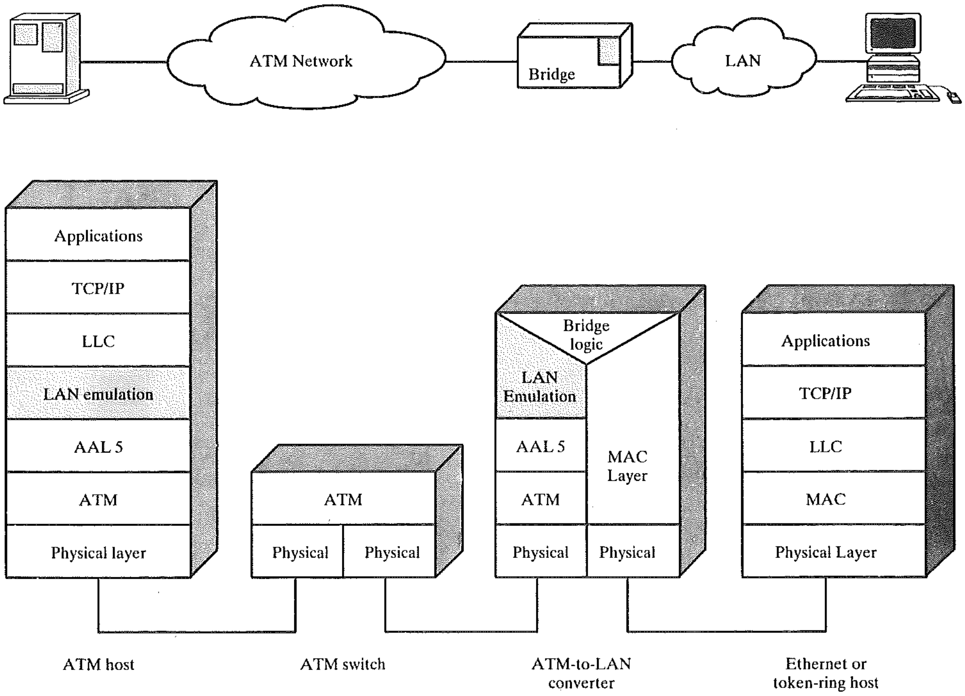


FIGURE 14.14 LAN emulation protocol architecture.

ATM network underneath. The key to this diagram is the use of a bridge device known as an ATM-to-LAN converter.

In Figure 14.14, the bridge logic must be augmented by the capability of converting MAC frames to and from ATM cells. This is one of the key functions of the LAN emulation module. The ATM Forum specification calls for making use of AAL 5 to segment MAC frames into ATM cells and to reassemble incoming ATM cells into MAC frames. For outgoing ATM cells, the ATM-to-LAN converter connects in the usual fashion to an ATM switch as part of an ATM network.

Figure 14.14 shows the case in which a host on a legacy LAN is exchanging data with a host attached directly to an ATM network. To accommodate this exchange, the ATM host must include a LAN emulation module that accepts MAC frames from AAL and must pass up the contents to an LLC layer. Thus, the host is indeed emulating a LAN because it can receive and transmit MAC frames in the same format as the distant legacy LAN. From the point of view of end systems on the legacy LAN, the ATM host is just another end system with a MAC address. The entire LAN emulation process is transparent to existing systems implementing LLC and MAC.

Emulated LANs

With the protocol architecture just described, it is possible to set up a number of logically independent, *emulated LANs.* An emulated LAN supports a single

MAC protocol, of which two types are currently defined: Ethernet/IEEE 802.3 and IEEE 802.5 (token ring). An emulated LAN consists of some combination of the following:

- End systems on one or more legacy LANs
- End systems attached directly to an ATM switch

Each end system on an emulated LAN must have a unique MAC address. Data interchange between end systems on the same emulated LAN involves the use of the MAC protocol and is transparent to the upper layers. That is, it appears to LLC that all of the end systems on an emulated LAN are on the same shared-medium LAN. Communication between end systems on different emulated LANs is possible only through routers or bridges. Note that the bridges or routers have to reassemble the cells into packets and then chop them up into cells to send them to another emulated LAN.

LAN Emulation Clients and Servers

The discussion so far leaves out a number of issues that must be addressed, including the following:

1. Devices attached directly to ATM switches and to ATM-to-LAN converter systems have ATM-based addresses. How are translations made between these addresses and MAC addresses?
2. ATM makes use of a connection-oriented protocol involving virtual channels and virtual paths. How can the connectionless LAN MAC protocol be supported over this connection-oriented framework?
3. Multicasting and broadcasting on a shared-medium LAN is easily achieved. How is this capability carried over into the ATM environment?

To address these issues, the ATM Forum developed a capability based on a client/server approach, which is discussed next.

ATM LAN emulation requires two types of components: clients and servers. Clients operate on behalf of devices that are attached to legacy LANs and that use MAC addresses. A client is responsible for adding its MAC entities into the overall configuration and for dealing with the tasks associated with translating between MAC addresses and ATM addresses. Typically, a client would be provided in a router, in an ATM-attached server (see Figure 14.13), or perhaps in an ATM switch that directly connects to one of the above (referred to as an *edge switch*). Servers are responsible for integrating MAC entities into the overall configuration and for managing all of the associated tasks, such as finding addresses and emulating broadcasting. Servers may be implemented in separate components or in ATM switches. Each emulated LAN consists of one or more clients and a single LAN emulation service.

The LAN emulation service in fact comprises three types of servers, which perform separate tasks: the LAN Emulation Configuration Server (LECS), the LAN Emulation Server (LES), and the Broadcast and Unknown Server (BUS). The reason for breaking the server up into three modules is that a manager may decide to have more of one kind of server than another, for efficient operation, and may decide to distribute the servers physically to minimize the communications burden. Table 14.2 provides a brief definition of the three types of servers and of the client.

TABLE 14.2 LAN emulation client and servers.

Entity	Description
LAN Emulation Client (LEC)	Sets up control connections to LAN emulation servers; sets up data connections to other clients; maps MAC addresses to ATM addresses.
LAN Emulation Configuration Server (LECS)	Assists client in selecting an LES.
LAN Emulation Server (LES)	Performs initial address mapping; accepts clients.
Broadcast and Unknown Server (BUS)	Performs multicasting.

Figure 14.15 indicates the way in which clients and the three types of servers interact. The client can establish virtual channel connections, called control connections, to the LECS and the LES. The link to the LECS is used by an LEC to gain

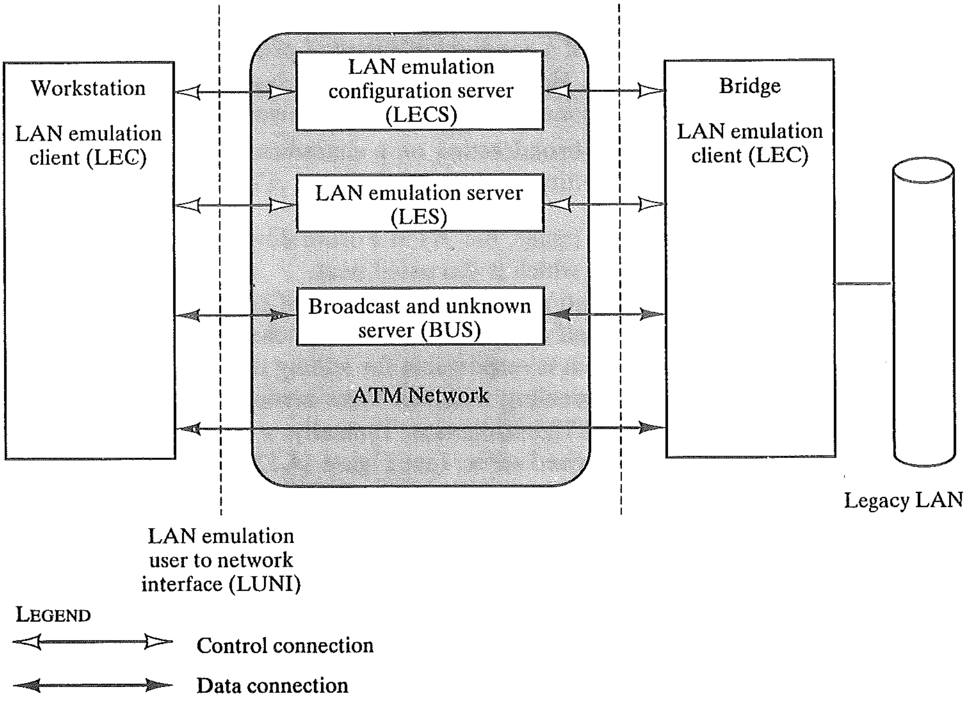

Workstation
LAN emulation client (LEC)
LAN emulation configuration server (LECS)
LAN emulation server (LES)
Broadcast and unknown server (BUS)
ATM Network
Bridge
LAN emulation client (LEC)
Legacy LAN
LAN emulation user to network interface (LUNI)
LEGEND
Control connection
Data connection

FIGURE 14.15 LAN emulation client connections across LUNI.

entrance to an emulated LAN and to locate an LES. The LES is responsible for registering new clients and their MAC addresses into an emulated LAN, as well as for mapping between MAC addresses and ATM addresses.

Once a client and its end systems have joined an emulated LAN, most of the work is done across virtual channel connections called data connections. MAC frames, segmented into ATM cells, are transmitted via data connections between end systems on the same emulated LAN. For a unicast transmission, a virtual channel connection is set up between two clients; this is the protocol setup illustrated in Figure 14.14. Finally, the data connection between a client and a BUS carries transmissions intended for broadcast or multicast, and it also is used to handle transmissions in which the sending client does not know the address of the receiving client.

LAN Emulation Scenario

To clarify the concepts involved in LAN emulation, let us follow a typical sequence of events.

Initialization

To join an emulated LAN, a client must begin by obtaining the ATM address of the LAN emulation server (LES) for that emulated LAN. Typically, the way in which this is done is that the client establishes a virtual channel connection to the LAN emulation configuration server (LECS).

There are three possible techniques by which the client can discover the LECS ATM address so that it can perform initialization:

1. The client can use a network management procedure defined as part of the ATM Forum's Interim Local Management Interface (ILMI). This procedure takes place between the client and ILMI software in the associated ATM switch. If the ILMI software has the LECS address for the requested emulated LAN, it provides that address to the client. The client then establishes a virtual channel connection to the LECS.
2. If the ILMI procedure fails, the client tries a predefined address listed in the specification, called the *well-known address*. This address is supposed to correspond to an LECS on any ATM network that conforms to the ATM Forum specification. The client uses this address to establish a virtual channel connection to the LECS.
3. If the well-known address fails, the client tries the well-known *virtual path identifier/virtual channel identifier* defined in the ATM Forum specification. When the ATM network is configured, the network manager can establish this permanent virtual path/virtual channel.

Configuration

Once a connection is established between the client and the LECS, the client can engage in a dialogue with the LECS. Based upon its own policies, configuration database, and information provided by the client, the LECS assigns the client to a particular emulated LAN service by giving the client the LES's ATM address. The LECS returns information to the client about the emulated LAN, including MAC protocol, maximum frame size, and the name of the emulated LAN. The name may be something defined by the configuration manager to be meaningful in defining logical workgroups (e.g., finance, personnel).

Joining

The client now has the information it needs to join an emulated LAN. It proceeds by setting up a control connection to the LES. The client then issues a JOIN REQUEST to the LES, which includes the client's ATM address, its MAC address, LAN type, maximum frame size, a client identifier, and a proxy indication. This latter parameter indicates whether this client corresponds to an end system attached directly to an ATM switch or is a LAN-to-ATM converter supporting end systems on a legacy LAN.

If the LES is prepared to accept this client, it sends back a JOIN RESPONSE, indicating acceptance. Otherwise, it sends back a JOIN RESPONSE indicating rejection.

Registration and BUS Initialization

Once a client has joined an emulated LAN, it goes through a registration procedure. If the client is a proxy for a number of end systems on a legacy LAN, it sends a list of all MAC addresses on the legacy LAN that are to be part of this emulated LAN to the LES.

Next, the client sends a request to the LES for the ATM address of the BUS. This address functions as the broadcast address for the emulated LAN and is used when a MAC frame is to be broadcast to all stations on the emulated LAN. The client then sets up a data connection to the BUS.

Data Transfer

Once a client is registered, it is able to send and receive MAC frames. First, consider the sending of MAC frames. In the case of an end system attached to an ATM switch, the end system generates its own MAC frames for transmission to one or more other end systems on the emulated LAN. In the case of a proxy client, it functions as a bridge that receives MAC frames from end systems on its legacy LAN and then transmits those MAC frames. In both cases, an outgoing MAC frame must be segmented into ATM cells and transmitted over a virtual channel. There are three cases to consider:

- Unicast MAC frame, ATM address known
- Unicast MAC frame, address unknown
- Multicast or broadcast MAC frame

If the client knows the ATM address of the unicast frame, it checks whether it has a virtual data connection already established to the destination client. If so, it sends the frame over that connection (as a series of ATM cells); otherwise, it uses ATM signaling to set up the connection and then sends the frame.

If the address is unknown, the sending client performs two actions. First, the client sends the frame to the BUS over the data connection that it maintains with the BUS. The BUS, in turn, either transmits the frame to the intended MAC destination or else broadcasts the frame to all MAC destinations on the emulated LAN. In the latter case, the intended destination will recognize its MAC address and accept the frame. Second, the client attempts to learn the ATM address for this MAC for future reference. It does this by sending an LE_ARP_REQUEST (LAN Emulation Address Resolution Protocol Request) command to the LES; the command includes the MAC address for which an ATM address is desired. If the LES knows the ATM address, it returns the address to the client in an LE_ARP_RESPONSE. Otherwise, the LES holds the request while it attempts to learn the ATM address. The LES sends out its own LE_ARP_REQUEST to all clients on the emulated LAN. The client that represents the MAC address in question will return its ATM address to the LES, which can then send that address back to the original requesting client.

Finally, if the MAC frame is a multicast or broadcast frame, the sending client transmits the frame to the BUS over the virtual data connection it has to the BUS. The bus then replicates that frame and sends it over virtual data connections to all of the clients on the emulated LAN.

14.4 RECOMMENDED READING

The definitive study of bridges is [PERL92]. [BERT92] and [SPRA91] examine the performance implications of transparent bridging and source routing.

LAN emulation is explained in [TRUO95] and [JEFF94].

BERT92 Bertsekas, D. and Gallager, R. *Data Networks*. Englewood Cliffs, NJ: Prentice Hall, 1992.

JEFF94 Jeffries, R. "ATM LAN Emulation: The Inside Story." *Data Communications*, September 21, 1994.

PERL92 Perlman, R. *Interconnections: Bridges and Routers*. Reading, MA: Addison Wesley, 1992.

SPRA91 Spragins, J., Hammond, J., and Pawlikowski, K. *Telecommunications Protocols and Design*. Reading, MA: Addison-Wesley, 1991.

TRUO95 Truong, H. et al. "LAN Emulation on an ATM Network." *IEEE Communications Magazine*, May 1995.

14.5 PROBLEMS

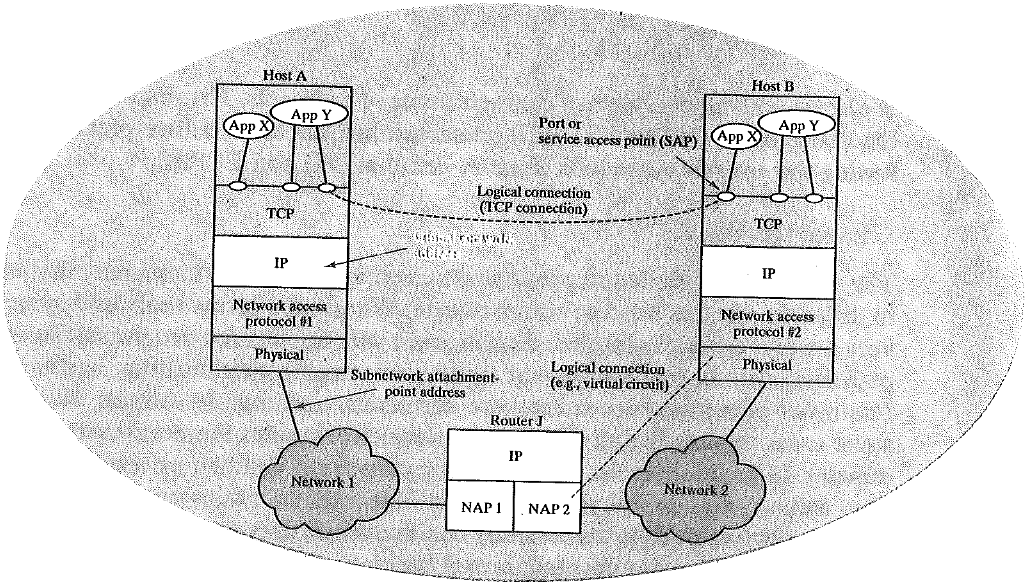
14.1 The token ring MAC protocol specifies that the A and C bits may be set by a station on the ring to indicate address recognized and frame copied, respectively. This information is then available to the source station when the frame returns after circulating around the ring. If a bridge captures a frame and forwards it, should it set the A and C bits or not? Make a case for each policy.

14.2 Draw a figure similar to Figure 14.14 that shows the protocol architecture for the interconnection of two end systems on separate LANs (with the same MAC protocol) across an ATM switch.

PART FOUR Communications Architecture and Protocols

CHAPTER 15

PROTOCOLS AND ARCHITECTURE


Host A
App X
App Y
TCP
IP
Network access protocol #1
Physical
Global network address
Logical connection (TCP connection)
Port or service access point (SAP)
Host B
App X
App Y
TCP
IP
Network access protocol #2
Physical
Subnetwork attachment-point address
Logical connection (e.g., virtual circuit)
Router J
IP
NAP 1
NAP 2
Network 1
Network 2

The purpose of this chapter is to serve as an overview and necessary background to the detailed material that follows in the remainder of Part Four. It will also serve to show how the concepts of Parts One through Three fit into the broader area of computer networks and computer communications.

We begin with an exposition of the concept of a communications protocol. It is shown that protocols are fundamental to all data communications. Next, we look at a way of systematically describing and implementing the communications function by viewing the communications task in terms of a column of layers, each of which contains protocols; this is the view of the now-famous Open Systems Interconnection (OSI) model.

Although the OSI model is almost universally accepted as the framework for discourse in this area, there is another model, known as the TCP/IP protocol architecture, which has gained commercial dominance at the expense of OSI. Most of the specific protocols described in Part Four are part of the TCP/IP suite of protocols. In this chapter, we provide an overview.

15.1 PROTOCOLS

We begin with an overview of characteristics of protocols. The reader should review the concepts of OSI and TCP/IP presented in Chapter 1 before proceeding. Following this overview, we look in more detail at OSI and TCP/IP.

Characteristics

The concepts of distributed processing and computer networking imply that entities in different systems need to communicate. We use the terms *entity* and *system* in a very general sense. Examples of entities are user application programs, file transfer packages, data base management systems, electronic mail facilities, and terminals. Examples of systems are computers, terminals, and remote sensors. Note that in some cases the entity and the system in which it resides are coextensive (e.g., terminals). In general, an entity is anything capable of sending or receiving information, and a system is a physically distinct object that contains one or more entities.

For two entities to successfully communicate, they must "speak the same language." What is communicated, how it is communicated, and when it is communicated must conform to some mutually acceptable set of conventions between the entities involved. The set of conventions is referred to as a protocol, which may be defined as a set of rules governing the exchange of data between two entities. The key elements of a protocol are

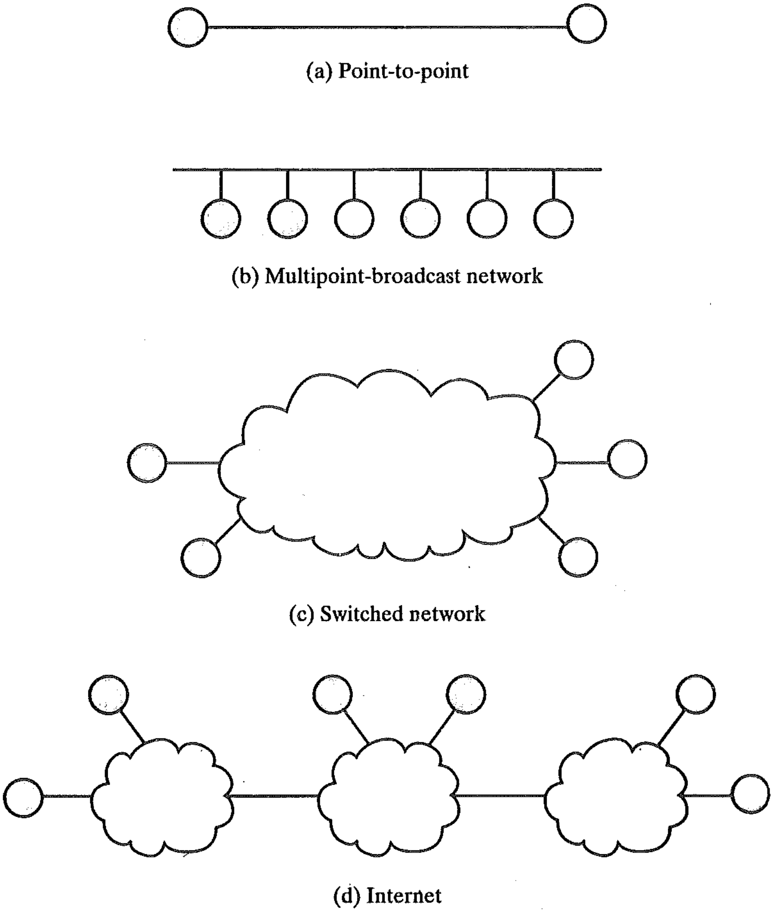
- **Syntax.** Includes such things as data format, coding, and signal levels.
- **Semantics.** Includes control information for coordination and error handling:
- **Timing.** Includes speed matching and sequencing.

HDLC is an example of a protocol. The data to be exchanged must be sent in frames of a specific format (syntax). The control field provides a variety of regulatory functions, such as setting a mode and establishing a connection (semantics). Provisions are also included for flow control (timing). Most of Part Four will be devoted to discussing other examples of protocols.

Some important characteristics of a protocol are

- Direct/indirect
- Monolithic/structured
- Symmetric/asymmetric
- Standard/nonstandard

Communication between two entities may be direct or indirect. Figure 15.1 depicts possible situations. If two systems share a point-to-point link, the entities in these systems may communicate directly; that is, data and control information pass directly between entities with no intervening active agent. The same may be said of



(a) Point-to-point

(b) Multipoint-broadcast network

(c) Switched network

(d) Internet

FIGURE 15.1 Means of connection of communicating systems.

a multipoint configuration, although here the entities must be concerned with the issue of access control, making the protocol more complex. If systems connect through a switched communication network, a direct protocol is no longer possible. The two entities must depend on the functioning of other entities to exchange data. A more extreme case is a situation in which two entities do not even share the same switched network, but are indirectly connected through two or more networks. A set of such interconnected networks is termed an internet.

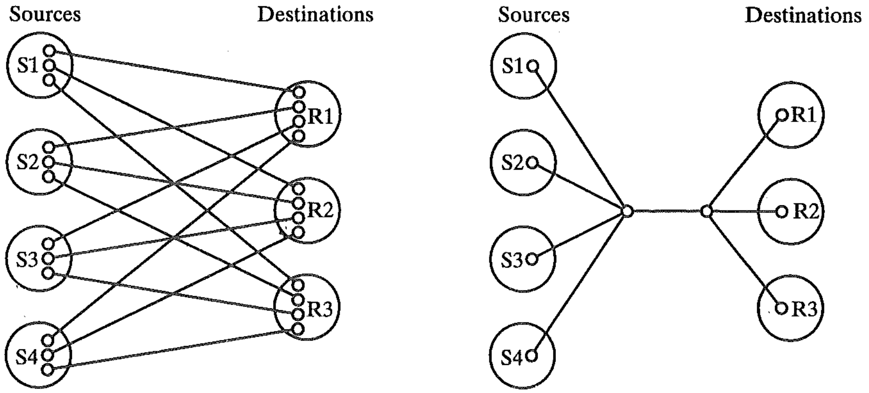
A protocol is either monolithic or structured. It should become clear as Part Four proceeds that the task of communication between entities on different systems is too complex to be handled as a unit. For example, consider an electronic mail package running on two computers connected by a synchronous HDLC link. To be truly monolithic, the package would need to include all of the HDLC logic. If the connection were over a packet-switched network, the package would still need the HDLC logic (or some equivalent) to attach to the network. It would also need logic for breaking up mail into packet-sized chunks, logic for requesting a virtual circuit, and so forth. Mail should only be sent when the destination system and entity are active and ready to receive; logic is needed for that kind of coordination, and, as we shall see, the list goes on. A change in any aspect means that this huge package must be modified, with the risk of introducing difficult-to-find bugs.

An alternative is to use structured design and implementation techniques. Instead of a single protocol, there is a set of protocols that exhibit a hierarchical or layered structure. Primitive functions are implemented in lower-level entities that provide services to higher-level entities. For example, there could be an HDLC module (entity) that is invoked by an electronic mail facility when needed, which is just another form of indirection; higher-level entities rely on lower-level entities to exchange data.

When structured protocol design is used, we refer to the hardware and software that implements the communications function as a communications architecture; the remainder of this chapter, after this section, is devoted to this concept.

A protocol may be either symmetric or asymmetric. Most of the protocols that we shall study are symmetric; that is, they involve communication between peer entities. Asymmetry may be dictated by the logic of an exchange (e.g., a client and a server process), or by the desire to keep one of the entities or systems as simple as possible. An example of the latter situation is the normal response mode of HDLC. Typically, this involves a computer that polls and selects a number of terminals. The logic on the terminal end is quite straightforward.

Finally, a protocol may be either standard or nonstandard. A nonstandard protocol is one built for a specific communications situation or, at most, a particular model of a computer. Thus, if K different kinds of information sources have to communicate with L types of information receivers, $K \times L$ different protocols are needed without standards and a total of $2 \times K \times L$ implementations are required (Figure 15.2a). If all systems shared a common protocol, only $K + L$ implementations would be needed (Figure 15.2b). The increasing use of distributed processing and the decreasing inclination of customers to remain locked into a single vendor dictate that all vendors implement protocols that conform to an agreed-upon standard.



(a) Without standards: 12 different protocols, 24 protocol implementations

(b) With standards: 1 protocol, 7 implementations

FIGURE 15.2 The use of standard protocols.

Functions

Before turning to a discussion of communications architecture and the various levels of protocols, let us consider a rather small set of functions that form the basis of all protocols. Not all protocols have all functions; this would involve a significant duplication of effort. There are, nevertheless, many instances of the same type of function being present in protocols at different levels.

This discussion will, of necessity, be rather abstract; it does, however, provide an integrated overview of the characteristics and functions of communications protocols. The concept of protocol is fundamental to all of the remainder of Part Four, and, as we proceed, specific examples of all these functions will be seen.

We can group protocol functions into the following categories:

- Segmentation and reassembly
- Encapsulation
- Connection control
- Ordered delivery
- Flow control
- Error control
- Addressing
- Multiplexing
- Transmission services

Segmentation and Reassembly[1]

A protocol is concerned with exchanging streams of data between two entities. Usually, the transfer can be characterized as consisting of a sequence of blocks of data

[1] In most protocol specifications related to the TCP/IP protocol suite, the term fragmentation rather than segmentation is used. The meaning is the same.

of some bounded size. At the application level, we refer to a logical unit of data transfer as a message. Now, whether the application entity sends data in messages or in a continuous stream, lower-level protocols may need to break the data up into blocks of some smaller bounded size; this process is called segmentation. For convenience, we shall refer to a block of data exchanged between two entities via a protocol as a protocol data unit (PDU).

There are a number of motivations for segmentation, depending on the context. Among the typical reasons for segmentation are

- The communications network may only accept blocks of data up to a certain size. For example, an ATM network is limited to blocks of 53 octets; Ethernet imposes a maximum size of 1526 cctets.
- Error control may be more efficient with a smaller PDU size. For example, fewer bits need to be retransmitted using smaller blocks with the selective repeat technique.
- More equitable access to shared transmission facilities, with shorter delay, can be provided. For example, without a maximum block size, one station could monopolize a multipoint medium.
- A smaller PDU size may mean that receiving entities can allocate smaller buffers.
- An entity may require that data transfer comes to some sort of closure from time to time, for checkpoint and restart/recovery operations.

There are several disadvantages to segmentation that argue for making blocks as large as possible:

- Each PDU, as we shall see, contains a fixed minimum amount of control information. Hence, the smaller the block, the greater the percentage overhead.
- PDU arrival may generate an interrupt that must be serviced. Smaller blocks result in more interrupts.
- More time is spent processing smaller, more numerous PDUs.

All of these factors must be taken into account by the protocol designer in determining minimum and maximum PDU size.

The counterpart of segmentation is reassembly. Eventually, the segmented data must be reassembled into messages appropriate to the application level. If PDUs arrive out of order, the task is complicated.

The process of segmentation was illustrated in Figure 1.7.

Encapsulation

Each PDU contains not only data but control information. Indeed, some PDUs consist solely of control information and no data. The control information falls into three general categories:

- **Address.** The address of the sender and/or receiver may be indicated.
- **Error-detecting code.** Some sort of frame check sequence is often included for error detection.

- **Protocol control.** Additional information is included to implement the protocol functions listed in the remainder of this section.

The addition of control information to data is referred to as encapsulation. Data are accepted or generated by an entity and encapsulated into a PDU containing that data plus control information (See Figures 1.7 and 1.8). An example of this is the HDLC frame (Figure 6.10).

Connection Control

An entity may transmit data to another entity in such a way that each PDU is treated independently of all prior PDUs. This process is known as connectionless data transfer; an example is the use of the datagram. While this mode is useful, an equally important technique is connection-oriented data transfer, of which the virtual circuit is an example.

Connection-oriented data transfer is to be preferred (even required) if stations anticipate a lengthy exchange of data and/or certain details of their protocol must be worked out dynamically. A logical association, or connection, is established between the entities. Three phases occur (Figure 15.3):

- Connection establishment
- Data transfer
- Connection termination

With more sophisticated protocols, there may also be connection interrupt and recovery phases to cope with errors and other sorts of interruptions.

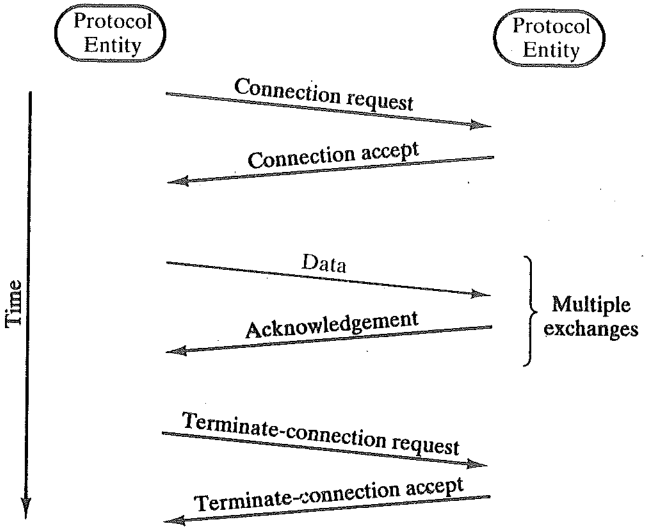


FIGURE 15.3 The phases of a connection-oriented data transfer.

During the connection establishment phase, two entities agree to exchange data. Typically, one station will issue a connection request (in connectionless fashion!) to the other. A central authority may or may not be involved. In simpler protocols, the receiving entity either accepts or rejects the request and, in the former case, away they go. In more complex proposals, this phase includes a negotiation concerning the syntax, semantics, and timing of the protocol. Both entities must, of course, be using the same protocol. But the protocol may allow certain optional features, and these must be agreed upon by means of negotiation. For example, the protocol may specify a PDU size of up to 8000 octets; one station may wish to restrict this to 1000 octets.

Following connection establishment, the data transfer phase is entered; here, both data and control information (e.g., flow control, error control) are exchanged. The figure shows a situation in which all of the data flows in one direction, with acknowledgments returned in the other direction. More typically, data and acknowledgments flow in both directions. Finally, one side or the other wishes to terminate the connection and does so by sending a termination request. Alternatively, a central authority might forcibly terminate a connection.

The key characteristic of connection-oriented data transfer is that sequencing is used. Each side sequentially numbers the PDUs that it sends to the other side. Because each side remembers that it is engaged in a logical connection, it can keep track of both outgoing numbers, which it generates, and incoming numbers, which are generated by the other side. Indeed, one can essentially define a connection-oriented data transfer as one in which both sides number PDUs and keep track of the incoming and outgoing numbers. Sequencing supports three main functions: ordered deliver, flow control, and error control.

Ordered Delivery

If two communicating entities are in different hosts connected by a network, there is a risk that PDUs will not arrive in the order in which they were sent, because they may traverse different paths through the network. In connection-oriented protocols, it is generally required that PDU order be maintained. For example, if a file is transferred between two systems, we would like to be assured that the records of the received file are in the same order as those of the transmitted file, and not shuffled. If each PDU is given a unique number, and numbers are assigned sequentially, then it is a logically simple task for the receiving entity to reorder received PDUs on the basis of sequence number. A problem with this scheme is that, with a finite sequence number field, sequence numbers repeat (modulo some maximum number). Evidently, the maximum sequence number must be greater than the maximum number of PDUs that could be outstanding at any time. In fact, the maximum number may need to be twice the maximum number of PDUs that could be outstanding (e.g., selective-repeat ARQ; see Chapter 6).

Flow Control

Flow control was introduced in Chapter 6. In essence, flow control is a function performed by a receiving entity to limit the amount or rate of data that is sent by a transmitting entity.

The simplest form of flow control is a stop-and-wait procedure, in which each PDU must be acknowledged before the next can be sent. More efficient protocols involve some form of credit provided to the transmitter, which is the amount of data that can be sent without an acknowledgment. The sliding-window technique is an example of this mechanism.

Flow control is a good example of a function that must be implemented in several protocols. Consider again Figure 1.6. The network will need to exercise flow control over station 1's network services module via the network access protocol, in order to enforce network traffic control. At the same time, station 2's network services module has only limited buffer space and needs to exercise flow control over station 1's network services module via the process-to-process protocol. Finally, even though station 2's network service module can control its data flow, station 2's application may be vulnerable to overflow. For example, the application could be hung up waiting for disk access. Thus, flow control is also needed over the application-oriented protocol.

Error Control

Another previously introduced function is error control. Techniques are needed to guard against loss or damage of data and control information. Most techniques involve error detection, based on a frame check sequence, and PDU retransmission. Retransmission is often activated by a timer. If a sending entity fails to receive an acknowledgment to a PDU within a specified period of time, it will retransmit.

As with flow control, error control is a function that must be performed at various levels of protocol. Consider again Figure 1.6. The network access protocol should include error control to assure that data are successfully exchanged between station and network. However, a packet of data may be lost inside the network, and the process-to-process protocol should be able to recover from this loss.

Addressing

The concept of addressing in a communications architecture is a complex one and covers a number of issues. At least four separate issues need to be discussed:

- Addressing level
- Addressing scope
- Connection identifiers
- Addressing mode

During the discussion, we illustrate the concepts using Figure 15.4, which shows a configuration using the TCP/IP protocol architecture. The concepts are essentially the same for the OSI architecture or any other communications architecture.

Addressing level refers to the level in the communications architecture at which an entity is named. Typically, a unique address is associated with each end system (e.g., host or terminal) and each intermediate system (e.g., router) in a configuration. Such an address is, in general, a network-level address. In the case of the TCP/IP architecture, this is referred to as an IP address, or simply an internet

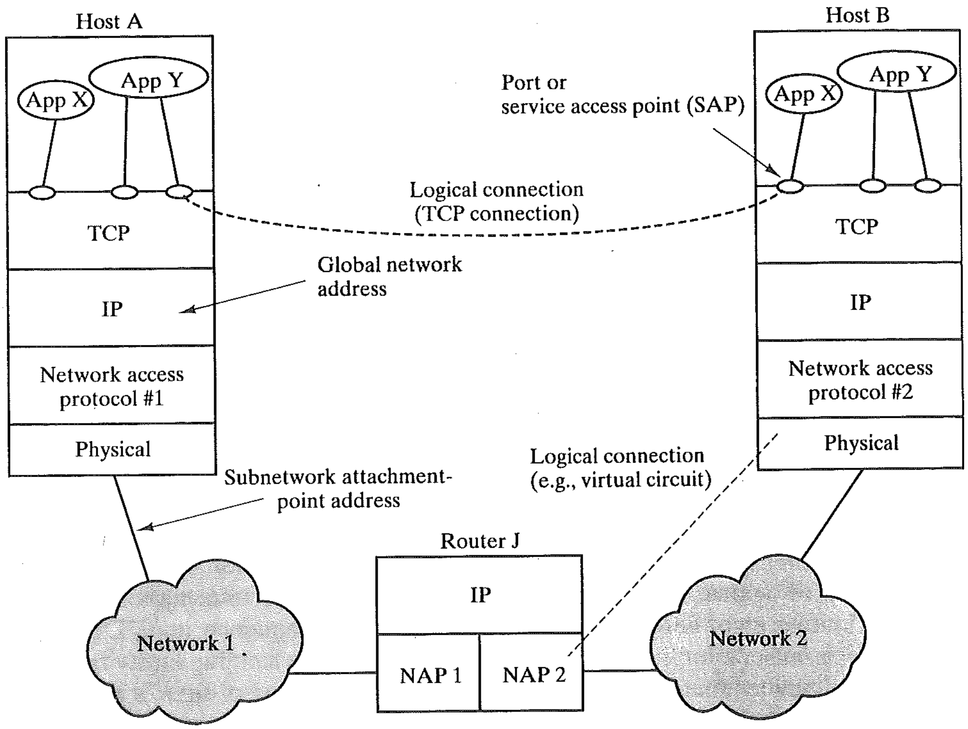

Host A
App X
App Y
TCP
IP
Network access protocol #1
Physical
Host B
App X
App Y
Port or service access point (SAP)
Logical connection (TCP connection)
TCP
IP
Network access protocol #2
Physical
Global network address
Subnetwork attachment-point address
Logical connection (e.g., virtual circuit)
Router J
IP
NAP 1
NAP 2
Network 1
Network 2

FIGURE 15.4 Addressing concepts.

address. In the case of the OSI architecture, this is referred to as a network service access point (NSAP). The network-level address is used to route a PDU through a network or networks to a system indicated by a network-level address in the PDU.

Once data arrives at a destination system, it must be routed to some process or application in the system. Typically, a system will support multiple applications, and an application may support multiple users. Each application and, perhaps, each concurrent user of an application, is assigned a unique identifier, referred to as a port in the TCP/IP architecture, and as a service access point (SAP) in the OSI architecture. For example, a host system might support both an electronic mail application and a file transfer application. At minimum, each application would have a port number or SAP that is unique within that system. Further, the file transfer application might support multiple simultaneous transfers, in which case, each transfer is dynamically assigned a unique port number or SAP.

Figure 15.4 illustrates two levels of addressing within a system; this is typically the case for the TCP/IP architecture. However, there can be addressing at each level of an architecture. For example, a unique SAP can be assigned to each level of the OSI architecture.

Another issue that relates to the address of an end system or intermediate system is *addressing scope.* The internet address or NSAP address referred to above is a global address. The key characteristics of a global address are

- **Global nonambiguity.** A global address identifies a unique system. Synonyms are permitted. That is, a system may have more than one global address.
- **Global applicability.** It is possible at any global address to identify any other global address, in any system, by means of the global address of the other system.

Because a global address is unique and globally applicable, it enables an internet to route data from any system attached to any network to any other system attached to any other network.

Figure 15.4 illustrates that another level of addressing may be required. Each subnetwork must maintain a unique address for each device interface on the subnetwork. Examples are a MAC address on an IEEE 802 network and an X.25 DTE address, both of which enable the subnetwork to route data units (e.g., MAC frames, X.25 packets) through the subnetwork and deliver them to the intended attached system. We can refer to such an address as a *subnetwork attachment-point address.*

The issue of addressing scope is generally only relevant for network-level addresses. A port or SAP above the network level is unique within a given system but need not be globally unique. For example, in Figure 15.4, there can be a port 1 in system A and a port 1 in system B. The full designation of these two ports could be expressed as A.1 and B.1, which are unique designations.

The concept of *connection identifiers* comes into play when we consider connection-oriented data transfer (e.g., virtual circuit) rather than connectionless data transfer (e.g., datagram). For connectionless data transfer, a global name is used with each data transmission. For connection-oriented transfer, it is sometimes desirable to use only a connection name during the data transfer phase. The scenario is this: Entity 1 on system A requests a connection to entity 2 on system B, perhaps using the global address B.2. When B.2 accepts the connection, a connection name (usually a number) is provided and is used by both entities for future transmissions. The use of a connection name has several advantages:

- **Reduced overhead.** Connection names are generally shorter than global names. For example, in the X.25 protocol (discussed in Chapter 9) used over packet-switched networks, connection-request packets contain both source and destination address fields, each with a system-defined length that may be a number of octets. After a virtual circuit is established, data packets contain just a 12-bit virtual circuit number.
- **Routing.** In setting up a connection, a fixed route may be defined. The connection name serves to identify the route to intermediate systems, such as packet-switching nodes, for handling future PDUs.
- **Multiplexing.** We address this function in more general terms below. Here we note that an entity may wish to enjoy more than one connection simultaneously. Thus, incoming PDUs must be identified by connection name.
- **Use of state information.** Once a connection is established, the end systems can maintain state information relating to the connection; this enables such functions as flow control and error control using sequence numbers, as we saw with HDLC (Chapter 6) and X.25 (Chapter 9).

Figure 15.4 shows several examples of connections. The logical connection between router J and host B is at the network level. For example, if network 2 is a packet-switching network using X.25, then this logical connection would be a virtual circuit. At a higher level, many transport-level protocols, such as TCP, support logical connections between users of the transport service. Thus, TCP can maintain a connection between two ports on different systems.

Another addressing concept is that of *addressing mode.* Most commonly, an address refers to a single system or port; in this case, it is referred to as an individual or *unicast address*. It is also possible for an address to refer to more than one entity or port. Such an address identifies multiple simultaneous recipients for data. For example, a user might wish to send a memo to a number of individuals. The network control center may wish to notify all users that the network is going down. An address for multiple recipients may be *broadcast*—intended for all entities within a domain—or *multicast*—intended for a specific subset of entities. Table 15.1 illustrates the possibilities.

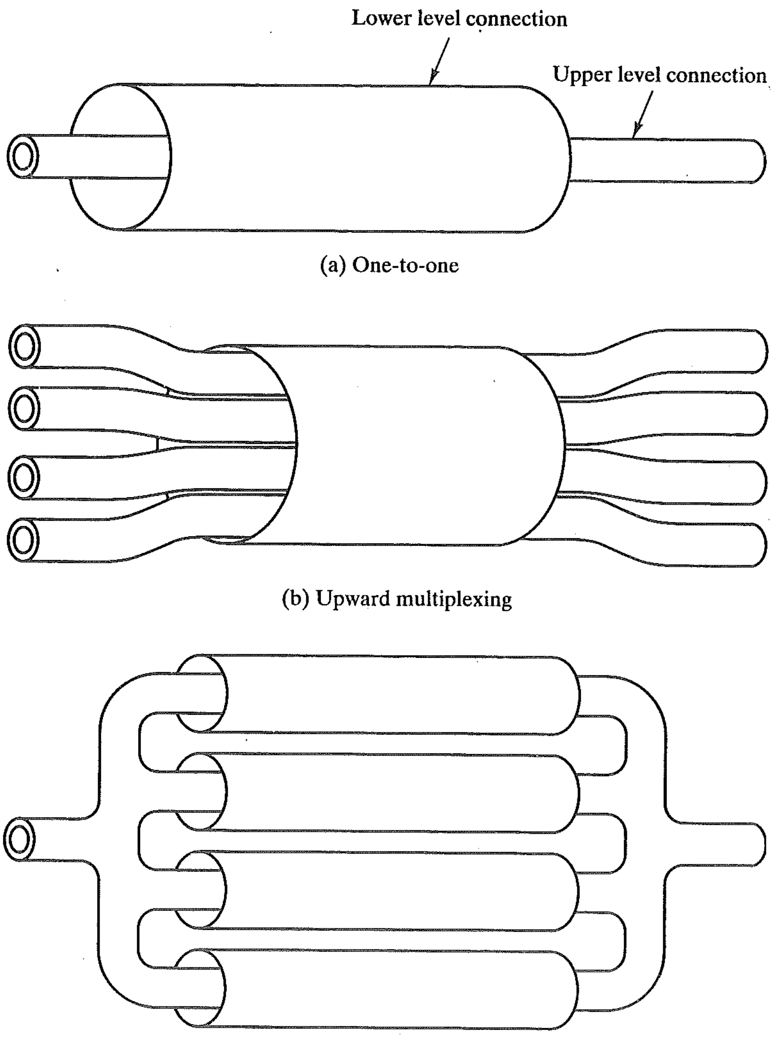
Multiplexing

Related to the concept of addressing is that of multiplexing. One form of multiplexing is supported by means of multiple connections into a single system. For example, with X.25, there can be multiple virtual circuits terminating in a single end system; we can say that these virtual circuits are multiplexed over the single physical interface between the end system and the network. Multiplexing can also be accomplished via port names, which also permit multiple simultaneous connections. For example, there can be multiple TCP connections terminating in a given system, each connection supporting a different pair of ports.

Multiplexing is used in another context as well, namely the mapping of connections from one level to another. Consider again Figure 15.4. Network A might provide a virtual circuit service. For each process-to-process connection established at the network services level, a virtual circuit could be created at the network access level. This is a one-to-one relationship, but need not be so. Multiplexing can be used in one of two directions (Figure 15.5). Upward multiplexing occurs when multiple higher-level connections are multiplexed on, or share, a single lower-level connec-

TABLE 15.1 Addressing modes.

Destination	Network address	System address	Port/SAP address
Unicast	Individual	Individual	Individual
Multicast	Individual Individual All	Individual All All	Group Group Group
Broadcast	Individual Individual All	Individual All All	All All All



(a) One-to-one

(b) Upward multiplexing

(c) Downward multiplexing

FIGURE 15.5 Multiplexing and protocol connections.

tion in order to make more efficient use of the lower-level service or to provide several higher-level connections in an environment where only a single lower-level connection exists. Figure 15.5 shows an example of upward multiplexing. Downward multiplexing, or splitting, means that a single higher-level connection is built on top of multiple lower-level connections, the traffic on the higher connection being divided among the various lower connections. This technique may be used to provide reliability, performance, or efficiency.

Transmission Services

A protocol may provide a variety of additional services to the entities that use it. We mention here three common examples:

- **Priority.** Certain messages, such as control messages, may need to get through to the destination entity with minimum delay. An example would be a close-connection request. Thus, priority could be assigned on a per-message basis. Additionally, priority could be assigned on a per-connection basis.
- **Grade of service.** Certain classes of data may require a minimum throughput or a maximum delay threshold.
- **Security.** Security mechanisms, restricting access, may be invoked.

All of these services depend on the underlying transmission system and on any intervening lower-level entities. If it is possible for these services to be provided from below, the protocol can be used by the two entities to exercise such services.

15.2 OSI

As discussed in Chapter 1, standards are needed to promote interoperability among vendor equipment and to encourage economies of scale. Because of the complexity of the communications task, no single standard will suffice. Rather, the functions should be broken down into more manageable parts and organized as a communications architecture, which would then form the framework for standardization.

This line of reasoning led ISO in 1977 to establish a subcommittee to develop such an architecture. The result was the Open Systems Interconnection (OSI) reference model. Although the essential elements of the model were put into place quickly, the final ISO standard, ISO 7498, was not published until 1984. A technically compatible version was issued by CCITT as X.200 (now ITU-T).

The Model

A widely accepted structuring technique, and the one chosen by ISO, is layering. The communications functions are partitioned into a hierarchical set of layers. Each layer performs a related subset of the functions required to communicate with another system, relying on the next-lower layer to perform more primitive functions, and to conceal the details of those functions, as it provides services to the next-higher layer. Ideally, the layers should be defined so that changes in one layer do not require changes in the other layers. Thus, we have decomposed one problem into a number of more manageable subproblems.

The task of ISO was to define a set of layers and to delineate the services performed by each layer. The partitioning should group functions logically, and should have enough layers to make each one manageably small, but should not have so many layers that the processing overhead imposed by their collection is burdensome. The principles that guided the design effort are summarized in Table 15.2. The resulting reference model has seven layers, which are listed with a brief defin-

TABLE 15.2 Principles used in defining the OSI layers (ISO 7498).

1. Do not create so many layers as to make the system engineering task of describing and integrating the layers more difficult than necessary.
2. Create a boundary at a point where the description of services can be small and the number of interactions across the boundary are minimized.
3. Create separate layers to handle functions that are manifestly different in the process performed or the technology involved.
4. Collect similar functions into the same layer.
5. Select boundaries at a point which past experience has demonstrated to be succesful.
6. Create a layer of easily localized functions so that the layer could be totally redesigned and its protocols changed in a major way to take advantage of new advances in architecture, hardware or software technology without changing the services expected from and provided to the adjacent layers.
7. Create a boundary where it may be useful at some point in time to have the corresponding interface standardized.[1,2]
8. Create a layer where there is a need for a different level of abstraction in the handling of data, for example morphology, syntax, semantic.
9. Allow changes of functions or protocols to be made within a layer without affecting other layers.
10. Create for each layer boundaries with its upper and lower layer only.

Similar principles have been applied to sublayering:

11. Create further subgrouping and organization of functions to form sublayers within a layer in cases where distinct communication services need it.
12. Create, where needed, two or more sublayers with a common, and therefore minimal functionality to allow interface operation with adjacent layers.
13. Allow by-passing of sublayers.

[1] Advantages and drawbacks of standardizing internal interfaces within open systems are not considered in this International Standard. In particular, mention of, or references to principle (7) should not be taken to imply usefulness of standards for such internal interfaces.

[2] It is important to note that OSI *per se* does not require interfaces within open systems to be standardized. Moreover, whenever standards for such interfaces are defined, adherence to such internal interface standards can in no way be considered as a condition of openness.

ition in Figure 1.10. Table 15.3 provides ISO's justification for the selection of these layers.

Figure 15.6 illustrates the OSI architecture. Each system contains the seven layers. Communication is between applications in the two computers, labeled application X and application Y in the figure. If application X wishes to send a message to application Y, it invokes the application layer (layer 7). Layer 7 establishes a peer relationship with layer 7 of the target computer, using a layer-7 protocol (application protocol). This protocol requires services from layer 6, so the two layer-6 entities use a protocol of their own, and so on down to the physical layer, which actually transmits bits over a transmission medium.

Note that there is no direct communication between peer layers except at the physical layer. That is, above the physical layer, each protocol entity sends data down to the next-lower layer to get the data across to its peer entity. Even at the physical layer, the OSI model does not stipulate that two systems be directly connected. For example, a packet-switched or circuit-switched network may be used to provide the communication link.

TABLE 15.3 Justification of the OSI layers (ISO 7498).

1. It is essential that the architecture permits usage of a realistic variety of physical media for interconnection with different control procedures (for example V.24, V.25, etc . . .). Application of principles 3, 5, and 8 (Table 15.2) leads to identification of a **Physical Layer** as the lowest layer in the architecture.
2. Some physical communication media (for example telephone line) require specific techniques to be used in order to transmit data between systems despite a relatively high error rate (i.e., an error rate not acceptable for the great majority of applications). These specific techniques are used in data-link control procedures which have been studied and standardized for a number of years. It must also be recognized that new physical communication media (for example fiber optics) will require different data-link control procedures. Application of principles 3, 5, and 8 leads to identification of a **Data Link Layer** on top of the Physical Layer in the architecture.
3. In the open systems architecture, some open systems will act as the final destination of data. Some open systems may act only as intermediate nodes (forwarding data to other systems). Application of principles 3, 5, and 7 leads to identification of a **Network Layer** on top of the data link layer. Network oriented protocols such as routing, for example, will be grouped in this layer. Thus, the Network Layer will provide a connection path (network-connection) between a pair of transport entities; including the case where intermediate nodes are involved.
4. Control of data transportation from source end open system to destination end open system (which is not performed in intermediate nodes) is the last function to be performed in order to provide the totality of the transport service. Thus, the upper layer in the transport service part of the architecture is the **Transport Layer**, on top of the Network Layer. This Transport Layer relieves higher layer entities from any concern with the transportation of data between them.
5. There is a need to organize and synchronize dialogue, and to manage the exchange of data. Application of principles 3 and 4 leads to the identification of a **Session Layer** on top of the Transport Layer.
6. The remaining set of general interests functions are those related to representation and manipulation of structured data for the benefit of application programs. Application of principles 3 and 4 leads to the identification of a **Presentation Layer** on top of the Session Layer.
7. Finally, there are applications consisting of application processes which perform information processing. An aspect of these application processes and the protocols by which they communicate comprise the **Application Layer** as the highest layer of the architecture.

Figure 15.6 also highlights the use of protocol data units (PDUs) within the OSI architecture. First, consider the most common way in which protocols are realized. When application X has a message to send to application Y, it transfers those data to an application entity in the application layer. A header is appended to the data that contains the required information for the peer–layer-7 protocol (encapsulation). The original data, plus the header, are now passed as a unit to layer 6. The presentation entity treats the whole unit as data and appends its own header (a second encapsulation). This process continues down through layer 2, which generally adds both a header and a trailer (e.g., HDLC). This layer-2 unit, called a *frame*, is then passed by the physical layer onto the transmission medium. When the frame is received by the target system, the reverse process occurs. As the data ascend, each layer strips off the outermost header, acts on the protocol information contained therein, and passes the remainder up to the next layer.

At each stage of the process, a layer may fragment the data unit it receives from the next-higher layer into several parts, in order to accommodate its own requirements. These data units must then be reassembled by the corresponding peer layer before being passed up.

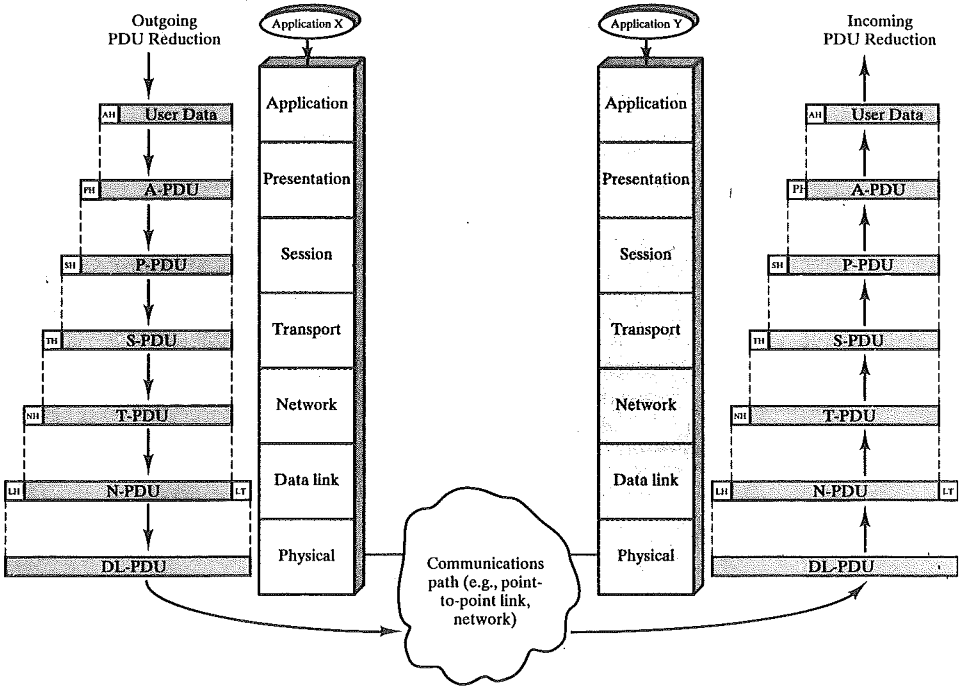


FIGURE 15.6 The OSI environment.

Standardization Within the OSI Framework

The principal motivation for the development of the OSI model was to provide a framework for standardization. Within the model, one or more protocol standards can be developed at each layer. The model defines, in general terms, the functions to be performed at that layer and facilitates the standards-making process in two ways:

- Because the functions of each layer are well-defined, standards can be developed independently and simultaneously for each layer, thereby speeding 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.

Figure 15.7 illustrates the use of the OSI model as such a framework. The overall communications function is decomposed into seven distinct layers, using the principles outlined in Table 15.2. These principles essentially amount to the use of modular design. That is, the overall function is broken up into a number of modules, making the interfaces between modules as simple as possible. In addition, the design principle of information-hiding is used: Lower layers are concerned with greater levels of detail; upper layers are independent of these details. Within each

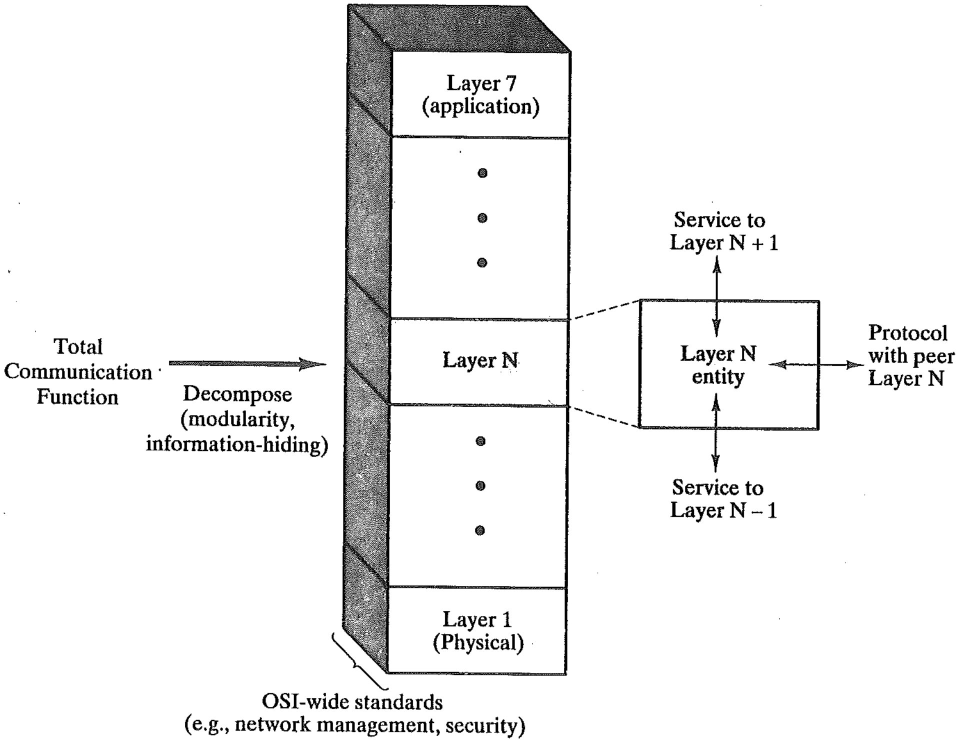


FIGURE 15.7 The OSI architecture as a framework for standardization.

layer, there exist both the service provided to the next higher layer and the protocol to the peer layer in other systems.

Figure 15.8 shows more specifically the nature of the standardization required at each layer. Three elements are key:

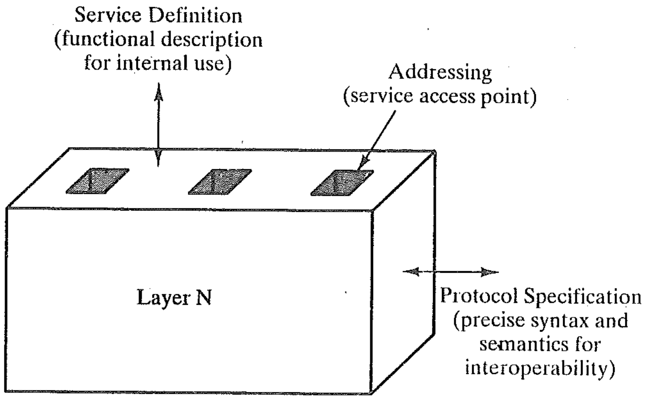


FIGURE 15.8 Layer-specific standards.

- **Protocol specification.** Two entities at the same layer in different systems cooperate and interact by means of a protocol. Because two different open systems are involved, the protocol must be specified precisely; this includes the format of the protocol data units exchanged, the semantics of all fields, and the allowable sequence of PDUs.
- **Service definition.** In addition to the protocol or protocols that operate at a given layer, standards are needed for the services that each layer provides to the next-higher layer. Typically, the definition of services is equivalent to a functional description that defines *what* services are provided, but not *how* the services are to be provided.
- **Addressing.** Each layer provides services to entities at the next-higher layer. These entities are referenced by means of a service access point (SAP). Thus, a network service access point (NSAP) indicates a transport entity that is a user of the network service.

The need to provide a precise protocol specification for open systems is self-evident. The other two items listed above warrant further comment. With respect to service definitions, the motivation for providing only a functional definition is as follows. First, the interaction between two adjacent layers takes places within the confines of a single open system and is not the concern of any other open system. Thus, as long as peer layers in different systems provide the same services to their next-higher layers, the details of how the services are provided may differ from one system to another without loss of interoperability. Second, it will usually be the case that adjacent layers are implemented on the same processor. In that case, we would like to leave the system programmer free to exploit the hardware and operating system to provide an interface that is as efficient as possible.

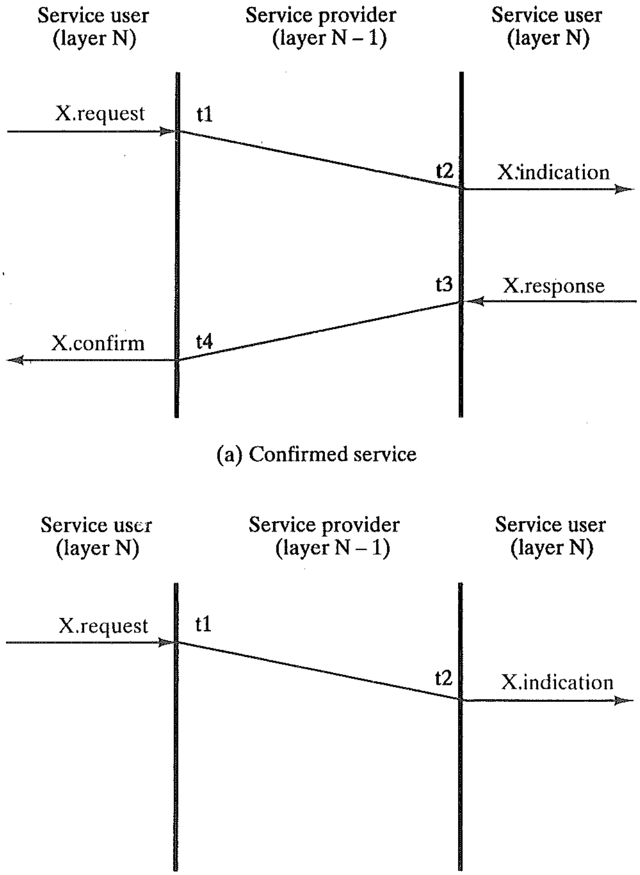
Service Primitives and Parameters

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 a parameter is 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 (X.210). These are defined in Table 15.4. The layout of Figure 15.9a suggests the time ordering of these events. For

TABLE 15.4 Service primitive types.

REQUEST	A primitive issued by a service user to invoke some service and to pass the parameters needed to fully specify the requested service.
INDICATION	A primitive issued by a service provider to either 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.



(a) Confirmed service

(b) Nonconfirmed service

FIGURE 15.9 Time sequence diagrams for service primitives.

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 DATA.request primitive. Associated with the primitive are the needed parameters, 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 a DATA.indication, which includes the data and source address as parameters.
4. If an acknowledgment is called for, the destination (N) entity issues a DATA.response 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 DATA.confirm.

This sequence of events is referred to as a *confirmed service*, as the initiator receives confirmation that the requested service has had the desired effect at the other end. If only request and indication primitives are involved (corresponding to steps 1 through 3), then the service dialogue is a *nonconfirmed service*; the initiator receives no confirmation that the requested action has taken place (Figure 15.9b).

The OSI Layers

In this section, we discuss briefly each of the layers and, where appropriate, give examples of standards for protocols at those layers.

Physical Layer

The physical layer covers the physical interface between devices and the rules by which bits are passed from one to another. The physical layer has four important characteristics:

- **Mechanical.** Relates to the physical properties of the interface to a transmission medium. Typically, the specification is of a pluggable connector that joins one or more signal conductors, called *circuits*.
- **Electrical.** Relates to the representation of bits (e.g., in terms of voltage levels) and the data transmission rate of bits.
- **Functional.** Specifies the functions performed by individual circuits of the physical interface between a system and the transmission medium.
- **Procedural.** Specifies the sequence of events by which bit streams are exchanged across the physical medium.

We have already covered physical layer protocols in some detail in Section 5.3. Examples of standards at this layer are EIA-232-E, as well as portions of ISDN and LAN standards.

Data Link Layer

Whereas the physical layer provides only a raw bit-stream service, the data link layer attempts to make the physical link reliable while providing the means to activate, maintain, and deactivate the link. The principal service provided by the data link layer to higher layers is that of error detection and control. Thus, with a fully functional data-link-layer protocol, the next higher layer may assume error-free transmission over the link. However, if communication is between two systems that are not directly connected, the connection will comprise a number of data links in tandem, each functioning independently. Thus, the higher layers are not relieved of any error control responsibility.

Chapter 6 was devoted to data link protocols; examples of standards at this layer are HDLC, LAPB, LLC, and LAPD.

Network Layer

The network layer provides for the transfer of information between end systems across some sort of communications network. It relieves higher layers of the need to know anything about the underlying data transmission and switching technologies used to connect systems. At this layer, the computer system engages in a dialogue with the network to specify the destination address and to request certain network facilities, such as priority.

There is a spectrum of possibilities for intervening communications facilities to be managed by the network layer. At one extreme, there is a direct point-to-point link between stations. In this case, there may be no need for a network layer because the data link layer can perform the necessary function of managing the link.

Next, the systems could be connected across a single network, such as a circuit-switching or packet-switching network. As an example, the packet level of the X.25 standard is a network layer standard for this situation. Figure 15.10 shows how the presence of a network is accommodated by the OSI architecture. The lower three layers are concerned with attaching to and communicating with the network. The packets created by the end system pass through one or more network nodes

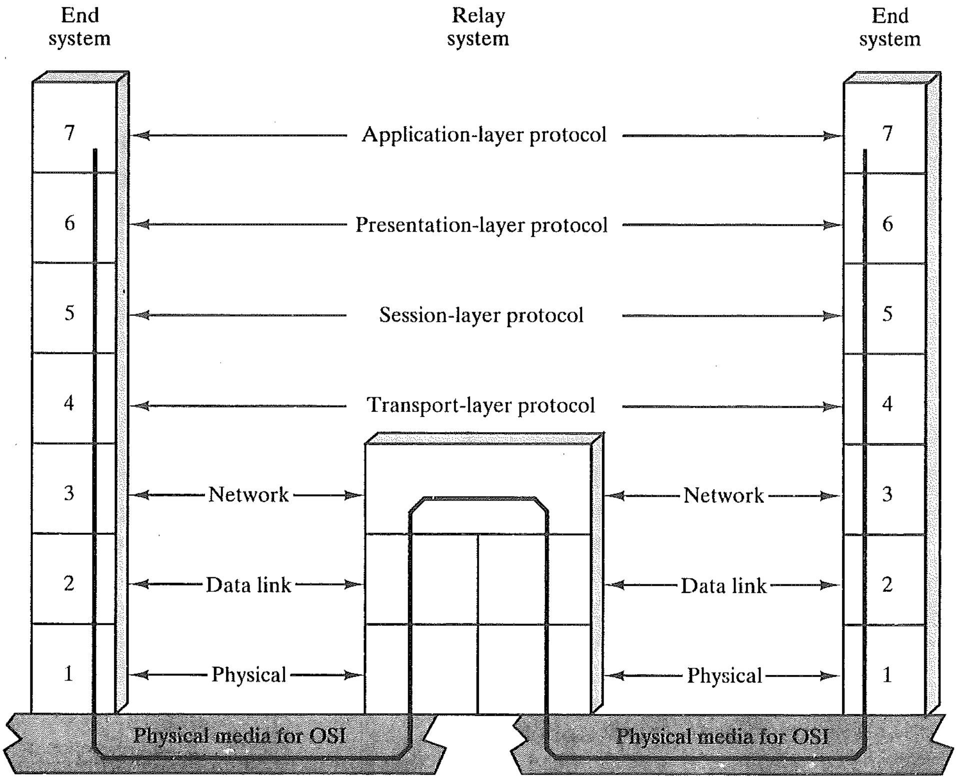


FIGURE 15.10 The use of a relay.

that act as relays between the two end systems. The network nodes implement layers 1–3 of the architecture. In the figure, two end systems are connected through a single network node. Layer 3 in the node performs a switching and routing function. Within the node, there are two data link layers and two physical layers, corresponding to the links to the two end systems. Each data link (and physical) layer operates independently to provide service to the network layer over its respective link. The upper four layers are *end-to-end* protocols between the attached end systems.

At the other extreme, two end systems might wish to communicate but are not even connected to the same network. Rather, they are connected to networks that, directly or indirectly, are connected to each other. This case requires the use of some sort of internetworking technique; we explore this approach in Chapter 16.

Transport Layer

The transport layer provides a mechanism for the exchange of data between end systems. The connection-oriented transport service ensures that data are delivered error-free, in sequence, with no losses or duplications. The transport layer may also be concerned with optimizing the use of network services and with providing a requested quality of service to session entities. For example, the session entity may specify acceptable error rates, maximum delay, priority, and security.

The size and complexity of a transport protocol depend on how reliable or unreliable the underlying network and network layer services are. Accordingly, ISO has developed a family of five transport protocol standards, each oriented toward a different underlying service. In the TCP/IP protocol suite, there are two common transport-layer protocols: the connection-oriented TCP (transmission control protocol) and the connectionless UDP (user datagram protocol).

Session Layer

The lowest four layers of the OSI model provide the means for the reliable exchange of data and provide an expedited data service. For many applications, this basic service is insufficient. For example, a remote terminal access application might require a half-duplex dialogue. A transaction-processing application might require checkpoints in the data-transfer stream to permit backup and recovery. A message-processing application might require the ability to interrupt a dialogue in order to prepare a new portion of a message and later to resume the dialogue where it was left off.

All these capabilities could be embedded in specific applications at layer 7. However, because these types of dialogue-structuring tools have widespread applicability, it makes sense to organize them into a separate layer: the session layer.

The session layer provides the mechanism for controlling the dialogue between applications in end systems. In many cases, there will be little or no need for session-layer services, but for some applications, such services are used. The key services provided by the session layer include

- **Dialogue discipline.** This can be two-way simultaneous (full duplex) or two-way alternate (half duplex).

- **Grouping.** The flow of data can be marked to define groups of data. For example, if a retail store is transmitting sales data to a regional office, the data can be marked to indicate the end of the sales data for each department; this would signal the host computer to finalize running totals for that department and start new running counts for the next department.
- **Recovery.** The session layer can provide a checkpointing mechanism, so that if a failure of some sort occurs between checkpoints, the session entity can retransmit all data since the last checkpoint.

ISO has issued a standard for the session layer that includes, as options, services such as those just described.

Presentation Layer

The presentation layer defines the format of the data to be exchanged between applications and offers application programs a set of data transformation services. The presentation layer also defines the syntax used between application entities and provides for the selection and subsequent modification of the representation used. Examples of specific services that may be performed at this layer include data compression and encryption.

Application Layer

The application layer provides a means for application programs to access the OSI environment. This layer contains management functions and generally useful mechanisms that support distributed applications. In addition, general-purpose applications such as file transfer, electronic mail, and terminal access to remote computers are considered to reside at this layer.

15.3 TCP/IP PROTOCOL SUITE

For many years, the technical literature on protocol architectures was dominated by discussions related to OSI and to the development of protocols and services at each layer. Throughout the 1980s, the belief was widespread that OSI would come to dominate commercially, both over architectures such as IBM's SNA, as well as over competing multivendor schemes such as TCP/IP; this promise was never realized. In the 1990s, TCP/IP has become firmly established as the dominant commercial architecture and as the protocol suite upon which the bulk of new protocol development is to be done.

There are a number of reasons for the success of the TCP/IP protocols over OSI:

1. TCP/IP protocols were specified, and enjoyed extensive use, prior to ISO standardization of alternative protocols. Thus, organizations in the 1980s with an immediate need were faced with the choice of waiting for the always-

promised, never-delivered complete OSI package, and the up-and-running, plug-and-play TCP/IP suite. Once the obvious choice of TCP/IP was made, the cost and technical risks of migrating from an installed base inhibited OSI acceptance.

2. The TCP/IP protocols were initially developed as a U.S. military research effort funded by the Department of Defense (DOD). Although DOD, like the rest of the U.S. government, was committed to international standards, DOD had immediate operational needs that could not be met during the 1980s and early 1990s by off-the-shelf OSI-based products. Accordingly, DOD mandated the use of TCP/IP protocols for virtually all software purchases. Because DOD is the largest consumer of software products in the world, this policy created an enormous market, encouraging vendors to develop TCP/IP-based products.
3. The Internet is built on the foundation of the TCP/IP suite. The dramatic growth of the Internet, and especially the World Wide Web, has cemented the victory of TCP/IP over OSI.

The TCP/IP Approach

The TCP/IP protocol suite recognizes that the task of communications is too complex and too diverse to be accomplished by a single unit. Accordingly, the task is broken up into modules or entities that may communicate with peer entities in another system. One entity within a system provides services to other entities and, in turn, uses the services of other entities. Good software-design practice dictates that these entities be arranged in a modular and hierarchical fashion.

The OSI model is based on this system of communication, but takes it one step further, recognizing that, in many respects, protocols at the same level of the hierarchy have certain features in common. This thinking yields the concept of rows or layers, as well as the attempt to describe in an abstract fashion what features are held in common by the protocols within a given row.

As an explanatory tool, a layered model has significant value and, indeed, the OSI model is used for precisely that purpose in many books on data communications and telecommunications. The objection sometimes raised by the designers of the TCP/IP protocol suite and its protocols is that the OSI model is prescriptive rather than descriptive. It dictates that protocols within a given layer perform certain functions, which may not be always desirable. It is possible to define more than one protocol at a given layer, and the functionality of those protocols may not be the same or even similar. Rather, what is common about a set of protocols at the same layer is that they share the same set of support protocols at the next lower layer.

Furthermore, there is the implication in the OSI model that, because interfaces between layers are well-defined, a new protocol can be substituted for an old one at a given layer with no impact on adjacent layers (see principle 6, Table 15.2); this is not always desirable or even possible. For example, a LAN lends itself easily to multicast and broadcast addressing at the link level. If the IEEE 802 link level were inserted below a network protocol entity that did not support multicasting and broadcasting, that service would be denied to upper layers of the hierarchy. To get around some of these problems, OSI proponents talk of null layers and sublayers. It sometimes seems that these artifacts save the model at the expense of good protocol design.

In the TCP/IP model, as we shall see, the strict use of all layers is not mandated. For example, there are application-level protocols that operate directly on top of IP.

TCP/IP Protocol Architecture

The TCP/IP protocol suite was introduced in Chapter 1. As we pointed out, there is no official TCP/IP protocol model. However, it is useful to characterize the protocol suite as involving five layers. To summarize from Chapter 1, these layers are

- **Application layer.** Provides communication between processes or applications on separate hosts.
- **Host-to-host, or transport layer.** Provides end-to-end, data-transfer service. This layer may include reliability mechanisms. It hides the details of the underlying network or networks from the application layer.
- **Internet layer.** Concerned with routing data from source to destination host through one or more networks connected by routers.
- **Network access layer.** Concerned with the logical interface between an end system and a subnetwork.
- **Physical layer.** Defines characteristics of the transmission medium, signaling rate, and signal encoding scheme.

Operation of TCP and IP

Figure 15.4 indicates how the TCP/IP 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 token ring, 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, acting 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. Each process with 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 1 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 that

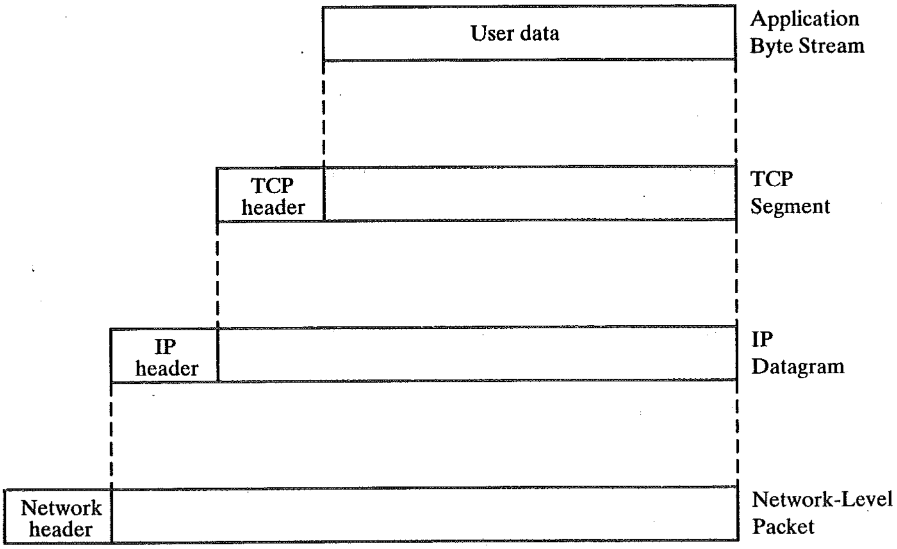


FIGURE 15.11 Protocol data units in the TCP/IP architecture.

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 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 15.11. 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, thereby forming a TCP segment. The control information is to be used by the peer TCP protocol entity at host B. Examples of items that are included in this header are

- **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 X, 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 process.

Protocol Interfaces

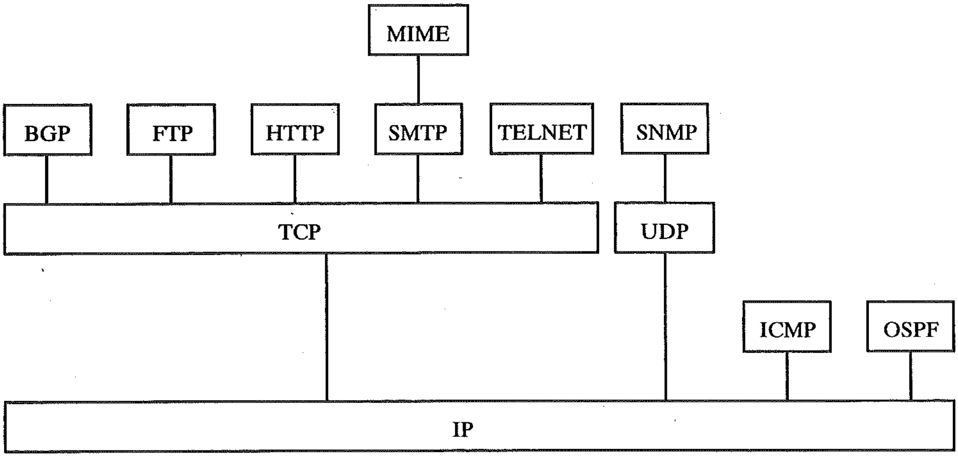
Each layer in the TCP/IP protocol suite interacts with its immediate adjacent layers. At the source, the process layer makes use of the services of the host-to-host layer and provides data down to that layer. A similar relationship exists at the interface of the host-to-host and internet layers and at the interface of the internet and network access layers. At the destination, each layer delivers data up to the next-higher layer.

This use of each individual layer is not required by the architecture. As Figure 15.12 suggests, it is possible to develop applications that directly invoke the services of any one of the layers. Most applications require a reliable end-to-end protocol and thus make use of TCP; some special-purpose applications, however, do not need such services, for example, the simple network management protocol (SNMP) that uses an alternative host-to-host protocol known as the *user datagram protocol* (UDP); others may make use of IP directly. Applications that do not involve internetworking and that do not need TCP have been developed to invoke the network access layer directly.

The Applications

Figure 15.12 shows the position of some of the key protocols commonly implemented as part of the TCP/IP protocol suite. Most of these protocols are discussed in the remainder of Part Four. In this section, we briefly highlight three protocols that have traditionally been considered mandatory elements of TCP/IP, and which were designated as military standards, along with TCP and IP, by DOD.

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



LEGEND

BGP	= Border Gateway Protocol
FTP	= File Transfer Protocol
HTTP	= Hyptertext Transfer Protocol
ICMP	= Internet Control Message Protocol
IP	= Internet Protocol
OSPF	= Open Shortest Path First
MIME	= Multi-Purpose Internet Mail Extension
SMTP	= Simple Mail Transfer Protocol
SNMP	= Simple Network Management Protocol
TCP	= Transmission Control Protocol
UDP	= User Datagram Protocol

FIGURE 15.12 Some protocols in the TCP/IP protocol suite.

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 requests a 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 the completion and to accept new file transfer commands.

TELNET provides a remote log-on capability, which enables a user at a terminal or personal computer to log on 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.

15.4 RECOMMENDED READING

For the reader interested in greater detail on TCP/IP, there are two three-volume works that are more than adequate. The works by Comer and Stevens have become classics and are considered definitive [COME95, COME94a, COME94b]. The works by Stevens and Wright are equally worthwhile [STEV94, STEV96, WRIG95]. A more compact and very useful reference work is [MURP95], which covers the spectrum of TCP/IP related protocols in a technically concise but thorough fashion, including coverage of some protocols not found in the other two works.

One of the best treatments of OSI and OSI-related protocols is [JAIN93]. [HALS96] also provides good coverage.

COME94a Comer, D. and Stevens, D. *Internetworking with TCP/IP, Volume II: Design Implementation, and Internals.* Englewood Cliffs, NJ: Prentice Hall, 1994.

COME94b Comer, D. and Stevens, D. *Internetworking with TCP/IP, Volume III: Client-Server Programming and Applications.* Englewood Cliffs, NJ: Prentice Hall, 1994.

COME95 Comer, D. *Internetworking with TCP/IP, Volume I: Principles, Protocols, and Architecture.* Upper Saddle River, NJ: Prentice Hall, 1995.

HALS96 Halsall, F. *Data Communications, Computer Networks, and Open Systems.* Reading, MA: Addison-Wesley, 1996.

JAIN93 Jain, B. and Agrawala, A. *Open Systems Interconnection.* New York: McGraw-Hill, 1993.

MURP95 Murphy, E., Hayes, S., and Enders, M. *TCP/IP: Tutorial and Technical Overview.* Upper Saddle River: NJ: Prentice Hall, 1995.

STEV94 Stevens, W. *TCP/IP Illustrated, Volume 1: The Protocols.* Reading, MA: Addison-Wesley, 1994.

STEV96 Stevens, W. *TCP/IP Illustrated, Volume 3: TCP for Transactions, HTTP, NNTP, and the UNIX(R) Domain Protocol.* Reading, MA: Addison-Wesley, 1996.

WRIG95 Wright, G. and Stevens, W. *TCP/IP Illustrated, Volume 2: The Implementation.* Reading, MA: Addison-Wesley, 1995.

15.4 PROBLEMS

15.1 List the major disadvantages with the layered approach to protocols.

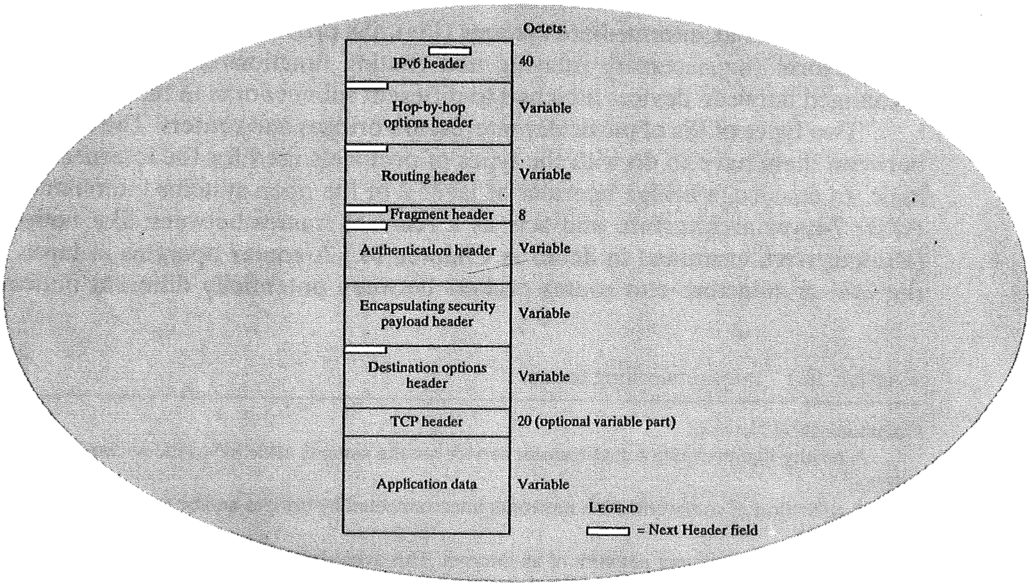
15.2 Based on the principles enunciated in Table 15.2, design an architecture with eight layers and make a case for it. Design one with six layers and make a case for that.

15.3 Two blue armies are each poised on opposite hills preparing to attack a single red army in the valley. The red army can defeat either of the blue armies separately but will fail to defeat both blue armies if they attack simultaneously. The blue armies communicate via an unreliable communications system (a foot soldier). The commander, with one of the blue armies, would like to attack at noon. His problem is this: If he sends a message ordering the attack, he cannot be sure it will get through. He could ask for acknowledgment but that might not get through. Is there a protocol that the two blue armies can use to avoid defeat?

15.4 Discuss the need or lack of need for a network layer (OSI layer 3) in a broadcast network.

CHAPTER 16

INTERNETWORKING



Packet-switched and packet broadcasting networks grew out of a need to allow the computer user to have access to resources beyond those available in a single system. In a similar fashion, the resources of a single network are often inadequate to meet users' needs. Because the networks that might be of interest exhibit so many differences, it is impractical to consider merging them into a single network. Rather, what is needed is the ability to interconnect various networks so that any two stations on any of the constituent networks can communicate.

Table 16.1 lists some commonly used terms relating to the interconnection of networks, or internetworking. An interconnected set of networks, from a user's point of view, may appear simply as a larger network. However, if each of the constituent networks retains its identity, and special mechanisms are needed for communicating across multiple networks, then the entire configuration is often referred to as an **internet**, and each of the constituent networks as a **subnetwork**.

Each constituent subnetwork in an internet supports communication among the devices attached to that subnetwork; these devices are referred to as **end systems** (ESs). In addition, subnetworks are connected by devices referred to in the ISO documents as **intermediate systems** (ISs). ISs provide a communications path and perform the necessary relaying and routing functions so that data can be exchanged between devices attached to different subnetworks in the internet.

Two types of ISs of particular interest are bridges and routers. The differences between them have to do with the types of protocols used for the internetworking logic. In essence, a **bridge** operates at layer 2 of the open systems interconnection (OSI) 7-layer architecture and acts as a relay of frames between like networks. (Bridges were examined in detail in Chapter 14.) A **router** operates at layer 3 of the OSI architecture and routes packets between potentially different networks.

TABLE 16.1 Internetworking terms.

Communication Network
    A facility that provides a data transfer service among stations attached to the network.

Internet
    A collection of communication networks interconnected by bridges and/or routers.

Subnetwork
    Refers to a constituent network of an internet. This avoids ambiguity since the entire internet, from a user's point of view, is a single network.

End System (ES)
    A device attached to one of the subnetworks of an internet that is used to support end-user applications or services.

Intermediate System (IS)
    A device used to connect two subnetworks and permit communication between end systems attached to different subnetworks.

Bridge
    An IS used to connect two LANs that use identical LAN protocols. The bridge acts as an address filter, picking up packets from one LAN that are intended for a destination on another LAN and passing those packets on. The bridge does not modify the contents of the packets and does not add anything to the packet. The bridge operates at layer 2 of the OSI model.

Router
    An IS used to connect two networks that may or may not be similar. The router employs an internet protocol present in each router and each host of the network. The router operates at layer 3 of the OSI model.

Both the bridge and the router assume that the same upper-layer protocols are in use.

We begin our examination with a discussion of the principles underlying various approaches to internetworking. We then examine the most important architectural approach to internetworking: the connectionless router. As an example, we describe the most widely used internetworking protocol, called simply the Internet Protocol (IP). These three approaches are explored in some detail. The chapter then turns to the issue of internetwork routing algorithms. Finally, we look at the newest standardized internetworking protocol, known as IPv6.

Figure 16.1 highlights the position of the protocols discussed in this chapter within the TCP/IP protocol.

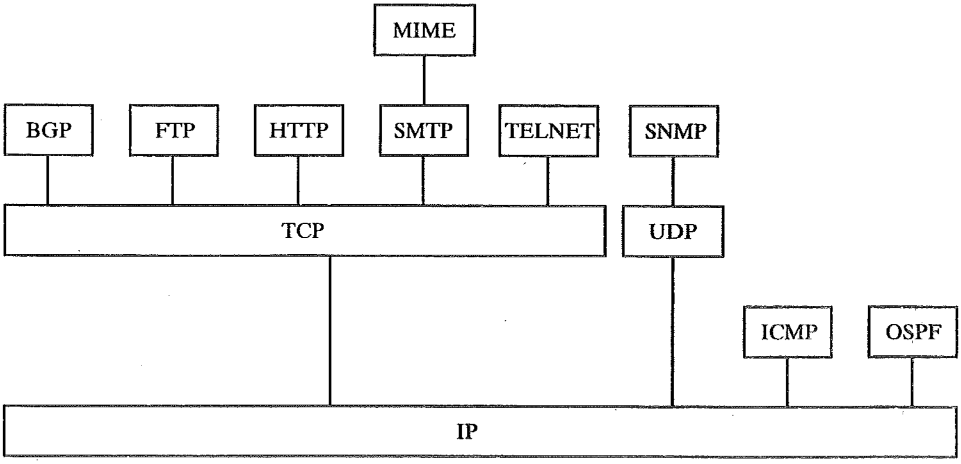


FIGURE 16.1 Internetworking protocols in context.

16.1 PRINCIPLES OF INTERNETWORKING

Requirements

Although a variety of approaches have been taken to provide internetwork service, the overall requirements on the internetworking facility can be stated in general; these include

1. Providing a link between networks. At minimum, a physical and link control connection is needed.
2. Providing for the routing and delivery of data between processes on different networks.
3. Providing an accounting service that keeps track of the use of the various networks and routers and that maintains status information.

4. Providing the services listed above in such a way as not to require modifications to the networking architecture of any of the constituent networks; this means that the internetworking facility must accommodate a number of differences among networks, including
   a) **Different addressing schemes.** The networks may use different endpoint names and addresses and directory maintenance schemes. Some form of global network-addressing must be provided, as well as a directory service.
   b) **Different maximum packet size.** Packets from one network may have to be broken up into smaller pieces for another. This process is referred to as segmentation, or fragmentation.
   c) **Different network-access mechanisms.** The network-access mechanism between station and network may be different for stations on different networks.
   d) **Different timeouts.** Typically, a connection-oriented transport service will await an acknowledgment until a timeout expires, at which time it will retransmit its block of data. In general, longer times are required for successful delivery across multiple networks. Internetwork timing procedures must allow for successful transmission that avoids unnecessary retransmissions.
   e) **Error recovery.** Intranetwork procedures may provide anything from no error recovery up to reliable end-to-end (within the network) service. The internetwork service should not depend on, nor be interfered with, by the nature of the individual network's error recovery capability.
   f) **Status reporting.** Different networks report status and performance differently. Yet it must be possible for the internetworking facility to provide such information on internetworking activity to interested and authorized processes.
   g) **Routing techniques.** Intranetwork routing may depend on fault detection and congestion control techniques peculiar to each network; the internetworking facility must be able to coordinate these to adaptively route data between stations on different networks.
   h) **User-access control.** Each network will have its own user-access control technique (authorization for use of the network) that must be invoked by the internetwork facility as needed. Further, a separate internetwork access control technique may be required.
   i) **Connection, connectionless.** Individual networks may provide connection-oriented (e.g., virtual circuit) or connectionless (datagram) service. It may be desirable for the internetwork service not to depend on the nature of the connection service of the individual networks.

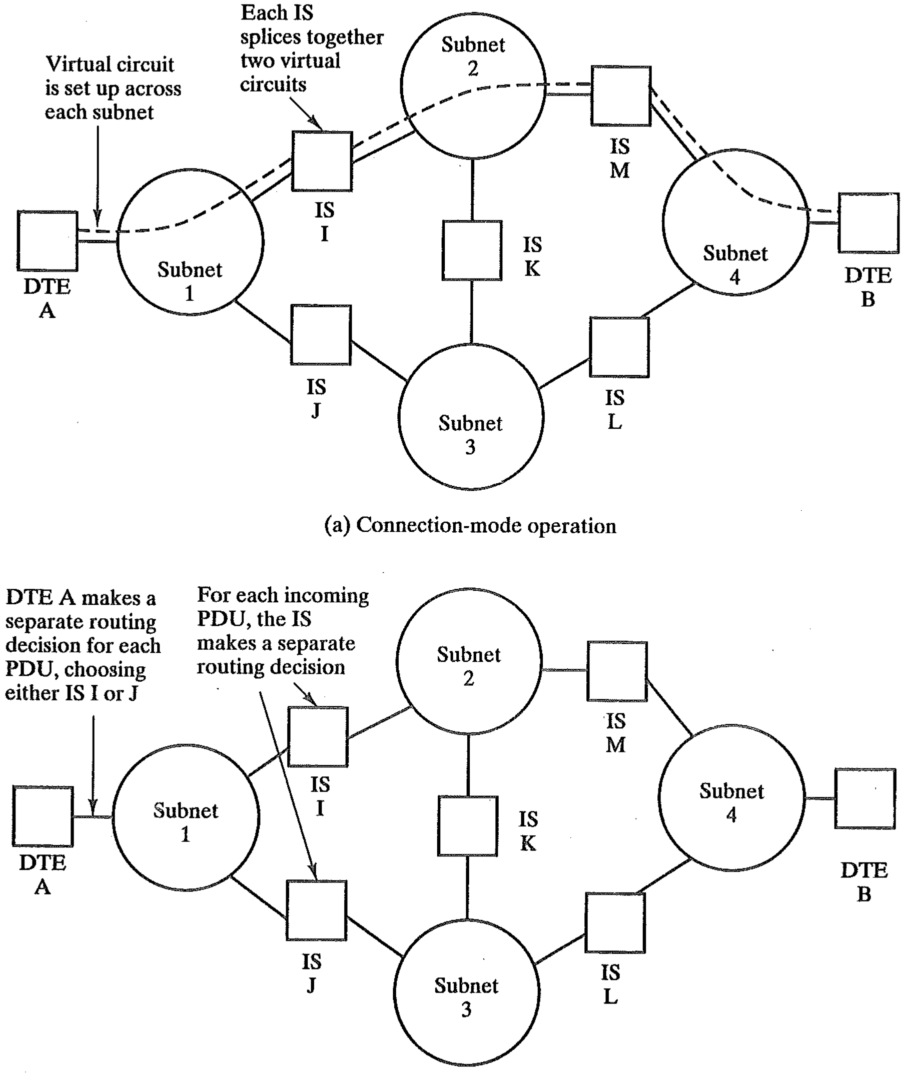
These points are worthy of further comment but are best pursued in the context of specific architectural approaches. We outline these approaches next, and then turn to a more detailed discussion of the router-based connectionless approach.

Architectural Approaches

In describing the interworking function, two dimensions are important:

- The mode of operation (connection-mode or connectionless)
- The protocol architecture

The mode of operation determines the protocol architecture. There are two general approaches, depicted in Figure 16.2.


Virtual circuit is set up across each subnet
Each IS splices together two virtual circuits
Subnet 2
IS M
IS I
DTE A
Subnet 1
IS K
Subnet 4
DTE B
IS J
Subnet 3
IS L
DTE A makes a separate routing decision for each PDU, choosing either IS I or J
For each incoming PDU, the IS makes a separate routing decision
Subnet 2
IS M
IS I
Subnet 1
Subnet 4
IS K
DTE A
DTE B
IS J
Subnet 3
IS L

(a) Connection-mode operation

(b) Connectionless-mode operation

FIGURE 16.2 Internetworking approaches.

Connection-Mode Operation

In the connection-mode operation, it is assumed that each subnetwork provides a connection-mode form of service. That is, it is possible to establish a logical network connection (e.g., virtual circuit) between any two DTEs attached to the same subnetwork. With this in mind, we can summarize the connection-mode approach as follows:

1. ISs are used to connect two or more subnetworks; each IS appears as a DTE to each of the subnetworks to which it is attached.
2. When DTE A wishes to exchange data with DTE B, a logical connection is set up between them. This logical connection consists of the concatenation of a sequence of logical connections across subnetworks. The sequence is such that it forms a path from DTE A to DTE B.
3. The individual subnetwork logical connections are spliced together by ISs. For example, there is a logical connection from DTE A to IS I across subnetwork 1 and another logical connection from IS I to IS M across subnetwork 2. Any traffic arriving at IS I on the first logical connection is retransmitted on the second logical connection, and vice versa.

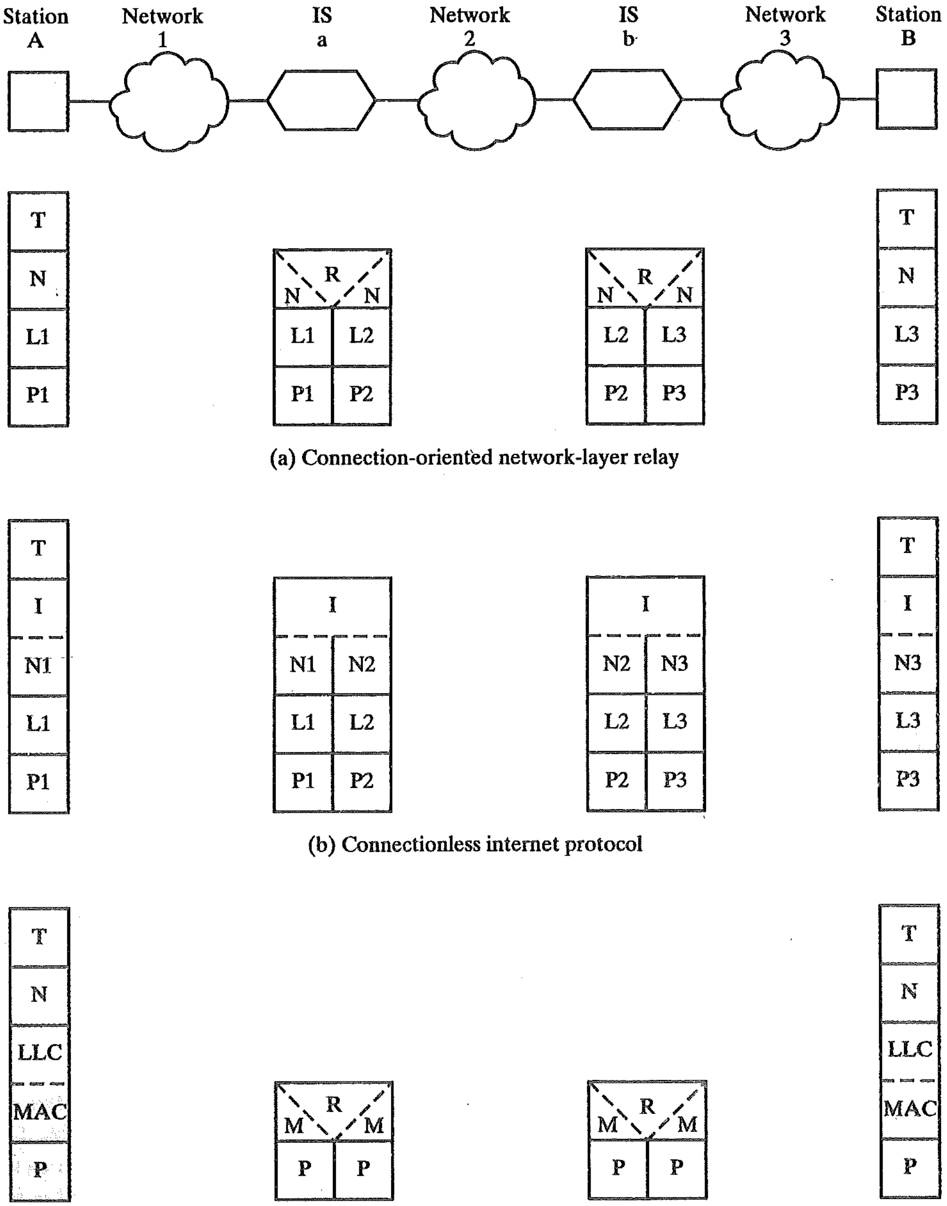
Several additional points can be made about this form of operation. First, this approach is suited to providing support for a connection-mode network service. From the point of view of network users in DTEs A and B, a logical network connection is established between them that provides all of the features of a logical connection across a single network.

The second point to be made is that this approach assumes that there is a connection-mode service available from each subnetwork and that these services are equivalent; clearly, this may not always be the case. For example, an IEEE 802 or FDDI local area network provides a service defined by the logical link control (LLC). Two of the options with LLC provide only connectionless service. Therefore, in this case, the subnetwork service must be enhanced. An example of how this would be done is for the ISs to implement X.25 on top of LLC across the LAN.

Figure 16.3a illustrates the protocol architecture for connection-mode operation. Access to all subnetworks, either inherently or by enhancement, is by means of the same network layer protocol. The interworking units operate at layer 3. As was mentioned, layer 3 ISs are commonly referred to as routers. A connection-oriented router performs the following key functions:

- **Relaying.** Data units arriving from one subnetwork via the network layer protocol are relayed (retransmitted) on another subnetwork. Traffic is over logical connections that are spliced together at the routers.
- **Routing.** When an end-to-end logical connection, consisting of a sequence of logical connections, is to be set up, each router in the sequence must make a routing decision that determines the next hop in the sequence.

Thus, at layer 3, a relaying operation is performed. It is assumed that all of the end systems share common protocols at layer 4 (transport), and above, for successful end-to-end communication.



(a) Connection-oriented network-layer relay

(b) Connectionless internet protocol

(c) Bridge

FIGURE 16.3 Internetwork architectures.

Connectionless-Mode Operation

Figure 16.3b illustrates the connectionless mode of operation. Whereas connection-mode operation corresponds to the virtual circuit mechanism of a packet-switching network (Figure 9.4c), connectionless-mode operation corresponds to the datagram mechanism of a packet-switching network (Figure 9.4d). Each network protocol data unit is treated independently and routed from source DTE to destination DTE through a series of routers and networks. For each data unit transmitted by A, A makes a decision as to which router should receive the data unit. The data unit hops across the internet from one router to the next until it reaches the destination subnetwork. At each router, a routing decision is made (independently for each data unit) concerning the next hop. Thus, different data units may travel different routes between source and destination DTE.

Figure 16.3b illustrates the protocol architecture for connectionless-mode operation. All DTEs and all routers share a common network layer protocol known generically as the internet protocol (IP). An internet protocol was initially developed for the DARPA internet project and published as RFC 791, and has become an Internet Standard. The ISO standard, ISO 8473, provides similar functionality. Below this internet protocol, a protocol is needed to access the particular subnetwork. Thus, there are typically two protocols operating in each DTE and router at the network layer: an upper sublayer that provides the internetworking function, and a lower sublayer that provides subnetwork access.

Bridge Approach

A third approach that is quite common is the use of a bridge. The bridge, also known as a MAC-level relay, uses a connectionless mode of operation (Figure 16.2b), but does so at a lower level than a router.

The protocol architecture for a bridge is illustrated in Figure 16.3c. In this case, the end systems share common transport and network protocols. In addition, it is assumed that all of the networks use the same protocols at the link layer. In the case of IEEE 802 and FDDI LANs, this means that all of the LANs share a common LLC protocol and a common MAC protocol. For example, all of the LANs are IEEE 802.3 using the unacknowledged connectionless form of LLC. In this case, MAC frames are relayed through bridges between the LANs.

The bridge approach is examined in Chapter 14.

16.2 CONNECTIONLESS INTERNETWORKING

The internet protocol (IP) was developed as part of the DARPA internet project. Somewhat later, when the international standards community recognized the need for a connectionless approach to internetworking, the ISO connectionless network protocol (CLNP) was standardized. The functionality of IP and CLNP is very similar; they differ in the formats used and in some minor functional features. In this section, we examine the essential functions of an internetworking protocol, which apply to both CLNP and IP. For convenience, we refer to IP, but it should be understood that the narrative in this section applies to both IP and CLNP.

Operation of a Connectionless Internetworking Scheme

IP provides a connectionless, or datagram, service between end systems. There are a number of advantages to this connectionless approach:

- A connectionless internet facility is flexible. It can deal with a variety of networks, some of which are themselves connectionless. In essence, IP requires very little from the constituent networks.
- A connectionless internet service can be made highly robust. This is basically the same argument made for a datagram network service versus a virtual circuit service. For a further discussion, the reader is referred to Section 9.2.
- A connectionless internet service is best for connectionless transport protocols.

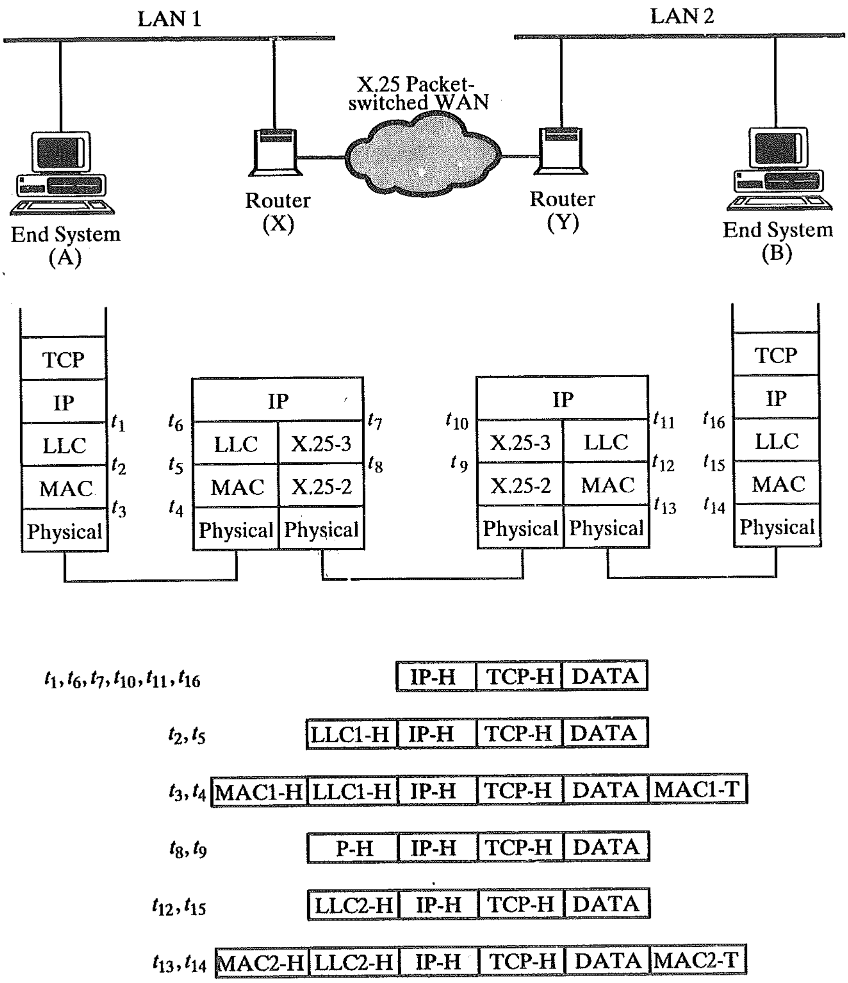
Figure 16.4 depicts a typical example of IP, in which two LANs are interconnected by an X.25 packet-switched WAN. The figure depicts the operation of the internet protocol for data exchange between host A on one LAN (subnetwork 1) and host B on another departmental LAN (subnetwork 2) through the WAN. The figure shows the format of the data unit at each stage. The end systems and routers must all share a common internet protocol. In addition, the end systems must share the same protocols above IP. The intermediate routers need only implement up through IP.

The IP at A receives blocks of data to be sent to B from the higher layers of software in A. IP attaches a header specifying, among other things, the global internet address of B. That address is logically in two parts: network identifier and end system identifier. The result is called an internet-protocol data unit, or simply a datagram. The datagram is then encapsulated with the LAN protocol and sent to the router, which strips off the LAN fields to read the IP header. The router then encapsulates the datagram with the X.25 protocol fields and transmits it across the WAN to another router. This router strips off the X.25 fields and recovers the datagram, which it then wraps in LAN fields appropriate to LAN 2 and sends it to B.

Let us now look at this example in more detail. End system A has a datagram to transmit to end system B; the datagram includes the internet address of B. The IP module in A recognizes that the destination (B) is on another subnetwork. So, the first step is to send the data to a router, in this case router X. To do this, IP passes the datagram down to the next lower layer (in this case LLC) with instructions to send it to router X. LLC in turn passes this information down to the MAC layer, which inserts the MAC-level address of router X into the MAC header. Thus, the block of data transmitted onto LAN 1 includes data from a layer or layers above TCP, plus a TCP header, an IP header, and LLC header, and a MAC header and trailer.

Next, the packet travels through subnetwork 1 to router X. The router removes MAC and LLC fields and analyzes the IP header to determine the ultimate destination of the data, in this case B. The router must now make a routing decision. There are three possibilities:

1. The destination station Y is connected directly to one of the subnetworks to which the router is attached. In this case, the router sends the datagram directly to the destination.
2. To reach the destination, one or more additional routers must be traversed. In this case, a routing decision must be made: To which router should the datagram be sent? In both cases, the IP module in the router sends the datagram



LEGEND

TCP-H	=	TCP header	MACi-H	=	MAC header
IP-H	=	IP header	MACi-T	=	MAC trailer
LLCi-H	=	LLC header	P-H	=	X.25 packet header

FIGURE 16.4 Internet protocol operation.

down to the next lower layer with the destination subnetwork address. Please note that we are speaking here of a lower-layer address that refers to this network.

3. The router does not know the destination address. In this case, the router returns an error message to the source of the datagram.

In this example, the data must pass through router Y before reaching the destination. Router X, then, constructs a new packet by appending an X.25 header, containing the address of router Y, to the IP data unit. When this packet arrives at router Y, the packet header is stripped off. The router determines that this IP data unit is destined for B, which is connected directly to a network to which this router is attached. The router therefore creates a frame with a destination address of B and sends it out onto LAN 2. The data finally arrive at B, where the LAN and IP headers can be stripped off.

At each router, before the data can be forwarded, the router may need to segment the data unit to accommodate a smaller maximum packet-size limitation on the outgoing network. The data unit is split into two or more segments, each of which becomes an independent IP data unit. Each new data unit is wrapped in a lower-layer packet and queued for transmission. The router may also limit the length of its queue for each network to which is attaches so as to avoid having a slow network penalize a faster one. Once the queue limit is reached, additional data units are simply dropped.

The process described above continues through as many routers as it takes for the data unit to reach its destination. As with a router, the destination end system recovers the IP data unit from its network wrapping. If segmentation has occurred, the IP module in the destination end system buffers the incoming data until the entire original data field can be reassembled. This block of data is then passed to a higher layer in the end system.

This service offered by the internet protocol is an unreliable one. That is, the internet protocol does not guarantee that all data will be delivered or that the data that are delivered will arrive in the proper order. It is the responsibility of the next higher layer (e.g., TCP) to recover from any errors that occur. This approach provides for a great deal of flexibility.

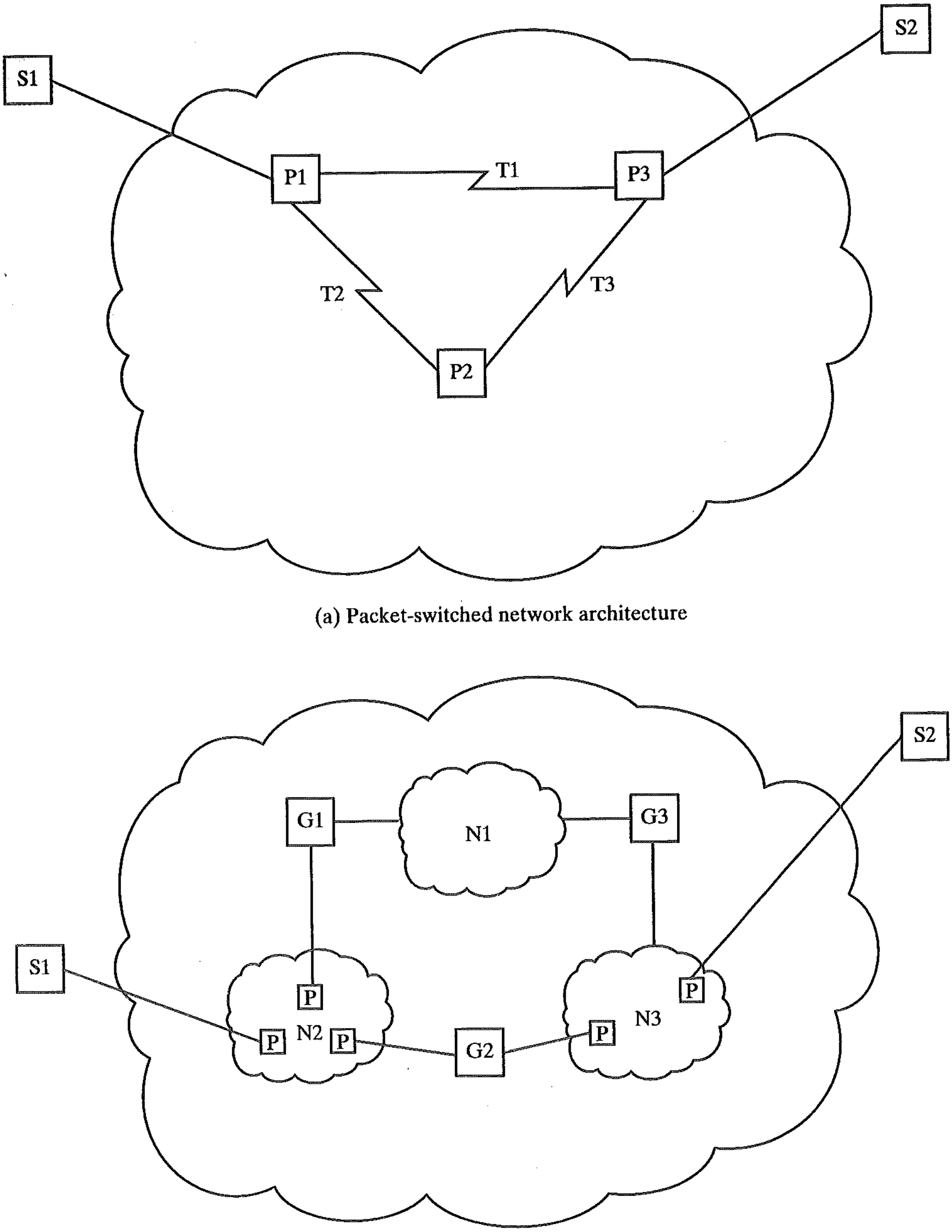
With the internet protocol approach, each unit of data is passed from router to router in an attempt to get from source to destination. Because delivery is not guaranteed, there is no particular reliability requirement on any of the subnetworks; thus, the protocol will work with any combination of subnetwork types. And, since the sequence of delivery is not guaranteed, successive data units can follow different paths through the internet; this allows the protocol to react to both congestion and failure in the internet by changing routes.

Design Issues

With that brief sketch of the operation of an IP-controlled internet, we can now go back and examine some design issues in greater detail:

- Routing
- Datagram lifetime
- Segmentation and reassembly
- Error control
- Flow control

As we proceed with this discussion, the reader will note many similarities between the design issues and techniques relevant to packet-switched networks. To



(a) Packet-switched network architecture

(b) Internet architecture

FIGURE 16.5 The internet as a network (based on HIND831).

see the reason for these parallels, consider Figure 16.5, which compares an internet architecture with a packet-switched network architecture. The routers (R1, R2, R3) in the internet correspond to the packet-switched nodes (P1, P2, P3) in the network, and the networks (N1, N2, N3) in the internet correspond to the transmission links

(T1, T2, T3) in the networks. The routers perform essentially the same functions as packet-switched nodes, and use the intervening networks in a manner analogous to transmission links.

Routing

Routing is generally accomplished by maintaining a routing table in each end system and router that gives, for each possible destination network, the next router to which the internet datagram should be sent.

The routing table may be static or dynamic. A static table, however, could contain alternate routes if a router is unavailable. A dynamic table is more flexible in responding to both error and congestion conditions. In the Internet, for example, when a router goes down, all of its neighbors will send out a status report, allowing other routers and stations to update their routing tables. A similar scheme can be used to control congestion; this is a particularly important function because of the mismatch in capacity between local and wide-area networks. Section 16.4 discusses routing protocols.

Routing tables may also be used to support other internetworking services, such as those governing security and priority. For example, individual networks might be classified to handle data up to a given security classification. The routing mechanism must assure that data of a given security level are not allowed to pass through networks not cleared to handle such data.

Another routing technique is source routing. The source station specifies the route by including a sequential list of routers in the datagram. This, again, could be useful for security or priority requirements.

Finally, we mention a service related to routing: route recording. To record a route, each router appends its internet address to a list of addresses in the datagram. This feature is useful for testing and debugging purposes.

Datagram Lifetime

If dynamic or alternate routing is used, the potential exists for a datagram to loop indefinitely through the internet. This is undesirable for two reasons. First, an endlessly circulating datagram consumes resources. Second, we will see in Chapter 17 that a transport protocol may depend on there being an upper bound on datagram lifetime. To avoid these problems, each datagram can be marked with a lifetime. Once the lifetime expires, the datagram is discarded.

A simple way to implement lifetime is to use a hop count. Each time that a datagram passes through a router, the count is decremented. Alternatively, the lifetime could be a true measure of time; this requires that the routers must somehow know how long it has been since the datagram or fragment last crossed a router, in order to know by how much to decrement the lifetime field. This would seem to require some global clocking mechanism. The advantage of using a true time measure is that it can be used in the reassembly algorithm, described next.

Segmentation and Reassembly

Individual subnetworks within an internet may specify different maximum packet sizes. It would be inefficient and unwieldy to try to dictate uniform packet size across networks. Thus, routers may need to segment incoming datagrams into smaller pieces, called fragments, before transmitting on to the next subnetwork.

If datagrams can be segmented (perhaps more than once) in the course of their travels, the question arises as to where they should be reassembled, The easiest solution is to have reassembly performed at the destination only. The principal disadvantage of this approach is that fragments can only get smaller as data move through the internet; this may impair the efficiency of some networks. Also, if intermediate router reassembly is allowed, the following disadvantages result:

1. Large buffers are required at routers, and there is the risk that all of the buffer space will be used up in the storing partial datagrams.
2. All fragments of a datagram must pass through the same router, thereby inhibiting the use of dynamic routing.

In IP, datagram fragments are reassembled at the destination end system. The IP segmentation technique uses the following fields in the IP header:

- Data Unit Identifier (ID)
- Data Length
- Offset
- More-flag

The ID is a means of uniquely identifying an end-system-originated datagram. In IP, it consists of the source and destination addresses, an identifier of the protocol layer that generated the data (e.g., TCP), and a sequence number supplied by that protocol layer. Data length indicates the length of the user data field in octets, and the offset is the position of a fragment of user data in the data field of the original datagram, in multiples of 64 bits.

The source end system creates a datagram with a data length equal to the entire length of the data field, with Offset = 0, and a more-flag set to 0 (false). To segment a long datagram, an IP module in a router performs the following tasks:

1. Create two new datagrams and copy the header fields of the incoming datagram into both.
2. Divide the incoming user data field into two approximately equal portions along a 64-bit boundary, placing one portion in each new datagram. The first portion must be a multiple of 64 bits.
3. Set the Data Length of the first new datagram to the length of the inserted data, and set more-flag to 1 (true). The Offset field is unchanged.
4. Set the data length of the second new datagram to the length of the inserted data, and add the length of the first data portion divided by 8 to the Offset field. The more-flag remains the same.

Table 16.2 gives an example. The procedure can easily be generalized to an *n*-way split.

To reassemble a datagram, there must be sufficient buffer space at the reassembly point. As fragments with the same ID arrive, their data fields are inserted in the proper position in the buffer until the entire data field is reassem-

TABLE 16.2 Segmentation example.

Original datagram	First segment	Second segment
Data Length = 472 Segment Offset = 0 More = 0	Data Length = 240 Segment Offset = 0 More = 1	Data Length = 232 Segment Offset = 30 More = 0

bled, which is achieved when a contiguous set of data exists, starting with an offset of zero and ending with data from a fragment with a false more-flag.

One eventuality that must be dealt with is that one or more of the fragments may not get through; the IP service does not guarantee delivery. Some means is needed to decide whether to abandon a reassembly effort to free up buffer space. Two approaches are commonly used. First, one can assign a reassembly lifetime to the first fragment to arrive. This is a local, real-time clock assigned by the reassembly function and decremented while the fragments of the original datagram are being buffered. If the time expires prior to complete reassembly, the received fragments are discarded. A second approach is to make use of the datagram lifetime, which is part of the header of each incoming fragment. The lifetime field continues to be decremented by the reassembly function; as with the first approach, if the lifetime expires prior to complete reassembly, the received fragments are discarded.

Error Control

The internetwork facility does not guarantee successful delivery of every datagram. When a datagram is discarded by a router, the router should attempt to return some information to the source, if possible. The source internet protocol entity may use this information to modify its transmission strategy, and it may notify higher layers. To report that a specific datagram has been discarded, some means of datagram identification is needed.

Datagrams may be discarded for a number of reasons, including lifetime expiration, congestion, and FCS error. In the latter case, notification is not possible, as the source address field may have been damaged.

Flow Control

Internet flow control allows routers and/or receiving stations to limit the rate at which they receive data. For the connectionless type of service we are describing, flow control mechanisms are limited. The best approach would seem to be to send flow control packets, requesting reduced data flow, to other routers and source stations.

16.3 THE INTERNET PROTOCOL

The Internet Protocol (IP) is part of the TCP/IP protocol suite, and is the most widely-used internetworking protocol. It is functionally similar to the ISO standard connectionless network protocol (CLNP). As with any protocol standard, IP is specified in two parts:

- The interface with a higher layer (e.g., TCP), specifying the services that IP provides
- The actual protocol format and mechanisms

In this section, we first examine IP services and then the IP protocol. This is followed by a discussion of IP address formats. Finally, the Internet Control Message Protocol (ICMP), which is an integral part of IP, is described.

IP Services

IP provides two service primitives at the interface to the next-higher layer (Figure 16.6). The Send primitive is used to request transmission of a data unit. The Deliver primitive is used by IP to notify a user of the arrival of a data unit. The parameters associated with the two primitives are

- **Source address.** Internetwork address of sending IP entity.
- **Destination address.** Internetwork address of destination IP entity.
- **Protocol.** Recipient protocol entity (an IP user).
- **Type of service indicators.** Used to specify the treatment of the data unit in its transmission through component networks.
- **Identifier.** Used in combination with the source and destination addresses and user protocol to identify the data unit uniquely. This parameter is needed for reassembly and error reporting.
- **Don't-fragment identifier.** Indicates whether IP can segment (called fragment in the standard) data to accomplish delivery.
- **Time to live.** Measured in network hops.
- **Data length.** Length of data being transmitted.
- **Option data.** Options requested by the IP user.
- **Data.** User data to be transmitted.

```
Send (                              Deliver (
       Source address                       Source address
       Destination address                  Destination address
       Protocol                             Protocol
       Type of service indicators           Type of service indicators
       Identifier
       Don't-fragment identifier
       Time to live
       Data length                          Data length
       Option data                          Option data
       Data                                 Data
       )                                    )
```

FIGURE 16.6 IP service primitives and parameters.

Note that the *identifier, don't-fragment identifier*, and *time-to-live* parameters are present in the Send primitive but not in the Deliver primitive. These three parameters provide instructions to IP that are not of concern to the recipient IP user.

The sending IP user includes the *type-of-service* parameter to request a particular quality of service. The user may specify one or more of the services listed in Table 16.3. This parameter can be used to guide routing decisions. For example, if a router has several alternative choices for the next hop in routing a datagram, it may choose a network of a higher data rate if the high throughput option has been selected. This parameter, if possible, is also passed down to the network access protocol for use over individual networks. For example, if a precedence level is selected, and if the subnetwork supports precedence or priority levels, the precedence level will be mapped onto the network level for this hop.

The options parameter allows for future extensibility and for inclusion of parameters that are usually not invoked. The currently defined options are

- **Security.** Allows a security label to be attached to a datagram.
- **Source routing.** A sequenced list of router addresses that specifies the route to be followed. Routing may be strict (only identified routers may be visited) or loose (other intermediate routers may be visited).
- **Route recording.** A field is allocated to record the sequence of routers visited by the datagram.
- **Stream identification.** Names reserved resources used for stream service. This service provides special handling for volatile periodic traffic (e.g., voice).
- **Timestamping.** The source IP entity and some or all intermediate routers add a timestamp (precision to milliseconds) to the data unit as it goes by.

IP Protocol

The protocol between IP entities is best described with reference to the IP datagram format, shown in Figure 16.7. The fields are

- **Version (4 bits).** Indicates the version number, to allow evolution of the protocol.
- **Internet header length (IHL) (4 bits).** Length of header in 32-bit words. The minimum value is five, for a minimum header length of 20 octets.

TABLE 16.3 IP Service quality options.

Precedence	A measure of a datagram's relative importance. Eight levels of precedence are used. IP will attempt to provide preferential treatment for higher precedence datagrams.
Reliability	One of two levels may be specified: normal or high. A high value indicates a request that attempts be made to minimize the likelihood that this datagram will be lost or damaged.
Delay	One of two levels may be specified: normal or low. A low value indicates a request to minimize the delay that this datagram will experience.
Throughput	One of two levels may be specified: normal or high. A high value indicates a request to maximize the throughput for this datagram.

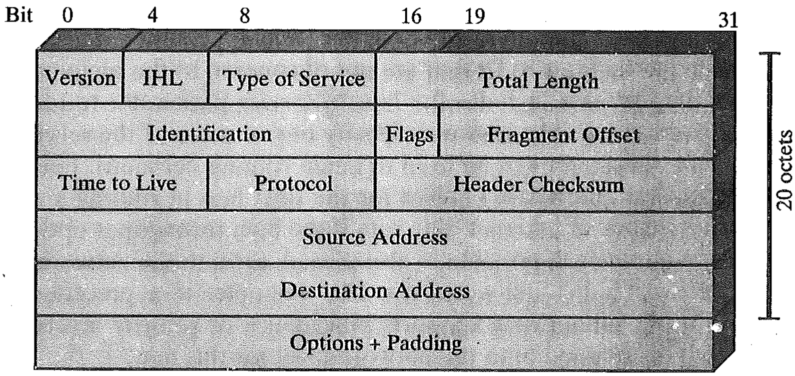


FIGURE 16.7 IP header.

- **Type of service (8 bits).** Specifies reliability, precedence, delay, and throughput parameters.
- **Total length (16 bits).** Total datagram length, in octets.
- **Identifier (16 bits).** A sequence number that, together with the source address, destination address, and user protocol, is intended to uniquely identify a datagram. Thus, the identifier should be unique for the datagram's source address, destination address, and user protocol for the time during which the datagram will remain in the internet.
- **Flags (3 bits).** Only two of the bits are currently defined. The More bit is used for segmentation (fragmentation) and reassembly, as previously explained. The Don't-Fragment bit prohibits fragmentation when set. This bit may be useful if it is known that the destination does not have the capability to reassemble fragments. However, if this bit is set, the datagram will be discarded if it exceeds the maximum size of an en route subnetwork. Therefore, if the bit is set, it may be advisable to use source routing to avoid subnetworks with small maximum packet size.
- **Fragment offset (13 bits).** Indicates where in the original datagram this fragment belongs, measured in 64-bit units, implying that fragments other than the last fragment must contain a data field that is a multiple of 64 bits.
- **Time to live (8 bits).** Measured in router hops.
- **Protocol (8 bits).** Indicates the next higher level protocol that is to receive the data field at the destination.
- **Header checksum (16 bits).** An error-detecting code applied to the header only. Because some header fields may change during transit (e.g., time to live, segmentation-related fields), this is reverified and recomputed at each router. The checksum field is the 16-bit one's complement addition of all 16-bit words in the header. For purposes of computation, the checksum field is itself initialized to a value of zero.
- **Source address (32 bits).** Coded to allow a variable allocation of bits to specify the network and the end system attached to the specified network (7 and 24 bits, 14 and 16 bits, or 21 and 8 bits).

- **Destination address (32 bits).** As above.
- **Options (variable).** Encodes the options requested by the sending user.
- **Padding (variable).** Used to ensure that the datagram header is a multiple of 32 bits.
- **Data (variable).** The data field must be an integer multiple of 8 bits. The maximum length of the datagram (data field plus header) is 65,535 octets.

It should be clear how the IP services specified in the Send and Deliver primitives map into the fields of the IP datagram.

IP Addresses

The source and destination address fields in the IP header each contain a 32-bit global internet address, generally consisting of a network identifier and a host identifier. The address is coded to allow a variable allocation of bits to specify network and host, as depicted in Figure 16.8. This encoding provides flexibility in assigning addresses to hosts and allows a mix of network sizes on an internet. In particular, the three network classes are best suited to the following conditions:

- **Class A.** Few networks, each with many hosts.
- **Class B.** Medium number of networks, each with a medium number of hosts.
- **Class C.** Many networks, each with a few hosts.

In a particular environment, it may be best to use addresses all from one class. For example, a corporate internetwork that consists of a large number of departmental local area networks may need to use class C addresses exclusively. However, the format of the addresses is such that it is possible to mix all three classes of

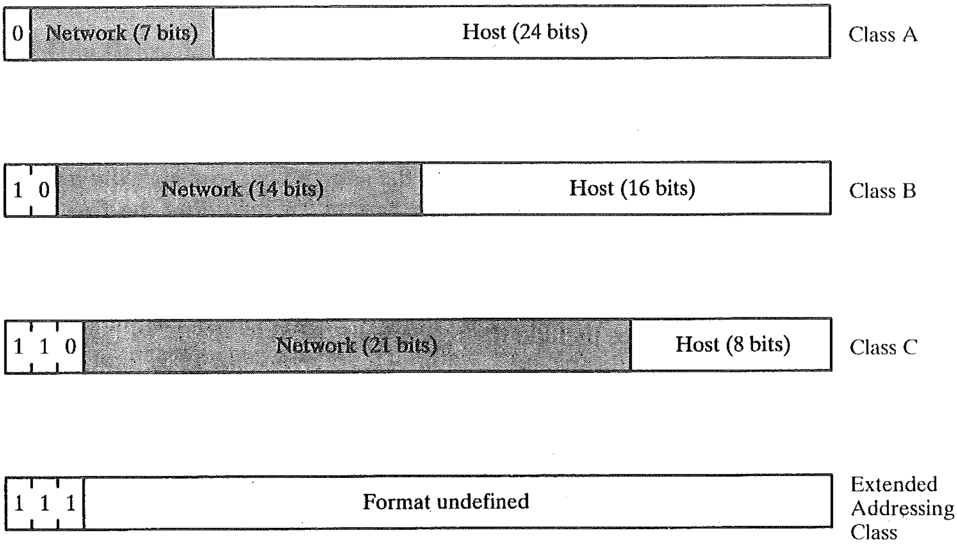


FIGURE 16.8 IP address formats.

addresses on the same internetwork; this is what is done in the case of the Internet itself. A mixture of classes is appropriate for an internetwork consisting of a few large networks, many small networks, plus some medium-sized networks.

The Internet Control Message Protocol (ICMP)

The IP standard specifies that a compliant implementation must also implement ICMP (RFC 792). ICMP provides a means for transferring messages from routers and other hosts to a host. In essence, ICMP provides feedback about problems in the communication environment. Examples of its use are: When a datagram cannot reach its destination, when the router does not have the buffering capacity to forward a datagram, and when the router can direct the station to send traffic on a shorter route. In most cases, an ICMP message is sent in response to a datagram, either by a router along the datagram's path, or by the intended destination host.

Although ICMP is, in effect, at the same level as IP in the TCP/IP architecture, it is a user of IP. An ICMP message is constructed and then passed down to IP, which encapsulates the message with an IP header and then transmits the resulting datagram in the usual fashion. Because ICMP messages are transmitted in IP datagrams, their delivery is not guaranteed and their use cannot be considered reliable.

Figure 16.9 shows the format of the various ICMP message types. All ICMP message start with a 64-bit header consisting of the following:

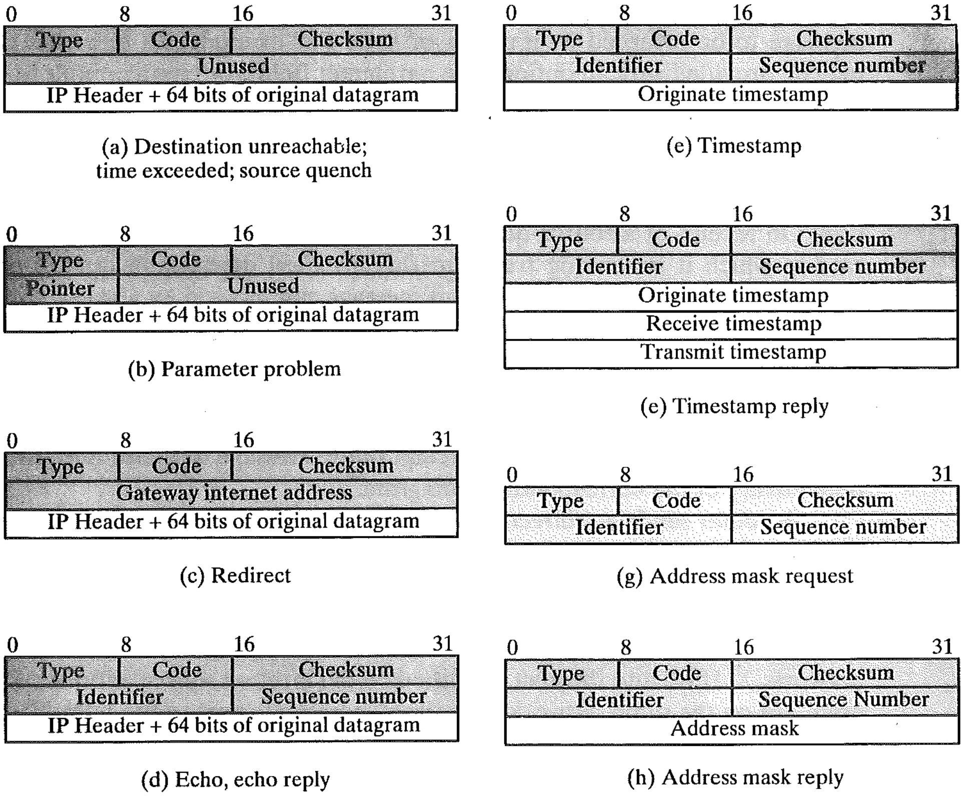
- **Type (8 bits).** Specifies the type of ICMP message.
- **Code (8 bits).** Used to specify parameters of the message that can be encoded in one or a few bits.
- **Checksum (16 bits).** Checksum of the entire ICMP message. This is the same checksum algorithm used for IP.
- **Parameters (32 bits).** Used to specify more lengthy parameters.

These fields are generally followed by additional information fields that further specify the content of the message.

In those cases in which the ICMP message refers to a prior datagram, the information field includes the entire IP header plus the first 64 bits of the data field of the original datagram. This enables the source host to match the incoming ICMP message with the prior datagram. The reason for including the first 64 bits of the data field is that this will enable the IP module in the host to determine which upper-level protocol or protocols were involved. In particular, the first 64 bits would include a portion of the TCP header or other transport-level header.

ICMP messages include the following:

- Destination unreachable
- Time exceeded
- Parameter problem
- Source quench
- Redirect



(a) Destination unreachable; time exceeded; source quench

(e) Timestamp

(b) Parameter problem

(e) Timestamp reply

(c) Redirect

(g) Address mask request

(d) Echo, echo reply

(h) Address mask reply

FIGURE 16.9 ICMP message formats.

- Echo
- Echo reply
- Timestamp
- Timestamp reply
- Address mask request
- Address mask reply

The **destination-unreachable** message covers a number of contingencies. A router may return this message if it does not know how to reach the destination network. In some networks, an attached router may be able to determine if a particular host is unreachable, and then return the message. The destination host itself may return this message if the user protocol or some higher-level service access point is unreachable. This could happen if the corresponding field in the IP header was set incorrectly. If the datagram specifies a source route that is unusable, a message is returned. Finally, if a router must fragment a datagram but the Don't-Fragment flag is set, a message is returned.

A router will return a **time-exceeded** message if the lifetime of the datagram expires. A host will send this message if it cannot complete reassembly within a time limit.

A syntactic or semantic error in an IP header will cause a *parameter-problem* message to be returned by a router or host. For example, an incorrect argument may be provided with an option. The parameter field contains a pointer to the octet in the original header where the error was detected.

The *source-quench* message provides a rudimentary form of flow control. Either a router or a destination host may send this message to a source host, requesting that it reduce the rate at which it is sending traffic to the internet destination. On receipt of a source-quench message, the source host should cut back the rate at which it is sending traffic to the specified destination until it no longer receives source-quench messages; this message can be used by a router or host that must discard datagrams because of a full buffer. In this case, the router or host will issue a source-quench message for every datagram that it discards. In addition, a system may anticipate congestion and issue such messages when its buffers approach capacity. In that case, the datagram referred to in the source-quench message may well be delivered. Thus, receipt of the message does not imply delivery or nondelivery of the corresponding datagram.

A router sends a *redirect* message to a host on a directly connected router to advise the host of a better route to a particular destination; the following is an example of its use. A router, R1, receives a datagram from a host on a network to which the router is attached. The router, R1, checks its routing table and obtains the address for the next router, R2, on the route to the datagram's internet destination network, X. If R2 and the host identified by the internet source address of the datagram are on the same network, a redirect message is sent to the host. The redirect message advises the host to send its traffic for network X directly to router R2, as this is a shorter path to the destination. The router forwards the original datagram to its internet destination (via R2). The address of R2 is contained in the parameter field of the redirect message.

The *echo* and *echo-reply* messages provide a mechanism for testing that communication is possible between entities. The recipient of an echo message is obligated to return the message in an echo-reply message. An identifier and sequence number are associated with the echo message to be matched in the echo-reply message. The identifier might be used like a service access point to identify a particular session, and the sequence number might be incremented on each echo request sent.

The *timestamp* and *timestamp-reply* messages provide a mechanism for sampling the delay characteristics of the internet. The sender of a timestamp message may include an identifier and sequence number in the parameters field and include the time that the message is sent (originate timestamp). The receiver records the time it received the message and the time that it transmits the reply message in the timestamp-reply message. If the timestamp message is sent using strict source routing, then the delay characteristics of a particular route can be measured.

The *address-mask-request* and *address-mask-reply* messages are useful in an environment that includes what are referred to as subnets. The concept of the subnet was introduced to address the following requirement. Consider an internet that includes one or more WANs and a number of sites, each of which has a number of LANs. We would like to allow arbitrary complexity of interconnected LAN structures within an organization, while insulating the overall internet against explosive growth in network numbers and routing complexity. One approach to this problem is to assign a single network number to all of the LANs at a site. From the point of view of the rest of the internet, there is a single network at that site, which simplifies addressing and routing. To allow the routers within the site to function properly, each LAN is assigned a subnet number. The host portion of the internet address is partitioned into a subnet number and a host number to accommodate this new level of addressing.

Within the subnetted network, the local routers must route on the basis of an extended network number consisting of the network portion of the IP address and the subnet number. The bit positions containing this extended network number are indicated by the address mask. The address-mask request and reply messages allow a host to learn the address mask for the LAN to which it connects. The host broadcasts an address-mask request message on the LAN. The router on the LAN responds with an address-mask reply message that contains the address mask. The use of the address mask allows the host to determine whether an outgoing datagram is destined for a host on the same LAN (send directly) or another LAN (send datagram to router). It is assumed that some other means (e.g., manual configuration) is used to create address masks and to make them known to the local routers.

16.4 ROUTING PROTOCOLS

The routers in an internet are responsible for receiving and forwarding packets through the interconnected set of subnetworks. Each router makes routing decisions based on knowledge of the topology and on the conditions of the internet. In a simple internet, a fixed routing scheme is possible. In more complex internets, a degree of dynamic cooperation is needed among the routers. In particular, the router must avoid portions of the network that have failed and should avoid portions of the network that are congested. In order to make such dynamic routing decisions, routers exchange routing information using a special routing protocol for that purpose. Information is needed about the status of the internet, in terms of which networks can be reached by which routes, and in terms of the delay characteristics of various routes.

In considering the routing function of routers, it is important to distinguish two concepts:

- **Routing information.** Information about the topology and delays of the internet.
- **Routing algorithm.** The algorithm used to make a routing decision for a particular datagram, based on current routing information.

There is another way to partition the problem that is useful from two points of view: allocating routing functions properly and effective standardization; this is to partition the routing function into

- Routing between end systems (ESs) and routers
- Routing between routers

The reason for the partition is that there are fundamental differences between what an ES must know to route a packet and what a router must know. In the case of an ES, the router must first know whether the destination ES is on the same subnet. If the answer is yes, then data can be delivered directly using the subnetwork access protocol; otherwise, the ES must forward the data to a router attached to the same subnetwork. If there is more than one such router, it is simply a matter of choosing one. The router forwards datagrams on behalf of other systems and needs to have some idea of the overall topology of the network in order to make a global routing decision.

In this section, we look at an example of an ES-to-router and router-to-router protocol.

Autonomous Systems

In order to proceed in our discussion of router-router protocols, we need to introduce the concept of an *autonomous system.* An autonomous system is an internet connected by homogeneous routers; generally, the routers are under the administrative control of a single entity. An *interior router protocol* (IRP) passes routing information between routers within an autonomous system. The protocol used within the autonomous system does not need to be implemented outside of the system. This flexibility allows IRPs to be custom-tailored to specific applications and requirements.

It may happen, however, that an internet will be constructed of more than one autonomous system. For example, all of the LANs at a site, such as an office complex or campus, could be linked by routers to form an autonomous system. This system might be linked through a wide-area network to other autonomous systems. The situation is illustrated in Figure 16.10. In this case, the routing algorithms and routing tables used by routers in different autonomous systems may differ themselves. Nevertheless, the routers in one autonomous system need at least a minimal level of information concerning networks that can be reached outside the system. The protocol used to pass routing information between routers in different autonomous systems is referred to as an *exterior router protocol* (ERP).

We can expect that an ERP will need to pass less information and be simpler than an IRP, for the following reason. If a datagram is to be transferred from a host in one autonomous system to a host in another autonomous system, a router in the first system need only determine the target autonomous system and devise a route to get into that system. Once the datagram enters the target autonomous system, the routers there can cooperate to finally deliver the datagram.

In the remainder of this section, we look at what are perhaps the most important examples of these two types of routing protocols.

Border Gateway Protocol

The Border Gateway Protocol (BGP) was developed for use in conjunction with internets that employ the TCP/IP protocol suite, although the concepts are applicable to any internet. BGP has become the standardized exterior router protocol for the Internet.

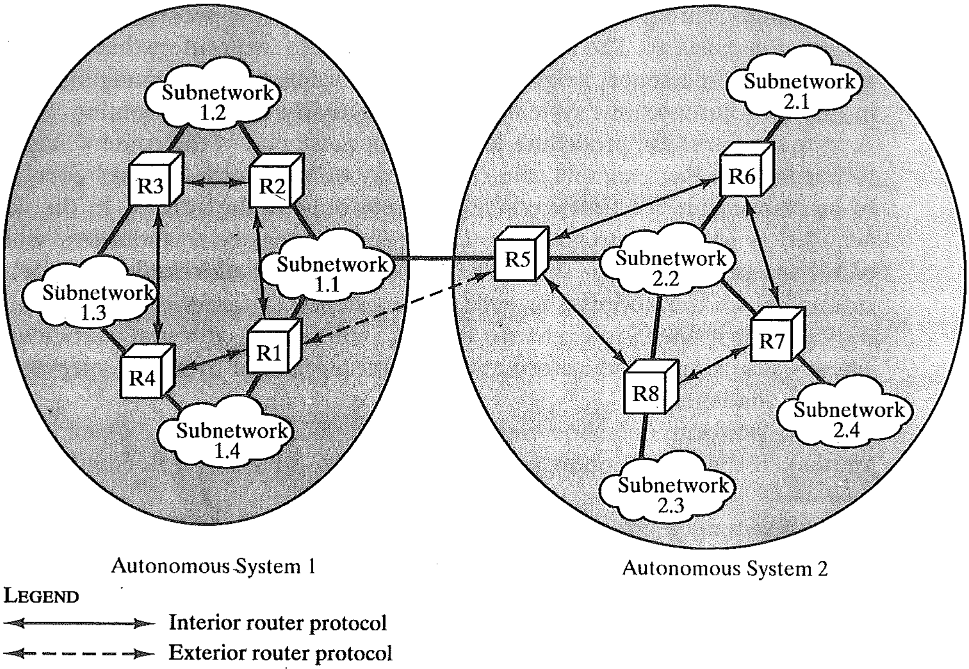


FIGURE 16.10 Application of exterior and interior routing protocols.

Functions

BGP was designed to allow routers, called gateways in the standard, in different autonomous systems (ASs) to cooperate in the exchange of routing information. The protocol operates in terms of messages, which are sent over TCP connections. The repertoire of messages is summarized in Table 16.4.

TABLE 16.4 BGP-4 messages.

Open	Used to open a neighbor relationship with another router.
Update	Used to (1) transmit information about a single route and/or (2) to list multiple routes to be withdrawn.
Keepalive	Used to (1) acknowledge an Open message and (2) periodically confirm the neighbor relationship.
Notification	Send when an error condition is detected.

Three functional procedures are involved in BGP:

- Neighbor acquisition
- Neighbor reachability
- Network reachability

Two routers are considered to be neighbors if they are attached to the same subnetwork. If the two routers are in different autonomous systems, they may wish to exchange routing information. For this purpose, it is necessary to first perform *neighbor acquisition.* The term "neighbor" refers to two routers that share the same subnetwork. In essence, neighbor acquisition occurs when two neighboring routers in different autonomous systems agree to regularly exchange routing information. A formal acquisition procedure is needed because one of the routers may not wish to participate. For example, the router may be overburdened and does not want to be responsible for traffic coming in from outside the system. In the neighbor-acquisition process, one router sends a request message to the other, which may either accept or refuse the offer. The protocol does not address the issue of how one router knows the address, or even the existence of, another router, nor how it decides that it needs to exchange routing information with that particular router. These issues must be addressed at configuration time or by active intervention of a network manager.

To perform neighbor acquisition, one router sends an Open message to another. If the target router accepts the request, it returns a Keepalive message in response.

Once a neighbor relationship is established, the **neighbor-reachability** procedure is used to maintain the relationship. Each partner needs to be assured that the other partner still exists and is still engaged in the neighbor relationship. For this purpose, the two routers periodically issue Keepalive messages to each other.

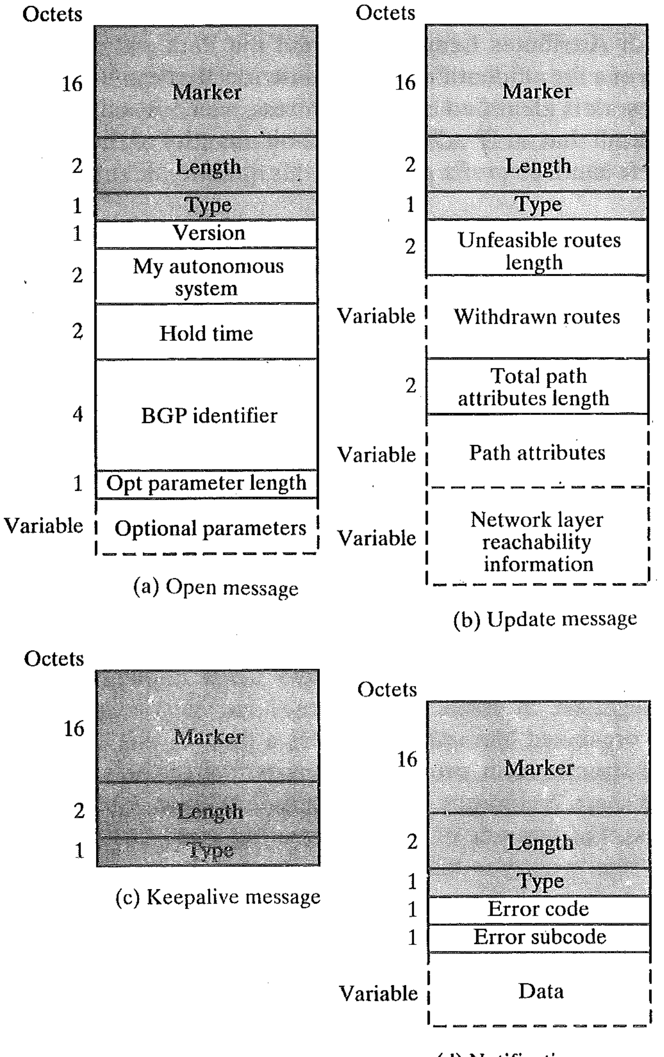
The final procedure specified by BGP is **network reachability**. Each router maintains a database of the subnetworks that it can reach and the preferred route for reaching that subnetwork. Whenever a change is made to this database, the router issues an Update message that is broadcast to all other routers implementing BGP. By the broadcasting of these Update message, all of the BGP routers can build up and maintain routing information.

BGP Messages

Figure 16.11 illustrates the formats of all of the BGP messages. Each message begins with a 19-octet header containing three fields, as indicated by the shaded portion of each message in the figure:

- **Marker.** Reserved for authentication. The sender may insert a value in this field that would be used as part of an authentication mechanism to enable the recipient to verify the identity of the sender.
- **Length.** Length of message in octets.
- **Type.** Type of message: Open, Update, Notification, Keepalive.

To acquire a neighbor, a router first opens a TCP connection to the neighbor router of interest. It then sends an Open message. This message identifies the AS to which the sender belongs and provides the IP address of the router. It also includes a Hold Time parameter, which indicates the number of seconds that the sender proposes for the value of the Hold Timer. If the recipient is prepared to open a neighbor relationship, it calculates a value of Hold Timer that is the minimum of its Hold Time and the Hold Time in the Open message. This calculated value is the maximum number of seconds that may elapse between the receipt of successive Keepalive and/or Update messages by the sender.



(a) Open message

(b) Update message

(c) Keepalive message

(d) Notification message

FIGURE 16.11 BGP message formats.

The Keepalive message consists simply of the header. Each router issues these messages to each of its peers often enough to prevent the Hold Time from expiring.

The Update message communicates two types of information:

1. Information about a single route through the internet. This information is available to be added to the database of any recipient router.
2. A list of routes previously advertised by this router that are being withdrawn.

An Update message may contain one or both types of information. Let us consider the first type of information first. Information about a single route through the network involves three fields: the Network Layer Reachability Information (NLRI) field, the Total Path Attributes Length field, and the Path Attributes field. The NLRI field consists of a list of identifiers of subnetworks that can be reached by this route. Each subnetwork is identified by its IP address, which is actually a portion of a full IP address. Recall that an IP address is a 32-bit quantity of the form {network, end system}. The left-hand, or prefix portion of this quantity, identifies a particular subnetwork.

The Path Attributes field contains a list of attributes that apply to this particular route. The following are the defined attributes:

- **Origin.** Indicates whether this information was generated by an interior router protocol (e.g., OSPF) or an exterior router protocol (in particular, BGP).
- **AS_Path.** A list of the ASs that are traversed for this route.
- **Next_Hop.** The IP address of the border router that should be used as the next hop to the destinations listed in the NLRI field.
- **Multi_Exit_Disc.** Used to communicate some information about routes internal to an AS. This is described later in this section.
- **Local_Pref.** Used by a router to inform other routers within the same AS of its degree of preference for a particular route. It has no significance to routers in other ASs.
- **Atomic_Aggregate, Aggregator.** These two fields implement the concept of route aggregation. In essence, an internet and its corresponding address space can be organized hierarchically, or as a tree. In this case, subnetwork addresses are structured in two or more parts. All of the subnetworks of a given subtree share a common partial internet address. Using this common partial address, the amount of information that must be communicated in NLRI can be significantly reduced.

The AS_Path attribute actually serves two purposes. Because it lists the ASs that a datagram must traverse if it follows this route, the AS_Path information enables a router to perform policy routing. That is, a router may decide to avoid a particular path in order to avoid transiting a particular AS. For example, information that is confidential may be limited to certain kinds of ASs. Or, a router may have information about the performance or quality of the portion of the internet that is included in an AS that leads the router to avoid that AS. Examples of performance or quality metrics include link speed, capacity, tendency to become congested, and overall quality of operation. Another criterion that could be used is minimizing the number of transit ASs.

The reader may wonder about the purpose of the Next_Hop attribute. The requesting router will necessarily want to know which networks are reachable via the responding router, but why provide information about other routers? This is best explained with reference to Figure 16.11. In this example, router R1 in autonomous system 1, and router R5 in autonomous system 2, implement BGP and acquire a neighbor relationship. R1 issues Update messages to R5 indicating which networks it could reach and the distances (network hops) involved. R1 also provides the same information on behalf of R2. That is, R1 tells R5 what networks are reachable via R2. In this example, R2 does not implement BGP. Typically, most of the routers in an autonomous system will not implement BGP; only a few will be assigned responsibility for communicating with routers in other autonomous systems. A final point: R1 is in possession of the necessary information about R2, as R1 and R2 share an interior router protocol (IRP).

The second type of update information is the withdrawal of one or more routes. In each case, the route is identified by the IP address of the destination subnetwork.

Finally, the notification message is sent when an error condition is detected. The following errors may be reported:

- **Message header error.** Includes authentication and syntax errors.
- **Open message error.** Includes syntax errors and options not recognized in an Open message. This message can also be used to indicate that a proposed Hold Time in an Open message is unacceptable.
- **Update message error.** Includes syntax and validity errors in an Update message.
- **Hold timer expired.** If the sending router has not received successive Keepalive and/or Update and/or Notification messages within the Hold Time period, then this error is communicated and the connection is closed.
- **Finite state machine error.** Includes any procedural error.
- **Cease.** Used by a router to close a connection with another router in the absence of any other error.

BGP Routing Information Exchange

The essence of BGP is the exchange of routing information among participating routers in multiple ASs. This process can be quite complex. In what follows, we provide a simplified overview.

Let us consider router R1 in autonomous system 1 (AS1), in Figure 16.10. To begin, a router that implements BGP will also implement an internal routing protocol, such as OSPF. Using OSPF, R1 can exchange routing information with other routers within AS1 and build up a picture of the topology of the subnetworks and routers in AS1 and construct a routing table. Next, R1 can issue an Update message to R5 in AS2. The Update message could include the following:

- AS_Path: the identity of AS1
- Next_Hop: the IP address of R1
- NLRI: a list of all of the subnetworks in AS1

This message informs R5 that all of the subnetworks listed in NLRI are reachable via R1 and that the only autonomous system traversed is AS1.

Suppose now that R5 also has a neighbor relationship with another router in another autonomous system, say R9 in AS3. R5 will forward the information just received from R1 to R9 in a new Update message. This message includes the following:

- AS_Path: the list of identifiers {AS2, AS1}
- Next_Hop: the IP address of R5
- NLRI: a list of all of the subnetworks in AS1

This message informs R9 that all of the subnetworks listed in NLRI are reachable via R5 and that the autonomous systems traversed are AS2 and AS1. R9 must now decide if this is its preferred route to the subnetworks listed. It may have knowledge of an alternate route to some or all of these subnetworks that it prefers for reasons of performance or some other policy metric. If R9 decides that the route provided in R5's update message is preferable, then R9 incorporates that routing information into its routing database and forwards this new routing information to other neighbors. This new message will include an AS_Path field of {AS1, AS2, AS3}.

In the above fashion, routing update information is propagated through the larger internet consisting of a number of interconnected autonomous systems. The AS_Path field is used to assure that such messages do not circulate indefinitely: If an Update message is received by a router in an AS that is included in the AS_Path field, that router will not forward the update information to other routers, thereby preventing looping of messages.

The preceding discussion leaves out several details that are briefly summarized here. Routers within the same AS, called internal neighbors, may exchange BGP information. In this case, the sending router does not add the identifier of the common AS to the AS_Path field. When a router has selected a preferred route to an external destination, it transmits this route to all of its internal neighbors. Each of these routers then decides if the new route is preferred; if so, the new route is added to its database and a new Update message goes out.

When there are multiple entry points into an AS that are available to a border router in another AS, the Multi_Exit_Disc attribute may be used to choose among them. This attribute contains a number that reflects some internal metric for reaching destinations within an AS. For example, suppose in Figure 16.10 that both R1 and R2 implemented BGP and that both had a neighbor relationship with R5. Each provides an Update message to R5 for subnetwork 1.3 that includes a routing metric used internally to AS1, such as a routing metric associated with the OSPF internal router protocol. R5 could then use these two metrics as the basis for choosing between the two routes.

Open Shortest Path First (OSPF) Protocol

The history of interior routing protocols on the Internet mirrors that of packet-switching protocols on ARPANET. Recall that ARPANET began with a protocol based on the Bellman-Ford algorithm. The resulting protocol required each node to exchange path-delay information with its neighbors. Information about a change in network conditions would gradually ripple through the network. A second generation protocol was based on Dijkstra's algorithm and required each node to exchange link-delay information with all other nodes using flooding. It was found that this latter technique was more effective.

Similarly, the initial interior routing protocol on the DARPA internet was the Routing Information Protocol (RIP), which is essentially the same protocol as the first-generation ARPANET protocol. This protocol requires each router to transmit its entire routing table. Although the algorithm is simple and easy to implement, as the internet expands, routing updates grow larger and consume significantly more network bandwidth. Accordingly, OSPF operates in a fashion similar to the revised ARPANET routing algorithm. OSPF uses what is known as a link-state routing algorithm. Each router maintains descriptions of the state of its local links to subnetworks, and from time to time transmits updated state information to all of the routers of which it is aware. Every router receiving an update packet must acknowledge it to the sender. Such updates produce a minimum of routing traffic because the link descriptions are small and rarely need to be sent.

The OSPF protocol (RFC 1583) is now widely used as the interior router protocol in TCP/IP networks. OSPF computes a route through the internet that incurs the least cost based on a user-configurable metric of cost. The user can configure the cost to express a function of delay, data rate, dollar cost, or other factors. OSPF is able to equalize loads over multiple equal-cost paths.

Each router maintains a database that reflects the known topology of the autonomous system of which it is a part. The topology is expressed as a directed graph. The graph consists of

- Vertices, or nodes, of two types:
  1. router
  2. network, which is, in turn, of two types:
     a) transit, if it can carry data that neither originates nor terminates on an end system attached to this network.
     b) stub, if it is not a transit network.
- Edges of two types:
  1. graph edges that connect two router vertices when the corresponding routers are connected to each other by a direct point-to-point link.
  2. graph edges that connect a router vertex to a network vertex when the router is directly connected to the network.

Figure 16.12 shows an example of an autonomous system, and Figure 16.13 is the resulting directed graph. The mapping is straightforward:

- Two routers joined by a point-to-point link are represented in the graph as being directly connected by a pair of edges, one in each direction (e.g., routers 6 and 10).
- When multiple routers are attached to a network (such as a LAN or packet-switching network), the directed graph shows all routers bidirectionally connected to the network vertex (e.g., routers 1, 2, 3, and 4 all connect to network 3).
- If a single router is attached to a network, the network will appear in the graph as a stub connection (e.g., network 7).

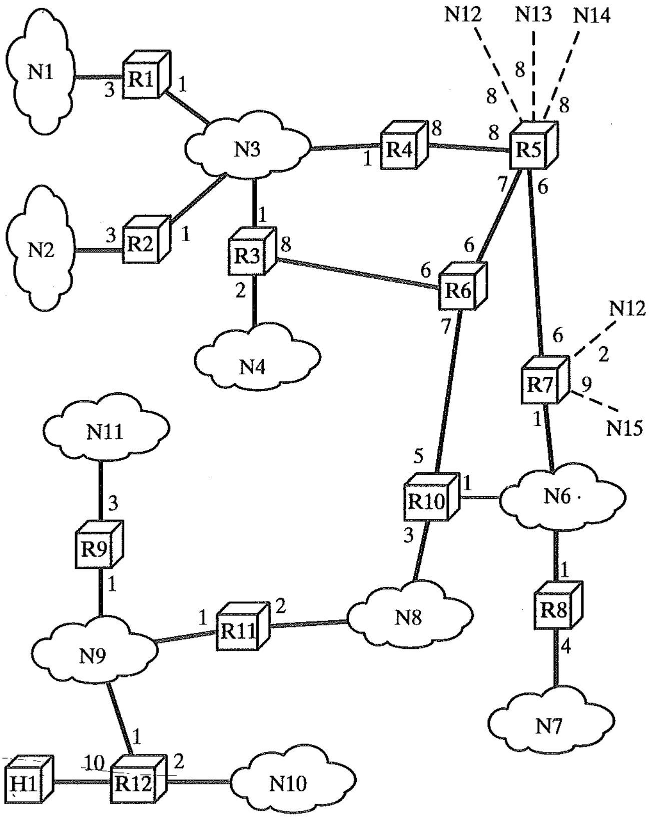


FIGURE 16.12 A sample autonomous system.

- An end system, called a host, can be directly connected to a router; such a case is depicted in the corresponding graph (e.g., host 1).
- If a router is connected to other autonomous systems, then the path cost to each network in the other system must be obtained by some exterior routing protocol (ERP). Each such network is represented on the graph by a stub and an edge to the router with the known path cost (e.g., networks 12 through 15).

A cost is associated with the output side of each router interface. This cost is configurable by the system administrator. Arcs on the graph are labeled with the cost of the corresponding router-output interface. Arcs having no labeled cost have a cost of 0. Note that arcs leading from networks to routers always have a cost of 0.

A database corresponding to the directed graph is maintained by each router. It is pieced together from link-state messages from other routers in the internet. Using Dijkstra's algorithm (see Appendix 9A), a router calculates the least-cost path to all destination networks. The result for router 6 of Figure 16.12 is shown as a tree in Figure 16.14, with R6 as the root of the tree. The tree gives the entire route to any destination network or host. However, only the next hop to the destination is used in the forwarding process. The resulting routing table for router 6 is shown in Table 16.5. The table includes entries for routers advertising external routes

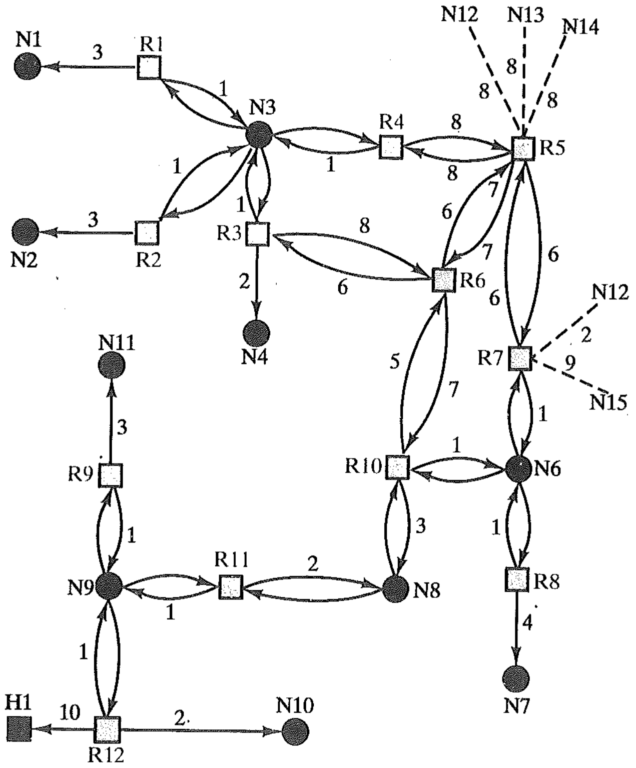


FIGURE 16.13 Directed graph of autonomous system of Figure 16.12.

TABLE 16.5 Routing Table for RT6.

Destination	Next hop	Distance
N1	RT3	10
N2	RT3	10
N3	RT3	7
N4	RT3	8
N6	RT10	8
N7	RT10	12
N8	RT10	10
N9	RT10	11
N10	RT10	13
N11	RT10	14
H1	RT10	21
RT5	RT5	6
RT7	RT10	8
N12	RT10	10
N13	RT5	14
N14	RT5	14
N15	RT10	17

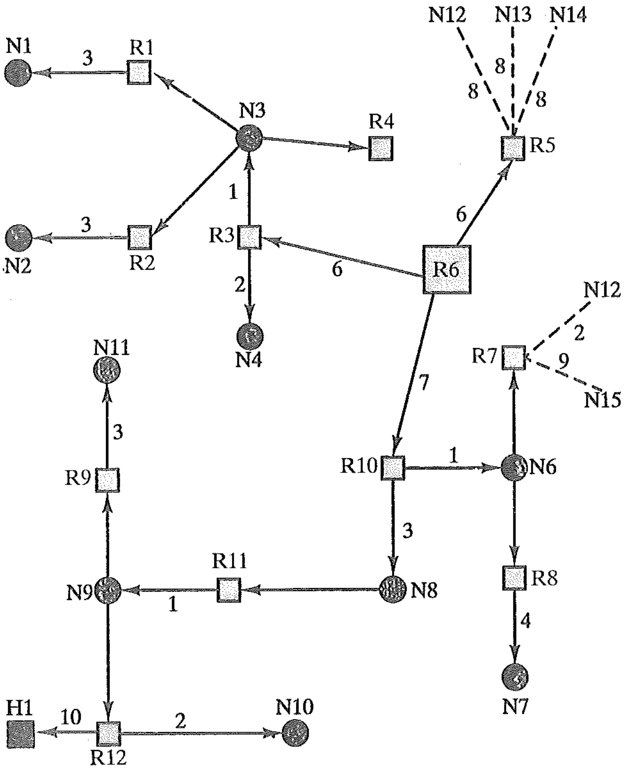


FIGURE 16.14 The SPF tree for router R6.

(routers 5 and 7). For external networks whose identity is known, entries are also provided.

16.5 IPv6 (IPNG)

The Internet Protocol (IP) has been the foundation of the Internet and of virtually all multivendor private internetworks. This protocol is reaching the end of its useful life, and a new protocol, known as IPv6 (IP version 6), has been defined to ultimately replace IP.[1]

We first look at the motivation for developing a new version of IP, and then examine some of its details.

[1] You may think a few versions in this narrative have been skipped. The currently deployed version of IP is actually IP version 4; previous versions of IP (1 through 3) were successively defined and replaced to reach IPv4. Version 5 was the number assigned to the Stream Protocol, a connection-oriented internet-level protocol—hence the use of the label, version 6.

IP Next Generation

The driving motivation for the adoption of a new version of IP was the limitation imposed by the 32-bit address field in IPv4. With a 32-bit address field, it is, in principle, possible to assign 2^{32} different addresses, which represents over 4 billion possibilities. One might think that this number of addresses was more than adequate to meet addressing needs on the Internet. However, in the late 1980s, it was perceived that there would be a problem, and this problem began to manifest itself in the early 1990s. Some of the reasons for the inadequacy of 32-bit addresses include

- The two-level structure of the IP address (network number, host number) is convenient but wasteful of the address space. Once a network number is assigned to a network, all of the host-number addresses for that network number are assigned to that network. The address space for that network may be sparsely used, but as far as the effective IP address space is concerned, if a network number is used, then all addresses within the network are used.
- The IP addressing model generally requires that a unique network number be assigned to each IP network whether or not it is actually connected to the Internet.
- Networks are proliferating rapidly. Most organizations boast multiple LANs, not just a single LAN system; wireless networks have rapidly assumed a major role; and the Internet itself has grown explosively for years.
- Growth of TCP/IP usage into new areas will result in a rapid growth in the demand for unique IP addresses. Examples include using TCP/IP for the interconnection of electronic point-of-sale terminals and for cable television receivers.
- Typically, a single IP address is assigned to each host. A more flexible arrangement is to allow multiple IP addresses per host; this, of course, increases the demand for IP addresses.

So, the need for an increased address space dictated that a new version of IP was needed. In addition, IP is a very old protocol, and new requirements in the areas of security, routing flexibility, and traffic support had been defined.

In response to these needs, the Internet Engineering Task Force (IETF) issued a call for proposals for a next generation IP (IPng) in July of 1992. A number of proposals were received, and by 1994, the final design for IPng emerged. A major milestone was reached with the publication of RFC 1752, "The Recommendation for the IP Next Generation Protocol," issued in January 1995. RFC 1752 outlines the requirements for IPng, specifies the PDU formats, and highlights the IPng approach in the areas of addressing, routing, and security. A number of other Internet documents defined details of the protocol, now officially called IPv6; these include an overall specification of IPv6 (RFC 1883), an RFC discussing the Flow Label in the IPv6 header (RFC 1809), and several RFCs dealing with addressing aspects of IPv6 (RFC 1884, RFC 1886, RFC 1887).

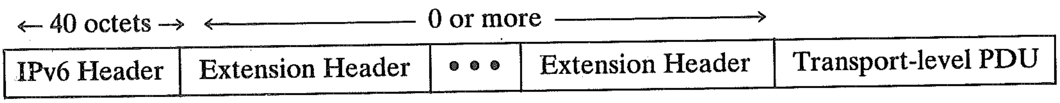
IPv6 includes the following enhancements over IPv4:

- **Expanded Address Space.** IPv6 uses 128-bit addresses instead of the 32-bit addresses of IPv4. This is an increase of address space by a factor of 2^{96}! It has been pointed out [HIND95] that this allows on the order of 6×10^{23} unique addresses per square meter of the surface of the earth! Even if addresses are very inefficiently allocated, this address space seems secure.
- **Improved Option Mechanism.** IPv6 options are placed in separate optional headers that are located between the IPv6 header and the transport-layer header. Most of these optional headers are not examined or processed by any router on the packet's path; this simplifies and speeds up router processing of IPv6 packets, as compared to IPv4 datagrams.[2] It also makes it easier to add additional options.
- **Address Autoconfiguration.** This capability provides for dynamic assignment of IPv6 addresses.
- **Increased Addressing Flexibility.** IPv6 includes the concept of an anycast address, for which a packet is delivered to just one of a set of nodes. The scalability of multicast routing is improved by adding a scope field to multicast addresses.
- **Support for Resource Allocation.** Instead of the type-of-service field in IPv4, IPv6 enables the labeling of packets belonging to a particular traffic flow for which the sender requests special handling; this aids in the support of specialized traffic, such as real-time video.
- **Security Capabilities.** IPv6 includes features that support authentication and privacy.

All of these features are explored in the remainder of this section, except for the security features, which are discussed in Chapter 18.

IPv6 Structure

An IPv6 protocol data unit (known as a packet) has the following general form:



The only header that is required is referred to simply as the IPv6 header. This is of fixed size with a length of 40 octets, compared to 20 octets for the mandatory portion of the IPv4 header (Figure 16.7). The following extension headers have been defined:

- **Hop-by-Hop Options Header.** Defines special options that require hop-by-hop processing.

[2] The protocol data unit for IPv6 is referred to as a packet rather than as a datagram, which is the term used for IPv4 PDUs.

- **Routing Header.** Provides extended routing, similar to IPv4 source routing.
- **Fragment Header.** Contains fragmentation and reassembly information.
- **Authentication Header.** Provides packet integrity and authentication.
- **Encapsulating Security Payload Header.** Provides privacy.
- **Destination Options Header.** Contains optional information to be examined by the destination node.

The IPv6 standard recommends that, when multiple extension headers are used, the IPv6 headers appear in the following order:

1. IPv6 header: Mandatory, must always appear first.
2. Hop-by-Hop Options header.
3. Destination Options header: For options to be processed by the first destination that appears in the IPv6 Destination Address field plus subsequent destinations listed in the Routing header.
4. Routing Header.
5. Fragment Header.
6. Authentication Header.
7. Encapsulating Security Payload header.
8. Destination Options header: For options to be processed only by the final destination of the packet.

Figure 16.15 shows an example of an IPv6 packet that includes an instance of each header. Note that the IPv6 header and each extension header include a Next Header field. This field identifies the type of the immediately following header. If the next header is an extension header, then this field contains the type identifier of that header; otherwise, this field contains the protocol identifier of the upper-layer protocol using IPv6 (typically a transport-level protocol), using the same values as the IPv4 Protocol field. In the figure, the upper-layer protocol is TCP, so that the upper-layer data carried by the IPv6 packet consists of a TCP header followed by a block of application data.

We first look at the main IPv6 header, and then examine each of the extensions in turn.

IPv6 Header

The IPv6 header has a fixed length of 40 octets, consisting of the following fields (Figure 16.16):

- **Version (4 bits).** Internet Protocol version number; the value is 6.
- **Priority (4 bits).** Priority value, discussed below.
- **Flow Label (24 bits).** May be used by a host to label those packets for which it is requesting special handling by routers within a network; discussed below.
- **Payload Length (16 bits).** Length of the remainder of the IPv6 packet following the header, in octets. In other words, this is the total length of all of the extension headers plus the transport-level PDU.

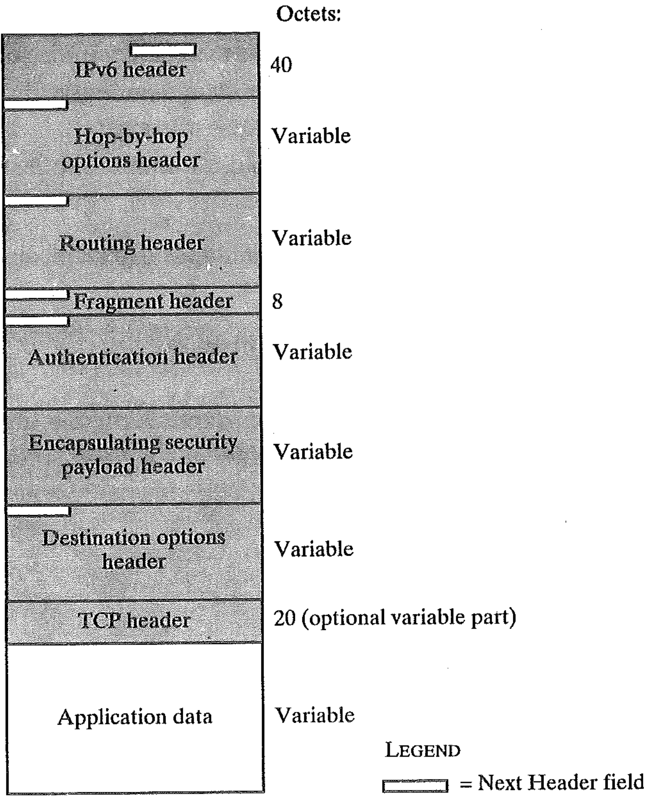


FIGURE 16.15 IPv6 packet with all extension headers.

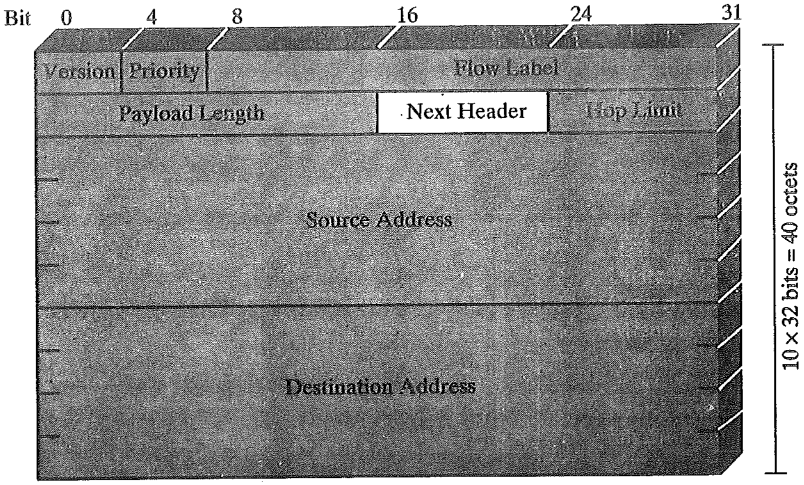


FIGURE 16.16 IPv6 header.

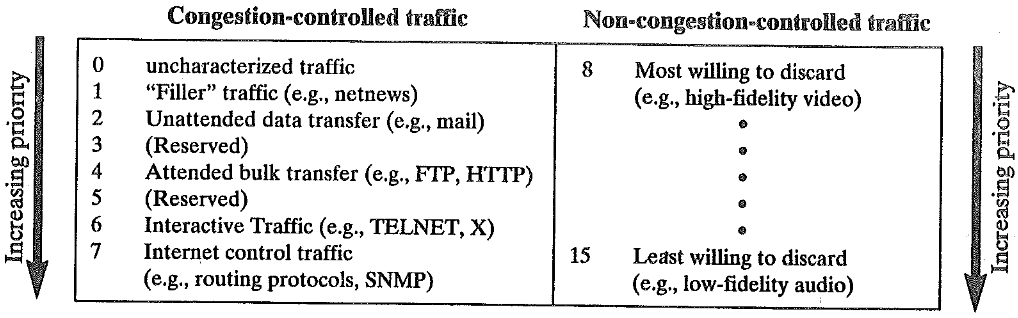
- **Next Header (8 bits).** Identifies the type of header immediately following the IPv6 header.
- **Hop Limit (8 bits).** The remaining number of allowable hops for this packet. The hop limit is set to some desired maximum value by the source, and decremented by 1 by each node that forwards the packet. The packet is discarded if Hop Limit is decremented to zero; this is simpler than the processing required for the time-to-live field of IPv4. The consensus was that the extra effort in accounting for time intervals in IPv4 added no significant value to the protocol.
- **Source Address (128 bits).** The address of the originator of the packet.
- **Destination Address (128 bits).** The address of the intended recipient of the packet. This may not in fact be the intended ultimate destination if a Routing header is present, as explained below.

Although the IPv6 header is longer than the mandatory portion of the IPv4 header (40 octets versus 20 octets), it contains fewer fields (8 versus 12). Thus, routers have less processing to do per header, which should speed up routing.

Priority Field

The 4-bit priority field enables a source to identify the desired transmit and delivery priority of each packet relative to other packets from the same source. In fact, the field enables the source to specify separate priority-related characteristics for each packet. First, packets are classified as being part of either traffic for which the source is providing congestion control or traffic for which the source is not providing congestion control; and second, packets are assigned one of eight levels of relative priority within each classification. Figure 16.17 illustrates the interpretation of priority-field values.

Congestion-controlled traffic refers to traffic for which the source "backs off" in response to congestion; an example is TCP. Let us consider what this means. If there is congestion in the network, TCP segments will take longer to arrive at the destination, and, hence, acknowledgments from the destination back to the source will take longer. As congestion increases, it becomes necessary for segments to be discarded en route. The discarding could be done by a router when that router experiences buffer overflow, or it could be done by an individual network along the


Increasing priority
Increasing priority

Congestion-controlled traffic		Non-congestion-controlled traffic	
0	uncharacterized traffic	8	Most willing to discard (e.g., high-fidelity video)
1	"Filler" traffic (e.g., netnews)		•
2	Unattended data transfer (e.g., mail)		•
3	(Reserved)		•
4	Attended bulk transfer (e.g., FTP, HTTP)		•
5	(Reserved)		•
6	Interactive Traffic (e.g., TELNET, X)		
7	Internet control traffic (e.g., routing protocols, SNMP)	15	Least willing to discard (e.g., low-fidelity audio)

FIGURE 16.17 IPv6 priorities.

route, when a switching node within the network becomes congested. Whether it is a data segment or an acknowledgment segment, the effect is that TCP does not receive an acknowledgment to its transmitted data segment. TCP responds by retransmitting the segment and by reducing the flow of segments that it generates.

The nature of congestion-controlled traffic is that it is acceptable for there to be a variable amount of delay in the delivery of packets and even for packets to arrive out of order. IPv6 defines the following categories of congestion-controlled traffic, in order of decreasing priority:

- **Internet Control Traffic.** This is the most important traffic to deliver, especially during times of high congestion. For example, routing protocols, such as OSPF and BGP, need to receive updates concerning traffic conditions so that they can adjust routes to try to relieve congestion. Management protocols, such as SNMP, need to be able to report congestion to network-management applications, as well as to be able to perform dynamic reconfiguration and to alter performance-related parameters to cope with congestion.
- **Interactive Traffic.** After Internet-control traffic, the most important traffic to support is interactive traffic, such as on-line user-to-host connections. User efficiency is critically dependent on rapid response time during interactive sessions, so delay must be minimized.
- **Attended Bulk Transfer.** These are applications that may involve the transfer of a large amount of data and for which the user is generally waiting for the transfer to complete. This category differs from interactive traffic in the sense that the user is prepared to accept a longer delay for the bulk transfer than experienced during an interactive dialogue; a good example of this is FTP. Another example is hypertext transfer protocol (HTTP), which supports web browser-server interaction (discussed in Chapter 19).[3]
- **Unattended Data Transfer.** These are applications that are initiated by a user but which are not expected to be delivered instantly. Generally, the user will not wait for the transfer to be complete, but will go on to other tasks. The best example of this category is electronic mail.
- **Filler Traffic.** This is traffic that is expected to be handled in the background, when other forms of traffic have been delivered. USENET messages are good examples.
- **Uncharacterized Traffic.** If the upper-layer application gives IPv6 no guidance about priority, then the traffic is assigned this lowest-priority value.

Non-congestion-control traffic is traffic for which a constant data rate and a constant delivery delay are desirable; examples are real-time video and audio. In these cases, it makes no sense to retransmit discarded packets, and, further, it is important to maintain a smooth delivery flow. Eight levels of priority are allocated for this type of traffic from the highest priority 8 (most willing to discard) to the low-

[3] In the case of HTTP, the duration of the underlying connection may be too brief for the sender to receive feedback and therefore HTTP may not do congestion control.

est priority 15 (least willing to discard). In general, the criterion is how much the quality of the received traffic will deteriorate in the face of some dropped packets. For example, low-fidelity audio, such as a telephone voice conversation, would typically be assigned a high priority. The reason is that the loss of a few packets of audio is readily apparent as clicks and buzzes on the line. On the other hand, a high-fidelity video signal contains a fair amount of redundancy, and the loss of a few packets will probably not be noticeable; therefore, this traffic is assigned a relatively low priority.

Please note that there is no priority relationship implied between the congestion-controlled priorities on the one hand and the non–congestion-controlled priorities on the other hand. Priorities are relative only within each category.

Flow Label

The IPv6 standard defines a flow as a sequence of packets sent from a particular source to a particular (unicast or multicast) destination, for which the source desires special handling by the intervening routers. A flow is uniquely identified by the combination of a source address and a nonzero 24-bit flow label. Thus, all packets that are to be part of the same flow are assigned the same flow label by the source.

From the source's point of view, a flow, typically, will be a sequence of packets that are generated from a single application instance at the source and that have the same transfer service requirements. A flow may comprise a single TCP connection or even multiple TCP connections; an example of the latter is a file transfer application, which could have one control connection and multiple data connections. A single application may generate a single flow or multiple flows; an example of the latter is multimedia conferencing, which might have one flow for audio and one for graphic windows, each with different transfer requirements in terms of data rate, delay, and delay variation.

From the router's point of view, a flow is a sequence of packets that share attributes that affect how those pockets are handled by the router; these include path, resource allocation, discard requirements, accounting, and security attributes. The router may treat packets from different flows differently in a number of ways, including allocating different buffer sizes, giving different precedence in terms of forwarding, and requesting different qualities of service from subnetworks.

There is no special significance to any particular flow label; instead, the special handling to be provided for a packet flow must be declared in some other way. For example, a source might negotiate or request special handling ahead of time, from routers by means of a control protocol, or at transmission time, by information in one of the extension headers in the packet, such as the Hop-by-Hop Options header. Examples of special handling that might be requested are some non-default quality of service and a form of real-time service.

In principle, all of a user's requirements for a particular flow could be defined in an extension header and included with each packet. If we wish to leave the concept of flow open to include a wide variety of requirements, this design approach could result in very large packet headers. The alternative, adopted for IPv6, is the flow label, in which the flow requirements are defined prior to flow commencement and in which a unique flow label is assigned to the flow. In this case, the router must save flow-requirement information about each flow.

The following rules apply to the flow label:

1. Hosts or routers that do not support the Flow Label field must set the field to zero when originating a packet, pass the field unchanged when forwarding a packet, and ignore the field when receiving a packet.
2. All packets originating from a given source with the same non-zero Flow Label must have the same Destination Address, Source Address, Hop-by-Hop Options header contents (if this header is present), and Routing header contents (if this header is present). The intent is that a router can decide how to route and process the packet by simply looking up the flow label in a table, without having to examine the rest of the header.
3. The source assigns a flow label to a flow. New flow labels must be chosen (pseudo-) randomly and uniformly in the range 1 to 2^{24}–1, subject to the restriction that a source must not re-use a flow label for a new flow within the lifetime of the existing flow. The zero flow label is reserved to indicate that no flow label is being used.

This last point requires some elaboration. The router must maintain information about the characteristics of each active flow that may pass through it, presumably in some sort of table. In order to be able to forward packets efficiently and rapidly, table lookup must be efficient. One alternative is to have a table with 2^{24} (about 16 million) entries, one for each possible flow label; this imposes an unnecessary memory burden on the router. Another alternative is to have one entry in the table per active flow, include the flow label with each entry, and require the router to search the entire table each time a packet is encountered; this imposes an unnecessary processing burden on the router. Instead, most router designs are likely to use some sort of hash-table approach. With this approach, a moderate-sized table is used, and each flow entry is mapped into the table using a hashing function on the flow label. The hashing function might simply be the low-order few bits (say 10 or 12) of the flow label or some simple calculation on the 24 bits of the flow label. In any case, the efficiency of the hash approach typically depends on the flow labels being uniformly distributed over their possible range—hence requirement number 3 listed above.

IPv6 Addresses

IPv6 addresses are 128 bits in length. Addresses are assigned to individual interfaces on nodes, not to the nodes themselves.[4] A single interface may have multiple unique unicast addresses. Any of the unicast addresses associated with a node's interface may be used to uniquely identify that node.

The combination of long addresses and multiple addresses per interface enables improved routing efficiency over IPv4. In IPv4, addresses generally do not have a structure that assists routing, and, therefore, a router may need to maintain huge tables of routing paths. Longer internet addresses allow for aggregating addresses by hierarchies of network, access provider, geography, corporation, and so on. Such aggregation should make for smaller routing tables and faster table

[4] In IPv6, a node is any device that implements IPv6; this includes hosts and routers.

look-ups. The allowance for multiple addresses per interface would allow a subscriber that uses multiple access providers across the same interface to have separate addresses aggregated under each provider's address space.

The first field of any IPv6 address is the variable-length Format Prefix, which identifies various categories of addresses. Table 16.6 indicates the current allocation of addresses based on the Format Prefix, and Figure 16.18 shows some IPv6 address formats.

TABLE 16.6 Address allocation.

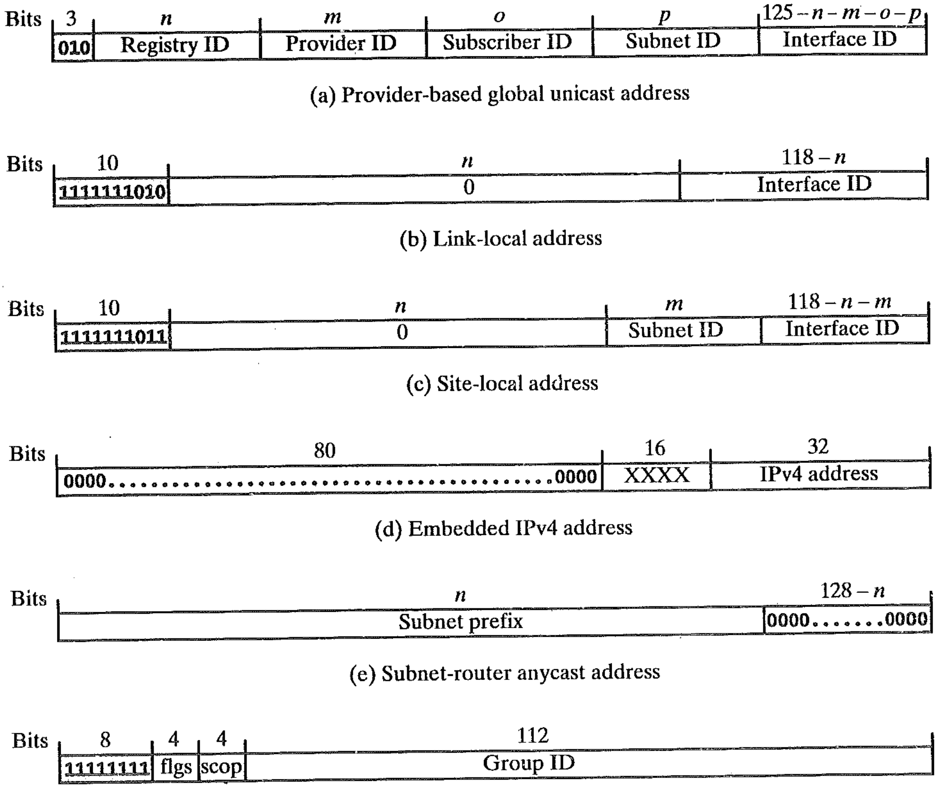
Allocation space	Prefix (binary)	Fraction of address space
Reserved	0000 0000	1/256
Unassigned	0000 0001	1/256
Reserved for NSAP Allocation	0000 001	1/128
Reserved for IPX Allocation	0000 010	1/128
Unassigned	0000 011	1/128
Unassigned	0000 1	1/32
Unassigned	0001	1/16
Unassigned	001	1/8
Provider-Based Unicast Address	010	1/8
Unassigned	011	1/8
Reserved for Geographic-Based Unicast Addresses	100	1/8
Unassigned	101	1/8
Unassigned	110	1/8
Unassigned	1110	1/16
Unassigned	1111 0	1/32
Unassigned	1111 10	1/64
Unassigned	1111 110	1/128
Unassigned	1111 1110 0	1/512
Link Local Use Addresses	1111 1110 10	1/1024
Site Local Use Addresses	1111 1110 11	1/1024
Multicast Addresses	1111 1111	1/256

IPv6 allows three types of addresses:

- **Unicast.** An identifier for a single interface. A packet sent to a unicast address is delivered to the interface identified by that address.
- **Anycast.** An identifier for a set of interfaces (typically belonging to different nodes). A packet sent to an anycast address is delivered to one of the interfaces identified by that address (the "nearest" one, according to the routing protocols' measure of distance).
- **Multicast.** An identifier for a set of interfaces (typically belonging to different nodes). A packet sent to a multicast address is delivered to all interfaces identified by that address.

Unicast Addresses

Unicast addresses may be structured in a number of ways. The following have been identified:



(a) Provider-based global unicast address

(b) Link-local address

(c) Site-local address

(d) Embedded IPv4 address

(e) Subnet-router anycast address

(f) Multicast address

FIGURE 16.18 IPv6 address formats.

- Provider-Based Global
- Link-Local
- Site-Local
- Embedded IPv4
- Loopback

A **provider-based global unicast address** provides for global addressing across the entire universe of connected hosts. The address has five fields after the Format Prefix (Figure 16.18a):

- **Registry ID.** Identifies the registration authority, which assigns the provider portion of the address.
- **Provider ID.** A specific internet service provider, which assigns the subscriber portion of the address.
- **Subscriber ID.** Distinguishes among multiple subscribers attached to the provider portion of the address.

- **Subnet ID.** A topologically connected group of nodes within the subscriber network.
- **Node ID.** Identifies a single node interface among the group of interfaces identified by the subnet prefix.

At present, there is no fixed length assigned to any of these fields. However, from the point of view of a network manager or network designer responsible for a subscriber's installation, the Subnet ID and Node ID are the principal concern. The subscriber can deal with these fields in a number of ways. One possibility for a LAN-based installation is to use a 48-bit node field and use the IEEE 802 medium access control (MAC) address for that field. The remainder of the available bits would then be the Subnet ID field, identifying a particular LAN at that subscriber site.

IPv6 accommodates local-use unicast addresses. Packets with such addresses can only be routed locally—that is, within a subnetwork or set of subnetworks of a given subscriber.

Two types of local-use addresses have been defined: link-local and site-local.

Link-local addresses are to be used for addressing on a single link or subnetwork (Figure 16.18b). They cannot be integrated into the global addressing scheme. Examples of their use include auto-address configuration, discussed presently, and neighbor discovery.

Site-local addresses are designed for local use but formatted in such a way that they can later be integrated into the global address scheme. The advantage of such addresses is that they can be used immediately by an organization that expects to transition to the use of global addresses. The structure of these addresses (Figure 16.18c) is such that the meaningful portion of the address is confined to the low-order bits not used for global addresses. The remaining bits consist of a local-use Format Prefix (1111 1101 10 or 1111 1101 11), followed by a zero field. When an organization is ready for global connection, these remaining bits can be replaced with a global prefix (e.g., 010 + Registry ID + Provider ID + Subscriber ID + ID).

A key issue in deploying IPv6 is the transition from IPv4 to IPv6. It is not practical to simply replace all IPv4 routers in the Internet or a private internet with IPv6 routers and then replace all IPv4 addresses with IPv6 addresses. Instead, there will be a lengthy transition period when IPv6 and IPv4 must coexist, and, therefore, IPv6 addresses and IPv4 addresses must coexist. *Embedded IPv4 addresses* accommodate this coexistence period. An embedded IPv4 address consists of a 32-bit IPv4 address in the lower-order 32 bits prefixed by either 96 zeroes or prefixed by 80 zeros followed by 16 ones (Figure 16.18d). The first form is known as an IPv4-compatible IPv6 address, and the latter as an IPv4-mapped IPv6 address.

During the transition, the following types of devices will be supported:

- Dual-use routers: These are capable of routing both IPv6 and IPv4 packets.
- Dual-use hosts: These implement both IPv6 and IPv4 and have both IPv6 and IPv4 addresses; the IPv6 address is an *IPv4-compatible IPv6 address.*
- IPv4-only routers: Can only recognize and route IPv4 packets and only know about hosts with IPv4 addresses.

- IPv4-only hosts: Implement only IPv4 and have only an IPv4 address. This address can be represented in the IPv6 realm using an *IPv4-mapped IPv6 address.*

With this configuration, full interoperability is maintained. The details of this interoperability are rather complex (see [GILL95] for a discussion). However, the following general principles apply:

1. Traffic generated by IPv4-only hosts. If the traffic passes through an IPv6 portion of the Internet, a dual-use router must convert the IPv4 destination address to an embedded IPv4 address of the appropriate type and then construct the IPv6 header.
2. Traffic generated by a dual-use host. A technique known as tunneling may be employed to forward such traffic through IPv4 routing topologies. In essence, an IPv6 packet is encapsulated in an IPv4 datagram.

Full coexistence can be maintained so long as all IPv6 nodes employ an IPv4-compatible address. Once more general IPv6 addresses come into use, coexistence will be more difficult to maintain.

The unicast address 0:0:0:0:0:0:0:1 is called the *loopback address.*[5] It may be used by a node to send an IPv6 packet to itself; such packets are not to be sent outside a single node.

Anycast Addresses

An anycast address enables a source to specify that it wants to contact any one node from a group of nodes via a single address. A packet with such an address will be routed to the nearest interface in the group, according to the router's measure of distance. A possible use of an anycast address might be found within a routing header to specify an intermediate address along a route. The anycast address could refer to the group of routers associated with a particular provider or particular subnet, thus dictating that the packet be routed through that provider or internet in the most efficient manner.

Anycast addresses are allocated from the same address space as unicast addresses. Thus, members of an anycast group must be configured to recognize that addresses and routers must be configured to be able to map an anycast address to a group of unicast interface addresses.

One particular form of anycast address, the *subnet-router anycast address,* is predefined (Figure 16.18e). The subnet prefix field identifies a specific subnetwork. For example, in a provider-based global address space, the subnet prefix is of the form (010 + Registry ID + Provider ID + Subscriber ID + Subnet ID). Thus, the anycast address is identical to a unicast address for an interface on this subnetwork, with the interface ID portion set to zero. Any packet sent to this address will be

[5] The preferred form for representing IPv6 addresses as text strings is x:x:x:x:x:x:x:x, where each x is the hexadecimal value of a 16-bit portion of the address; it is not necessary to show leading zeros, but each position must have at least one digit.

delivered to one router on the subnetwork; all that is required is to insert the correct interface ID into the anycast address to form the unicast destination address.

Multicast Addresses

IPv6 includes the capability of addressing a predefined group of interfaces with a single multicast address. A packet with a multicast address is to be delivered to all members of the group; this address consists of an 8-bit format prefix of all ones, a 4-bit flags field, a 4-bit scope files, and a 112-bit group ID.

At present, the flags field consists of 3 zero bits followed by a T bit with

- T = 0: Indicates a permanently-assigned, or well-known, multicast address, assigned by the global-internet numbering authority.
- T = 1: Indicates a non-permanently-assigned, or transient, multicast address.

The scope value is used to limit the scope of the multicast group. The values are

0	reserved	4	unassigned	8	organization-local	12	unassigned
1	node-local	5	site-local	9	unassigned	13	unassigned
2	link-local	6	unassigned	10	unassigned	14	global
3	unassigned	7	unassigned	11	unassigned	15	reserved

The group ID identifies a multicast group, either permanent or transient, within the given scope. In the case of a permanent multicast address, the address itself is independent of the scope field, but that field limits the scope of addressing for a particular packet. For example, if the "NTP servers group" is assigned a permanent multicast address with a group ID of 43 (hex), then

FF05:0:0:0:0:0:0:43 means all NTP servers at the same site as the sender.

FF0E:0:0:0:0:0:0:43 means all NTP servers in the internet.

Non-permanently-assigned multicast addresses are only meaningful within a given scope, thereby enabling the same group ID to be reused, with different interpretations, at different sites.

Multicasting is a useful capability in a number of contexts. For example, it allows hosts and routers to send neighbor discovery messages only to those machines that have registered to receive them, removing the necessity for all other machines to examine and discard irrelevant packets. Another example is that most LANs provide a natural broadcast capability. Therefore, a multicast address can be assigned that has a scope of link-local and with a group ID that is configured on all nodes on the LAN to be a subnet broadcast address.

Address Autoconfiguration

Typically, in an IPv4 environment, users or network managers must manually configure IPv4 addresses on nodes. This is a task that is error-prone if done by individual users and time-consuming if done by network managers. It also requires that every node be manually reconfigured in order to change network addresses. Address autoconfiguration is a feature defined as part of the IPv6 specification that enables a host to configure automatically one or more addresses per interface. The aim of this feature is to support a "plug-and-play" capability, which will allow a user to attach a host to a subnetwork and have IPv6 addresses automatically assigned to its interfaces, with no user intervention.

Three models of address assignment have been defined: local scope, stateless server, and stateful server. The **local scope model** is designed for use on a network without routers, isolated from other networks. In this case, the IPv6 address could simply be the MAC address (FE:[MAC]), or it could be an address assigned by some simple dynamic address assignment capability; this would be a link-local address (Figure 16.18b) that is not useful for later integration into the global environment.

With the **stateless server model**, a new device sends a request packet to a local well-known multicast address. This request includes something to identify the device, such as its MAC address or some other key value. An address server receives this request, constructs an IPv6 address for the device based on its knowledge of the network, and sends the address back to the device in a response packet. The address server need not retain any memory of the transaction. This is a very simple configuration mechanism and requires minimal system administration support.

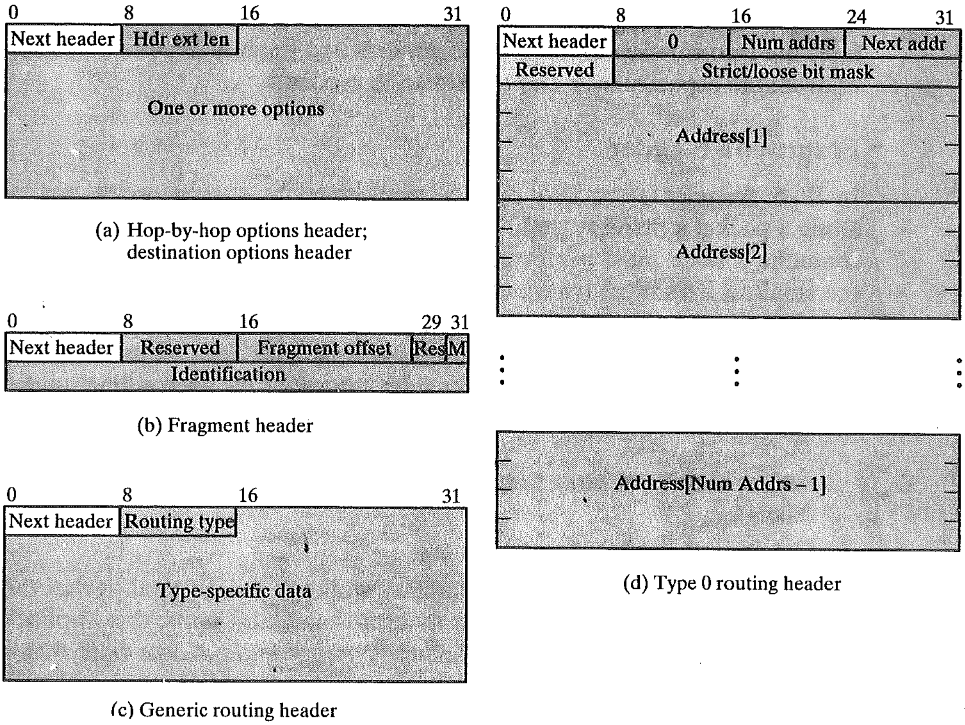
The **stateful server model** supports greater administrative system control by retaining information about address-assignment transactions. In this case, the MAC address or other key value received in a request packet is used to look up information in a database and create a new IPv6 address. This new address is given a lifetime value, after which the host must request again. The result is that subnetwork-wide address reconfiguration is more easily supported.

Hop-by-Hop Options Header

The hop-by-hop options header carries optional information that, if present, must be examined by every router along the path. This header consists of (Figure 16.19a):

- **Next Header (8 bits).** Identifies the type of header immediately following this header.
- **Header Extension Length (8 bits).** Length of this header in 64-bit units, not including the first 64 bits.
- **Options.** A variable-length field consisting of one or more option definitions. Each definition is in the form of three subfields: *option type* (8 bits), which identifies the option; *length* (8 bits), which specifies the length of the *option data* field in octets; and *option data*, which is a variable-length specification of the option.

It is actually the lowest-order five bits of the option type field that are used to specify a particular option. The high-order two bits indicate the particular action to be taken by a node that does not recognize this option type, as follows:



(a) Hop-by-hop options header; destination options header

(b) Fragment header

(c) Generic routing header

(d) Type 0 routing header

FIGURE 16.19 IPv6 extension headers.

00 Skip over this option and continue processing the header.

01 Discard the packet.

10 Discard the packet and send an ICMP parameter problem, code 2, message to the packet's source address, pointing to the unrecognized option type.

11 Discard the packet and, only if the packet's Destination Address is not a multicast address, send an ICMP parameter problem, code 2, message to the packet's source address, pointing to the unrecognized option type.

The third highest-order bit specifies whether the Option Data field does not change (0) or may change (1) en route from source to destination. Data that may change must be excluded from authentication calculations, as discussed in Section 18.4.

These conventions for the option type field also apply to the destination options header.

In the IPv6 standard, only one option is so far specified: the *jumbo payload* option, used to send IPv6 packets with payloads longer than $2^{16} = 65{,}536$ octets. The option data field of this option is 32 bits long and gives the length of the packet in octets, excluding the IPv6 header. For such packets, the payload length field in the IPv6 header must be set to zero, and there must be no fragment header. With this option, IPv6 supports packet sizes up to more than 4 billion octets; this facilitates the transmission of large video packets and enables IPv6 to make the best use of available capacity over any transmission medium.

Fragment Header

In IPv6, fragmentation may only be performed by source nodes, not by routers along a packet's delivery path. To take full advantage of the internetworking environment, a node must perform a path-discovery algorithm that enables it to learn the smallest maximum transmission unit (MTU) supported by any subnetwork on the path. With this knowledge, the source node will fragment, as required, for each given destination address. Otherwise the source must limit all packets to 576 octets, which is the minimum MTU that must be supported by each subnetwork.

The fragment header consists of (Figure 16.19b) the following:

- **Next Header (8 bits).** Identifies the type of header immediately following this header.
- **Reserved (8 bits).** For future use.
- **Fragment Offset (13 bits).** Indicates where in the original packet the payload of this fragment belongs. It is measured in 64-bit units; this implies that fragments (other than the last fragment) must contain a data field that is a multiple of 64 bits.
- **Res (2 bits).** Reserved for future use.
- **M Flag (1 bit).** 1 = more fragments; 0 = last fragment.
- **Identification (32 bits).** Intended to uniquely identify the original packet. The identifier must be unique for the packet's source address and destination address, for the time during which the packet will remain in the internet.

The fragmentation algorithm is the same as that described in Section 16.1.

Routing Header

The routing header contains a list of one or more intermediate nodes to be visited on the way to a packet's destination. All routing headers start with an 8-bit next-header field and an 8-bit routing-type field, followed by routing data specific to a given routing type (Figure 16.19c). If a router does not recognize the routing-type value, it must discard the packet. Type-0 routing has been defined and has the following fields (Figure 16.19d):

- **Next Header (8 bits).** Identifies the type of header immediately following this header.
- **Routing Type (8 bits).** Set to zero.
- **Num Addrs (8 bits).** Number of addresses in the routing header; the maximum value is 23.
- **Next Addr (8 bits).** Index of the next address to be processed; initialized to zero by the originating node.

- **Reserved (8 bits).** For future use.
- **Strict/Loose Bit Mask (24 bits).** Numbered from left to right (bit 0 through bit 23), with each bit corresponding to one of the hop. Each bit indicates whether the corresponding next destination address must be a neighbor of the preceding address (1 = strict, must be a neighbor; 0 = loose, need not be a neighbor).

When using the routing header, the source node does not place the ultimate destination address in the IPv6 header. Instead, that address is the last one listed in the routing header (Address[Num Addr - a] in Figure 16.19d), and the IPv6 header contains the destination address of the first desired router on the path. The routing header will not be examined until the packet reaches the node identified in the IPv6 header. At that point, the packet IPv6 and routing header contents are updated, and the packet is forwarded.

IPv6 requires an IPv6 node to reverse routes in a packet it receives containing a routing header, in order to return a packet to the sender; with this in mind, let us consider a set of examples from [HIND95], which illustrate the power and flexibility of the routing header feature. Figure 16.20 shows a configuration in which two hosts are connected by two providers, and the two providers are in turn connected by a wireless network. In the scenarios to follow, address sequences are shown in the following format:

$$\text{SRC}, I_1, I_2, \ldots \text{I}_n, \text{DST}$$

where SRC is the source address listed in the IPv6 header, I_i is the ith intermediate address, and DST is the ultimate destination. If there is no routing header, DST appears in the IPv6 header; otherwise, it is the last address in the routing header. Now consider three scenarios in which H1 sends a packet to H2.

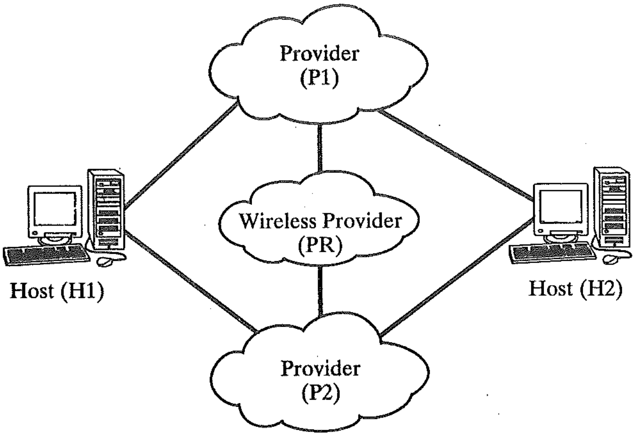


FIGURE 16.20 Example routing configuration.

1. No routing header is used. H1 sends a packet to H2 containing the sequence (H1, H2). H2 replies with a packet containing (H2, H1). In this case, either provider could be used on each transfer.
2. H1 wants to enforce a policy that all traffic between itself and H2 can only use P1. It constructs a packet with the sequence (H1, P1, H2) and dictates strict routing; this ensures that when H2 replies to H1, it will use the sequence (H2, P1, H1) with strict routing, thus enforcing H1's policy of using P1.
3. H1 becomes mobile and moves to provider PR. It could maintain communication (not breaking any TCP connections) with H2 by sending packets with (H1, PR, P1, H2); this would ensure that H2 would enforce the policy of using P1 by replying with (H2, PR, P1, H1).

These examples show the ability in IPv6 to select a particular provider, to maintain connections while mobile, and to route packets to new addresses dynamically.

Destination Options Header

The *destination options* header carries optional information that, if present, is examined only by the packet's destination node. The format of this header is the same as that of the hop-by-hop options header (Figure 16.19a).

16.6 ICMPv6

As with IPv4, IPv6 requires the use of ICMP, the original version of which does not meet all of the needs for IPv6. Hence, a new version, known as ICMPv6, has been specified (RFC 1885). Key features of ICMPv6 are as follows:

- ICMPv6 uses a new protocol number to distinguish it from the ICMP that works with IPv4.
- Both protocols use the same header format.
- Some little-used ICMP messages have been omitted from ICMPv6.
- The maximum size of ICMPv6 messages is larger (576 octets, including IPv6 headers) so as to exploit the increased size of packets that IPv6 guarantees will be transmitted without fragmentation.

As with ICMP for IPv4, ICMPv6 provides a means for transferring error messages and informational messages among IPv6 nodes. In most cases, an ICMPv6 message is sent in response to an IPv6 packet, either by a router along the packet's path or by the intended destination node. ICMPv6 messages are carried encapsulated in an IPv6 packet.

All ICMPv6 messages begin with the same common header fields:

- **Type (8 bits).** Identifies the type of ICMPv6 message. Currently, there are four types of error messages and five types of informational messages.

- **Code (8 bits).** Used to specify parameters of the message that can be encoded in a few bits.
- **Checksum (16 bits).** The 16-bit one's complement of the one's complement sum of the entire ICMPv6 message, plus a pseudo-header, described next.
- **Parameters (32 bits).** Used to specify more lengthy parameters.

The checksum field applies to the entire ICMPv6 message, plus a pseudo-header prefixed to the header at the time of calculation (at both transmission and reception). The pseudo-header includes the following fields from the IPv6 header: source and destination addresses, payload length, and the next-header field (Figure 16.21). By including the pseudo-header, ICMPv6 protects itself from misdelivery by IP. That is, if IP delivers a message to the wrong host, even if the message contains no bit errors, the receiving ICMPv6 entity will detect the delivery error.

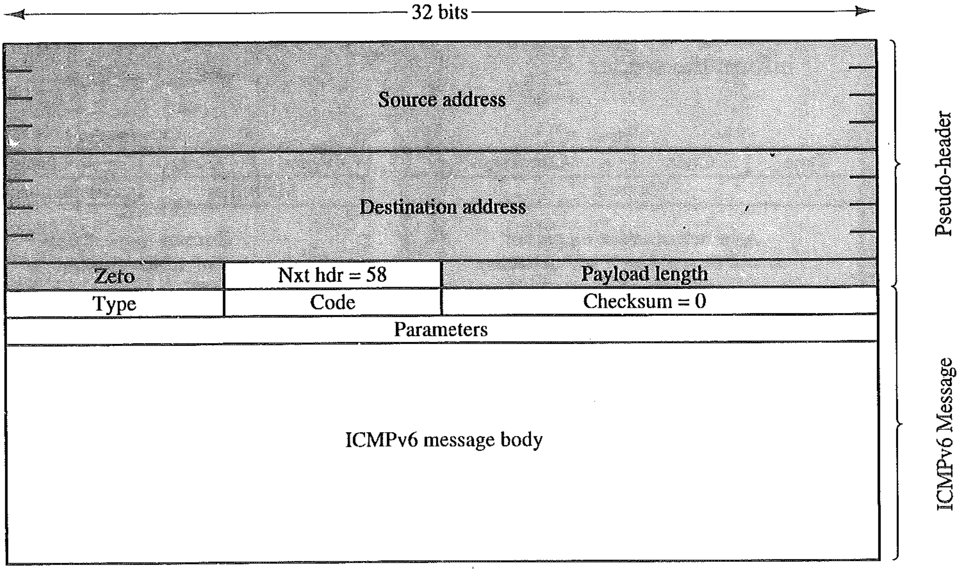


FIGURE 16.21 Scope of checksum for ICMPv6.

Error Messages

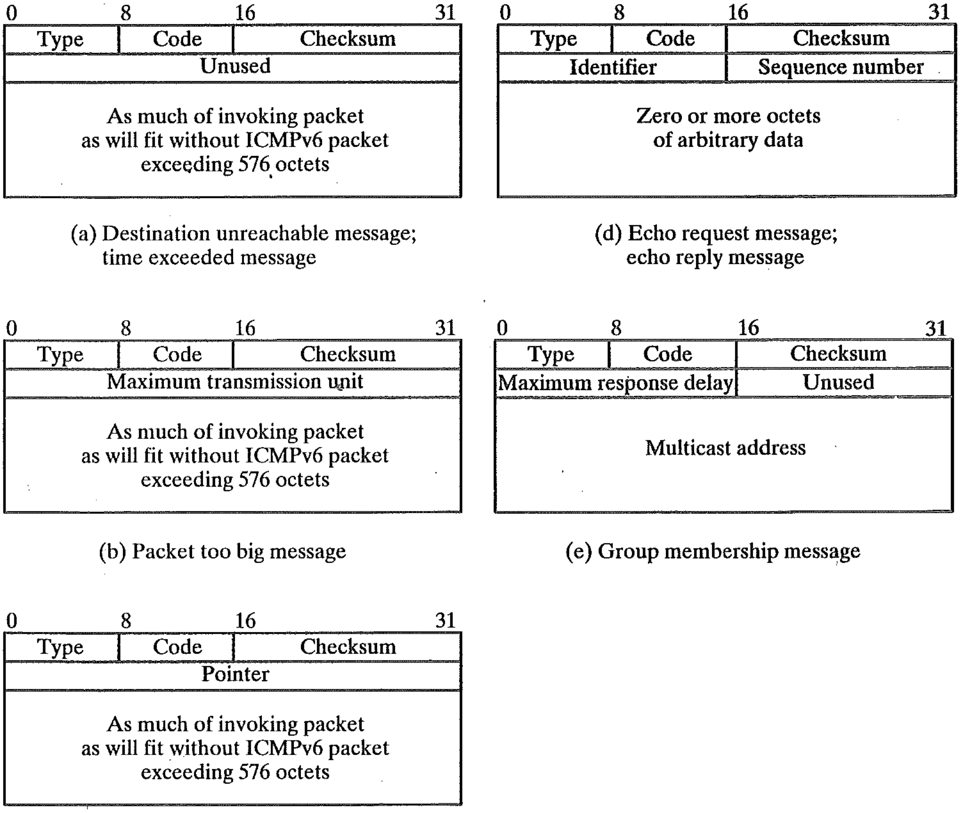
ICMPv6 includes four error messages:

- Destination unreachable
- Packet too big
- Time exceeded
- Parameter problem

Each of these messages refers to a prior IPv6 packet and is sent to the originator of the message. The message body includes as much of the original packet as possible, up to a limit on the size of the IPv6 packet carrying this message of 576 octets (Figure 16.22a to c); this enables the source node to match the incoming ICMPv6 message with the prior datagram.

The *destination unreachable* message covers a number of contingencies, which are indicated by the code field value:

- **No route to destination (0).** Router does not know how to reach the destination subnetwork.
- **Communication with destination administratively prohibited (1).** Delivery prevented by a firewall of other administrative policy.
- **Not a neighbor (2).** The router field of the packet identifies the next node on the path with the strict bit set, but that node is not a neighbor.
- **Address unreachable (3).** Unable to resolve the IPv6 destination address into a link address, or a link-specific problem of some kind.
- **Port unreachable (4).** The destination transport protocol (e.g., UDP) is not listening on that port, and that transport protocol has no other means to inform the sender.


0
8
16
31
Type
Code
Checksum
Unused
As much of invoking packet as will fit without ICMPv6 packet exceeding 576 octets
0
8
16
31
Type
Code
Checksum
Identifier
Sequence number
Zero or more octets of arbitrary data
0
8
16
31
Type
Code
Checksum
Maximum transmission unit
As much of invoking packet as will fit without ICMPv6 packet exceeding 576 octets
0
8
16
31
Type
Code
Checksum
Maximum response delay
Unused
Multicast address
0
8
16
31
Type
Code
Checksum
Pointer
As much of invoking packet as will fit without ICMPv6 packet exceeding 576 octets

(a) Destination unreachable message; time exceeded message

(d) Echo request message; echo reply message

(b) Packet too big message

(e) Group membership message

(c) Parameter problem message

FIGURE 16.22 ICMPv6 message formats.

The *packet-too-big* message is sent by a router in response to a packet whose size exceeds the MTU of the outgoing link. The MTU field in this message (Figure 16.22b) is the MTU of the next-hop link or subnetwork. This message can be used as part of a path MTU discovery process, by which a source node learns of the MTU along various paths.

The *time exceeded* message is sent by a router if it receives a packet with a hop limit of zero, or if it decrements the hop limit to zero. The code is set to 0. This message may also be sent by a destination node, with a code of 1, if reassembly timeout occurs. For IPv6, it is expected that each node will set a local reassembly timeout value for the reassembly process.

A syntactic or semantic error in an IPv6 header will cause a *parameter problem* message to be returned by a router or host. Three kinds of error conditions are identified by the code field: erroneous header field (0), unrecognized next-header type (1), and unrecognized IPv6 option (2). The pointer field contains a pointer to the octet in the original header where the error was detected.

Informational Messages

ICMPv6 includes three informational messages (Figure 16.22d and e):

- Echo request
- Echo reply
- Group membership

The *echo request* and *echo reply* messages provide a mechanism for testing that communication is possible between entities. The recipient of an echo request message is obligated to return the message body in an echo reply message. An identifier and sequence number are associated with the echo request message to be matched in the echo reply message. The identifier might be used like a service access point to identify a particular session, and the sequence number might be incremented on each echo request sent.

The *group-management* message implements the procedures of the Internet Group Management Protocol (IGMP), defined in RFC 1112. IGMP is an extension of ICMP that provides a mechanism for deciding whether a router should forward a multicast IPv4 datagram. In ICMPv6, there are actually three different messages with different type values:

- Group-membership query (type = 130)
- Group-membership report (type = 131)
- Group-membership termination (type = 132)

A host may join a multicast group by sending a group-membership report on a subnetwork with the multicast address in the body of the message; the IPv6 packet containing this message is addressed to that same multicast address. Routers on the subnetwork receive the report and store the information that at least one node on that subnetwork is a member of the group. A host may terminate its membership by sending a group-membership termination message.

At regular intervals, routers send out group-membership query messages. The IPv6 destination address can be a specific multicast address, or it can be the link-local–all-nodes multicast address. The maximum response delay value is the maximum time that a responding report message may be delayed, in milliseconds. Each host which still wishes to be a member of the group or groups replies for each appropriate group with a group-membership report.

16.7 RECOMMENDED READING

Good coverage of internetworking and IP can be found in [COME95]. Detailed coverage of OSPF and other routing algorithms is provided by [HUIT95]; these topics are also treated in [STEE95] and [PERL92].

[BRAD96] is the most thorough treatment of IPv6-related issues. However, the actual technical treatment of IPv6 is limited to less than is covered in this book; instead, the book provides a relatively nontechnical discussion of the requirements for IPv6 and of the history of the IPv6 development. [GILL95] provides a detailed description of the IPv4-to-IPv6 transition, with emphasis on tunneling mechanisms; [MURP95] also contains a useful discussion of the topic.

BRAD96 Bradner, S. and Mankin, A. *IPng: Internet Protocol Next Generation.* Reading, MA: Addison-Wesley, 1996.

COME95 Comer, D. *Internetworking with TCP/IP, Volume I: Principles, Protocols, and Architecture.* Englewood Cliffs, NJ: Prentice Hall, 1995.

GILL95 Gilligan, R. and Callon, R. "IPv6 Transition Mechanisms Overview." *Connexions,* October, 1995.

HUIT95 Huitema, C. *Routing in the Internet.* Englewood Cliffs, NJ: Prentice Hall, 1995.

MURP95 Murphy, E. Hayes, S. and Enders, M. *TCP/IP: Tutorial and Technical Overview.* Upper Saddle River, NJ: Prentice Hall, 1995.

PERL92 Perlman, R. *Interconnections: Bridges and Routers.* Reading, MA: Addison-Wesley, 1992.

STEE95 Steenstrup, M. *Routing in Communications Networks.* Englewood Cliffs, NJ: Prentice Hall, 1995.

Recommended Web Site

- http://playground.sun.com/pub/ipng/html/ipng-main.html: Contains information about IPv6 and related topics.

16.8 PROBLEMS

16.1 In the discussion of IP, it was mentioned that the *identifier, don't-fragment identifier,* and *time to live* parameters are present in the Send primitive but not in the Deliver primitive because they are only of concern to IP. For each of these primitives, indicate whether it is of concern to the IP entity in the source, to the IP entities in any intermediate routers, and to the IP entity in the destination end systems. Justify your answer.

16.2 What are the pros and cons of intermediate reassembly of an internet segmented datagram versus reassembly at the final destination?

16.3 What is the header overhead in the IP protocol?

16.4 Describe some circumstances where it might be desirable to use source routing rather than let the routers make the routing decision.

16.5 Because of segmentation, an IP datagram can arrive in several pieces, not necessarily in the correct order. The IP entity at the receiving-end system must accumulate these segments until the original datagram is reconstituted.

a. Consider that the IP entity creates a buffer for assembling the data field in the original datagram. As assembly proceeds, the buffer will contain blocks of data and "holes" between the data blocks. Describe an algorithm for reassembly based on this concept.

b. For the algorithm in part (a), it is necessary to keep track of the holes; describe a simple mechanism for doing this.

16.6 The IP checksum needs to be recalculated at routers because of changes to the IP header, such as the lifetime field. It is possible to recalculate the checksum from scratch. Suggest a procedure that involves less calculation. Hint: Suppose that the value in octet k is changed by Z = new_value–old_value; consider the effect of this change on the checksum.

16.7 An IP datagram is to be segmented. Which options in the option field need to be copied into the header of each segment, and which need only be retained in the first segment? Justify the handling of each option.

16.8 A transport layer message consisting of 1500 bits of data and 160 bits of header is sent to an internet layer which appends another 160 bits of header; this is then transmitted through two networks, each of which uses a 24-bit packet header. The destination network has a maximum packet size of 800 bits. How many bits, including headers, are delivered to the network layer protocol at the destination?

16.9 The ICMP format includes the first 64 bits of the datagram data field. What might be the purpose of including these bits?

16.10 The architecture suggested by Figure 16.4 is to be used. What functions could be added to the routers to alleviate some of the problems caused by the mismatched local and long-haul networks?

16.11 Would the spanning tree approach be good for an internet including routers?

16.12 Should internetworking be concerned with a network's internal routing? Why or why not?

16.13 Compare the individual fields of the IPv4 header with the IPv6 header. Account for the functionality provided by each IPv4 field by showing how the same functionality is provided in IPv6.

16.14 Justify the recommended order in which IPv6 extension headers appear (i.e., why is the hop-by-hop options header first, why is the routing header before the fragment header, and so on).

16.15 The IPv6 standard states that if a packet with a non-zero flow label arrives at a router, and the router has no information for that flow label, the router should ignore the flow label and forward the packet.

a. What are the disadvantages of treating this event as an error, discarding the packet and sending an ICMP message?

b. Are there situations in which routing the packet as if its flow label were zero will cause the wrong result? Explain.

16.16 The IPv6 flow mechanism assumes that the state associated with a given flow label is stored in routers, so they know how to handle packets that carry that flow label. A design requirement is to flush flow labels that are no longer being used (stale flow label) from routers.

a. Assume that a source always sends a control message to all affected routers deleting a flow label when the source finishes with that flow. In that case, how could a stale flow label persist?

b. Suggest router and source mechanisms to overcome the problem of stale flow labels.

16.17 The question arises as to which packets generated by a source should carry non-zero IPv6 flow labels. For some applications, the answer is obvious. Small exchanges of data should have a zero flow label because it is not worth creating a flow for a few packets. Real-time flows should have a flow label; such flows are a primary reason flow labels were created. A more difficult issue is what to do with peers sending large amounts of best-effort traffic (e.g., TCP connections). Make a case for assigning a unique flow label to each long-term TCP connection. Make a case for not doing this.

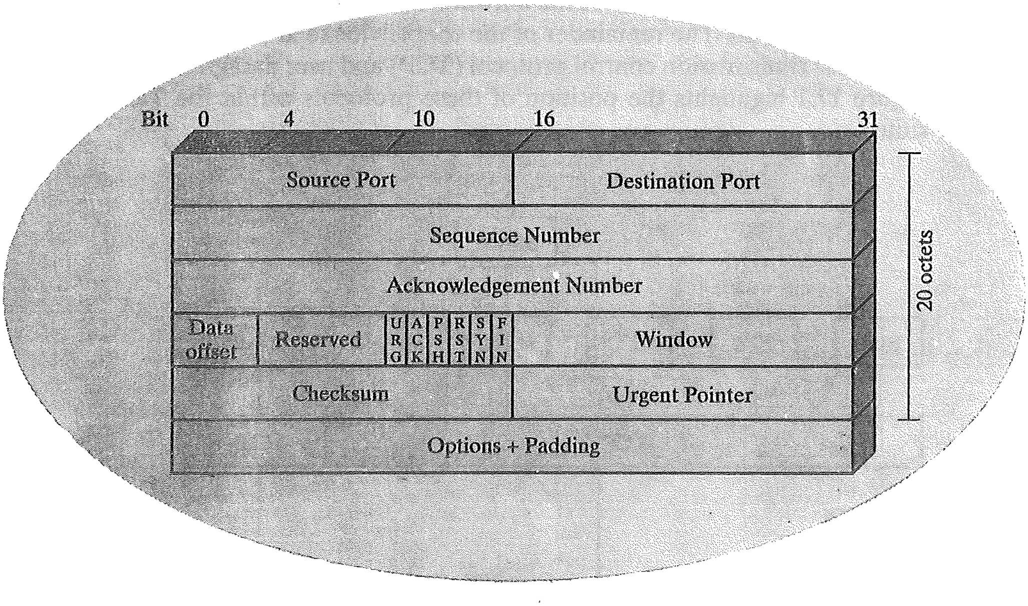
16.18 The original IPv6 specifications combined the priority and flow label fields into a single 28-bit flow label field. This allowed flows to redefine the interpretation of different values of priority. Suggest reasons why the final specification includes the Priority field as a distinct field.

16.19 Based on Table 16.6, what percentage of the total address space has already been assigned?

16.20 For Type 0 IPv6 routing, specify the algorithm for updating the IPv6 and routing headers by intermediate nodes.

CHAPTER 17

TRANSPORT PROTOCOLS



The transport protocol is the keystone of the whole concept of a computer-communications architecture. Lower-layer protocols are needed, to be sure, but they are less important for 1) pedagogical purposes, and 2) designing purposes. For one thing, lower-level protocols are better understood and, on the whole, less complex than transport protocols. Also, standards have settled out quite well for most kinds of layer 1 to 3 transmission facilities, and there is a large body of experience behind their use.

Viewed from the other side, upper-level protocols depend heavily on the transport protocol. The transport protocol provides the basic end-to-end service of transferring data between users and relieves applications and other upper-layer protocols from the need to deal with the characteristics of intervening communications networks and services.

We begin by looking at the services that one might expect from a transport protocol. Next, we examine the protocol mechanisms required to provide these services. We find that most of the complexity relates to connection-oriented services. As might be expected, the less the network service provides, the more the transport protocol must do. The remainder of the chapter looks at two widely used transport protocols: transmission control protocol (TCP) and user datagram protocol (UDP). Figure 17.1 highlights the position of these protocols within the TCP/IP protocol suite.

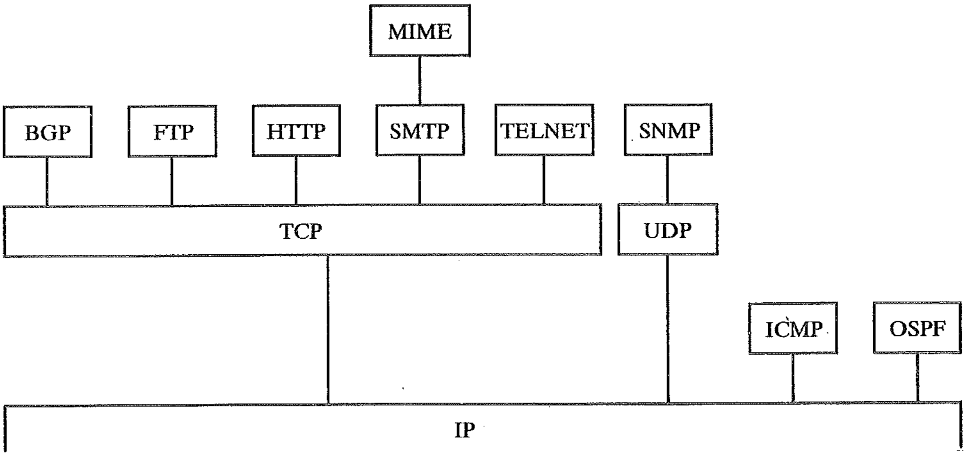


FIGURE 17.1 Transport-level protocols in context.

17.1 TRANSPORT SERVICES

We begin by looking at the kinds of services that a transport protocol can or should provide to higher-level protocols. Figure 17.2 places the concept of transport services in context. In a system, there is a transport entity that provides services to TS

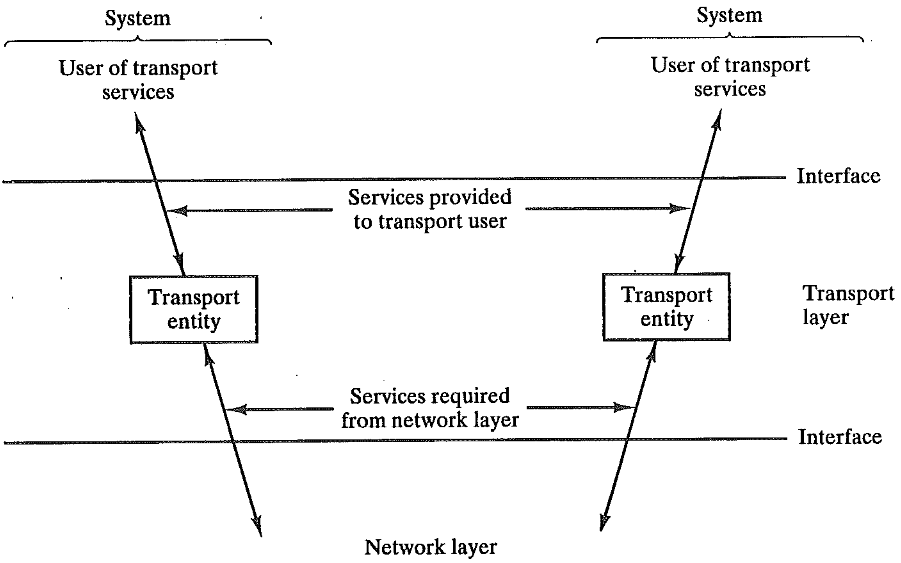

System
User of transport services
System
User of transport services
Interface
Services provided to transport user
Transport entity
Transport entity
Transport layer
Services required from network layer
Interface
Network layer

FIGURE 17.2 Transport entity context.

users,[1] which might be an application process or a session-protocol entity. This local transport entity communicates with some remote-transport entity, using the services of some lower layer, such as the network layer.

We have already mentioned that the general service provided by a transport protocol is the end-to-end transport of data in a way that shields the TS user from the details of the underlying communications systems. To be more specific, we must consider the specific services that a transport protocol can provide. The following categories of service are useful for describing the transport service:

- Type of service
- Quality of service
- Data transfer
- User interface
- Connection management
- Expedited delivery
- Status reporting
- Security

Type of Service

Two basic types of service are possible: connection-oriented and connectionless, or datagram service. A connection-oriented service provides for the establishment,

[1] For brevity, we will refer to the user of a transport service as a TS user; this term is used in ISO documents.

maintenance, and termination of a logical connection between TS users. This has, so far, been the most common type of protocol service available and has a wide variety of applications. The connection-oriented service generally implies that the service is reliable.

The strengths of the connection-oriented approach are clear. It allows for connection-related features such as flow control, error control, and sequenced delivery. Connectionless service, however, is more appropriate in some contexts. At lower layers (internet, network), connectionless service is more robust (e.g., see discussion in Section 9.1). In addition, it represents a "least common denominator" of service to be expected at higher layers. Further, even at transport and above, there is justification for a connectionless service. There are instances in which the overhead of connection establishment and maintenance is unjustified or even counterproductive. Some examples follow:

- **Inward data collection.** Involves the periodic active or passive sampling of data sources, such as sensors, and automatic self-test reports from security equipment or network components. In a real-time monitoring situation, the loss of an occasional data unit would not cause distress, as the next report should arrive shortly.
- **Outward data dissemination.** Includes broadcast messages to network users, the announcement of a new node or the change of address of a service, and the distribution of real-time clock values.
- **Request-response.** Applications in which a transaction service is provided by a common server to a number of distributed TS users, and for which a single request-response sequence is typical. Use of the service is regulated at the application level, and lower-level connections are often unnecessary and cumbersome.
- **Real-time applications.** Such as voice and telemetry, involving a degree of redundancy and/or a real-time transmission requirement; these must not have connection-oriented functions, such as retransmission.

Thus, there is a place at the transport level for both a connection-oriented and a connectionless type of service.

Quality of Service

The transport protocol entity should allow the TS user to specify the quality of transmission service to be provided. The transport entity will attempt to optimize the use of the underlying link, network, and internet resources to the best of its ability, so as to provide the collective requested services.

Examples of services that might be requested are

- Acceptable error and loss levels
- Desired average and maximum delay
- Desired average and minimum throughput
- Priority levels

Of course, the transport entity is limited to the inherent capabilities of the underlying service. For example, IP does provide a quality-of-service parameter. It allows for specification of eight levels of precedence or priority as well as a binary specification for normal or low delay, normal or high throughput, and normal or high reliability. Thus, the transport entity can "pass the buck" to the internetwork entity. However, the internet protocol entity is itself limited; routers have some freedom to schedule items preferentially from buffers, but beyond that are still dependent on the underlying transmission facilities. Here is another example: X.25 provides for throughput class negotiation as an optional user facility. The network may alter flow control parameters and the amount of network resources allocated on a virtual circuit to achieve desired throughput.

The transport layer may also resort to other mechanisms to try to satisfy TS user requests, such as splitting one transport connection among multiple virtual circuits to enhance throughput.

The TS user of the quality-of-service feature needs to recognize that

- Depending on the nature of the transmission facility, the transport entity will have varying degrees of success in providing a requested grade of service.
- There is bound to be a trade-off among reliability, delay, throughput, and cost of services.

Nevertheless, certain applications would benefit from, or even require, certain qualities of service and, in a hierarchical or layered architecture, the easiest way for an application to extract this quality of service from a transmission facility is to pass the request down to the transport protocol.

Examples of applications that might request particular qualities of service are as follows:

- A file transfer protocol might require high throughput. It may also require high reliability to avoid retransmissions at the file transfer level.
- A transaction protocol (e.g., web browser–web server) may require low delay.
- An electronic mail protocol may require multiple priority levels.

One approach to providing a variety of qualities of service is to include a quality-of-service facility within the protocol; we have seen this with IP and will see that transport protocols typically follow the same approach. An alternative is to provide a different transport protocol for different classes of traffic; this is to some extent the approach taken by the ISO-standard family of transport protocols.

Data Transfer

The whole purpose, of course, of a transport protocol is to transfer data between two transport entities. Both user data and control data must be transferred, either on the same channel or separate channels. Full-duplex service must be provided. Half-duplex and simplex modes may also be offered to support peculiarities of particular TS users.

User Interface

It is not clear that the exact mechanism of the user interface to the transport protocol should be standardized. Rather, it should be optimized to the station environment. As examples, a transport entity's services could be invoked by

- Procedure calls.
- Passing of data and parameters to a process through a mailbox.
- Use of direct memory access (DMA) between a host user and a front-end processor containing the transport entity.

A few characteristics of the interface may be specified, however. For example, a mechanism is needed to prevent the TS user from swamping the transport entity with data. A similar mechanism is needed to prevent the transport entity from swamping a TS user with data. Another aspect of the interface has to do with the timing and significance of confirmations. Consider the following: A TS user passes data to a transport entity to be delivered to a remote TS user. The local transport entity can acknowledge receipt of the data immediately, or it can wait until the remote transport entity reports that the data have made it through to the other end. Perhaps the most useful interface is one that allows immediate acceptance or rejection of requests, with later confirmation of the end-to-end significance.

Connection Management

When connection-oriented service is provided, the transport entity is responsible for establishing and terminating connections. A symmetric connection-establishment procedure should be provided, which allows either TS user to initiate connection establishment. An asymmetric procedure may also be provided to support simplex connections.

Connection termination can be either *abrupt* or *graceful.* With an abrupt termination, data in transit may be lost. A graceful termination prevents either side from shutting down until all data have been delivered.

Expedited Delivery

A service similar to that provided by priority classes is the expedited delivery of data. Some data submitted to the transport service may supersede data submitted previously. The transport entity will endeavor to have the transmission facility transfer the data as rapidly as possible. At the receiving end, the transport entity will interrupt the TS user to notify it of the receipt of urgent data. Thus, the expedited data service is in the nature of an interrupt mechanism, and is used to transfer occasional urgent data, such as a break character from a terminal or an alarm condition. In contrast, a priority service might dedicate resources and adjust parameters such that, on average, higher priority data are delivered more quickly.

Status Reporting

A status reporting service allows the TS user to obtain or be notified of information concerning the condition or attributes of the transport entity or a transport connection. Examples of status information are

- Performance characteristics of a connection (e.g., throughput, mean delay)
- Addresses (network, transport)
- Class of protocol in use
- Current timer values
- State of protocol "machine" supporting a connection
- Degradation in requested quality of service

Security

The transport entity may provide a variety of security services. Access control may be provided in the form of local verification of sender and remote verification of receiver. The transport service may also include encryption/decryption of data on demand. Finally, the transport entity may be capable of routing through secure links or nodes if such a service is available from the transmission facility.

17.2 PROTOCOL MECHANISMS

It is the purpose of this section to make good on our claim that a transport protocol may need to be very complex. For purposes of clarity, we present the transport protocol mechanisms in an evolutionary fashion. We begin with a network service that makes life easy for the transport protocol, by guaranteeing the delivery of all transport data units in order, as well as defining the required mechanisms. Then we will look at the transport protocol mechanisms required to cope with an unreliable network service.

Reliable Sequencing Network Service

In this case, we assume that the network service will accept messages of arbitrary length and will, with virtually 100% reliability, deliver them in sequence to the destination. Examples of such networks follow:

- A highly reliable packet-switching network with an X.25 interface
- A frame relay network using the LAPF control protocol
- An IEEE 802.3 LAN using the connection-oriented LLC service

The assumption of a reliable sequencing networking services allows the use of a quite simple transport protocol. Four issues need to be addressed:

- Addressing
- Multiplexing
- Flow control
- Connection establishment/termination

Addressing

The issue concerned with addressing is simply this: A user of a given transport entity wishes to either establish a connection with or make a connectionless data transfer to a user of some other transport entity. The target user needs to be specified by all of the following:

- User identification
- Transport entity identification
- Station address
- Network number

The transport protocol must be able to derive the information listed above from the TS user address. Typically, the user address is specified as *station* or *port*. The *port* variable represents a particular TS user at the specified station; in OSI, this is called a transport service access point (TSAP). Generally, there will be a single transport entity at each station, so a transport entity identification is not needed. If more than one transport entity is present, there is usually only one of each type. In this latter case, the address should include a designation of the type of transport protocol (e.g., TCP, UDP). In the case of a single network, *station* identifies an attached network device. In the case of an internet, station is a global internet address. In TCP, the combination of port and station is referred to as a socket.

Because routing is not a concern of the transport layer, it simply passes the station portion of the address down to the network service. Port is included in a transport header, to be used at the destination by the destination transport protocol.

One question remains to be addressed: How does the initiating TS user know the address of the destination TS user? Two static and two dynamic strategies suggest themselves:

1. The TS user must know the address it wishes to use ahead of time; this is basically a system configuration function. For example, a process may be running that is only of concern to a limited number of TS users, such as a process that collects statistics on performance. From time to time, a central network management routine connects to the process to obtain the statistics. These processes generally are not, and should not be, well-known and accessible to all.
2. Some commonly used services are assigned "well-known addresses" (for example, time sharing and word processing).
3. A name server is provided. The TS user requests a service by some generic or global name. The request is sent to the name server, which does a directory lookup and returns an address. The transport entity then proceeds with the connection. This service is useful for commonly used applications that change location from time to time. For example, a data entry process may be moved from one station to another on a local network in order to balance load.
4. In some cases, the target user is to be a process that is spawned at request time. The initiating user can send a process request to a well-known address. The user at that address is a privileged system process that will spawn the new process and return an address. For example, a programmer has developed a private application (e.g., a simulation program) that will execute on a remote

mainframe but be invoked from a local minicomputer. An RJE-type request can be issued to a remote job-management process that spawns tlie simulation process.

Multiplexing

We now turn to the concept of multiplexing, which was discussed in general terms in Section 10.1. With respect to the interface between the transport protocol and higher-level protocols, the transport protocol performs a multiplexing/demultiplexing function. That is, multiple users employ the same transport protocol, and are distinguished by either port numbers or service access points.

The transport entity may also perform a multiplexing function with respect to the network services that it uses. Recall that we defined upward multiplexing as the multiplexing of multiple connections on a single lower-level connection, and downward multiplexing as the splitting of a single connection among multiple lower-level connections.

Consider, for example, a transport entity making use of an X.25 service. Why should the transport entity employ upward multiplexing? There are, after all, 4095 virtual circuits available. In the typical case, this is more than enough to handle all active TS users. However, most X.25 networks base part of their charge on virtual-circuit connect time, as each virtual circuit consumes some node buffer resources. Thus, if a single virtual circuit provides sufficient throughput for multiple TS users, upward multiplexing is indicated.

On the other hand, downward multiplexing or splitting might be used to improve throughput. For example, each X.25 virtual circuit is restricted to a 3-bit or 7-bit sequence number. A larger sequence space might be needed for high-speed, high-delay networks. Of course, throughput can only be increased so far. If there is a single station-node link over which all virtual circuits are multiplexed, the throughput of a transport connection cannot exceed the data rate of that link.

Flow Control

Whereas flow control is a relatively simple mechanism at the link layer, it is a rather complex mechanism at the transport layer, for two main reasons:

- Flow control at the transport level involves the interaction of TS users, transport entities, and the network service.
- The transmission delay between transport entities is generally long compared to actual transmission time, and, what is worse, it is variable.

Figure 17.3 illustrates the first point. TS user A wishes to send data to TS user B over a transport connection. We can view the situation as involving four queues. A generates data and queues it up to send. A must wait to send that data until

- It has permission from B (peer flow control).
- It has permission from its own transport entity (interface flow control).

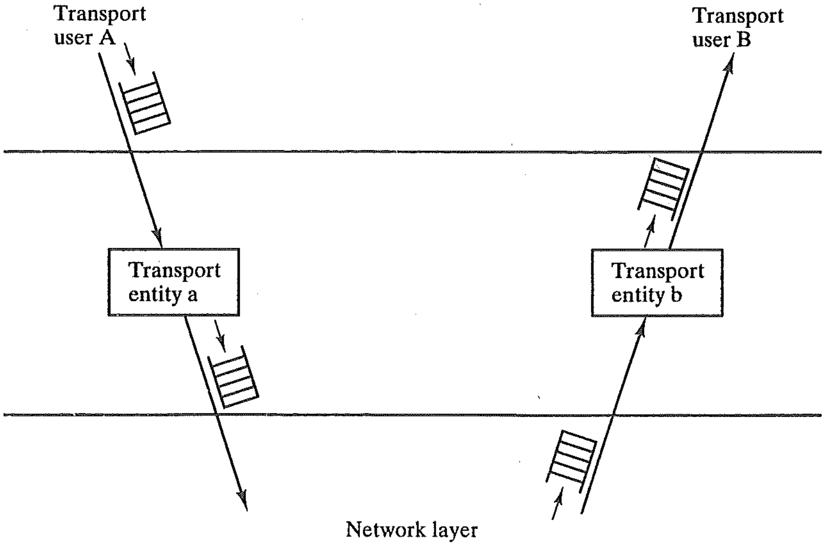


FIGURE 17.3 Queuing representation of connection-oriented data transfer.

As data flow down from A to transport entity a, a queues the data until it has permission to send it on from b and the network service. The data are then handed to the network layer for delivery to b. The network service must queue the data until it receives permission from b to pass them on. Finally, b must await B's permission before delivering the data to their destination.

To see the effects of delay, consider the possible interactions depicted in Figure 17.4. When a TS user wishes to transmit data, it sends these data to its transport

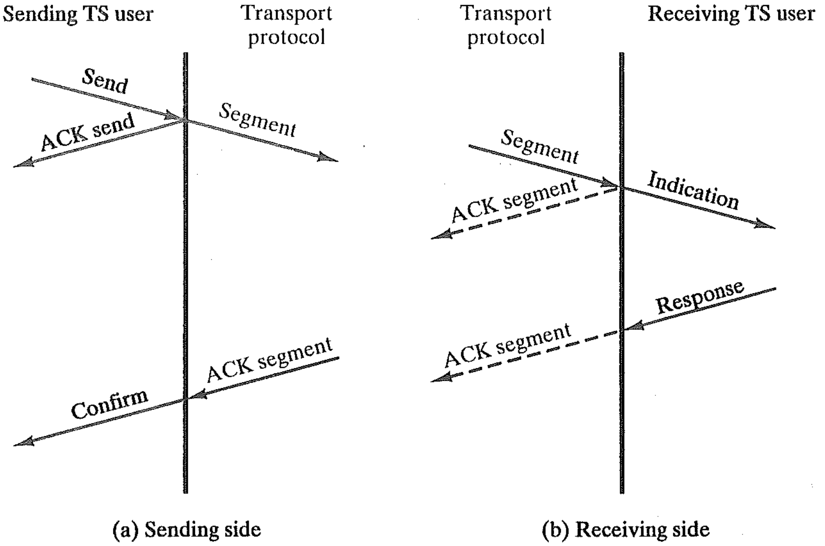


FIGURE 17.4 Interaction between transport service user and transport protocol.

entity (e.g., using a Send call); this triggers two events. The transport entity generates one or more transport-level protocol data units, which we will call segments,[2] and passes these on to the network service. It also in some way acknowledges to the TS user that it has accepted the data for transmission. At this point, the transport entity can exercise flow control across the user-transport interface by simply withholding its acknowledgment. The transport entity is most likely to do this if the entity itself is being held up by a flow control exercised by either the network service or the target transport entity.

In any case, once the transport entity has accepted the data, it sends out a segment. Some time later, it receives an acknowledgment that the data have been received at the remote end. It then sends a confirmation to the sender.

At the receiving end, a segment arrives at the transport entity, which unwraps the data and sends them on (e.g., by an Indication primitive) to the destination TS user. When the TS user accepts the data, it issues an acknowledgment (e.g., in the form of a Response primitive). The TS user can exercise flow control over the transport entity by withholding its response.

The target transport entity has two choices regarding acknowledgment. Either it can issue an acknowledgment as soon as it has correctly received the segment (the usual practice), or it can wait until it knows that the TS user has correctly received the data before acknowledging; the latter course is the safer, where the confirmation is in fact a confirmation that the destination TS user received the data. In the former case, the entity merely confirms that the data made it through to the remote transport entity.

With the discussion above in mind, we can cite two reasons why one transport entity would want to restrain the rate of segment transmission over a connection from another transport entity:

- The user of the receiving transport entity cannot keep up with the flow of data.
- The receiving transport entity itself cannot keep up with the flow of segments.

How do such problems manifest themselves? Well, presumably a transport entity has a certain amount of buffer space, to which incoming segments are added. Each buffered segment is processed (i.e., the transport header is examined) and the data are sent to the TS user. Either of the two problems mentioned above will cause the buffer to fill up. Thus, the transport entity needs to take steps to stop or slow the flow of segments so as to prevent buffer overflow. This requirement is not so easy to fulfill, because of the annoying time gap between sender and receiver. We return to this point in a moment. First, we present four ways of coping with the flow control requirement. The receiving transport entity can

1. Do nothing.
2. Refuse to accept further segments from the network service.
3. Use a fixed sliding-window protocol.
4. Use a credit scheme.

[2] In this chapter, we use the terminology of TCP for convenience. ISO standards simply refer to this object as a transport PDU, or TPDU.

Alternative 1 means that the segments that overflow the buffer are discarded. The sending transport entity, failing to get an acknowledgment, will retransmit. This is a shame, as the advantage of a reliable network is that one never has to retransmit. Furthermore, the effect of this maneuver is to exacerbate the problem! The sender has increased its output to include new segments, plus retransmitted old segments.

The second alternative is a backpressure mechanism that relies on the network service to do the work. When a buffer of a transport entity is full, it refuses additional data from the network service. This triggers flow control procedures within the network that throttle the network service at the sending end. This service, in turn, refuses additional segments from its transport entity. It should be clear that this mechanism is clumsy and coarse-grained. For example, if multiple transport connections are multiplexed on a single network connection (virtual circuit), flow control is exercised only on the aggregate of all transport connections.

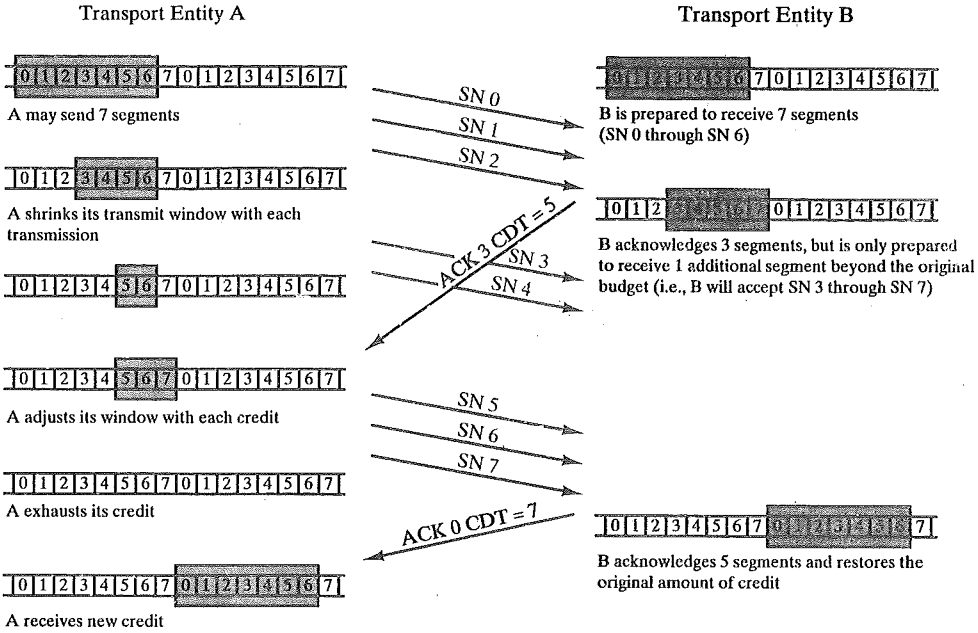
The third alternative is already familiar to you from our discussions of link layer protocols. The key ingredients, recall, are

- The use of sequence numbers on data units.
- The use of a window of fixed size.
- The use of acknowledgments to advance the window.

With a reliable network service, the sliding window technique would actually work quite well. For example, consider a protocol with a window size of 7. Whenever the sender receives an acknowledgment to a particular segment, it is automatically authorized to send the succeeding seven segments. (Of course, some may already have been sent.) Now, when the receiver's buffer capacity gets down to seven segments, it can withhold acknowledgment of incoming segments to avoid overflow. The sending transport entity can send, at most, seven additional segments and then must stop. Because the underlying network service is reliable, the sender will not time-out and retransmit. Thus, at some point, a sending transport entity may have a number of segments outstanding, for which no acknowledgment has been received. Because we are dealing with a reliable network, the sending transport entity can assume that the segments will get through and that the lack of acknowledgment is a flow control tactic. Such a strategy would not work well in an unreliable network, as the sending transport entity would not know whether the lack of acknowledgment is due to flow control or a lost segment.

The fourth alternative, a credit scheme, provides the receiver with a greater degree of control over data flow. Although it is not strictly necessary with a reliable network service, a credit scheme should result in a smoother traffic flow; further, it is a more effective scheme with an unreliable network service, as we shall see.

The credit scheme decouples acknowledgment from flow control. In fixed sliding-window protocols, such as X.25 and HDLC, the two are synonymous. In a credit scheme, a segment may be acknowledged without granting new credit, and vice versa. Figure 17.5 illustrates the protocol (compare Figure 6.4). For simplicity, we show a data flow in one direction only. In this example, data segments are numbered sequentially modulo 8 (e.g., SN 0 = segment with sequence number 0). Initially, through the connection-establishment process, the sending and receiving



Transport Entity A
Transport Entity B
0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
A may send 7 segments
A shrinks its transmit window with each transmission
A adjusts its window with each credit
A exhausts its credit
A receives new credit
SN 0
SN 1
SN 2
ACK 3 CDT = 5
SN 3
SN 4
SN 5
SN 6
SN 7
ACK 0 CDT = 7
B is prepared to receive 7 segments (SN 0 through SN 6)
B acknowledges 3 segments, but is only prepared to receive 1 additional segment beyond the original budget (i.e., B will accept SN 3 through SN 7)
B acknowledges 5 segments and restores the original amount of credit

FIGURE 17.5 Example of credit allocation mechanism.

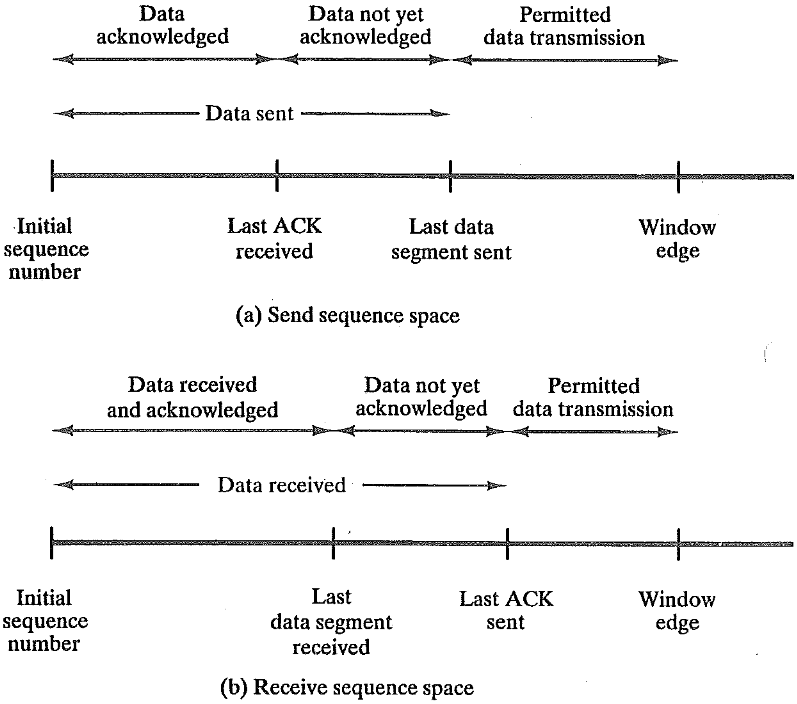
sequence numbers are synchronized, and A is granted a credit allocation of 7. A advances the trailing edge of its window each time that it transmits, and advances the leading edge only when it is granted credit.

Figure 17.6 shows the view of this mechanism from the sending and receiving sides; of course, both sides take both views because data may be exchanged in both directions. From the sending point of view, sequence numbers fall into four regions:

- **Data sent and acknowledged.** Beginning with the initial sequence number used on this connection through the last acknowledged number.
- **Data sent but not yet acknowledged.** Represents data that have already been transmitted, with the sender now awaiting acknowledgment.
- **Permitted data transmission.** The window of allowable transmissions, based on unused credit allocated from the other side.
- **Unused and unusable numbers.** Numbers above the window.

From the receiving point of view, the concern is for received data and for the window of credit that has been allocated. Note that the receiver is not required to immediately acknowledge incoming segments, but may wait and issue a cumulative acknowledgment for a number of segments; this is true for both TCP and the ISO transport protocol.

In both the credit allocation scheme and the sliding window scheme, the receiver needs to adopt some policy concerning the amount of data it permits the sender to transmit. The conservative approach is to only allow new segments up to the limit of available buffer space. If this policy were in effect in Figure 17.5, the first


Data acknowledged
Data not yet acknowledged
Permitted data transmission
Data sent
Initial sequence number
Last ACK received
Last data segment sent
Window edge

(a) Send sequence space

Data received and acknowledged
Data not yet acknowledged
Permitted data transmission
Data received
Initial sequence number
Last data segment received
Last ACK sent
Window edge

(b) Receive sequence space

FIGURE 17.6 Sending and receiving flow control perspectives.

credit message implies that B has five free buffer slots, and the second message that B has seven free slots.

A conservative flow control scheme may limit the throughput of the transport connection in long-delay situations. The receiver could potentially increase throughput by optimistically granting credit for space it does not have. For example, if a receiver's buffer is full but it anticipates that it can release space for two segments within a round-trip propagation time, it could immediately send a credit of 2. If the receiver can keep up with the sender, this scheme may increase throughput and can do no harm. If the sender is faster than the receiver, however, some segments may be discarded, necessitating a retransmission. Because retransmissions are not otherwise necessary with a reliable network service, an optimistic flow control scheme will complicate the protocol.

Connection Establishment and Termination

Even with a reliable network service, there is a need for connection establishment and termination procedures to support connection-oriented service. Connection establishment serves three main purposes:

- It allows each end to assure that the other exists.
- It allows negotiation of optional parameters (e.g., maximum segment size, maximum window size, quality of service).

- It triggers allocation of transport entity resources (e.g., buffer space, entry in connection table).

Connection establishment is by mutual agreement and can be accomplished by a simple set of user commands and control segments, as shown in the state diagram of Figure 17.7. To begin, a TS user is in an CLOSED state (i.e., it has no open transport connection). The TS user can signal that it will passively wait for a request with a Passive Open command. A server program, such as time sharing or a file transfer application, might do this. The TS user may change its mind by sending a Close command. After the Passive Open command is issued, the transport entity creates a connection object of some sort (i.e., a table entry) that is in the LISTEN state.

From the CLOSED state, the TS user may open a connection by issuing an Active Open command, which instructs the transport entity to attempt connection establishment with a designated user, which then triggers the transport entity to send an SYN (for synchronize) segment. This segment is carried to the receiving transport entity and interpreted as a request for connection to a particular port. If the destination transport entity is in the LISTEN state for that port, then a connection is established through the following actions by the receiving transport entity:

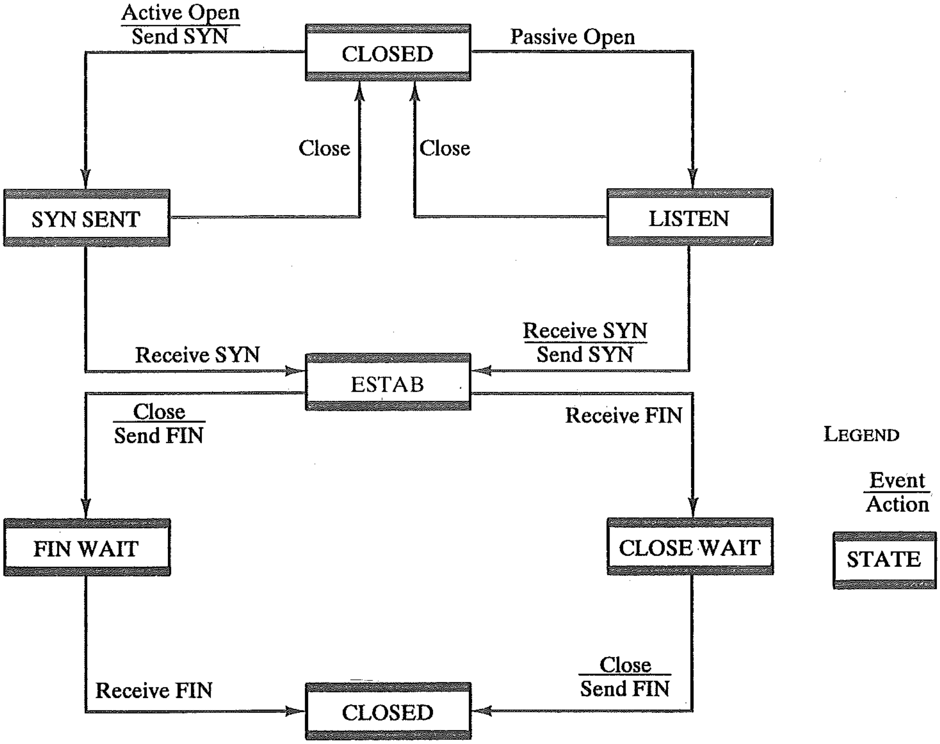


FIGURE 17.7 Simple connection state diagram.

- Signal the TS user that a connection is open.
- Send an SYN as confirmation to the remote transport entity.
- Put the connection object in an ESTAB (established) state.

When the responding SYN is received by the initiating transport entity, it too can move the connection to an ESTAB state. The connection is prematurely aborted if either TS user issues a Close command.

Figure 17.8 shows the robustness of this protocol. Either side can initiate a connection. Further, if both sides initiate the connection at about the same time, it is established without confusion; this is because the SYN segment functions both as a connection request and a connection acknowledgment.

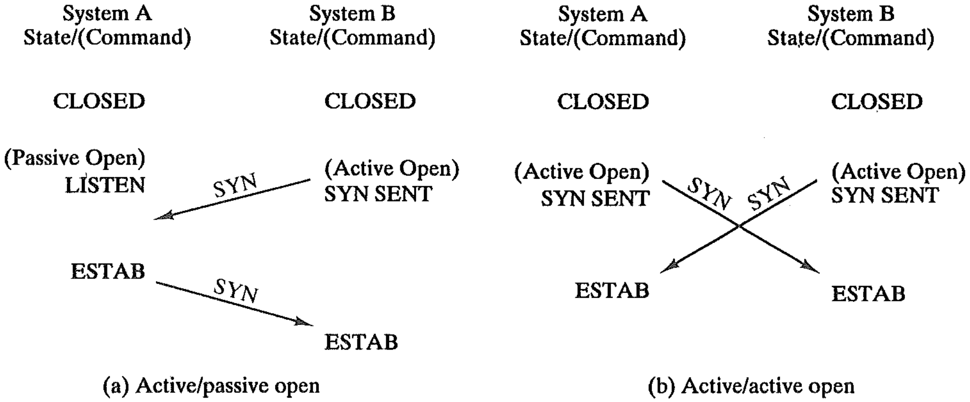


FIGURE 17.8 Connection establishment scenarios.

The reader may ask what happens if an SYN comes in while the requested TS user is idle (not listening). Three courses may be followed:

- The transport entity can reject the request by sending an RST (reset) segment back to the other transport entity.
- The request can be queued until a matching Open is issued by the TS user.
- The transport entity can interrupt or otherwise signal the TS user to notify it of a pending request.

Note that if the latter mechanism is used, a Passive Open command is not strictly necessary, but may be replaced by an Accept command, which is a signal from the user to the transport entity that it accepts the request for connection.

Connection termination is handled similarly. Either side, or both sides, may initiate a close. The connection is closed by mutual agreement. This strategy allows for either abrupt or graceful termination. To achieve the latter, a connection in the CLOSE WAIT state must continue to accept data segments until a FIN (finish) segment is received.

Similarly, the diagram defines the procedure for graceful termination. First, consider the side that initiates the termination procedure:

1. In response to a TS user's Close primitive, a FIN segment is sent to the other side of the connection, requesting termination.
2. Having sent the FIN, the transport entity places the connection in the FIN WAIT state. In this state, the transport entity must continue to accept data from the other side and deliver that data to its user.
3. When a FIN is received in response, the transport entity informs its user and closes the connection.

From the point of view of the side that does not initiate a termination,

1. When a FIN segment is received, the transport entity informs its user of the termination request and places the connection in the CLOSE WAIT state. In this state, the transport entity must continue to accept data from its user and transmit it in data segments to the other side.
2. When the user issues a Close primitive, the transport entity sends a responding FIN segment to the other side and closes the connection.

This procedure ensures that both sides have received all outstanding data and that both sides agree to connection termination before actual termination.

Unreliable Network Service

The most difficult case for a transport protocol is that of an unreliable network service. Examples of such networks are

- An internetwork using IP
- A frame relay network using only the LAPF core protocol
- An IEEE 802.3 LAN using the unacknowledged connectionless LLC service

The problem is not just that segments are occasionally lost, but that segments may arrive out of sequence due to variable transit delays. As we shall see, elaborate machinery is required to cope with these two interrelated network deficiencies. We shall also see that a discouraging pattern emerges. The combination of unreliability and nonsequencing creates problems with every mechanism we have discussed so far. Generally, the solution to each problem raises new problems, and although there are problems to be overcome for protocols at all levels, it seems that there are more difficulties with a reliable connection-oriented transport protocol than any other sort of protocol.

Seven issues need to be addressed:

- Ordered delivery
- Retransmission strategy
- Duplicate detection
- Flow control
- Connection establishment
- Connection termination
- Crash recovery

Ordered Delivery

With an unreliable network service, it is possible that segments, even if they are all delivered, may arrive out of order. The required solution to this problem is to number segments sequentially. We have seen that for data link control protocols, such as HDLC, and for X.25, that each data unit (frame, packet) is numbered sequentially with each successive sequence number being one more than the previous sequence number; this scheme is used in some transport protocols, such as the ISO transport protocols. However, TCP uses a somewhat different scheme in which each data octet that is transmitted is implicitly numbered. Thus, the first segment may have a sequence number of 0. If that segment has 1000 octets of data, then the second segment would have the sequence number 1000, and so on. For simplicity in the discussions of this section, we will assume that each successive segment's sequence number is one more than that of the previous segment.

Retransmission Strategy

Two events necessitate the retransmission of a segment. First, the segment may be damaged in transit but, nevertheless, could arrive at its destination. If a frame check sequence is included with the segment, the receiving transport entity can detect the error and discard the segment. The second contingency is that a segment fails to arrive. In either case, the sending transport entity does not know that the segment transmission was unsuccessful. To cover this contingency, we require that a positive acknowledgment (ACK) scheme be used: The receiver must acknowledge each successfully received segment. For efficiency, we do not require one ACK per segment. Rather, a cumulative acknowledgment can be used, as we have seen many times in this book. Thus, the receiver may receive segments numbered 1, 2, and 3, but only send ACK 4 back. The sender must interpret ACK 4 to mean that number 3 and all previous segments have been successfully received.

If a segment does not arrive successfully, no ACK will be issued and a retransmission becomes necessary. To cope with this situation, there must be a timer associated with each segment as it is sent. If the timer expires before the segment is acknowledged, the sender must retransmit.

So, the addition of a timer solves this first problem. Next, at what value should the timer be set? If the value is too small, there will be many unnecessary retransmissions, thereby wasting network capacity. If the value is too large, the protocol will be sluggish in responding to a lost segment. The timer should be set at a value a bit longer than the round trip delay (send segment, receive ACK). Of course this delay is variable even under constant network load. Worse, the statistics of the delay will vary with changing network conditions.

Two strategies suggest themselves. A fixed timer value could be used, based on an understanding of the network's typical behavior; this suffers from an inability to respond to changing network conditions. If the value is set too high, the service will always be sluggish. If it is set too low, a positive feedback condition can develop, in which network congestion leads to more retransmissions, which increase congestion.

An adaptive scheme has its own problems. Suppose that the transport entity keeps track of the time taken to acknowledge data segments and sets its retransmission timer based on the average of the observed delays. This value cannot be trusted for three reasons:

- The peer entity may not acknowledge a segment immediately; recall that we gave it the privilege of cumulative acknowledgments.
- If a segment has been retransmitted, the sender cannot know whether the received ACK is a response to the initial transmission or the retransmission.
- Network conditions may change suddenly.

Each of these problems is a cause for some further tweaking of the transport algorithm, but the problem admits of no complete solution. There will always be some uncertainty concerning the best value for the retransmission timer.

Incidentally, the retransmission timer is only one of a number of timers needed for proper functioning of a transport protocol; these are listed in Table 17.1, together with a brief explanation. Further discussion will be found in what follows.

Duplicate Detection

If a segment is lost and then retransmitted, no confusion will result. If, however, an ACK is lost, one or more segments will be retransmitted and, if they arrive successfully, will be duplicates of previously received segments. Thus, the receiver must be able to recognize duplicates. The fact that each segment carries a sequence number helps but, nevertheless, duplicate detection and handling is no easy thing. There are two cases:

- A duplicate is received prior to the close of the connection.
- A duplicate is received after the close of the connection.

Notice that we say "a" duplicate rather than "the" duplicate. From the sender's point of view, the retransmitted segment is the duplicate. However, the retransmitted segment may arrive before the original segment, in which case the receiver views the original segment as the duplicate. In any case, two tactics are needed to cope with a duplicate received prior to the close of a connection:

- The receiver must assume that its acknowledgment was lost and therefore must acknowledge the duplicate. Consequently, the sender must not get confused if it receives multiple ACKs to the same segment.

TABLE 17.1 Transport protocol timers.

Retransmission timer	Retransmit an unacknowledged segment
Reconnection timer	Minimum time between closing one connection and opening another with the same destination address
Window timer	Maximum time between ACK/CREDIT segments
Retransmit-SYN timer	Time between attempts to open a connection
Persistence timer	Abort connection when no segments are acknowledged
Inactivity timer	Abort connection when no segments are received

- The sequence number space must be long enough so as not to "cycle" in less than the maximum possible segment lifetime.

Figure 17.9 illustrates the reason for the latter requirement. In this example, the sequence space is of length 8. For simplicity, we assume a sliding-window protocol with a window size of 3. Suppose that A has transmitted data segments 0, 1, and 2 and receives no acknowledgments. Eventually, it times-out and retransmits segment 0. B has received 1 and 2, but 0 is delayed in transit. Thus, B does not send any ACKs. When the duplicate segment 0 arrives, B acknowledges 0, 1, and 2. Meanwhile, A has timed-out again and retransmits 1, which B acknowledges with another ACK 2. Things now seem to have sorted themselves out, and data transfer continues. When the sequence space is exhausted, A cycles back to sequence number 0 and continues. Alas, the old segment 0 makes a belated appearance and is accepted by B before the new segment 0 arrives.

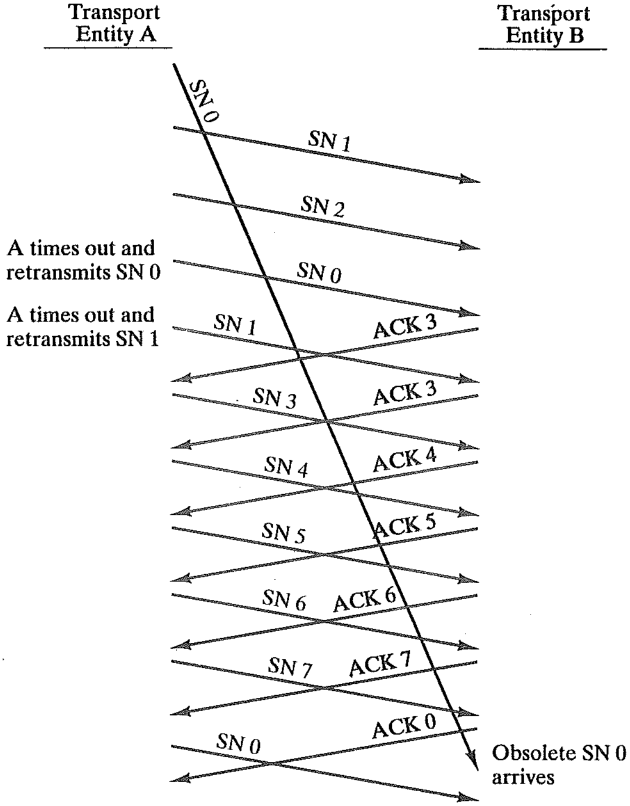


FIGURE 17.9 Example of incorrect duplicate detection.

It should be clear that the untimely emergence of the old segment would have caused no difficulty if the sequence numbers had not yet wrapped around. The problem is: How big must the sequence space be? This depends on, among other things, whether the network enforces a maximum packet lifetime, as well as the rate at which segments are being transmitted. Fortunately, each addition of a single bit to the sequence number field doubles the sequence space, so it is rather easy to select a safe size. As we shall see, the standard transport protocols allow stupendous sequence spaces.

Flow Control

The credit-allocation flow control mechanism described earlier is quite robust in the face of an unreliable network service and requires little enhancement. We assume that the credit allocation scheme is tied to acknowledgments in the following way: To both acknowledge segments and grant credit, a transport entity sends a control segment of the form (ACK *N*, CREDIT *M*), where ACK *N* acknowledges all data segments through number *N*, and CREDIT *M* allows segments number $N + 1$ though $N + M$ to be transmitted. This mechanism is quite powerful. Consider that the last control segment issued by B was (ACK *N*, CREDIT *M*). Then,

- To increase or decrease credit to *X* when no additional segments have arrived, B can issue (ACK *N*, CREDIT *X*).
- To acknowledge a new segment without increasing credit, B can issue (ACK $N + 1$, CREDIT $M - 1$).

If an ACK/CREDIT segment is lost, little harm is done. Future acknowledgments will resynchronize the protocol. Further, if no new acknowledgments are forthcoming, the sender times-out and retransmits a data segment, which triggers a new acknowledgment. However, it is still possible for deadlock to occur. Consider a situation in which B sends (ACK *N*, CREDIT 0), temporarily closing the window. Subsequently, B sends (ACK *N*, CREDIT *M*), but this segment is lost. A is awaiting the opportunity to send data, and B thinks that it has granted that opportunity. To overcome this problem, a window timer can be used. This timer is reset with each outgoing ACK/CREDIT segment. If the timer ever expires, the protocol entity is required to send an ACK/CREDIT segment, even if it duplicates a previous one. This breaks the deadlock and also assures the other end that the protocol entity is still alive.

An alternative or supplemental mechanism is to provide for acknowledgments to the ACK/CREDIT segment. With this mechanism in place, the window timer can have quite a large value without causing much difficulty.

Connection Establishment

As with other protocol mechanisms, connection establishment must take into account the unreliability of a network service. Recall that a connection establishment calls for the exchange of SYNs, a procedure sometimes referred to as a two-way handshake. Suppose that A issues an SYN to B. It expects to get an SYN back, confirming the connection. Two things can go wrong: A's SYN can be lost or B's answering SYN can be lost. Both cases can be handled by use of a retransmit-SYN timer. After A issues an SYN, it will reissue the SYN when the timer expires.

This situation gives rise, potentially,.to duplicate SYNs. If A's initial SYN was lost, there are no duplicates. If B's response was lost, then B may receive two SYNs from A. Further, if B's response was not lost, but simply delayed, A may get two responding SYNs; all of this means that A and B must simply ignore duplicate SYNs once a connection is established.

There are other problems with which to contend. Just as a delayed SYN or lost response can give rise to a duplicate SYN, a delayed data segment or lost acknowledgment can give rise to duplicate data segments, as we have seen in Figure 17.9). Such a delayed or duplicated data segment can interfere with connection establishment, as illustrated in Figure 17.10. Assume that with each new connection, each transport protocol entity begins numbering its data segments with sequence number 0. In the figure, a duplicate copy of segment 2 from an old connection arrives during the lifetime of a new connection and is delivered to B before delivery of the legitimate data segment number 2. One way of attacking this problem is to start each new connection with a different sequence number, far removed from the last sequence number of the most recent connection. For this purpose, the connection request is of the form SYN i, where i is the sequence number of the first data segment that will be sent on this connection.

Now, consider that a duplicate SYN *i* may survive past the termination of the connection. Figure 17.11 depicts the problem that may arise. An old SYN *i* arrives at B after the connection is terminated. B assumes that this is a fresh request and responds with SYN *j*. Meanwhile, A has decided to open a new connection with B and sends SYN *k*; B discards this as a duplicate. Now, both sides have transmitted and subsequently received a SYN segment, and therefore think that a valid con-

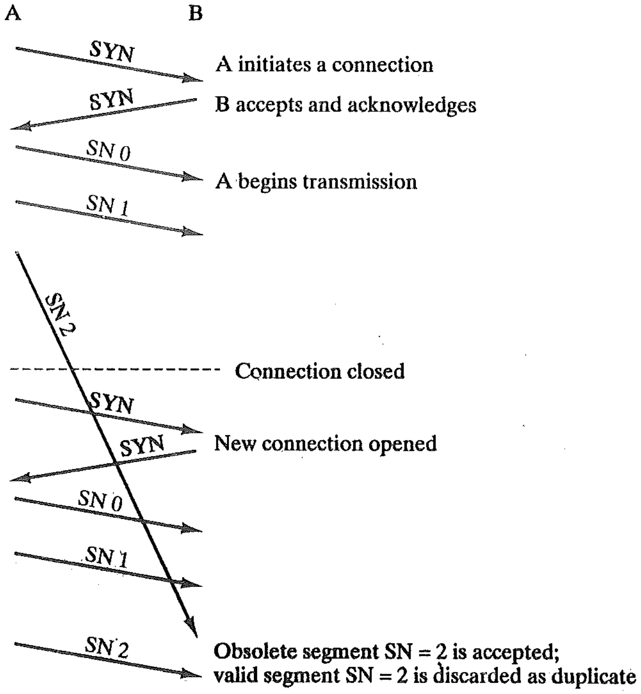


FIGURE 17.10 The two-way handshake: problem with obsolete data segment.

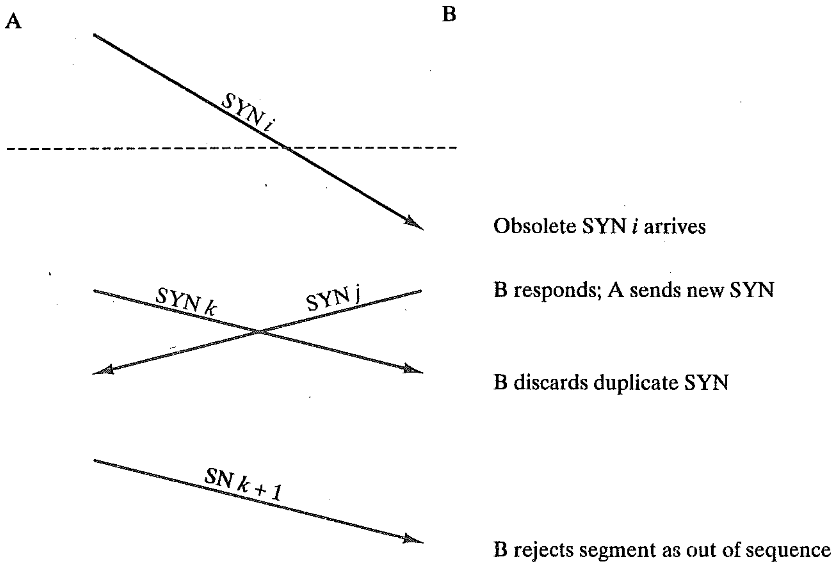


FIGURE 17.11 The two-way handshake: problem with obsolete SYN segments.

nection exists. However, when A initiates data transfer with a segment numbered *k*, B rejects the segment as being out of sequence.

The way out of this problem is for each side to acknowledge explicitly the other's SYN and sequence number. The procedure is known as a three-way handshake. The revised connection state diagram, which is the one employed by TCP, is shown in the upper part of Figure 17.12. A new state (SYN RECEIVED) is added, in which the transport entity hesitates during connection opening to assure that the SYN segments sent by the two sides have both been acknowledged before the connection is declared established. In addition to the new state, there is a control segment (RST) to reset the other side when a duplicate SYN is detected.

Figure 17.13 illustrates typical three-way handshake operations. Transport entity A initiates the connection; a SYN includes the sending sequence number, *i*. The responding SYN acknowledges that number and includes the sequence number for the other side. A acknowledges the SYN/ACK in its first data segment. Next is shown a situation in which an old SYN *X* arrives at B after the close of the relevant connection. B assumes that this is a fresh request and responds with SYN *j*, ACK *i*. When A receives this message, it realizes that it has not requested a connection and therefore sends an RST, ACK *j*. Note that the ACK *j* portion of the RST message is essential so that an old duplicate RST does not abort a legitimate connection establishment. The final example shows a case in which an old SYN, ACK arrives in the middle of a new connection establishment. Because of the use of sequence numbers in the acknowledgments, this event causes no mischief.

The upper part of Figure 17.12 does not include transitions in which RST is sent. This was done for simplicity. The basic rule is to send an RST if the connection state is not yet OPEN and an invalid ACK (one that does not reference something that was sent) is received. The reader should try various combinations of

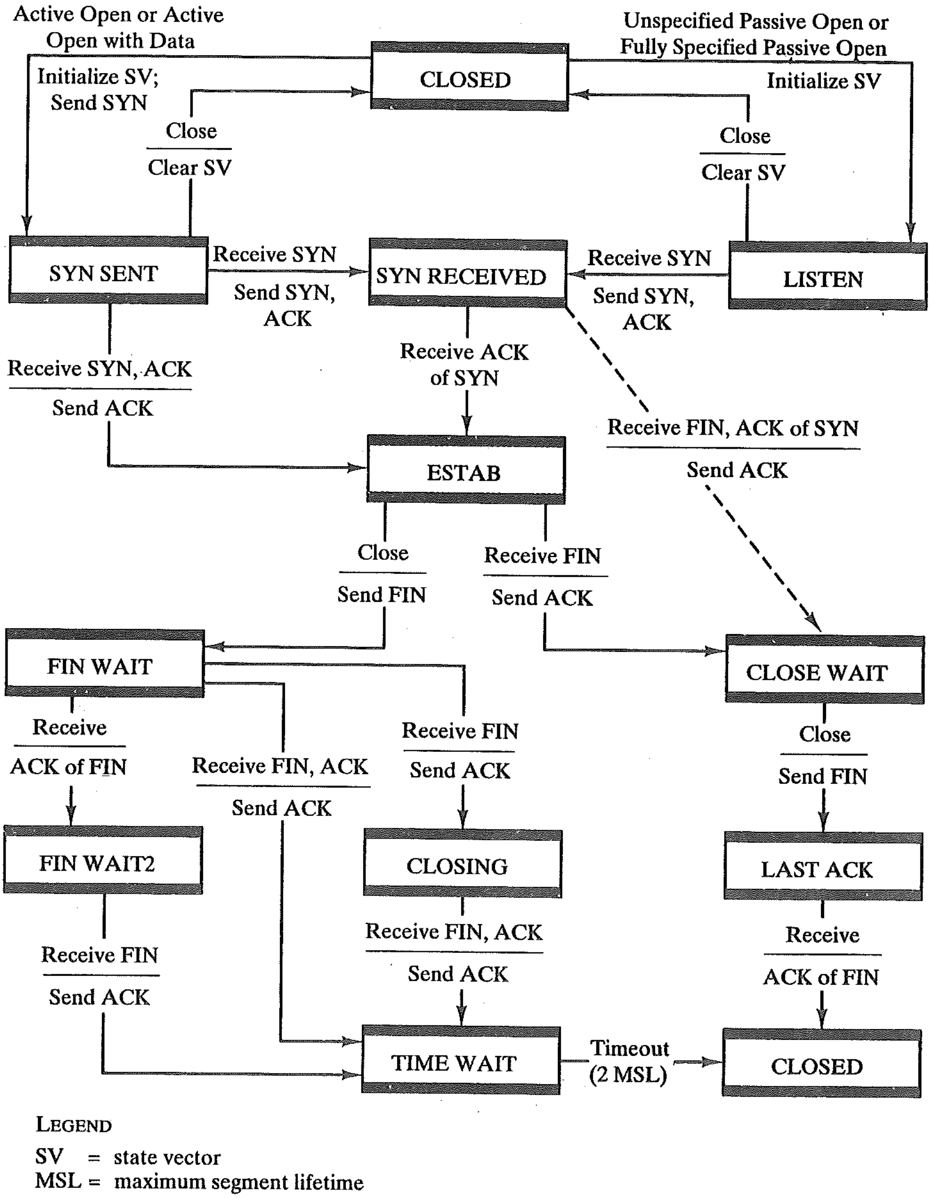
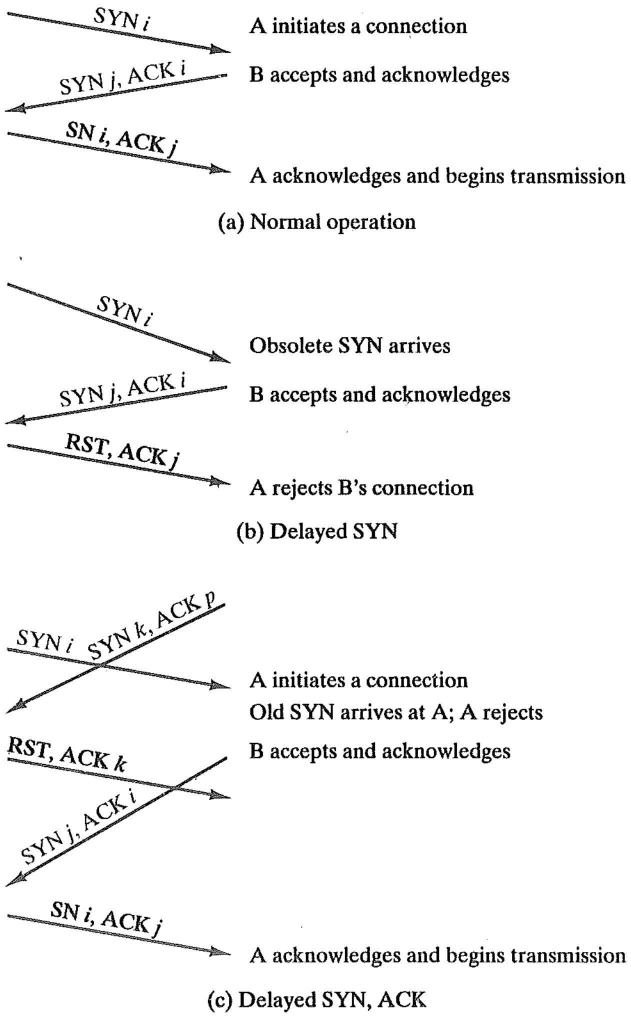


FIGURE 17.12 TCP entity state diagram.

events to see that this connection establishment procedure works in spite of any combination of old and lost segments.

Connection Termination

The state diagram of Figure 17.7 defines the use of a simple two-way handshake for connection establishment, which was found to be unsatisfactory in the face of an unreliable network service. Similarly, the two-way handshake defined in that dia-



(a) Normal operation

(b) Delayed SYN

(c) Delayed SYN, ACK

FIGURE 17.13 Examples of three-way handshake.

gram for connection termination is inadequate for an unreliable network service. The following scenario could be caused by a misordering of segments. A transport entity in the CLOSE WAIT state sends its last data segment, followed by a FIN segment, but the FIN segment arrives at the other side before the last data segment. The receiving transport entity will accept that FIN, close the connection, and lose the last segment of data. To avoid this problem, a sequence number can be associated with the FIN, which can be assigned the next sequence number after the last octet of transmitted data. With this refinement, the receiving transport entity, upon receiving a FIN, will wait if necessary for the late-arriving data before closing the connection.

A more serious problem is the potential loss of segments and the potential presence of obsolete segments. Figure 17.12 shows that the termination procedure adopts a similar solution to that used for connection establishment. Each side must explicitly acknowledge the FIN of the other, using an ACK with the sequence number of the FIN to be acknowledged. For a graceful close, a transport entity requires the following:

- It must send a FIN *i* and receive an ACK *i*.
- It must receive a FIN *j* and send an ACK *j*.
- It must wait an interval equal to twice the maximum-expected segment lifetime

Crash Recovery

When the system upon which a transport entity is running fails and subsequently restarts, the state information of all active connections is lost. The affected connections become *half-open*, as the side that did not fail does not yet realize the problem.

The still active side of a half-open connection can close the connection using a give-up timer. This timer measures the time the transport machine will continue to await an acknowledgment (or other appropriate reply) of a transmitted segment after the segment has been retransmitted the maximum number of times. When the timer expires, the transport entity assumes that either the other transport entity or the intervening network has failed. As a result, the timer closes the connection, and signals an abnormal close to the TS user.

In the event that a transport entity fails and quickly restarts, half-open connections can be terminated more quickly by the use of the RST segment. The failed side returns an RST *i* to every segment *i* that it receives. When the RST *i* reaches the other side, it must be checked for validity based on the sequence number *i*, as the RST could be in response to an old segment. If the reset is valid, the transport entity performs an abnormal termination.

These measures clean up the situation at the transport level. The decision as to whether to reopen the connection is up to the TS users. The problem is one of synchronization. At the time of failure, there may have been one or more outstanding segments in either direction. The TS user on the side that did not fail knows how much data it has received, but the other user may not if state information were lost. Thus, there is the danger that some user data will be lost or duplicated.

17.3 TCP

The TCP/IP protocol suite includes two transport-level protocols: the Transmission Control Protocol (TCP), which is connection-oriented, and the User Datagram Protocol (UDP), which is connectionless. In this section, we look at TCP (specified in RFC 793)—first at the service it provides to the TS user, and then at the internal protocol details.

TCP Services

TCP is designed to provide reliable communication between pairs of processes (TCP users) across a variety of reliable and unreliable networks and internets. Functionally, it is equivalent to Class 4 ISO Transport. In contrast to the ISO model, TCP is stream oriented. That is, TCP users exchange streams of data. The data are placed in allocated buffers and transmitted by TCP in segments. TCP supports security and precedence labeling. In addition, TCP provides two useful facilities for labeling data, *push* and *urgent*:

- **Data stream push.** Ordinarily, TCP decides when sufficient data have accumulated to form a segment for transmission. The TCP user can require TCP to transmit all outstanding data up to and including those labeled with a push flag. On the receiving end, TCP will deliver these data to the user in the same manner; a user might request this if it has come to a logical break in the data.
- **Urgent data signaling.** This provides a means of informing the destination TCP user that significant or "urgent" data is in the upcoming data stream. It is up to the destination user to determine appropriate action.

As with IP, the services provided by TCP are defined in terms of primitives and parameters. The services provided by TCP are considerably richer than those provided by IP, and, hence, the set of primitives and parameters is more complex. Table 17.2 lists TCP service request primitives, which are issued by a TCP user to TCP, and Table 17.3 lists TCP service response primitives, which are issued by TCP to a local TCP user. Table 17.4 provides a brief definition of the parameters involved. Several comments are in order.

The two Passive Open commands signal the TCP user's willingness to accept a connection request. The Active Open with Data allows the user to begin transmitting data with the opening of the connection.

TCP Header Format

TCP uses only a single type of protocol data unit, called a TCP segment. The header is shown in Figure 17.14. Because one header must serve to perform all protocol mechanisms, it is rather large, with a minimum length of 20 octets. The fields are

- **Source port (16 bits).** Source service access point.
- **Destination port (16 bits).** Destination service access point.
- **Sequence number (32 bits).** Sequence number of the first data octet in this segment except when SYN is present. If SYN is present, it is the initial sequence number (ISN), and the first data octet is ISN + 1.
- **Acknowledgment number (32 bits).** A piggybacked acknowledgment. Contains the sequence number of the next octet that the TCP entity expects to receive.
- **Data offset (4 bits).** Number of 32-bit words in the header.

TABLE 17.2 TCP service request primitives.

Primitive	Parameters	Description
Unspecified Passive Open	source-port, [timeout], [timeout-action], [precedence], [security-range]	Listen for connection attempt at specified security and precedence from any remote destination.
Fully Specified Passive Open	source-port, destination-port, destination-address, [timeout], [timeout-action], [precedence], [security-range]	Listen for connection attempt at specified security and precedence from specified destination.
Active Open	source-port, destination-port, destination-address, [timeout], [timeout-action], [precedence], [security]	Request connection at a particular security and precedence to a specified destination.
Active Open with Data	source-port, destination-port, destination-address, [timeout], [timeout-action], [precedence], [security], data, data-length, PUSH-flag, URGENT-flag	Request connection at a particular security and precedence to a specified destination and transmit data with the request
Send	local-connection-name, data, data-length, PUSH-flat, URGENT-flag, [timeout], [timeout-action]	Transfer data across named connection
Allocate	local-connection-name, data-length	Issue incremental allocation for receive data to TCP
Close	local-connection-name	Close connection gracefully
Abort	local-connection-name	Close connection abruptly
Status	local-connection-name	Query connection status

Note: Square brackets indicate optional parameters.

- **Reserved (6 bits).** Reserved for future use.
- **Flags (6 bits).**

  URG: Urgent pointer field significant.

  ACK: Acknowledgment field significant.

  PSH: Push function.

  RST: Reset the connection.

  SYN: Synchronize the sequence numbers.

  FIN: No more data from sender.
- **Window (16 bits).** Flow control credit allocation, in octets. Contains the number of data octets beginning with the one indicated in the acknowledgment field that the sender is willing to accept.
- **Checksum (16 bits).** The one's complement of the sum modulo 2^{16}–1 of all the 16-bit words in the segment, plus a pseudo-header. The situation is described below.
- **Urgent Pointer (16 bits).** Points to the octet following the urgent data; this allows the receiver to know how much urgent data are coming.
- **Options (Variable).** At present, only one option is defined, which specifies the maximum segment size that will be accepted.

TABLE 17.3 TCP service response primitives.

Primitive	Parameters	Description
Open ID	local-connection-name, source-port, destination-port*, destination-address*,	Informs TCP user of connection name assigned to pending connection requested in an Open primitive
Open Failure	local-connection-name	Reports failure of an Active Open request
Open Success	local-connection-name	Reports completion of pending Open request
Deliver	local-connection-name, data, data-length, URGENT-flag	Reports arrival of data
Closing	local-connection-name	Reports that remote TCP user has issued a Close and that all data sent by remote user have been delivered
Terminate	local-connection-name, description	Reports that the connection has been terminated; a description of the reason for termination is provided
Status Response	local-connection-name, source-port, source-address, destination-port, destination-address, connection-state, receive-window, send-window, amount-awaiting-ACK, amount-awaiting-receipt, urgent-state, precedence, security, timeout	Reports current status of connection
Error	local-connection-name, description	Reports service-request or internal error

* = Not used for Unspecified Passive Open.

Several of the fields in the TCP header warrant further elaboration. The *source port* and *destination port* specify the sending and receiving users of TCP. As with IP, there are a number of common users of TCP that have been assigned numbers; these numbers should be reserved for that purpose in any implementation. Other port numbers must be arranged by agreement between communicating parties.

The *sequence number* and *acknowledgment number* are bound to octets rather than to entire segments. For example, if a segment contains sequence number 1000 and includes 600 octets of data, the sequence number refers to the first octet in the data field; the next segment in logical order will have sequence number 1600. Thus, TCP is logically stream-oriented: It accepts a stream of octets from the user, groups them into segments as it sees fit, and numbers each octet in the stream.

The *checksum* field applies to the entire segment, plus a pseudo-header prefixed to the header at the time of calculation (at both transmission and reception). The pseudo-header includes the following fields from the IP header: source and destination internet address and protocol, plus a segment length field (Figure 17.15). By including the pseudo-header, TCP protects itself from misdelivery by IP. That is, if IP delivers a segment to the wrong host, even if the segment contains no bit errors, the receiving TCP entity will detect the delivery error. If TCP is used over IPv6, then the pseudo-header is different, and is depicted in Figure 16.21.

The reader may feel that some items are missing from the TCP header, and that is indeed the case. TCP is designed specifically to work with IP. Hence, some

TABLE 17.4 TCP Service parameters.

Source Port	Local TCP user.
Timeout	Longest delay allowed for data delivery before automatic connection termination or error report; user specified.
Timeout-action	Indicates whether the connection is terminated or an error is reported to the TCP user in the event of a timeout.
Precedence	Precedence level for a connection. Takes on values zero (lowest) through seven (highest); same parameter as defined for IP
Security-range	Allowed ranges in compartment, handling restriction, transmission control codes, and security levels.
Destination Port	Remote TCP user.
Destination Address	Internet address of remote host.
Security	Security information for a connection, including security level, compartment, handling restriction, and transmission control code; same parameter as defined for IP.
Data	Block of data sent by TCP user or delivered to a TCP user.
Data Length	Length of block of data sent or delivered.
PUSH flag	If set, indicates that the associated data are to be provided with the urgent data stream push service.
URGENT flag	If set, indicates that the associated data are to be provided with the urgent data signaling service.
Local Connection Name	Identifier of a connection defined by a (local socket, remote socket) pair; provided by TCP.
Description	Supplementary information in a Terminate or Error primitive.
Source Address	Internet address of the local host.
Connection State	State of referenced connection (CLOSED, ACTIVE OPEN, PASSIVE OPEN, ESTABLISHED, CLOSING).
Receive Window	Amount of data in octets the local TCP entity is willing to receive.
Send Window	Amount of data in octets permitted to be sent to remote TCP entity.
Amount Awaiting ACK	Amount of previously transmitted data awaiting acknowledgment.
Amount Awaiting Receipt	Amount of data in octets buffered at local TCP entity pending receipt by local TCP user.
Urgent State	Indicates to the receiving TCP user whether there are urgent data available or whether all urgent data, if any, have been delivered to the user.

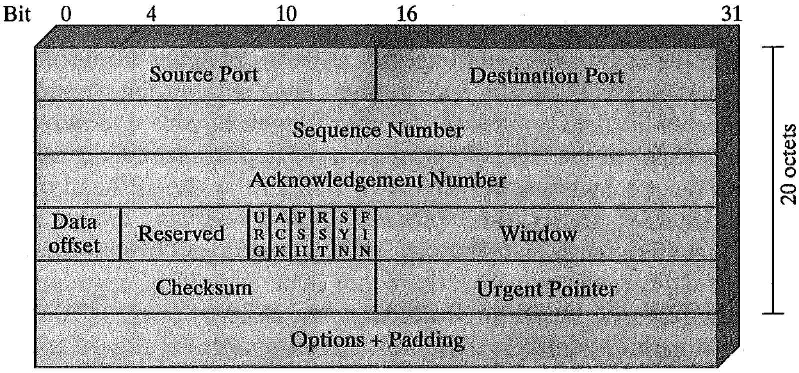


FIGURE 17.14 TCP header.

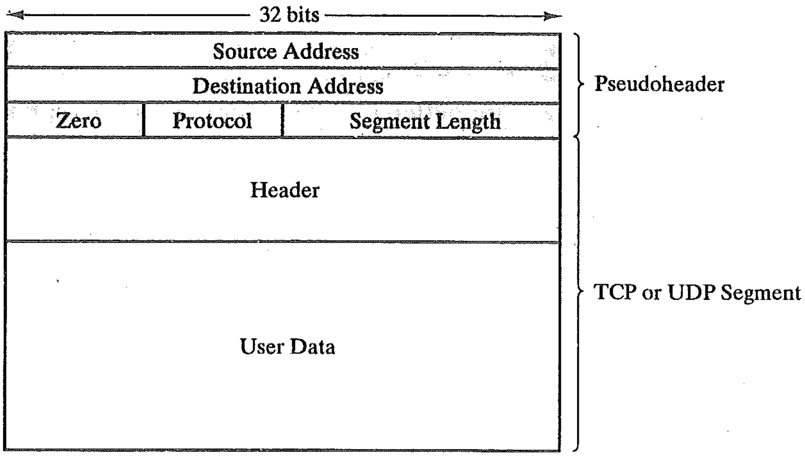


FIGURE 17.15 Scope of checksum for TCP and UDP.

user parameters are passed down by TCP to IP for inclusion in the IP header. The relevant ones are

- Precedence: a 3-bit field
- Normal-delay/low-delay
- Normal-throughput/high-throughput
- Normal-reliability/high-reliability
- Security: an 11-bit field

It is worth observing that this TCP/IP linkage means that the required minimum overhead for every data unit is actually 40 octets.

TCP Mechanisms

Connection Establishment

Connection establishment in TCP always uses a three-way handshake. When the SYN flag is set, the segment is essentially a request for connection (RFC), and thus functions as explained in Section 17.2. To initiate a connection, an entity sends an RFC *X*, where *X* is the initial sequence number. The receiver responds with RFC *Y*, ACK *X* by setting both the SYN and ACK flags. Finally, the initiator responds with ACK *Y*. If both sides issue crossing RFCs, no problem results; both sides respond with ACKs.

A connection is uniquely determined by the source and destination ports. Thus, at any one time, there can only be a single TCP connection between a unique pair of ports. However, a given port can support multiple connections, each with a different partner port.

Data Transfer

Although data are transferred in segments over a transport connection, data transfer is viewed logically as consisting of a stream of octets. Hence, every octet is numbered, modulo 2^{32}. Each segment contains the sequence number of the first octet in the data field. Flow control is exercised using a credit allocation scheme in which the credit is a number of octets rather than a number of segments.

As was mentioned, data are buffered by the transport entity on both transmission and reception. TCP normally exercises its own discretion as to when to construct a segment for transmission and when to release received data to the user. The PUSH flag is used to force the data so-far accumulated to be sent by the transmitter and passed on by the receiver. This serves an end-of-letter function.

The user may specify a block of data as urgent. TCP will designate the end of that block with an urgent pointer and send it out in the ordinary data stream. The receiving user is alerted that urgent data are being received.

If, during data exchange, a segment arrives that is apparently not meant for the current connection, the RST flag is set on an outgoing segment. Examples of this situation are delayed duplicate SYNs and an acknowledgment of data not yet sent.

Connection Termination

The normal means of terminating a connection consists of a graceful close. Each TCP user must issue a CLOSE primitive. The transport entity sets the FIN bit on the last segment that it sends out, which also contains the last of the data to be sent on this connection.

An abrupt termination occurs if the user issues an ABORT primitive. In this case, the entity abandons all attempts to send or receive data and discards data in its transmission and reception buffers. An RST segment is sent to the other side.

TCP Implementation Policy Options

The TCP standard provides a precise specification of the protocol to be used between TCP entities. However, certain aspects of the protocol admit several possible implementation options. Although two implementations that choose alternative options will be interoperable, there may be performance implications. The design-area options are the following:

- Send policy
- Deliver policy
- Accept policy
- Retransmit policy
- Acknowledge policy

Send Policy

In the absence of pushed data and a closed transmission window (see Figure 17.6a), a sending TCP entity is free to transmit data at its own convenience. As data are issued by the user, they are buffered in the transmit buffer. TCP may construct a segment for each batch of data provided by its user, or it may wait until a certain amount of data accumulates before constructing and sending a segment. The actual policy will depend on performance considerations. If transmissions are infrequent and large, there is low overhead in terms of segment generation and processing. On the other hand, if transmissions are frequent and small, then the system is providing quick response.

One danger of frequent, small transmissions is known as the silly window syndrome, which is discussed in Problem 17.19.

Deliver Policy

In the absence of a push, a receiving TCP entity is free to deliver data to the user at its own convenience. It may deliver data as each in-order segment is received, or it may buffer data from a number of segments in the receive buffer before delivery. The actual policy will depend on performance considerations. If deliveries are infrequent and large, the user is not receiving data as promptly as may be desirable. On the other hand, if deliveries are frequent and small, there may be unnecessary processing both in TCP and in the user software, as well as an unnecessary number of operating-system interrupts.

Accept Policy

When all data segments arrive in order over a TCP connection, TCP places the data in a receive buffer for delivery to the user. It is possible, however, for segments to arrive out of order. In this case, the receiving TCP entity has two options:

- **In-order.** Accept only segments that arrive in order; any segment that arrives out of order is discarded.
- **In-window.** Accept all segments that are within the receive window (see Figure 17.6b).

The in-order policy makes for a simple implementation but places a burden on the networking facility, as the sending TCP must time-out and retransmit segments that were successfully received but discarded because of misordering. Furthermore, if a single segment is lost in transit, then all subsequent segments must be retransmitted once the sending TCP times out on the lost segment.

The in-window policy may reduce transmissions but requires a more complex acceptance test and a more sophisticated data storage scheme to buffer and keep track of data accepted out of order.

For class 4 ISO transport (TP4), the in-window policy is mandatory.

Retransmit Policy

TCP maintains a queue of segments that have been sent but not yet acknowledged. The TCP specification states that TCP will retransmit a segment if it fails to receive an acknowledgment within a given time. A TCP implementation may employ one of three retransmission strategies:

- **First-only.** Maintain one retransmission timer for the entire queue. If an acknowledgment is received, remove the appropriate segment or segments from the queue and reset the timer. If the timer expires, retransmit the segment at the front of the queue and reset the timer.

- **Batch.** Maintain one retransmission timer for the entire queue. If an acknowledgment is received, remove the appropriate segment or segments from the queue and reset the timer. If the timer expires, retransmit all segments in the queue and reset the timer.
- **Individual.** Maintain one timer for each segment in the queue. If an acknowledgment is received, remove the appropriate segment or segments from the queue and destroy the corresponding timer or timers. If any timer expires, retransmit the corresponding segment individually and reset its timer.

The first-only policy is efficient in terms of traffic generated, as only lost segments (or segments whose ACK was lost) are retransmitted. Because the timer for the second segment in the queue is not set until the first segment is acknowledged, however, there can be considerable delays; the individual policy solves this problem at the expense of a more complex implementation. The batch policy also reduces the likelihood of long delays but may result in unnecessary retransmissions.

The actual effectiveness of the retransmit policy depends in part on the accept policy of the receiver. If the receiver is using an in-order accept policy, then it will discard segments received after a lost segment; this fits best with batch retransmission. If the receiver is using an in-window accept policy, then a first-only or individual retransmission policy is best. Of course, in a mixed network of computers, both accept policies may be in use.

The ISO TP4 specification outlines essentially the same options for a retransmit policy without mandating a particular one.

Acknowledge Policy

When a data segment arrives that is in sequence, the receiving TCP entity has two options concerning the timing of acknowledgment:

- **Immediate.** When data are accepted, immediately transmit an empty (no data) segment containing the appropriate acknowledgment number.
- **Cumulative.** When data are accepted, record the need for acknowledgment, but wait for an outbound segment with data on which to piggyback the acknowledgment. To avoid a long delay, set a window timer (see Table 17.1); if the timer expires before an acknowledgment is sent, transmit an empty segment containing the appropriate acknowledgment number.

The immediate policy is simple and keeps the sending TCP entity fully informed, which limits unnecessary retransmissions. However, this policy results in extra segment transmissions, namely, empty segments used only to ACK. Furthermore, the policy can case a further load on the network. Consider that a TCP entity receives a segment and immediately sends an ACK; then the data in the segment are released to the application, which expands the receive window, triggering another empty TCP segment to provide additional credit to the sending TCP entity.

Because of the potential overhead of the immediate policy, the cumulative policy is typically used. Recognize, however, that the use of this policy requires more processing at the receiving end and complicates the task of estimating round-trip delay by the sending TCP entity.

The ISO TP4 specification outlines essentially the same options for an acknowledge policy without mandating a particular one.

17.4 UDP

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), specified in RFC 768.

UDP provides a connectionless service for application-level procedures. Thus, UDP is basically an unreliable service; delivery and duplicate protection are not guaranteed. However, the overhead of the protocol is low, which may be adequate in many cases. An example of the use of UDP is in the context of network management, as described in Chapter 19.

UDP sits on top of IP. Because it is connectionless, UDP has very little to do. Essentially, it adds a port addressing capability to IP, best seen by examining the UDP header, shown in Figure 17.16. The header includes a source port and a destination port. The length field contains the length of the entire UDP segment, including header and data. The checksum is the same algorithm used for TCP and IP. For UDP, the checksum applies to the entire UDP segment plus a pseudo-header prefixed to the UDP header at the time of calculation and is the same one used for TCP (Figure 17.15). If an error is detected, the segment is discarded and no further action is taken.

The checksum field in UDP is optional. If it is not used, it is set to zero. However, it should be pointed out that the IP checksum applies only to the IP header and not to the data field, which in this case consists of the UDP header and the user data. Thus, if no checksum calculation is performed by UDP, then no check is made on the user data.

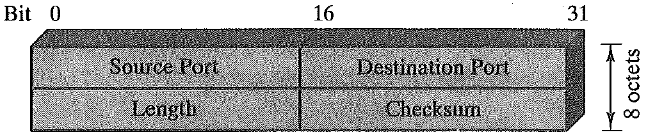


FIGURE 17.16 UDP header.

17.5 RECOMMENDED READING

[PART88] contains reprints of a number of key papers dealing with specific transport protocol design issues. Good accounts of transport protocols can be found in [SPRA91] and [HALS96].

HALS96 Halsall, F. *Data Communications, Computer Networks, and Open Systems*. Reading, MA: Addison-Wesley, 1996.

PART88 Partridge, C. *Innovations in Internetworking*. Norwood, MA: Artech House, 1988.

SPRA91 Spragins, J., Hammond, J., and Pawlikowski, K. *Telecommunications Protocols and Design*. Reading, MA: Addison-Wesley, 1991.

17.6 PROBLEMS

17.1 It is common practice in most transport protocols (indeed, most protocols at all levels) for control and data to be multiplexed over the same logical channel on a per–user-connection basis. An alternative is to establish a single control transport connection between each pair of communicating transport entities. This connection would be used to carry control signals relating to all user transport connections between the two entities. Discuss the implications of this strategy.

17.2 The discussion of flow control with a reliable network service referred to a backpressure mechanism utilizing a lower-level flow control protocol. Discuss the disadvantages of this strategy.

17.3 Two transport entities communicate across a reliable network. Let the normalized time to transmit a segment equal 1. Assume that the end-to-end propagation delay is 3, and that it takes a time 2 to deliver data from a received segment to the transport user. The sender initially granted a credit of seven segments. The receiver uses a conservative flow control policy and updates its credit allocation at every opportunity. What is the maximum achievable throughput?

17.4 Draw diagrams similar to Figure 17.8 for the following. (Assume a reliable sequenced network service.)

a. Connection termination: active/passive.
b. Connection termination: active/active.
c. Connection rejection.
d. Connection abortion: User issues an OPEN to a listening user, and then issues a CLOSE before any data are exchanged.

17.5 With a reliable sequencing network service, are segment sequence numbers strictly necessary? What, if any, capability is lost without them?

17.6 Consider a connection-oriented network service that suffers a reset. How could this problem be dealt with by a transport protocol that assumes that the network service is reliable except for resets?

17.7 The discussion of retransmission strategy made reference to three problems associated with dynamic timer calculation. What modifications to the strategy would help to alleviate those problems?

17.8 Consider a transport protocol that uses a connection-oriented network service. Suppose that the transport protocol uses a credit-allocation flow control scheme, and the network protocol uses a sliding window scheme. What relationship, if any, should there be between the dynamic window of the transport protocol and the fixed window of the network protocol?

17.9 In a network that has a maximum packet size of 128 bytes, a maximum packet lifetime of 30 s, and an 8-bit packet sequence number, what is the maximum data rate per connection?

17.10 Is a deadlock possible using only a two-way handshake instead of a three-way handshake? Give an example or prove otherwise.

17.11 Why is UDP needed? Why can't a user program directly access IP?

17.12 Listed below are four strategies that can be used to provide a transport user with the address of the destination transport user. For each one, describe an analogy with the Postal Service user.

a. Know the address ahead of time.
b. Make use of a "well-known address."
c. Use a name server.
d. Addressee is spawned at request time.

17.13 In a credit flow control scheme, what provision can be made for credit allocations that are lost or misordered in transit?

17.14 What happens in Figure 17.7 if an SYN comes in while the requested user is in CLOSED? Is there any way to get the attention of the user when it is not listening?

17.15 In Section 17.2, it was pointed out that an adaptive, or dynamic, calculation of the retransmission timer may be desirable. One approach would be to simply take the average of observed round-trip times over a number of segments, and then set the retransmission timer equal to a value slightly greater then the average. If the average accurately predicts future round-trip delays, then the resulting retransmission timer will yield good performance. The simple averaging method can be expressed as follows:

$$ARTT(K+1) = \frac{1}{K+1} \sum_{i=1}^{K+1} RTT(i)$$

where $RTT(K)$ is the round-trip time observed for the Kth transmitted segment, and $ARTT(K)$ is the average round-trip time for the first K segments. This expression can be rewritten as follows:

$$ARTT(K+1) = \frac{K}{K+1} ARTT(K) + \frac{1}{K+1} RTT(K+1)$$

a. The TCP standard recommends the following formulas for estimating round-trip time and setting the retransmission timer:

$$SRTT(K+1) = \alpha \times SRTT(K) + (1-\alpha) \times RTT(K+1)$$

$$RXT(K+1) = MIN(UBOUND, MAX(LBOUND, \beta \times \mathrm{SRTT}(K+1)))$$

where $SRTT$ is referred to as the smoothed round-trip time, $RXT(K)$ is the retransmission timer value assigned after the first K segments, UBOUND and LBOUND are pre-chosen fixed upper and lower bounds on the timer value, and α and β are constants. The recommended initial values for a new connection are $RTT = 0$ seconds, and $RXT = 3$ seconds.

Now, consider that the observed round-trip times for the first 20 segments on a connection have the following values: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10. Show the values for $ARRT(K)$, $SRTT(K)$, and $RXT(K)$ for this sequence. For the latter two parameters, include results for $\alpha = 0.25$, 0.75, and 0.875, and $\beta = 1.0$ and 1.25.

b. Repeat part (a) for the sequence 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10.

c. What function do α and β perform, and what is the effect of higher and lower values of each? Compare $SRTT$ with $ARTT$.

17.16 The technique specified in the TCP standard, and described in the previous problem, enables a TCP entity to adapt to changes in round-trip time. However, it does not cope well with a situation in which the round-trip time exhibits a relatively high variance. To cope with this, Von Jacobsen proposed a refinement to the standard algorithm that has now been officially adopted for use with TCP. The algorithm can be summarized as follows:

$$\begin{aligned}
SRTT(K+1) &= (1-\gamma) \times SRTT(K) + \gamma \times RTT(K+1) \\
ERR(K+1) &= RTT(K+1) - SRTT(K) \\
SDEV(K+1) &= SDEV(K) + \gamma \times (|ERR(K+1)| - SDEV(K)) \\
&= (1-\gamma) \times SDEV(K) + \gamma \times |ERR(K+1)| \\
RXT(K+1) &= SRTT(K+1) + 2 \times SDEV(K+1)
\end{aligned}$$

where $SDEV$ is the smoothed estimate of deviation of round-trip time, and ERR is the difference between the predicted and observed values of round-trip time.

a. Show the values of $SRTT(K)$ and $RXT(K)$ for the same sequences used in parts (a) and (b) of the preceding problem, for $\gamma = 0.75$, 0.25, and 0.125.

b. Compare Von Jacobsen's algorithm with the original TCP algorithm. What does the Von Jacobsen algorithm do? How is it superior to the original algorithm?

17.17 The Von Jacobsen algorithm is quite effective in tracking the round-trip time until there is a period in which the variability of round-trip times is high or until there is a need to retransmit segments due to timer expiration.

a. What problem does retransmission cause in estimating round-trip time?

b. Suggest and justify a way to compensate for retransmissions. Hint: The approach mandated for TCP, known as Karn's algorithm, is as follows: Follow the Von Jacobsen algorithm until a timer expires, necessitating retransmission. Then use a different strategy until an acknowledgment arrives before a timer expires. Now all you need to do is come up with this alternative strategy.

17.18 The primary objective of developing a dynamic retransmission timer algorithm is to coordinate the needs of sender and receiver so that the sender is not delivering data either too quickly or too slowly for the receiver. A dynamic retransmission timer has the side effect of helping to cope with network congestion. As network congestion increases, retransmission timers will lengthen, thereby reducing the total network load. However, dynamic timers alone are insufficient to fully cope with severe network congestion. Because the details of the network level are hidden from TCP, it is difficult for TCP to shape its behavior to respond to congestion. However, the fact that retransmission timers are lengthening gives the TCP entity some feel for the degree of congestion.

a. TCP mandates the use of a congestion window, whose size is altered to respond to congestion. At any time, TCP acts as if its window size is

allowed__window = MIN [received__credit, congestion__window].

During periods of normal traffic, the congestion window is the same as the actual window. Suggest a strategy for altering the congestion window during periods of congestion. Hint: Review the binary-exponential backoff technique used for CSMA/CD LANs.

b. When a period of congestion appears to be over, we would like to restore transmission to the use of the actual window. However, it is best to avoid an abrupt transition to the full window, as this might reawaken congestion. Suggest a conservative approach.

17.19 A poor implementation of TCP's sliding-window scheme can lead to extremely poor performance. There is a phenomenon known as the Silly Window Syndrome (SWS), which can easily cause degradation in performance by several factors of ten. As an example of SWS, consider an application that is engaged in a lengthy file transfer, and that TCP is transferring this file in 200-octet segments. The receiver initially provides a credit of 1000. The sender uses up this window with five segments of 200 octets. Now suppose that the receiver returns an acknowledgment to each segment and provides an additional credit of 200 octets for every received segment. From the receiver's point of view, this opens the window back up to 1000 octets. However, from the sender's point of view, if the first acknowledgment arrives after five segments have been sent, a window of only 200 octets becomes available. Assume that at some point, the receiver calculates a window of 200 octets but has only 50 octets to send until it reaches a "push" point. It therefore sends 50 octets in one segment, followed by 150 octets in the next segment, and then resumes transmission of 200-octet segments. What might now happen to cause a performance problem? State the SWS in more general terms.

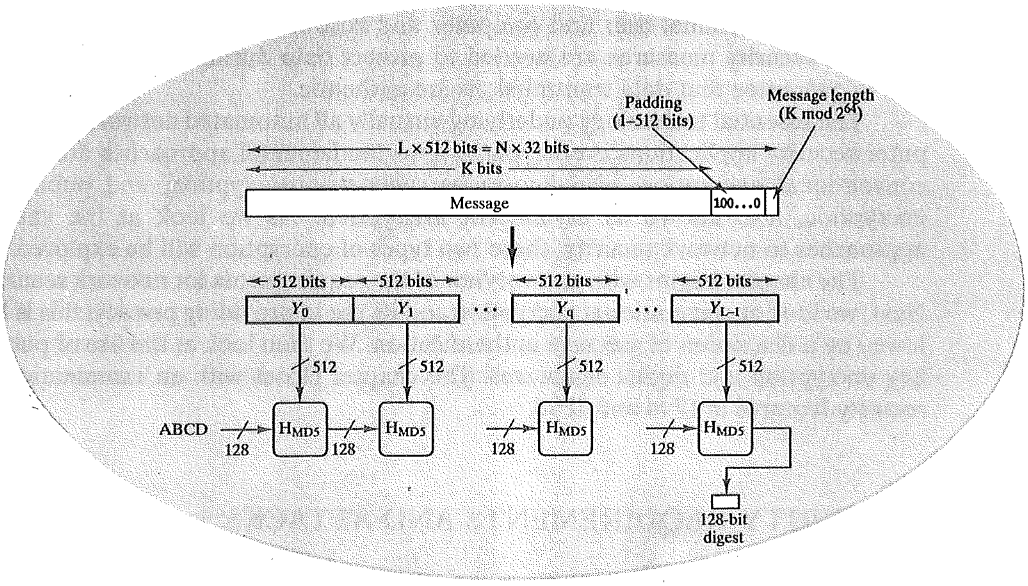
17.20 TCP mandates that both the receiver and the sender should incorporate mechanisms to cope with SWS.

a. Suggest a strategy for the receiver. Hint: Let the receiver "lie" about how much buffer space is available under certain circumstances; state a reasonable rule of thumb for this approach.

b. Suggest a strategy for the sender. Hint: Consider the relationship between the maximum possible send window and what is currently available to send.

CHAPTER 18

NETWORK SECURITY



The requirements of **information security** within an organization have undergone two major changes in the last several decades. Before the widespread use of data processing equipment, the security of information felt to be valuable to an organization was provided primarily by physical and administrative means; an example of the former is the use of rugged filing cabinets with a combination lock for storing sensitive documents; an example of the latter is personnel screening procedures used during the hiring process.

With the introduction of the computer, the need for automated tools for protecting files and other information stored on the computer became evident; this is especially the case for a shared system, such as a time-sharing system, and the need is even more acute for systems that can be accessed over a public telephone or data network. The generic name for the collection of tools designed to protect data and to thwart hackers is **computer security**. Although this is an important topic, it is beyond the scope of this book and will be dealt with only briefly.

The second major change that affected security is the introduction of distributed systems and the use of networks and communications facilities for carrying data between terminal user and computer and between computer and computer. **Network security** measures are needed to protect data during their transmission, and to guarantee that data transmissions are authentic.

The essential technology underlying virtually all automated network and computer security applications is encryption. Two fundamental approaches are in use: conventional encryption, also known as symmetric encryption, and public-key encryption, also known as asymmetric encryption. As we look at the various approaches to network security, these two types of encryption will be explored.

The chapter begins with an overview of the requirements for network security. Next, we look at conventional encryption and its use in providing privacy; this is followed by a discussion of message authentication. We then look at the use of public-key encryption and digital signatures. The chapter closes with an examination of security features in IPv4 and IPv6.

18.1 SECURITY REQUIREMENTS AND ATTACKS

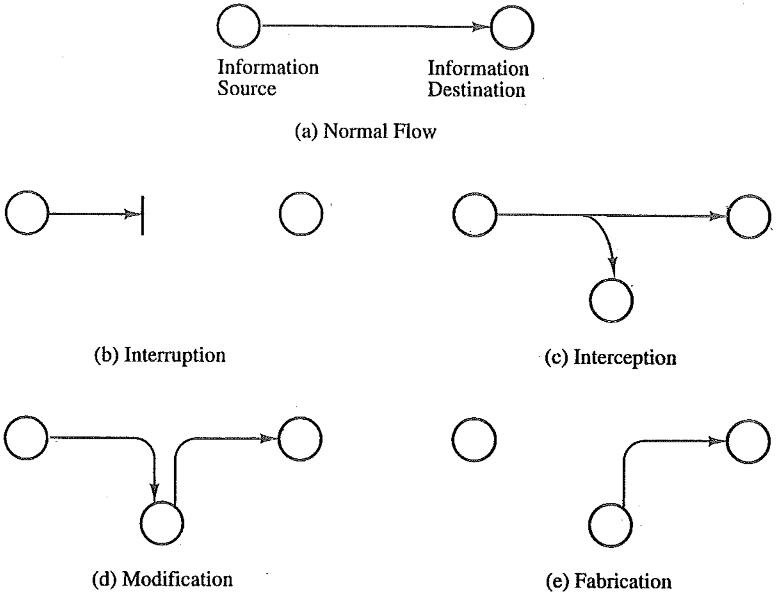
In order to be able to understand the types of threats that exist to security, we need to have a definition of security requirements. Computer and network security address three requirements:

- **Secrecy.** Requires that the information in a computer system only be accessible for reading by authorized parties. This type of access includes printing, displaying, and other forms of disclosure, including simply revealing the existence of an object.
- **Integrity.** Requires that computer system assets can be modified only by authorized parties. Modification includes writing, changing, changing status, deleting, and creating.
- **Availability.** Requires that computer system assets are available to authorized parties.

The types of attacks on the security of a computer system or network are best characterized by viewing the function of the computer system as providing information. In general, there is a flow of information from a source, such as a file or a region of main memory, to a destination, such as another file or a user. This normal flow is depicted in Figure 18.1a. The remaining parts of the figure show the following four general categories of attack:

- **Interruption.** An asset of the system is destroyed or becomes unavailable or unusable. This is an attack on **availability**. Examples include destruction of a piece of hardware, such as a hard disk, the cutting of a communication line, or the disabling of the file management system.
- **Interception.** An unauthorized party gains access to an asset. This is an attack on **confidentiality**. The unauthorized party could be a person, a program, or a computer. Examples include wiretapping to capture data in a network, and the illicit copying of files or programs.
- **Modification.** An unauthorized party not only gains access to but tampers with an asset. This is an attack on **integrity**. Examples include changing values in a data file, altering a program so that it performs differently, and modifying the content of messages being transmitted in a network.
- **Fabrication.** An unauthorized party inserts counterfeit objects into the system. This is an attack on **authenticity**. Examples include the insertion of spurious messages in a network or the addition of records to a file.

A useful categorization of these attacks is in terms of passive attacks and active attacks (Figure 18.2).



(a) Normal Flow

(b) Interruption

(c) Interception

(d) Modification

(e) Fabrication

FIGURE 18.1 Security threats.

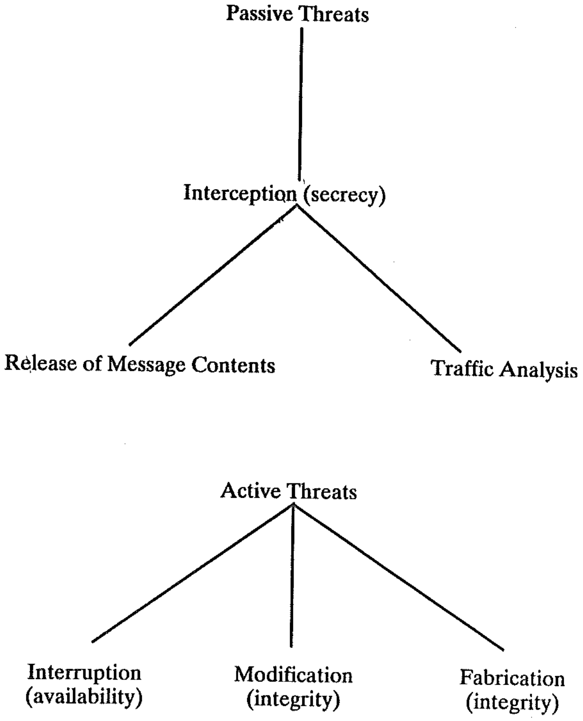


FIGURE 18.2 Active and passive network security threats.

Passive Attacks

Passive attacks mean the eavesdropping on, or monitoring of, transmissions. The goal of the opponent is to obtain information that is being transmitted. Two types of attacks are involved here: release-of-message contents and traffic analysis.

The *release-of-message contents* is easily understood. A telephone conversation, an electronic mail message, a transferred file may contain sensitive or confidential information. We would like to prevent the opponent from learning the contents of these transmissions.

The second passive attack, *traffic analysis,* is more subtle. Suppose that we had a way of masking the contents of messages or other information traffic so that opponents, even if they captured the message, could not extract the information from the message. The common technique for masking contents is encryption. If we had encryption protection in place, an opponent might still be able to observe the pattern of these messages. The opponent could determine the location and identity of communicating hosts and could observe the frequency and length of messages being exchanged. This information might be useful in guessing the nature of the communication that was taking place.

Passive attacks are very difficult to detect because they do not involve any alteration of the data. However, it is feasible to prevent the success of these attacks.

Thus, the emphasis in dealing with passive attacks is on prevention rather than detection.

Active Attacks

The second major category of attack is *active attacks*. These attacks involve some modification of the data stream or the creation of a false stream and can be subdivided into four categories: masquerade, replay, modification of messages, and denial of service.

A *masquerade* takes place when one entity pretends to be a different entity. A masquerade attack usually includes one of the other forms of active attack. For example, authentication sequences can be captured and replayed after a valid authentication sequence has taken place, thus enabling an authorized entity with few privileges to obtain extra privileges by impersonating an entity that has those privileges.

Replay involves the passive capture of a data unit and its subsequent retransmission to produce an unauthorized effect.

Modification of messages simply means that some portion of a legitimate message is altered, or that messages are delayed or reordered, to produce an unauthorized effect. For example, a message meaning "Allow John Smith to read confidential file *accounts*" is modified to mean "Allow Fred Brown to read confidential file *accounts*."

The *denial of service* prevents or inhibits the normal use or management of communications facilities. This attack may have a specific target; for example, an entity may suppress all messages directed to a particular destination (e.g., the security audit service). Another form of service denial is the disruption of an entire network, either by disabling the network or by overloading it with messages so as to degrade performance.

Active attacks present the opposite characteristics of passive attacks. Whereas passive attacks are difficult to detect, measures are available to prevent their success. On the other hand, it is quite difficult to prevent active attacks absolutely, as to do so would require physical protection of all communications facilities and paths at all times. Instead, the goal is to detect them and to recover from any disruption or delays caused by them. Because the detection has a deterrent effect, it may also contribute to prevention.

18.2 PRIVACY WITH CONVENTIONAL ENCRYPTION

The universal technique for providing privacy for transmitted data is conventional encryption. This section looks first at the basic concept of conventional encryption, followed by a discussion of the two most popular conventional encryption techniques: DES and triple DES. We then examine the application of these techniques to achieve privacy.

Conventional Encryption

Figure 18.3 illustrates the conventional encryption process. The original intelligible message, referred to as *plaintext*, is converted into apparently random nonsense, referred to as *ciphertext*. The encryption process consists of an algorithm and a key. The key is a value independent of the plaintext that controls the algorithm. The algorithm will produce a different output depending on the specific key being used at the time. Changing the key changes the output of the algorithm.

Once the ciphertext is produced, it is transmitted. Upon reception, the ciphertext can be transformed back to the original plaintext by using a decryption algorithm and the same key that was used for encryption.

The security of conventional encryption depends on several factors. First, the encryption algorithm must be powerful enough so that it is impractical to decrypt a message on the basis of the ciphertext alone. Beyond that, the security of conventional encryption depends on the secrecy of the key, not on the secrecy of the algorithm. That is, it is assumed that it is impractical to decrypt a message on the basis of the ciphertext *plus* knowledge of the encryption/decryption algorithm. In other words, we don't need to keep the algorithm secret; we only need to keep the key secret.

This feature of conventional encryption is what makes it feasible for widespread use. The fact that the algorithm need not be kept secret means that manufacturers can and have developed low-cost chip implementations of data encryption algorithms. These chips are widely available and incorporated into a number of products. With the use of conventional encryption, the principal security problem is maintaining the secrecy of the key.

Let us take a closer look at the essential elements of a conventional encryption scheme, again using Figure 18.3. There is some source for a message, which

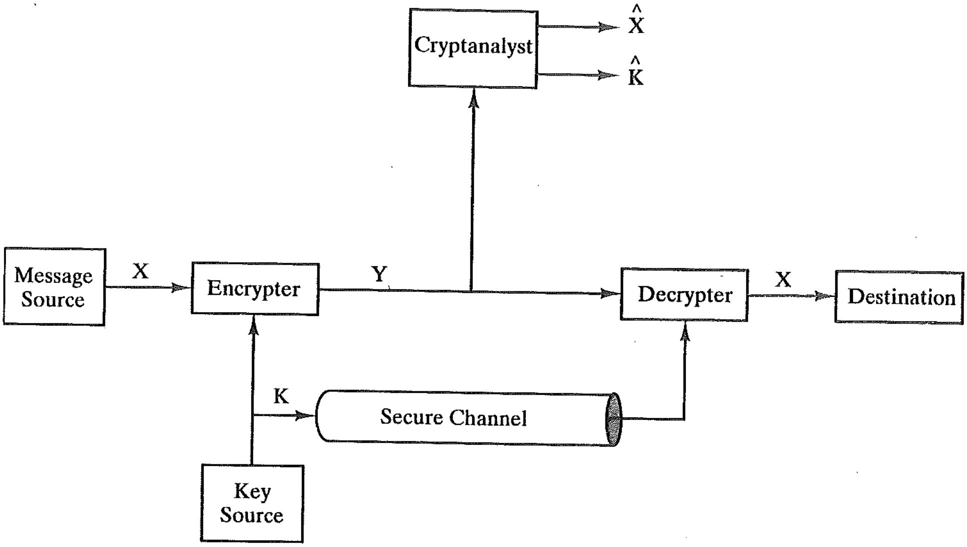


FIGURE 18.3 Model of conventional cryptosystem.

produces the following message in plaintext: $X = [X_1, X_2, \ldots, X_M]$. The M elements of X are letters in some finite alphabet. Traditionally, the alphabet usually consisted of the 26 capital letters. Nowadays, the binary alphabet $\{0, 1\}$ is typically used. For encryption, a key of the form $K = [K_1, K_2, \ldots, K_J]$ is generated. If the key is generated at the message source, then it must also be provided to the destination by means of some secure channel. Alternatively, a third party could generate the key and securely deliver it to both source and destination.

With the message X and the encryption key K as input, the encryption algorithm, or encrypter, forms the ciphertext $Y = [Y_1, Y_2, \ldots, Y_N]$. We can write this as

$$Y = \mathrm{E}_K(X)$$

This notation indicates that Y is produced by using encryption algorithm E as a function of the plaintext X, with the specific function determined by the value of the key K.

The intended receiver, in possession of the key, is able to invert the transformation:

$$X = \mathrm{D}_K(Y)$$

An opponent, observing Y but not having access to K or X, must attempt to recover X and K or both X and K. It is assumed that the opponent does have knowledge of the encryption (E) and decryption (D) algorithms. If the opponent is interested in only this particular message, then the focus of the effort is to recover X by generating a plaintext estimate $\hat{X}$. Often, however, the opponent is interested in being able to read future messages as well, in which case an attempt is made to recover K by generating a plaintext estimate $\hat{K}$.

Encryption Algorithms

The most commonly used conventional encryption algorithms are block ciphers. A block cipher processes the plaintext input in fixed-size blocks, and produces a block of ciphertext of equal size for each plaintext block. The two most important conventional algorithms, both of which are block ciphers, are DES and Triple DES.

The Data Encryption Standard (DES)

The most widely used encryption scheme is defined in the data encryption standard (DES) adopted in 1977 by the National Bureau of Standards, now the National Institute of Standards and Technology (NIST), as Federal Information Processing Standard 46 (FIPS PUB 46). In 1994, NIST "reaffirmed" DES for federal use for another five years [NIST94]; NIST recommends the use of DES for applications other than the protection of classified information.

The overall scheme for DES encryption is illustrated in Figure 18.4. As with any encryption scheme, there are two inputs to the encryption function: the plaintext to be encrypted and the key. In this case, the plaintext must be 64 bits in length, and the key is 56 bits in length.

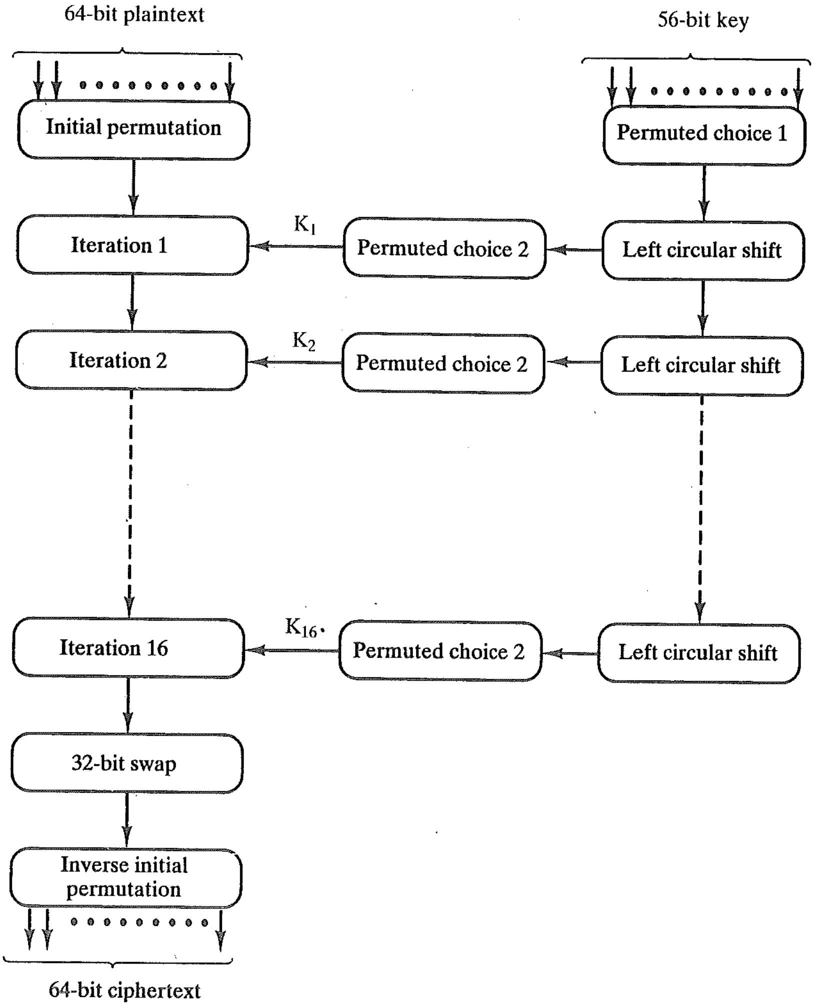


FIGURE 18.4 General depiction of DES algorithm.

The processing of the plaintext proceeds in three phases. First, the 64-bit plaintext passes through an initial permutation (IP) that rearranges the bits to produce the *permuted input.* This IP is followed by a phase consisting of 16 iterations of the same function. The output of the last (16th) iteration consists of 64 bits that are a function of the plaintext input and the key. The left and right halves of the output are swapped to produce the *preoutput.* Finally, the preoutput is passed through a permutation (IP^{-1}) that is the inverse of the initial permutation function, in order to produce the 64-bit ciphertext.

The right-hand portion of Figure 18.4 shows the way in which the 56-bit key is used. Initially, the key is passed through a permutation function. Then, for each of the 16 iterations, a *subkey* (K_i) is produced by the combination of a left circular shift and a permutation. The permutation function is the same for each iteration, but a different subkey is produced because of the repeated shifting of the key bits.

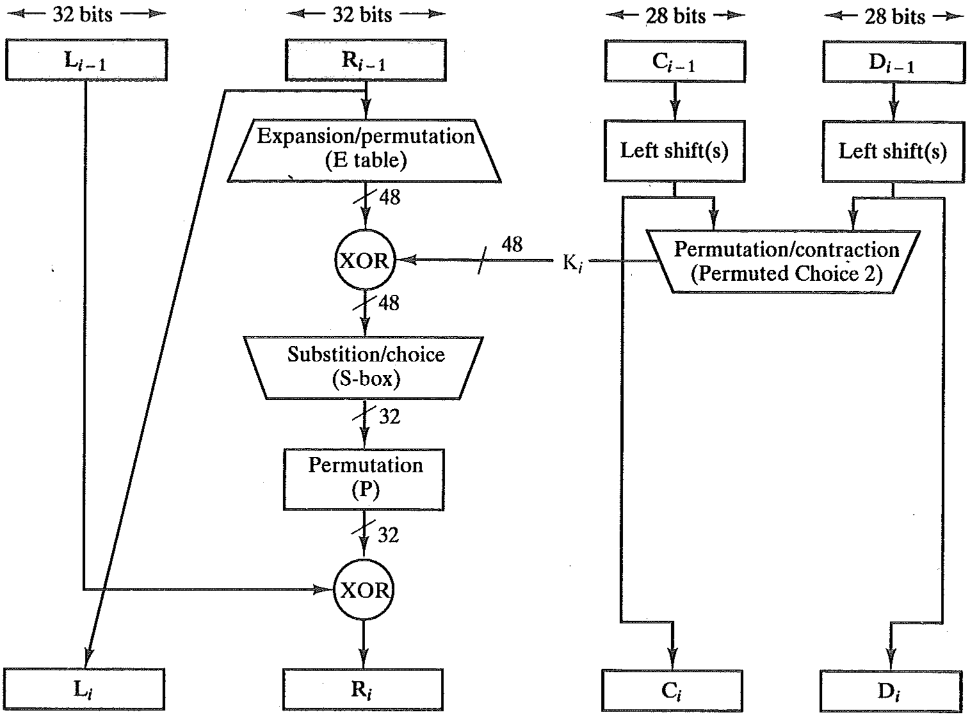


FIGURE 18.5 Single iteration of DES algorithm.

Figure 18.5 examines more closely the algorithm for a single iteration. The 64-bit permuted input passes through 16 iterations, producing an intermediate 64-bit value at the conclusion of each iteration. The left- and right-half of each 64-bit intermediate value are treated as separate 32-bit quantities, labeled L (left) and R (right). The overall processing at each iteration can be summarized in the following formulas:

$$L_i = R_{i-1}$$
$$R_i = L_{i-1} \oplus f(R_{i-1},K_i)$$

where $\oplus$ denotes the bitwise XOR function.

Thus, the left-hand output of an iteration (L_i) is simply equal to the right-hand input to that iteration (R_{i-1}). The right-hand output (R_i) is the exclusive-or of L_{i-1} and a complex function f of R_{i-1} and K_i. This complex function involves both permutation and substitution operations. The substitution operation, represented as tables called "S-boxes," simply maps each combination of 48 input bits into a particular 32-bit pattern.

Returning to Figure 18.4, we see that the 56-bit key used as input to the algorithm is first subjected to a permutation. The resulting 56-bit key is then treated as two 28-bit quantities, labeled C_0 and D_0. At each iteration, C and D are separately subjected to a circular left shift, or rotation, of 1 or 2 bits. These shifted values serve as input to the next iteration. They also serve as input to another permutation function, which produces a 48-bit output that serves as input to the function $f(R_{i-1},K_i)$.

The process of decryption with DES is essentially the same as the encryption process. The rule is as follows: Use the ciphertext as input to the DES algorithm, but use the keys K_i in reverse order. That is, use K_{16} on the first iteration, K_{15} on the second iteration, and so on until K_1 is used on the 16th and last iteration.

The Strength of DES

Since its adoption as a federal standard, there have been lingering concerns about the level of security provided by DES. These concerns, by and large, fall into two areas: the nature of the algorithm and key size.

For many years, the more important concern was the possibility of exploiting the characteristics of the DES algorithm to perform cryptanalysis. The focus of concern has been on the eight substitution tables, or S-boxes, that are used in each iteration. Because the design criteria for these boxes, and indeed for the entire algorithm, have never been made public, there is a suspicion that the boxes were constructed in such a way that cryptanalysis is possible for an opponent who knows the weaknesses in the S-boxes. This assertion is tantalizing, and over the years a number of regularities and unexpected behaviors of the S-boxes have been discovered. Despite this problem, no one has so far succeeded in discovering the supposed fatal weaknesses in the *S*-boxes. Indeed, as advances in cryptanalytic techniques have occurred, the underlying strength of the DES algorithm has become more apparent. As of this writing, no practical attack method for DES has been published. Given that the algorithm has survived years of intensive scrutiny unscathed, it is probably safe to say that DES is one of the strongest encryption algorithms ever devised.

The more serious concern, today, is the key size. With a key length of 56 bits, there are 2^{56} possible keys, which is approximately 7.6×10^{16} keys. Thus, on the face of it, a brute-force attack appears impractical. Assuming that on average half the key space has to be searched, a single machine performing one DES encryption per microsecond would take more than a thousand years to break the cipher.

However, the assumption of one encryption per microsecond is overly conservative. As far back as 1977, Diffie and Hellman, the inventors of public-key encryption, postulated that the technology existed to build a parallel machine with 1 million encryption devices, each of which could perform one encryption per microsecond. The authors estimated that the cost would be about $20 million in 1977 dollars.

The most rigorous recent analysis of the problem was done by Wiener [WIEN93] and is based on a known plaintext attack. That is, it is assumed that the attacker has at least one (plaintext, ciphertext) pair. Wiener takes care to provide the details of his design. To quote his paper,

> There have been numerous unverifiable claims about how fast the DES key space can be searched. To avoid adding to this list of questionable claims, a great deal of detail in the design of a key search machine is included in the appendices. This detailed work was done to obtain an accurate assessment of the cost of the machine and the time required to find a DES key. There are no plans to actually build such a machine.

Wiener reports on the design of a chip that uses pipelined techniques to achieve a key search rate of 50 million keys per second. Using 1993 costs, he designed a module that costs $100,000 and contains 5,760 key search chips. With this design, the following results are obtained:

Key Search Machine Unit Cost	Expected Search Time
$100,000	35 hours
$1,000,000	3.5 hours
$10,000,000	21 minutes

In addition, Wiener estimates a one-time development cost of about $500,000.

The Wiener design represents the culmination of years of concern about the security of DES and may in retrospect have been a turning point. As of the time of this writing, it still seems reasonable to rely on DES for personal and commercial applications. But the time has come to investigate alternatives for conventional encryption. One of the most promising and widely used candidates for replacing DES is triple DES.

Triple DES

Triple DES was first proposed by Tuchman [TUCH79], and first standardized for use in financial applications [ANSI85]. Triple DES uses two keys and three executions of the DES algorithm (Figure 18.6). The function follows an encrypt-decrypt-encrypt (EDE) sequence:

$$C = E_{K_1}[D_{K_2}[E_{K_1}[P]]]$$

There is no cryptographic significance to the use of decryption for the second stage. Its only advantage is that it allows users of triple DES to decrypt data encrypted by users of the older, single DES:

$$C = E_{K_1}[D_{K_1}[E_{K_1}[P]]] = E_{K_1}[P]$$

Although only two keys are used, three instances of the DES algorithm are required. It turns out that there is a simple technique, known as a meet-in-the-

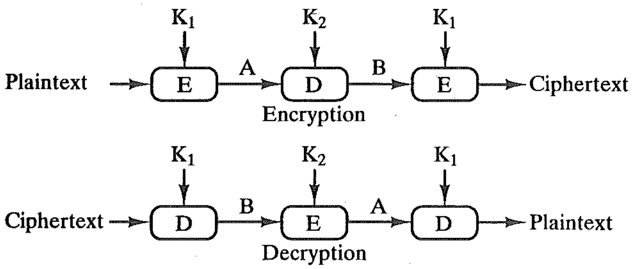


FIGURE 18.6 Triple DES.

middle attack, that would reduce a double DES system with two keys to the relative strength of ordinary single DES. With three iterations of the DES function, the effective key length is 112 bits.

Location of Encryption Devices

The most powerful, and most common, approach to countering the threats to network security is encryption. In using encryption we need to decide what to encrypt and where the encryption gear should be located. As Figure 18.7 indicates, there are two fundamental alternatives: link encryption and end-to-end encryption.

With link encryption, each vulnerable communications link is equipped on both ends with an encryption device. Thus, all traffic over all communications links is secured. Although this requires many encryption devices in a large network, the value of this approach is clear. However, one disadvantage of this approach is that the message must be decrypted each time it enters a packet switch; this is necessary because the switch must read the address (virtual circuit number) in the packet header in order to route the packet. Thus, the message is vulnerable at each switch. If this is a public packet-switching network, the user has no control over the security of the nodes.

With end-to-end encryption, the encryption process is carried out at the two end systems. The source host, or terminal, encrypts the data, which, in encrypted form, is then transmitted unaltered across the network to the destination terminal or host. The destination shares a key with the source and so is able to decrypt the

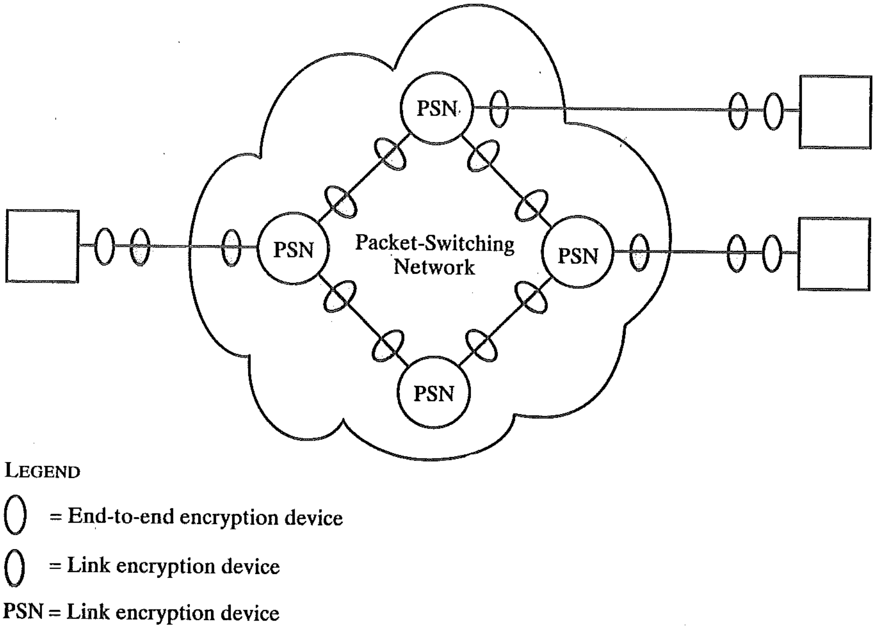


FIGURE 18.7 Encryption across a packet-switching network.

data. This approach would seem to secure the transmission against attacks on the network links or switches. There is, however, still a weak spot.

Consider the following situation. A host connects to an X.25 packet-switching network, sets up a virtual circuit to another host, and is prepared to transfer data to that other host using end-to-end encryption. Data are transmitted over such a network in the form of packets, consisting of a header and some user data. What part of each packet will the host encrypt? Suppose that the host encrypts the entire packet, including the header. This will not work because, remember, only the other host can perform the decryption. The packet-switching node will receive an encrypted packet and be unable to read the header. Therefore, it will not be able to route the packet! It follows that the host may only encrypt the user data portion of the packet and must leave the header in the clear, so that it can be read by the network.

Thus, with end-to-end encryption, the user data are secure; however, the traffic pattern is not, as packet headers are transmitted in the clear. To achieve greater security, both link and end-to-end encryption are needed, as is shown in Figure 18.7.

To summarize, when both forms are employed, the host encrypts the user data portion of a packet using an end-to-end encryption key. The entire packet is then encrypted using a link-encryption key. As the packet traverses the network, each switch decrypts the packet using a link-encryption key in order to read the header and then encrypts the entire packet again so as to send it out on the next link. Now the entire packet is secure, except for the time that the packet is actually in the memory of a packet switch, at which time the packet header is in the clear.

Key Distribution

For conventional encryption to work, the two parties in an exchange must have the same key, and that key must be protected from access by others. Furthermore, frequent key changes are usually desirable to limit the amount of data compromised if an attacker learns the key. Therefore, the strength of any cryptographic system rests with the key distribution technique, a term that refers to the means of delivering a key to two parties who wish to exchange data without allowing others to see the key. Key distribution can be achieved in a number of ways. For two parties *A* and *B*

1. A key could be selected by *A* and physically delivered to *B*.
2. A third party could select the key and physically deliver it to *A* and *B*.
3. If *A* and *B* have previously and recently used a key, one party could transmit the new key to the other, encrypted using the old key.
4. If *A* and *B* each have an encrypted connection to a third party *C*, *C* could deliver a key on the encrypted links to *A* and *B*.

Options 1 and 2 call for manual delivery of a key; for link encryption, this is a reasonable requirement, as each link encryption device is only going to be exchanging data with its partner on the other end of the link. However, for end-to-end encryption, manual delivery is awkward. In a distributed system, any given host or terminal may need to engage in exchanges with many other hosts and terminals over time. Thus, each device needs a number of keys, supplied dynamically. The problem is especially difficult in a wide-area distributed system.

Option 3 is a possibility for either link encryption or end-to-end encryption, but if an attacker ever succeeds in gaining access to one key, then all subsequent keys are revealed. Even if frequent changes are made to the link-encryption keys, these should be done manually. To provide keys for end-to-end encryption, option 4 is preferable.

Figure 18.8 illustrates an implementation that satisfies option 4 for end-to-end encryption. In the figure, link encryption is ignored. This can be added, or not, as required. For this scheme, two kinds of keys are identified:

- **Session key.** When two end systems (hosts, terminals, etc.) wish to communicate, they establish a logical connection (e.g., virtual circuit). For the duration of that logical connection, all user data are encrypted with a one-time session key. At the conclusion of the session, or connection, the session key is destroyed.
- **Permanent key.** A permanent key is one used between entities for the purpose of distributing session keys.

The configuration consists of the following elements:

- **Key distribution center.** The key distribution center determines which systems are allowed to communicate with each other. When permission is granted for two systems to establish a connection, the key distribution center provides a one-time session key for that connection.
- **Front-end processor.** The front-end processor performs end-to-end encryption and obtains session keys on behalf of its host or terminal.

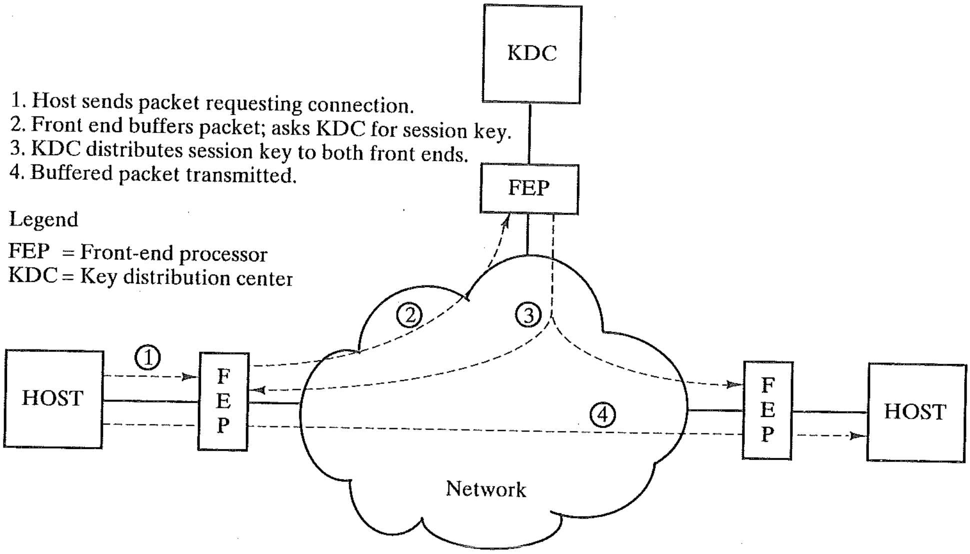

KDC
1. Host sends packet requesting connection.
2. Front end buffers packet; asks KDC for session key.
3. KDC distributes session key to both front ends.
4. Buffered packet transmitted.
FEP
Legend
FEP = Front-end processor
KDC = Key distribution center
②
③
①
HOST
F E P
④
F E P
HOST
Network

FIGURE 18.8 Automatic key distribution for connection-oriented protocol.

The steps involved in establishing a connection are shown in the figure. When one host wishes to set up a connection to another host, it transmits a connection-request packet (step 1). The front-end processor saves that packet and applies to the KDC for permission to establish the connection (step 2). The communication between the FEP and the KDC is encrypted using a master key shared only by the FEP and the KDC. If the KDC approves the connection request, it generates the session key and delivers it to the two appropriate front-end processors, using a unique permanent key for each front end (step 3). The requesting front-end processor can now release the connection-request packet, and a connection is set up between the two end systems (step 4). All user data exchanged between the two end systems are encrypted by their respective front-end processors using the one-time session key.

The automated key distribution approach provides the flexibility and dynamic characteristics needed both to allow a number of terminal users to access a number of hosts and for the hosts to exchange data with each other.

Of course, another approach to key distribution uses public-key encryption, which is discussed in Section 18.4.

Traffic Padding

We mentioned that, in some cases, users are concerned about security from traffic analysis. With the use of link encryption, packet headers are encrypted, reducing the opportunity for traffic analysis. However, it is still possible in those circumstances for an attacker to assess the amount of traffic on a network and to observe the amount of traffic entering and leaving each end system. An effective countermeasure to this attack is traffic padding, illustrated in Figure 18.9.

Traffic padding is a function that produces ciphertext output continuously, even in the absence of plaintext. A continuous random data stream is generated.

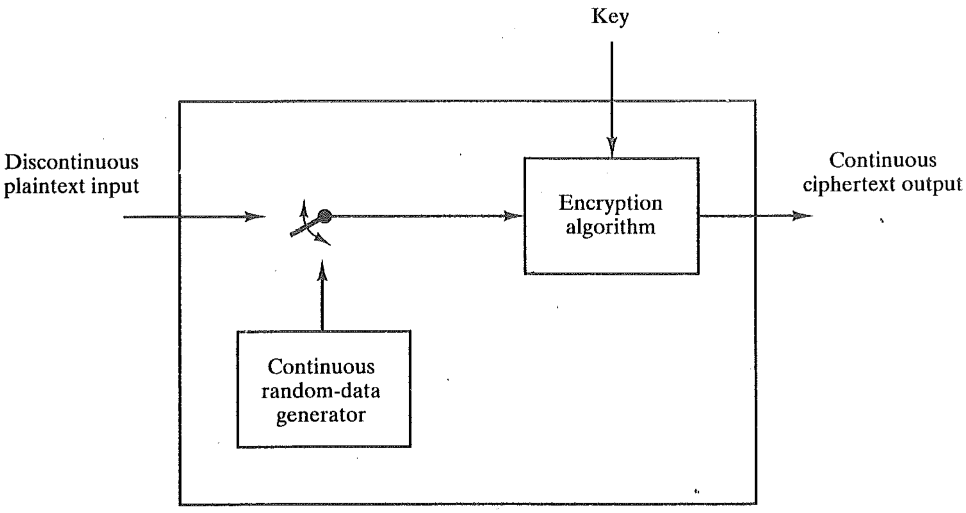

Key
Discontinuous plaintext input
Encryption algorithm
Continuous ciphertext output
Continuous random-data generator

FIGURE 18.9 Traffic-padding encryption device.

When plaintext is available, it is encrypted and transmitted. When input plaintext is not present, the random data are encrypted and transmitted. This makes it impossible for an attacker to distinguish between true data flow and noise, and it is therefore impossible for the intruder to deduce the amount of traffic.

18.3 MESSAGE AUTHENTICATION AND HASH FUNCTIONS

Encryption protects against passive attack (eavesdropping). A different requirement is to protect against active attack (falsification of data and transactions). Protection against such attacks is known as message authentication.

Approaches to Message Authentication

A message, file, document, or other collection of data is said to be authentic when it is genuine and actually coming from its alleged source. Message authentication is a procedure that allows communicating parties to verify that received messages are authentic. The two important aspects are to verify that the contents of the message have not been altered and that the source is authentic. We may also wish to verify a message's timeliness (it has not been artificially delayed and replayed) and sequence, relative to other messages flowing between two parties.

Authentication Using Conventional Encryption

It is possible to perform authentication simply by the use of conventional encryption. If we assume that only the sender and receiver share a key (which is as it should be), then only the genuine sender would be able to successfully encrypt a message for the other participant. Furthermore, if the message includes an error-detection code and a sequence number, the receiver is assured that no alterations have been made and that sequencing is proper. If the message also includes a timestamp, the receiver is assured that the message has not been delayed beyond that time normally expected for network transit.

Message Authentication Without Message Encryption

In this section, we examine several approaches to message authentication that do not rely on encryption but on a related family of functions. In all of these approaches, an authentication tag is generated and appended to each message for transmission. The message itself is not encrypted and can be read at the destination independent of the authentication function at the destination.

Because the approaches discussed in this section do not encrypt the message, message secrecy is not provided. Because conventional encryption will provide authentication, and because it is widely used with readily available products, why not simply use such an approach, which provides both secrecy and authentication? [DAVI90] suggests three situations in which message authentication without secrecy is preferable:

1. There are a number of applications in which the same message is broadcast to a number of destinations—for example, notification to users that the network is now unavailable or an alarm signal in a control center. It is cheaper and more reliable to have only one destination responsible for monitoring authenticity. Thus, the message must be broadcast in plaintext with an associated message authentication tag. The responsible system performs authentication. If a violation occurs, the other destination systems are alerted by a general alarm.
2. Another possible scenario is an exchange in which one side has a heavy load and cannot afford the time to decrypt all incoming messages. Authentication is carried out on a selective basis, with the messages being chosen at random for checking.
3. Authentication of a computer program in plaintext is an attractive service. The computer program can be executed without having to decrypt it every time, which would be wasteful of processor resources. However, if a message authentication tag were attached to the program, it could be checked whenever assurance is required of the integrity of the program.

Thus, there is a place for both authentication and encryption in meeting security requirements.

Message Authentication Code

One authentication technique involves the use of a secret key to generate a small block of data, known as a message authentication code, that is appended to the message. This technique assumes that two communicating parties, say A and B, share a common secret key K_{AB}. When A has a message to send to B, it calculates the message authentication code as a function of the message and the key: $MAC_M = F(K_{AB}, M)$. The message plus code are transmitted to the intended recipient. The recipient performs the same calculation on the received message, using the same secret key, to generate a new message authentication code. The received code is compared to the calculated code (Figure 18.10). If we assume that only the receiver and the sender know the identity of the secret key, and if the received code matches the calculated code, then,

1. The receiver is assured that the message has not been altered. If an attacker alters the message but does not alter the code, then the receiver's calculation of the code will differ from the received code. Because the attacker is assumed not to know the secret key, the attacker cannot alter the code to correspond to the alterations in the message.
2. The receiver is assured that the message is from the alleged sender. Because no one else knows the secret key, no one else could prepare a message with a proper code.
3. If the message includes a sequence number (such as is used with X.25, HDLC, TCP, and the ISO transport protocol), then the receiver can be assured of the proper sequence, as an attacker cannot successfully alter the sequence number.

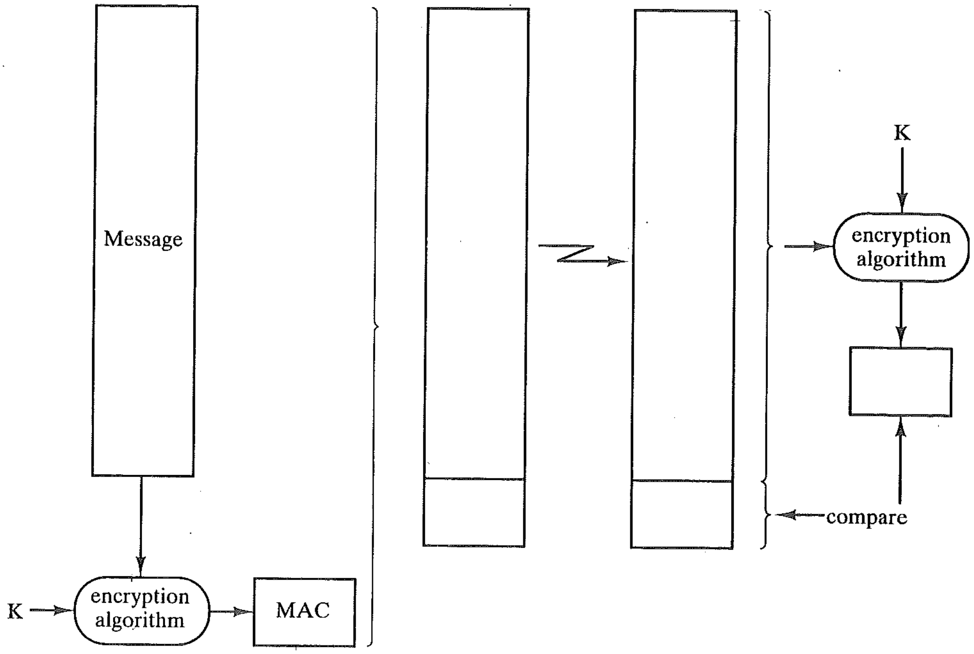


FIGURE 18.10 Message authentication using a message authentication code (MAC).

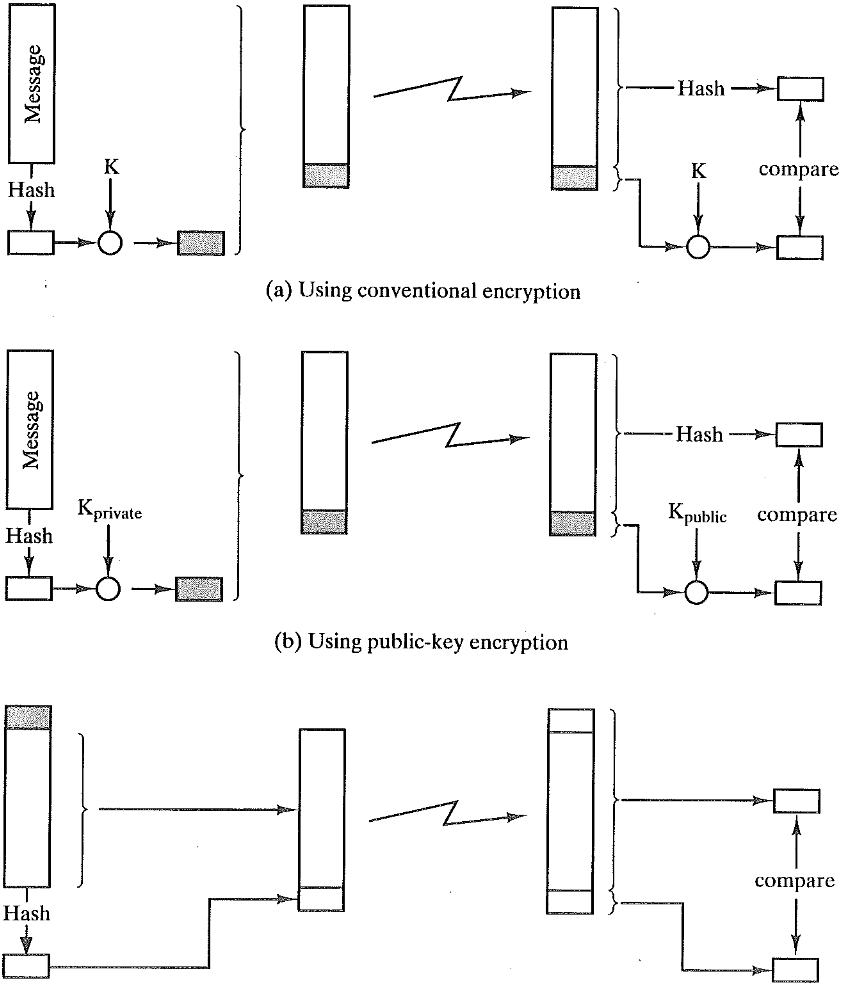
A number of algorithms could be used to generate the code. The National Bureau of Standards, in its publication *DES Modes of Operation*, recommends the use of the DES algorithm. The DES algorithm is used to generate an encrypted version of the message, and the last number of bits of ciphertext are used as the code. A 16- or 32-bit code is typical.

The process just described is similar to encryption. One difference is that the authentication algorithm need not be reversible, as it must for decryption. It turns out that because of the mathematical properties of the authentication function, it is less vulnerable to being broken than is encryption.

One-Way Hash Function.

A variation on the message authentication code that has received much attention recently is the one-way hash function. As with the message authentication code, a hash function accepts a variable-size message M as input and produces a fixed-size tag H(M), sometimes called a message digest, as output. To authenticate a message, the message digest is sent with the message in such a way that the message digest is authentic.

Figure 18.11 illustrates three ways in which the message digest can be authenticated. The message digest can be encrypted using conventional encryption (part (a)); if it is assumed that only the sender and receiver share the encryption key, then authenticity is assured. The message can also be encrypted using public-key encryption (part (b)). The public-key approach has two advantages: It provides a digital



(a) Using conventional encryption

(b) Using public-key encryption

(c) Using secret value

FIGURE 18.11 Message authentication using a one-way hash function.

signature as well as message authentication; and it does not require the distribution of keys to communicating parties.

These two approaches have an advantage over approaches that encrypt the entire message, in that less computation is required. Nevertheless, there has been an interest in developing a technique that avoids encryption altogether. Several reasons for this interest are pointed out in [TSUD92]:

- Encryption software is quite slow. Even though the amount of data to be encrypted per message is small, there may be a steady stream of messages into and out of a system.

- Encryption hardware costs are non-negligible. Low-cost chip implementations of DES are available, but the cost adds up if all nodes in a network must have this capability.
- Encryption hardware is optimized toward large data sizes. For small blocks of data, a high proportion of the time is spent in initialization/invocation overhead.
- Encryption algorithms may be covered by patents. Some encryption algorithms, such as the RSA public-key algorithm, are patented and must be licensed, adding a cost.
- Encryption algorithms may be subject to export control. This is true of DES.

Figure 18.11c shows a technique that uses a hash function but no encryption for message authentication. This technique assumes that two communicating parties, say A and B, share a common secret value S_{AB}. When A has a message to send to B, it calculates the hash function over the concatenation of the secret value and the message: $MD_M = H(S_{AB}||M)$.[1] It then sends $[M||MD_M]$ to B. Because B possesses S_{AB}, it can recompute $H(S_{AB}||M)$ and verify MD_M, and, because the secret value itself is not sent, it is not possible for an attacker to modify an intercepted message. As long as the secret value remains secret, it is also not possible for an attacker to generate a false message.

This third technique, using a shared secret value, is the one adopted for IP security (described in Chapter 19); it has also been tentatively specified for SNMPv2.[2]

Secure Hash Functions

The one-way hash function, or secure hash function, is important not only in message authentication but in digital signatures. In this section, we begin with a discussion of requirements for a secure hash function. Then we look at two very simple hash functions that are not secure, to gain an appreciation of the structure of such functions. Finally, we examine one of the most important hash functions, MD5.

Hash Function Requirements

The purpose of a hash function is to produce a "fingerprint" of a file, message, or other block of data. To be useful for message authentication, a hash function H must have the following properties, adapted from a list in [NECH91]:

1. H can be applied to a block of data of any size.
2. H produces a fixed-length output.
3. H(x) is relatively easy to compute for any given x, making both hardware and software implementations practical.

[1] || denotes concatenation.

[2] When SNMPv2 was first issued in 1993 as a set of proposed Internet standards, it included a security function that used the technique just discussed. SNMPv2 was reissued as draft Internet standards in 1996 without the security function because of lack of agreement on all the details. However, the final security standard for SNMP is likely to include the technique just discussed.

4. For any given code m, it is computationally infeasible to find x such that $H(x) = m$.
5. For any given block x, it is computationally infeasible to find $y \neq x$ with $H(y) = H(x)$.
6. It is computationally infeasible to find any pair (x, y) such that $H(x) = H(y)$.

The first three properties are requirements for the practical application of a hash function to message authentication.

The fourth property is the *one-way* property: It is easy to generate a code given a message, but virtually impossible to generate a message given a code. This property is important if the authentication technique involves the use of a secret value (Figure 18.11c), which is not itself sent; however, if the hash function is not one-way, an attacker can easily discover the secret value: If the attacker can observe or intercept a transmission, he or she obtains the message M and the hash code $MD_M = H(S_{AB}||M)$. The attacker then inverts the hash function to obtain $S_{AB}||M = H^{-1}(MD_M)$. Because the attacker now has both M and $S_{AB}||M$, it is a trivial matter to recover S_{AB}.

The fifth property guarantees that an alternative message hashing to the same value as a given message cannot be found; this prevents forgery when an encrypted hash code is used (Figure 18.11b and c). If this property were not true, an attacker would be capable of the following sequence: First, observe or intercept a message plus its encrypted hash code; second, generate an unencrypted hash code from the message; third, generate an alternate message with the same hash code.

A hash function that satisfies the first five properties in the preceding list is referred to as a weak hash function. If the sixth property is also satisfied, then it is referred to as a strong hash function. The sixth property protects against a sophisticated class of attack known as the birthday attack.[3]

In addition to providing authentication, a message digest also provides data integrity. It performs the same function as a frame check sequence: If any bits in the message are accidentally altered in transit, the message digest will be in error.

Simple Hash Functions

All hash functions operate using the following general principles. The input (message, file, etc.) is viewed as a sequence of n-bit blocks. The input is processed one block at a time in an iterative fashion to produce an n-bit hash function.

One of the simplest hash functions is to take the bit-by-bit–exclusive-or (XOR) of every block; this can be expressed as follows:

$$C_i = b_{i1} \oplus b_{i2} \oplus \ldots \oplus b_{im}$$

where

C_i = ith bit of the hash code, $1 \leq i \leq n$

m = number of n-bit blocks in the input

b_{ij} = ith bit in jth block

$\oplus$ = XOR operation

[3] See [STAL95b] for a discussion of birthday attacks.

	bit 1	bit 2	• • •	bit n
block 1	b_{11}	b_{21}		b_{n1}
block 2	b_{12}	b_{22}		b_{n2}
	• • •	• • •	• • •	• • •
block m	b_{1m}	b_{2m}		b_{nm}
hash code	C_1	C_2		C_n

FIGURE 18.12 Simple hash function using bitwise XOR.

```
main (int argc, char*argv[]) {
        unsigned long hash [4] = (0, 0, 0, 0), data [4] ;
        FILE *fp;
        int i;
        if ((fp = = fopen (argv [1], "rb")) != NULL) {
             while ((fread (data, 4,,4, fp) != NULL)
                  for (i=0; i<4; i++)
                       hash [i] ^=data [i];
             fclose (fp);
             for (i=0; i<4; i++)
                  printf ("%081x", hash [i]);
             printf ("\n") ;
        }    }
```

(a) XOR of every 128-bit block

```
main (int argc, char *argv []) {
        unsigned long hash [4] = (0, 0, 0, 0), data [4] ;
        FILE *fp;
        int i;
        if ((fp = = fopen (argv [1], "rb")) !=NULL) {
             while ((fread (data, 4,,4, fp) !=NULL)
                  for (i=0; i<4; i++)    {
                       hash [i] ^=data [i];
                       hash [i] = hash [i] >>1 ^ hash [i] <<31 ;
                       }
             fclose (fp);
             for (i=0; i<4; i++)
                  printf ("%081x", hash [i]) ;
             printf ("\n") ;
        }    }
```

(b) Rolling each longword 1 bit to the right

FIGURE 18.13 Simple hash functions (based on [SCHN91]).

Figure 18.12 illustrates this operation. Figure 18.13a is a C program that produces a 128-bit hash code. This type of code produces a simple parity for each bit position and is known as a longitudinal redundancy check. It is reasonably effective for random data as a data integrity check. Each 128-bit hash value is equally likely. Thus, the probability that a data error will result in an unchanged hash value is 2^{-128} With more predictably formatted data, the function is less effective. For example, in most normal text files, the high-order bit of each octet is always zero. Therefore, 16 bits in the hash will always be zero, and the effectiveness is reduced to 2^{-112}.

A simple way to improve matters is to perform a one-bit circular shift, or rotation on the hash value after each block is processed, as defined in Figure 18.13b. This has the effect of "randomizing" the input more completely and overcoming any regularities that appear in the input.

Although the second program provides a good measure of data integrity, it is virtually useless for data security. Consider the following task of a potential attacker: Given a hash code, produce a message that yields that hash code. The attacker would simply need to prepare the desired message and then append a 128-bit block that forces the new message, plus block, to yield the desired hash code.

Thus, we need a hash algorithm that is a much more complex function of the input bits.

MD5 Algorithm Description

The MD5 message-digest algorithm, specified in RFC 1321, was developed by Ron Rivest at MIT (the "R" in the RSA [Rivest-Shamir-Adelman] public-key encryption algorithm). The algorithm takes as input a message of arbitrary length and produces as output a 128-bit message digest. The input is processed in 512-bit blocks.

Figure 18.14 depicts the overall processing of a message to produce a digest. The processing consists of the following steps:

- **Step 1: Append Padding Bits**

The message is padded so that its length in bits is congruent to 448 modulo 512 (length = 448 mod 512). That is, the length of the padded message is 64 bits less than an integer multiple of 512 bits. Padding is always added, even if the message is already of the desired length. For example, if the message is 448 bits long, it is padded by 512 bits to a length of 960 bits. Thus, the number of padding bits is in the range of 1 to 512.

The padding consists of a single 1-bit followed by the necessary number of 0-bits.

- **Step 2: Append Length**

A 64-bit representation of the length in bits of the original message (before the padding) is appended to the result of step 1. If the original length is greater than 2^{64}, then only the low-order 64 bits of the length are used. Thus, the field contains the length of the original message, modulo 2^{64}. The inclusion of a length value at the end of the message makes more difficult a type of attack known as a padding attack [TSUD92].

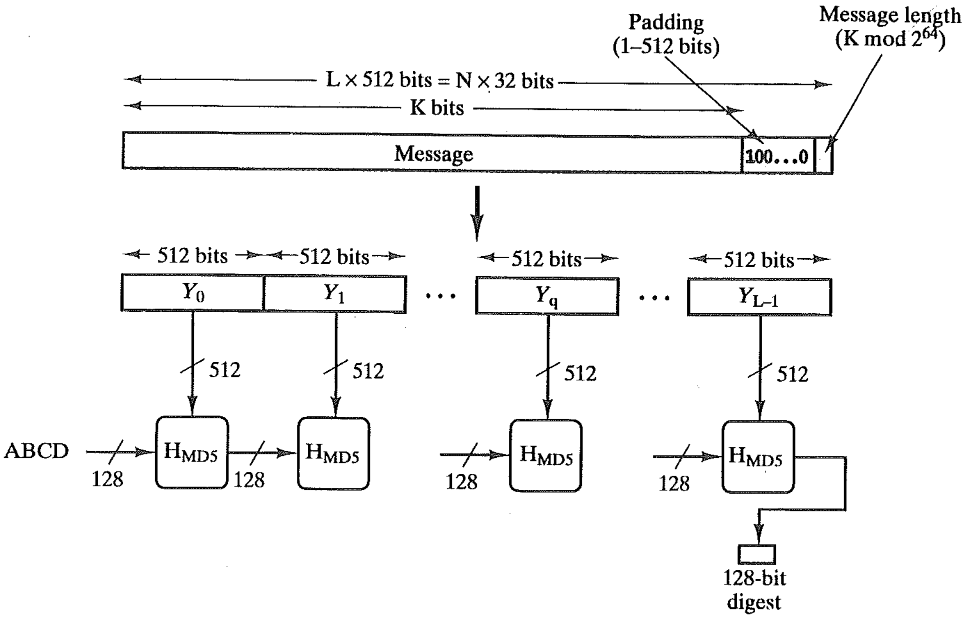


FIGURE 18.14 Message digest generation using MD5.

The outcome of the first two steps yields a message that is an integer multiple of 512 bits in length. In the figure, the expanded message is represented as the sequence of 512-bit blocks Y_0, Y_1, . . ., Y_{L-1}, so that the total length of the expanded message is $L \times 512$ bits. Equivalently, the result is a multiple of sixteen 32-bit words. Let M[0 . . . $N - 1$] denote the words of the resulting message, with N an integer multiple of 16. Thus, $N = L \times 16$.

- **Step 3: Initialize MD Buffer**

A 128-bit buffer is used to hold intermediate and final results of the hash function. The buffer can be represented as four 32-bit registers (A, B, C, D), which are initialized to the following hexadecimal values (low-order octets first):

```
A = 01234567
B = 89ABCDEF
C = FEDCBA98
D = 76543210
```

- **Step 4: Process Message in 512-Bit (16-Word) Blocks**

The heart of the algorithm is a module that consists of four "rounds" of processing; this module is labeled H_{MD5} in Figure 18.14, and its logic is illustrated in Figure 18.15. The four rounds have a similar structure, but each uses a different primitive logical function, referred to as F, G, H, and I in the specification. In the figure,

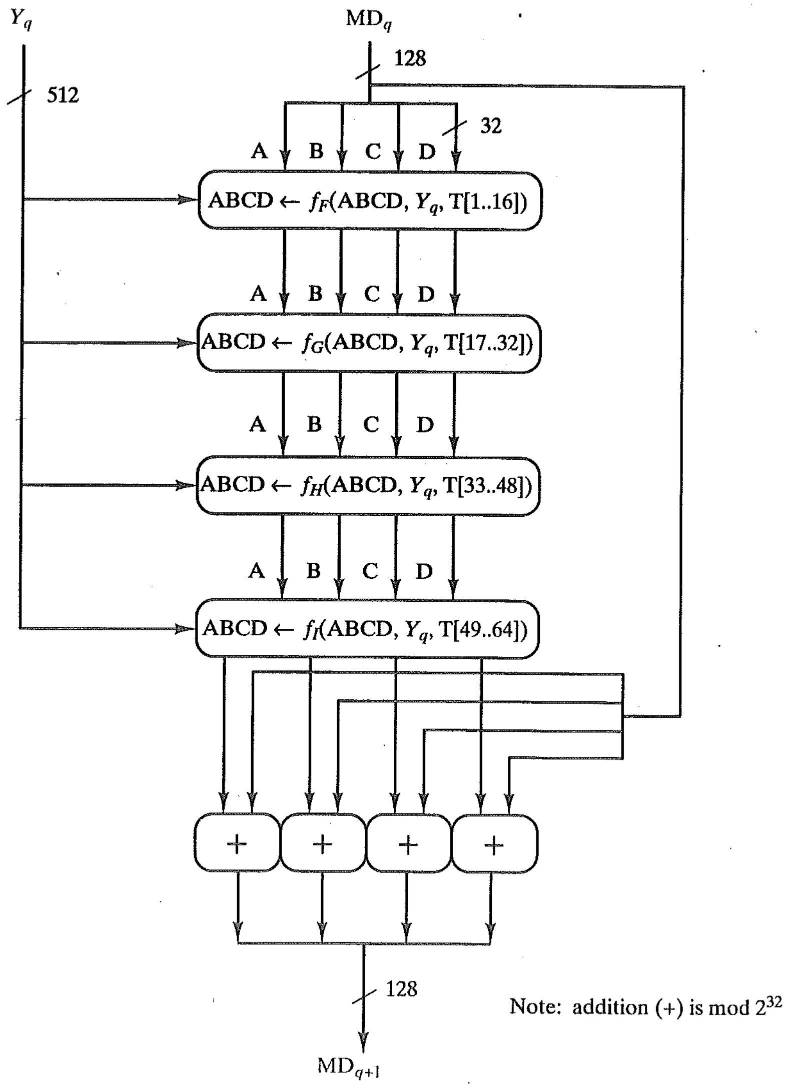

Y_q
MD_q
128
512
32
A
B
C
D
ABCD ← f_F(ABCD, Y_q, T[1..16])
ABCD ← f_G(ABCD, Y_q, T[17..32])
ABCD ← f_H(ABCD, Y_q, T[33..48])
ABCD ← f_I(ABCD, Y_q, T[49..64])
+
128
MD_{q+1}
Note: addition (+) is mod 2^{32}

FIGURE 18.15 MD5 processing of a single 512-bit block (H_{MD5}).

the four rounds are labeled f_F, f_G, f_H, f_I, to indicate that each round has the same general functional structure, f, but depends on a different primitive function (F, G, H, I).

Each primitive function takes three 32-bit words as input and produces a 32-bit word output. Each function performs a set of bitwise logical operations; that is, the *n*th bit of the output is a function of the *n*th bit of the three inputs. The functions are

$$F(X,Y,Z) = (X \cdot Y) + (X' \cdot Z)$$

$$G(X,Y,Z) = (X \cdot Z) + (Y \cdot Z')$$

$$H(X,Y,Z) = X \oplus Y \oplus Z$$

$$I(X,Y,Z) = Y \oplus (X + Z')$$

where the logical operators (AND, OR, NOT, XOR) are represented by the symbols ($\cdot$, $+$, $'$, $\oplus$). Function F is a conditional function: If X then Y else Z. Function H produces a parity bit. Table 18.1 is a truth table of the four functions.

Note that each round takes as input the current 512-bit block being processed (Y*q*) and the 128-bit buffer value ABCD and updates the contents of the buffer. Each round also makes use of one-fourth of a 64-element table T[1 . . . 64], constructed from the sine function. The *i*th element of T, denoted T[*i*], has a value equal to the integer part of $2^{32} \times$ abs[(sin(*i*)], where *i* is in radians. Because abs[(sin(*i*)] is a number between 0 and 1, each element of T is an integer that can be represented in 32 bits. The table provides a "randomized" set of 32-bit patterns, which should eliminate any regularities in the input data.

Overall, for block Y_q, the algorithm takes Y_q and an intermediate digest value MD_q as inputs. MD_q is placed into buffer ABCD. The output of the fourth round is added to MD_q to produce MD_{q+1}. The addition is done independently for each of the four words in the buffer with each of the corresponding words in MD_q, using addition modulo 2^{32}.

TABLE 18.1 Truth table of logical functions for MD5.

X	Y	Z	F	G	H	I
0	0	0	0	0	0	1
0	0	1	1	0	1	0
0	1	0	0	1	1	0
0	1	1	1	0	0	1
1	0	0	0	0	1	1
1	0	1	0	1	0	1
1	1	0	1	1	0	0
1	1	1	1	1	1	0

- **Step 5: Output**

After all *L* 512-bit blocks have been processed, the output from the *L*th stage is the 128-bit message digest.

The MD5 algorithm has the property that every bit of the hash code is a function of every bit in the input. The complex repetition of the basic functions (F, G, H, I) produces results that are well mixed; that is, it is unlikely that two messages chosen at random, even if they exhibit similar regularities, will have the same hash code. Rivest conjectures in the RFC that MD5 is as strong as possible for a 128-bit hash code; namely, the difficulty of coming up with two messages having the same message digest is on the order of 2^{64} operations, whereas the difficulty of finding a message with a given digest is on the order of 2^{128} operations. As of this writing, no analysis has been done to disprove these conjectures.

18.4 PUBLIC-KEY ENCRYPTION AND DIGITAL SIGNATURES

Of equal importance to conventional encryption is public-key encryption, which finds use in message authentication and key distribution. This section looks first at the basic concept of public-key encryption, followed by a discussion of the most widely used public-key algorithm: RSA. We then look at the problem of key distribution.

Public-Key Encryption

Public-key encryption, first publicly proposed by Diffie and Hellman in 1976 [DIFF76], is the first truly revolutionary advance in encryption in literally thousands of years. For one thing, public-key algorithms are based on mathematical functions rather than on substitution and permutation. But more importantly, public-key cryptography is asymmetric, involving the use of two separate keys, in contrast to the conventional symmetric encryption, which uses only one key. The use of two keys has profound consequences in the areas of confidentiality, key distribution, and authentication.

Before proceeding, we should mention several common misconceptions concerning public-key encryption. One such misconception is that public-key encryption is more secure from cryptanalysis than conventional encryption. In fact, the security of any encryption scheme depends on the length of the key and the computational work involved in breaking a cipher. There is nothing in principle about either conventional or public-key encryption that makes one superior to another from the point of view of resisting cryptanalysis. A second misconception is that public-key encryption is a general-purpose technique that has made conventional encryption obsolete. On the contrary, because of the computational overhead of current public-key encryption schemes, there seems no foreseeable likelihood that conventional encryption will be abandoned. Finally, there is a feeling that key distribution is trivial when using public-key encryption, compared to the rather cumbersome handshaking involved with key distribution centers for conventional encryption. In fact, some form of protocol is needed, often involving a central agent, and the procedures involved are no simpler nor any more efficient than those required for conventional encryption.

A public-key cryptographic algorithm relies on one key for encryption and a different but related key for decryption. Furthermore, these algorithms have the following important characteristic:

- It is computationally infeasible to determine the decryption key given only knowledge of the cryptographic algorithm and the encryption key.

In addition, some algorithms, such as RSA, also exhibit the following characteristic:

- Either of the two related keys can be used for encryption, with the other used for decryption.

The essential steps are the following:

1. Each end system in a network generates a pair of keys to be used for encryption and decryption of messages that it will receive.
2. Each system publishes its encryption key by placing it in a public register or file. This is the public key. The companion key is kept private.
3. If A wishes to send a message to B, it encrypts the message using B's public key.
4. When *B* receives the message, it decrypts it using B's private key. No other recipient can decrypt the message because only B knows B's private key.

With this approach, all participants have access to public keys, and private keys are generated locally by each participant and therefore need never be distributed. As long as a system controls its private key, its incoming communication is secure. At any time, a system can change its private key and publish the companion public key to replace its old public key.

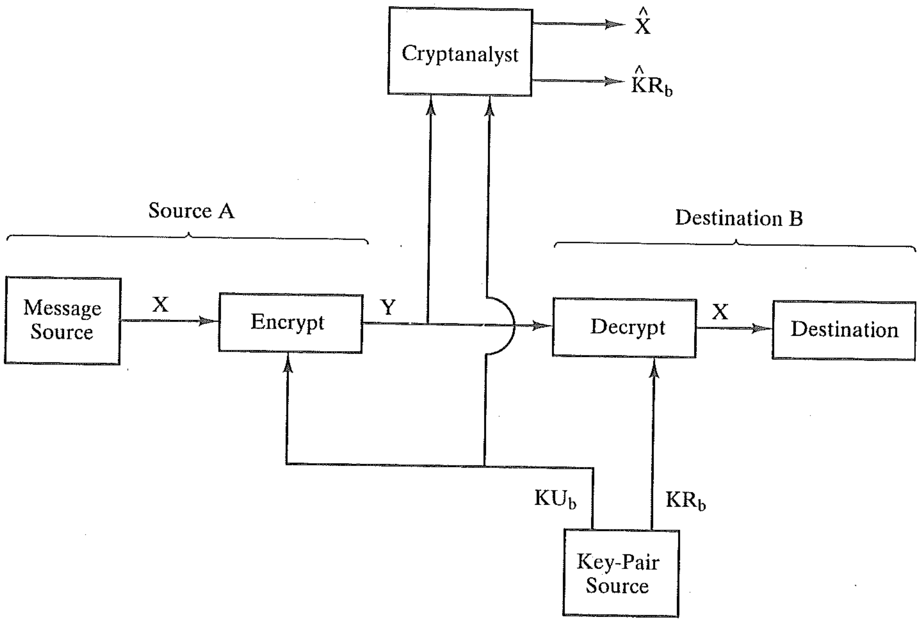


FIGURE 18.16 Public-key cryptosystem: secrecy.

Figure 18.16 illustrates the process (compare Figure 18.3). There is some source A for a message, which produces a message in plaintext, $X = [X_1, X_2, \ldots, X_M]$. The M elements of X are letters in some finite alphabet. The message is intended for destination B, which generates a related pair of keys: a public key KU_b, and a private key, KR_b. KR_b is secret, known only to B, whereas KU_b is publicly available, and therefore accessible by A.

With the message X and the encryption key KU_b as input, A forms the ciphertext $Y = [Y_1, Y_2, \ldots, Y_N]$:

$$Y = E_{KUb}(X)$$

The intended receiver, in possession of the matching private key, is able to invert the transformation:

$$X = D_{KRb}(Y)$$

An opponent, observing Y and having access to KU_b, but not having access to KR_b or X, must attempt to recover X and/or KR_b. It is assumed that the opponent does have knowledge of the encryption (E) and decryption (D) algorithms.

We mentioned earlier that either of the two related keys can be used for encryption, with the other being used for decryption. This enables a rather different cryptographic scheme to be implemented. Whereas the scheme illustrated in Figure 18.16 provides privacy, Figure 18.17 shows the use of public-key encryption to provide authentication:

$$Y = E_{KRa}(X)$$

$$X = D_{KUa}(Y)$$

In this case, A prepares a message to B and encrypts it using A's private key before transmitting it. B can decrypt the message using A's public key. Because the message was encrypted using A's private key, only A could have prepared the message. Therefore, the entire encrypted message serves as a **digital signature**. In addition, it is impossible to alter the message without access to A's private key, so the message is authenticated both in terms of source and in terms of data integrity.

In the preceding scheme, the entire message is encrypted, and, although it validates both author and contents, the encryption requires a great deal of storage; each document must be kept in plaintext to be used for practical purposes, and a copy also must be stored in ciphertext so that the origin and contents can be verified in case of a dispute. A more efficient way of achieving the same results is to encrypt a small block of bits that is a function of the document. Such a block, called an authenticator, must have the property that it is infeasible to change the document without changing the authenticator. If the authenticator is encrypted with the sender's private key, it serves as a signature that verifies origin, content, and sequencing. A secure hash code can serve this function.

It is important to emphasize that the encryption process just described does not provide confidentiality. That is, the message being sent is safe from alteration but not safe from eavesdropping. This is obvious in the case of a signature based on a portion of the message, as the rest of the message is transmitted in the clear. Even

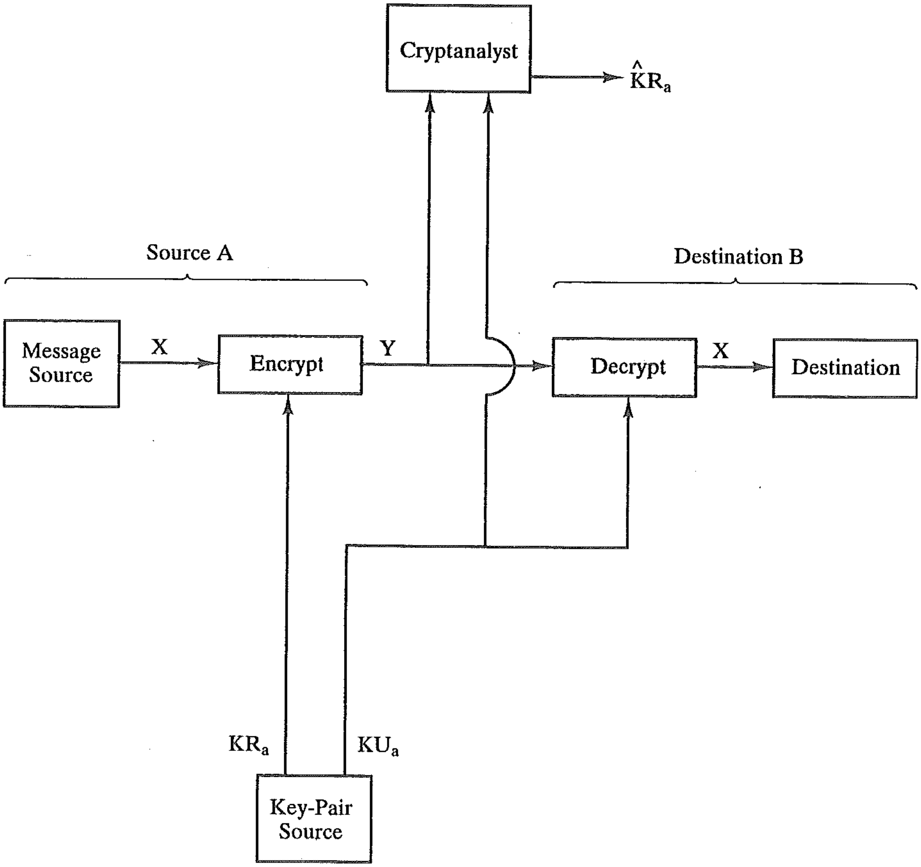


FIGURE 18.17 Public-key cryptosystem: authentication.

in the case of complete encryption, there is no protection of confidentiality because any observer can decrypt the message by using the sender's public key.

The RSA Public-Key Encryption Algorithm

One of the first public-key schemes was developed in 1977 by Ron Rivest, Adi Shamir, and Len Adleman at MIT, and first published in 1978 [RIVE78]. The RSA scheme has since reigned supreme as the only widely accepted and implemented approach to public-key encryption. RSA is a block cipher in which the plaintext and ciphertext are integers between 0 and n-1 for some n.

Encryption and decryption are of the following form, for some plaintext block M and ciphertext block C:

$$C = M^e \bmod n$$
$$M = C^d \bmod n = (M^e)^d \bmod n = M^{ed} \bmod n$$

Both sender and receiver must know the value of n. The sender knows the value of e, and only the receiver knows the value of d. Thus, this is a public-key encryption algorithm with a public key of KU = $\{e, n\}$ and a private key of KR = $\{d, n\}$. For this algorithm to be satisfactory for public-key encryption, the following requirements must be met:

1. Is it possible to find values of e, d, and n such that $M^{ed} = M \bmod n$ for all $M < n$?
2. Is it relatively easy to calculate M^e and C^d for all values of $M < n$?
3. Is it infeasible to determine d given e and n?

The answer to the first two questions is yes. The answer to the third question is yes, for large values of e and n.

Figure 18.18 summarizes the RSA algorithm. Begin by selecting two prime numbers, p and q, and calculating their product n, which is the modulus for encryption and decryption. Next, we need the quantity $\phi(n)$, which is referred to as the Euler totient of n, which is the number of positive integers less than n and relatively prime to n. Then select an integer d that is relatively prime to $\phi(n)$, (i.e., the greatest common divisor of d and $\phi(n)$ is 1). Finally, calculate e as the multiplicative inverse of d, modulo $\phi(n)$. It can be shown that d and e have the desired properties.

The private key consists of $\{d, n\}$, and the public key consists of $\{e, n\}$. Suppose that user A has published its public key and that user B wishes to send the message M to A. Then, B calculates $C = M^e \pmod{n}$ and transmits C. On receipt of this ciphertext, user A decrypts by calculating $M = C^d \pmod{n}$.

Key Generation

Select p, q	p and q both prime
Calculate $n = p \times q$	
Select integer d	$gcd(\phi(n), d) = 1; 1 < d < \phi(n)$
Calculate e	$e = d^{-1} \bmod \phi(n)$
Public key	KU = $\{e, n\}$
Private key	KR = $\{d, n\}$

Encryption

Plaintext: $M < n$
Ciphertext: $C = M^e \pmod{n}$

Decryption

Ciphertext: C
Plaintext: $M = C^d \pmod{n}$

FIGURE 18.18 The RSA algorithm.

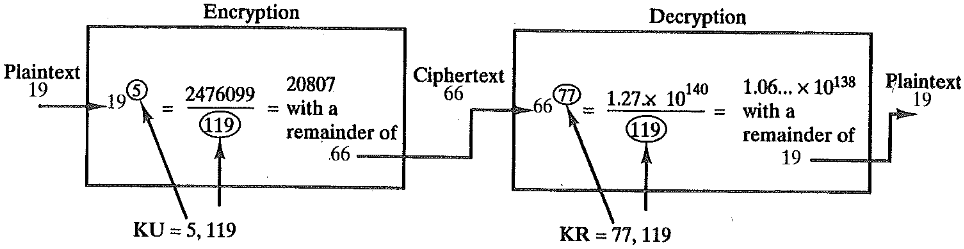


FIGURE 18.19 Example of RSA algorithm.

An example is shown in Figure 18.19. For this example, the keys were generated as follows:

1. Select two prime numbers, $p = 7$ and $q = 17$.
2. Calculate $n = pq = 7 \times 17 = 119$.
3. Calculate $\phi(n) = (p-1)(q-1) = 96$.
4. Select e such that e is relatively prime to $\phi(n) = 96$ and less than $\phi(n)$; in this case, $e = 5$.
5. Determine d such that $de = 1 \bmod 96$ and $d < 96$. The correct value is $d = 77$, because $77 \times 5 = 385 = 4 \times 96 + 1$.

The resulting keys are public key KU = {5, 119} and private key KR = {77, 119}. The example shows the use of these keys for a plaintext input of M = 19. For encryption, 19 is raised to the 5th power, yielding 2476099. Upon division by 119, the remainder is determined to be 66. Hence, $19^5 \equiv 66 \bmod 119$, and the ciphertext is 66. For decryption, it is determined that $66^{77} \equiv 19 \bmod 119$.

There are two possible approaches to defeating the RSA algorithm. The first is the brute force approach: Try all possible private keys. Thus, the larger the number of bits in e and d, the more secure the algorithm. However, because the calculations involved, both in key generation and in encryption/decryption, are complex, the larger the size of the key, the slower the system will run.

Most discussions of the cryptanalysis of RSA have focused on the task of factoring p into its two prime factors. Until recently, this was felt to be infeasible for numbers in the range of 100 decimal digits or so, which is about 300 or more bits. To demonstrate the strength of RSA, its three developers issued a challenge to decrypt a message that was encrypted using a 129-decimal-digit number as their public modulus. The authors predicted that it would take 40 quadrillion years with current technology to crack the code. Recently the code was cracked by a worldwide team cooperating over the Internet and using over 1600 computers after only eight months of work [LEUT94]. This result does not invalidate the use of RSA; it simply means that larger key sizes must be used. Currently, a 1024-bit key size (about 300 decimal digits) is considered strong enough for virtually all applications.

Key Management

With conventional encryption, a fundamental requirement for two parties to communicate securely is that they share a secret key. Suppose Bob wants to create a messaging application that will enable him to exchange e-mail securely with anyone who has access either to the Internet or to some other network that the two of them share (e.g., an on-line service such as Compuserve). Suppose Bob wants to do this using only conventional encryption. With conventional encryption, Bob and his correspondent, say, Alice, must come up with a way to share a unique secret key that no one else knows. How are they going to do that? If Alice is in the next room from Bob, Bob could generate a key and write it down on a piece of paper or store it on a diskette and hand it to Alice. But if Alice is on the other side of the continent, or maybe the world, what can Bob do? Well, he could encrypt this key using conventional encryption and e-mail it to Alice, but this means that Bob and Alice must share a secret key in order to encrypt this new secret key. Furthermore, Bob and everyone else who use this new e-mail package face the same problem with every potential correspondent: Each pair of correspondents must share a unique secret key.

How to distribute secret keys securely is the most difficult problem for conventional encryption. This problem is wiped away with public-key encryption by the simple fact that the private key is never distributed. If Bob wants to correspond with Alice and other people, he generates a single pair of keys, one private and one public. He keeps the private key secure and broadcasts the public key to all and sundry; if Alice does the same, then Bob has Alice's public key, Alice has Bob's public key, and they can now communicate securely. When Bob wishes to communicate with Alice, Bob can do the following:

1. Prepare a message.
2. Encrypt that message using conventional encryption with a one-time conventional session key.
3. Encrypt the session key using public-key encryption with Alice's public key.
4. Attach the encrypted session key to the message and send it to Alice.

Only Alice is capable of decrypting the session key and therefore recovering the original message.

It is only fair to point out, however, that we have replaced one problem.with another. Bob's private key is secure as he need never reveal it; however, Alice must be sure that the public key with Bob's name written all over it is in fact Bob's public key. Someone else could have broadcast a public key and said it was Bob's. The common way to overcome this problem is ingenious: Use public-key encryption to authenticate the public key. This encryption assumes the existence of some trusted signing authority or individual, and works as follows:

1. A public key is generated by Bob and submitted to Agency X for certification.
2. X determines by some procedure, such as a face-to-face meeting, that this is, authentically, Bob's public key.

3. X appends a timestamp to the public key, generates the hash code of the result, and encrypts that result with its private key forming the signature.
4. The signature is attached to the public key.

Anyone equipped with a copy of X's public key can now verify that Bob's public key is authentic.

18.5 IPv4 AND IPv6 SECURITY

In August 1995, the IETF published five security-related Proposed Standards that define a security capability at the internet level. The documents are

- RFC 1825: An overview of a security architecture
- RFC 1826: Description of a packet authentication extension to IP
- RFC 1828: A specific authentication mechanism
- RFC 1827: Description of a packet encryption extension to IP
- RFC 1829: A specific encryption mechanism

Support for these features is mandatory for IPv6 and optional for IPv4. In both cases, the security features are implemented as extension headers that follow the main IP header. The extension header for authentication is known as the Authentication header; the one designated for privacy is known as the Encapsulating Security Payload (ESP) header.

Security Associations

A key concept that appears in both the authentication and privacy mechanisms for IP is the security association. An association is a one-way relationship between a sender and a receiver. If a peer relationship is needed, for two-way secure exchange, then two security associations are required.

A security association is uniquely identified by an internet address and a security parameter index (SPI). Hence, in any IP packet,[4] the security association is uniquely identified by the Destination Address in the IPv4 or IPv6 header and the SPI in the enclosed extension header (Authentication header or Encapsulating Security Payload header).

A security association is normally defined by the following parameters:

- Authentication algorithm and algorithm mode being used with the IP Authentication Header (required for AH implementations).
- Key(s) used with the authentication algorithm in use with the Authentication Header (required for AH implementations).

[4] In the remainder of this section, the term IP packet refers to either an IPv4 datagram or an IPv6 packet.

- Encryption algorithm, algorithm mode, and transform being used with the IP Encapsulating Security Payload (required for ESP implementations).
- Key(s) used with the encryption algorithm in use with the Encapsulating Security Payload (required for ESP implementations).
- Presence/absence and size of a cryptographic synchronization or initialization vector field for the encryption algorithm (required for ESP implementations).
- Authentication algorithm and mode used with the ESP transform, if any is in use (recommended for ESP implementations).
- Authentication key(s) used with the authentication algorithm that is part of the ESP transform, if any (recommended for ESP implementations).
- Lifetime of the key or time when key change should occur (recommended for all implementations).
- Lifetime of this Security Association (recommended for all implementations).
- Source Address(es) of the Security Association, might be a wildcard address if more than one sending system shares the same Security Association with the destination (recommended for all implementations).
- Sensitivity level (for example, Secret or Unclassified) of the protected data (required for all systems claiming to provide multi-level security, recommended for all other systems).

The key management mechanism that is used to distribute keys is coupled to the authentication and privacy mechanisms only by way of the Security Parameters Index. Hence, authentication and privacy have been specified independently of any specific key-management mechanism.

Authentication

The Authentication header provides support for data integrity and authentication of IP packets. The Authentication Header consists of the following fields (Figure 18.20):

- **Next Header (8 bits).** Identifies the type of header immediately following this header.

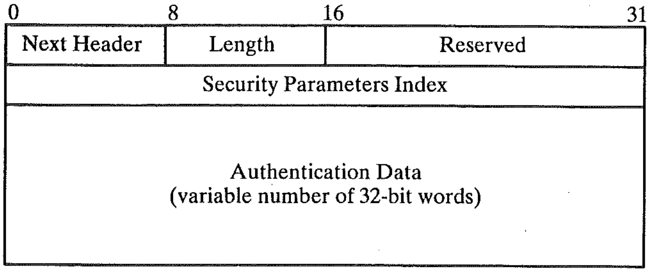


FIGURE 18.20 Authentication header.

- **Length (8 bits).** Length of Authentication Data field in 32-bit words.
- **Reserved (16 bits).** For future use.
- **Security Parameters Index (32 bits).** Identifies a security association.
- **Authentication Data (variable).** An integral number of 32-bit words.

The authentication data field contents will depend on the authentication algorithm specified. In any case, the authentication data is calculated over the entire IP packet, excluding any fields that may change in transit. Such fields are set to zero for purposes of calculation at both source and destination. The authentication calculation is performed prior to fragmentation at the source and after reassembly at the destination. Hence, fragmentation-related fields can be included in the calculation.

For IPv4, the Time-to-Live and Header Checksum fields are subject to change and are therefore set to zero for the authentication calculation. IPv4 options must be handled in accordance with the rule that any option whose value might change during transit must not be included in the calculation.

For IPv6, the Hop Limit field is the only field in the base IPv6 header subject to change; it is therefore set to zero for the calculation. For the Hop-by-Hop Options and Destination Options headers, the Option Type field for each option contains a bit that indicates whether the Option Data field for this option may change during transit; if so, this option is excluded from the authentication calculation.

Authentication Using Keyed MD5

RFC 1828 specifies the use of MD5 for authentication. The MD5 algorithm is performed over the IP packet plus a secret key by the source and then inserted into the IP packet. At the destination, the same calculation is performed on the IP packet plus the secret key and compared to the received value. This procedure provides both authentication and data integrity.

Specifically, the MD5 calculation is performed over the following sequence:

key, keyfill, IP packet, key, MD5fill

where

key = the secret key for this security association
keyfill = padding so that key + keyfill is an integer multiple of 512 bits
IP packet = with the appropriate fields set to zero
MD5fill = padding supplied by MD5 so that the entire block is an integer multiple of 512 bits

Figure 18.21 shows two ways in which the IP authentication service can be used. In one case, authentication is provided directly between a server and client workstations; the workstation can be either on the same network as the server or on an external network. So long as the workstation and the server share a protected

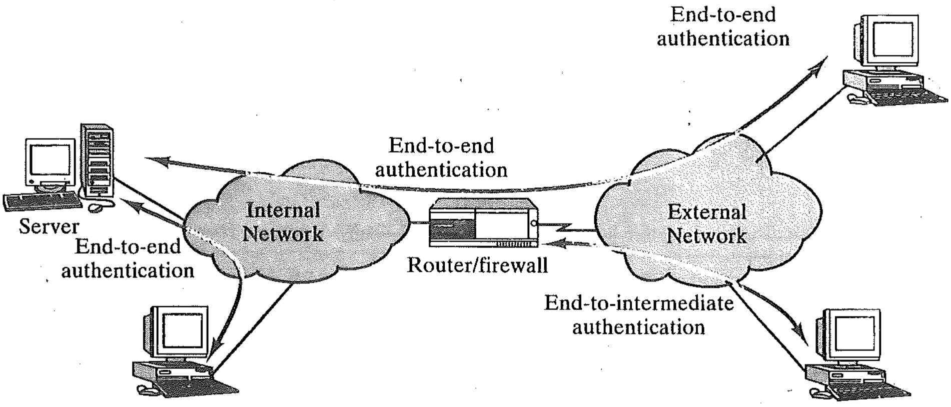


FIGURE 18.21 End-to-end versus end-to-intermediate authentication [DOTY95].

secret key, the authentication process is secure. In the other case, a remote workstation authenticates itself to the corporate firewall, either for access to the entire internal network or because the requested server does not support the authentication feature.

Encapsulating Security Payload

The use of the Encapsulating Security Payload provides support for privacy and data integrity for IP packets. Depending on the user's requirements, this mechanism can be used to encrypt either a transport-layer segment (e.g., TCP, UDP, ICMP), known as transport-mode ESP, or an entire IP packet, known as tunnel-mode ESP.

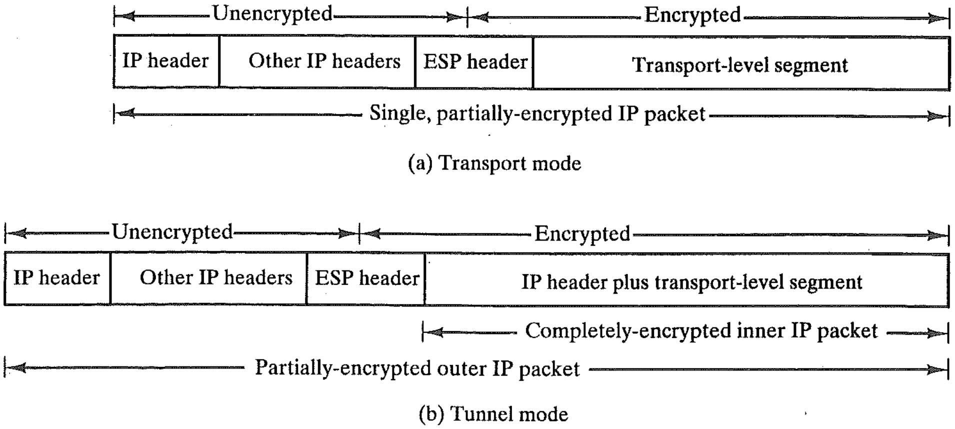
The ESP header begins with a 32-bit Security Parameters Index (SPI), which identifies a security association. The remainder of the header, if any, may contain parameters dependent on the encryption algorithm being used. In general, the first part of the header, including the SPI and possibly some parameters, is transmitted in unencrypted (plaintext) form, while the remainder of the header, if any, is transmitted in encrypted (ciphertext) form.

Transport-Mode ESP

Transport-mode ESP is used to encrypt the data carried by IP. Typically, these data are a transport-layer segment, such as a TCP or UDP segment, which in turn contains application-level data. For this mode, the ESP header is inserted into the IP packet immediately prior to the transport-layer header (e.g., TCP, UDP, ICMP). In the case of IPv6, if a Destination Options header is present, the ESP header is inserted immediately prior to that header (see Figure 16.16).

Transport-mode operation may be summarized as follows:

1. At the source, the block of data consisting of a trailing portion of the ESP header, plus the entire transport-layer segment, is encrypted, and the plaintext


Unencrypted
Encrypted
IP header
Other IP headers
ESP header
Transport-level segment
Single, partially-encrypted IP packet
Unencrypted
Encrypted
IP header
Other IP headers
ESP header
IP header plus transport-level segment
Completely-encrypted inner IP packet
Partially-encrypted outer IP packet

(a) Transport mode

(b) Tunnel mode

FIGURE 18.22 Secure IPv4 datagram or IPv6 packet.

of this block is replaced with its ciphertext to form the IP packet for transmission (Figure 18.22a).

2. This IP packet is then routed to the destination. Each intermediate router needs to examine and process the IP header plus any plaintext IP extension headers, but does not need to examine the ciphertext.
3. The destination node examines and processes the IP header plus any plaintext IP extension headers. Then, on the basis of the SPI in the ESP header, the destination node decrypts the remainder of the packet to recover the plaintext transport-layer segment.

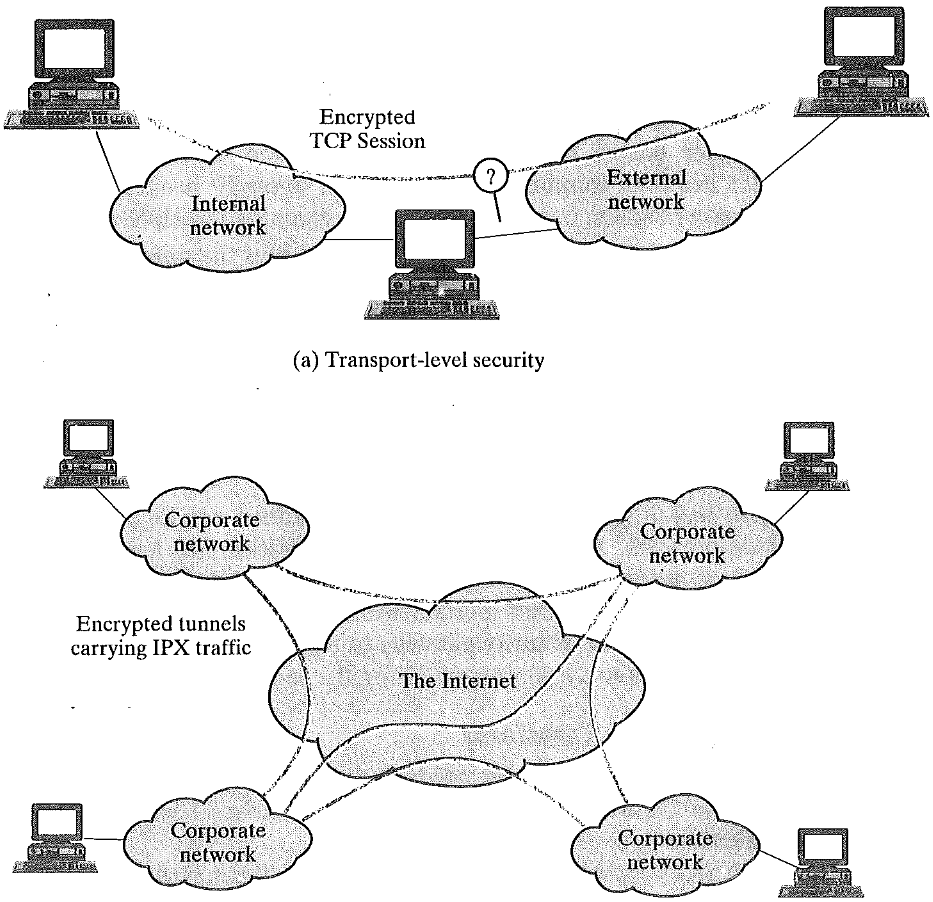
Transport-mode operation provides privacy for any application that uses it, thus avoiding the need to implement privacy in every individual application. This mode of operation is also reasonably efficient, adding little to the total length of the IP packet. One drawback to this mode is that it is possible to do traffic analysis on the transmitted packets. Figure 18.23a illustrates the protection provided by transport-mode operation.

Tunnel-Mode ESP

Tunnel-mode ESP is used to encrypt an entire IP packet. For this mode, the ESP is prefixed to the packet, and then the packet plus a trailing portion of the ESP header is encrypted. This method can be used to counter traffic analysis.

Because the IP header contains the destination address and possibly source routing directives and hop-by-hop option information, it is not possible to simply transmit the encrypted IP packet prefixed by the ESP header. Intermediate routes would be unable to process such a packet. Therefore, it is necessary to encapsulate the entire block (ESP header plus encrypted IP packet) with a new IP header that will contain sufficient information for routing but not for traffic analysis.

Whereas the transport-mode is suitable for protecting connections between hosts that support the ESP feature, the tunnel-mode is useful in a configuration that



(a) Transport-level security

(b) A virtual private network via tunnel mode

FIGURE 18.23 Applications of encapsulating security payload [DOTY95].

includes a firewall or other sort of security gateway that protects a trusted network from external networks. In this latter case, encryption occurs only between an external host and the security gateway, or between two security gateways; this relieves hosts on the internal network of the processing burden of encryption and also simplifies the key distribution task by reducing the number of needed keys; further, it thwarts traffic analysis based on ultimate destination.

Consider a case in which an external host wishes to communicate with a host on an internal network protected by a firewall, and in which ESP is implemented in the external host and the firewalls. The following steps occur for transfer of a transport-layer segment from the external host to the internal host:

1. The source prepares an inner IP packet with a destination address of the target internal host. This packet is prefixed by an ESP header; then the packet

and a portion of the ESP header are encrypted. The resulting block is encapsulated with a new IP header (base header plus optional extensions such as routing and hop-by-hop options) whose destination address is the firewall; this forms the outer IP packet (Figure 18.22b).

2. The outer packet is routed to the destination firewall. Each intermediate router needs to examine and process the outer IP header plus any outer IP extension headers, but does not need to examine the ciphertext.
3. The destination firewall examines and processes the outer IP header plus any outer IP extension headers. Then, on the basis of the SPI in the ESP header, the destination node decrypts the remainder of the packet to recover the plaintext inner IP packet. This packet is then transmitted in the internal network.
4. The inner packet is routed through zero or more routers in the internal network to the destination host.

Figure 18.22b shows how tunnel-mode operation can be used to set up a virtual private network. In this example, an organization has four private networks interconnected across the Internet. Hosts on the internal networks use the Internet for transport of data but don't interact with other Internet-based hosts. By terminating the tunnels at the security gateway to each internal network, the configuration allows the hosts to avoid implementing the security capability.

The ESP DES-CBC Transform

All implementations that claim conformance with the ESP specification must implement the DES-CBC (Data Encryption Standard-Cipher Block Chaining) method of encryption.

Figure 18.24 shows the format of the ESP header plus payload using DES-CBC, and indicates which portions are encrypted. The fields are

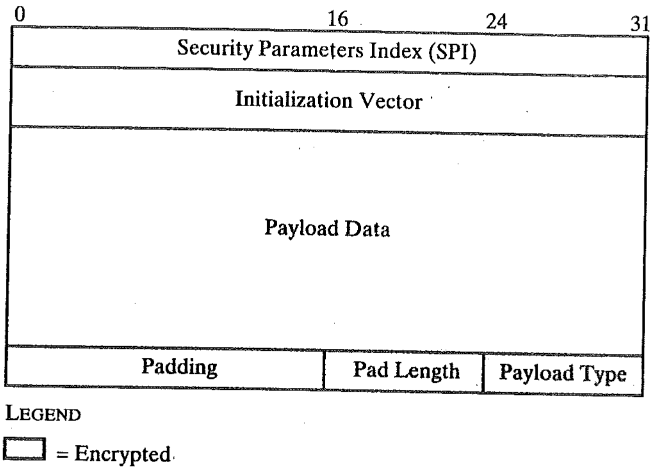


FIGURE 18.24 Encapsulating security payload format.

- **Security Parameters Index (32 bits).** Identifies a security association.
- **Initialization Vector (variable).** Input to the CBC algorithm, and a multiple of 32 bits.
- **Payload Data (variable).** Prior to encryption, this field contains the block of data to be encrypted, which may be a transport-layer segment (transport mode) or an IP packet (tunnel mode).
- **Padding.** Prior to encryption, filled with unspecified data to align Pad Length and Payload Type fields at a 64-bit boundary.
- **Pad Length (8 bits).** The size of the unencrypted padding field.
- **Payload Type (8 bits).** Indicates the protocol type of the Payload Data field (e.g., IP, TCP).

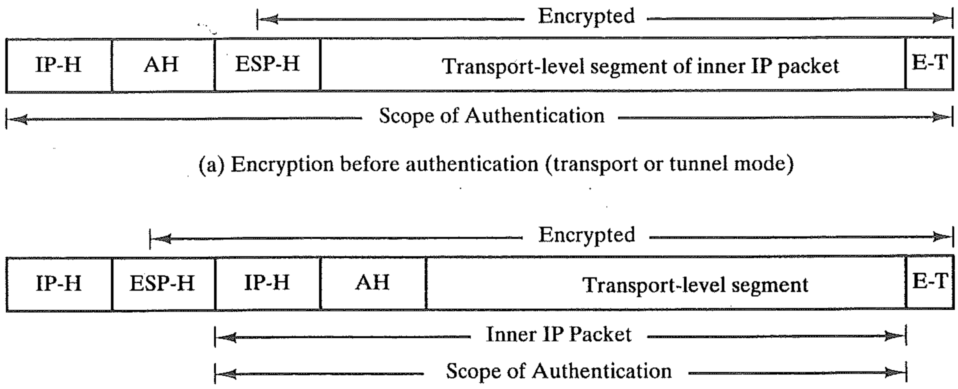
Note that the Initialization Vector is transmitted in plaintext. As was mentioned in our description of DES-CBC, this is not the most secure approach. However, for the purposes of this application, it should provide acceptable security.

Authentication Plus Privacy

The two IP security mechanisms can be combined in order to transmit an IP packet that has both privacy and authentication. There are two approaches that can be used, based on the order in which the two services are applied.

Encryption Before Authentication

Figure 18.25a illustrates the case of encryption applied before authentication. In this case, the entire transmitted IP packet is authenticated, including both



(a) Encryption before authentication (transport or tunnel mode)

(b) Authentication before encryption (tunnel mode)

LEGEND

IP-H = IP base header plus extentions headers
ESP-H= Encapsulating Security Payload header
E-T= Encapsulating Security Payload trailing fields
AH= Authentication header

FIGURE 18.25 Combining privacy and authentication.

encrypted and unencrypted parts. In this approach, the user first applies ESP to the data to be protected, then prepends the authentication header and the plaintext IP header(s). There are actually two subcases:

- **Transport-Mode ESP.** Authentication applies to the entire IP packet delivered to the ultimate destination, but only the transport-layer segment is protected by the privacy mechanism (encrypted).
- **Tunnel-Mode ESP.** Authentication applies to the entire IP packet delivered to the outer IP destination address (e.g., a firewall), and authentication is performed at that destination. The entire inner IP packet is protected by the privacy mechanism, for delivery to the inner IP destination.

Authentication Before Encryption

Figure 18.25b illustrates the case of authentication applied before encryption. This approach is only appropriate for tunnel-mode ESP. In this case, the authentication header is placed inside the inner IP packet. This inner packet is both authenticated and protected by the privacy mechanism.

As we have just seen, the functions of authentication and encryption can be applied in either order for tunnel-mode ESP. The use of authentication prior to encryption might be preferable for several reasons. First, because the AH is protected by ESP, it is impossible for anyone to intercept the message and alter the AH without detection. Secondly, it may be desirable to store the authentication information with the message and the destination for later reference. It is more convenient to perform this storage if the authentication information applies to the unencrypted message; otherwise, the message would have to be re-encrypted to verify the authentication information.

18.6 RECOMMENDED READING

The topics in this chapter are covered in greater detail in [STAL95]. For coverage of cryptographic algorithms, [SCHN96] is an essential reference work; it contains descriptions of virtually every cryptographic algorithm and protocol published in the last 15 years. Two very good collections of papers are [SIMM92], which provides a rigorous, mathematical treatment of cryptographic protocols and algorithms, and [ABRA95], which deals with the issues at more of a system level.

ABRA95 Abrams, M, Jajodia, S. and Podell, H. eds. *Information Security.* Los Alamitos, CA: IEEE Computer Society Press, 1995.

SCHN96 Schneier, B. *Applied Cryptography.* New York: Wiley, 1996.

SIMM92 Simmons, G., ed. *Contemporary Cryptology: The Science of Information Integrity.* Piscataway, NJ: IEEE Press, 1992.

STAL95 Stallings, W. *Network and Internetwork Security: Principles and Practice.* Upper Saddle River, NJ: Prentice-Hall, 1995.

Recommended Web Site

- http://www.tansu.com.au/Info/security.html: Contains pointers to FAQs, documents, USENET groups, and other web sites related to security.

18.7 PROBLEMS

18.1 One of the simplest forms of encryption is a substitution cipher. A substitution cipher is an encryption technique in which the letters of plaintext are replaced by other letters or by numbers or symbols. In a famous story by Edgar Allen Poe, the following ciphertext was generated using a simple substitution algorithm:

```
53‡‡†305))6*;4826)4‡.)4‡);806*;48†8¶60))85;;]8*;:‡*8†83
(88)5*†;46(;88*96*?;8)*‡(;485);5*†2:*‡(;4956*2(5*—4)8¶8*
;4069285);)6†8)4‡‡;1(‡9;48081;8;8‡1;48†85;4)485†528806*81
(‡9;48;(88;4(‡?34;48)4‡;161;:188;‡?;
```

Decrypt this message. Hints:

a. As you know, the most frequently occurring letter in English is e. Therefore, the first or second (or perhaps third?) most common character in the message is likely to stand for e. Also, e is often seen in pairs (e.g., meet, fleet, speed, seen, been, agree, etc.). Try to find a character in the ciphertext that decodes to e.

b. The most common word in English is "the." Use this fact to guess the characters that stand for t and h.

c. Decipher the rest of the message by deducing additional words.

Warning: the resulting message is in English but may not make much sense on a first reading.

18.2 One way to solve the key distribution problem is to use a line from a book that both the sender and the receiver possess to generate the key for a substitution cipher. Typically, at least in spy novels, the first sentence of a book serves as the key. The particular scheme discussed in this problem is from one of the best suspense novels involving secret codes, *Talking to Strange Men*, by Ruth Rendell. Work this problem without consulting that book!

Consider the following message:

SIDKHKDM AF HCRKIABIE SHIMC KD LFEAILA

This ciphertext was produced using the first sentence of *The Other Side of Silence* (a book about the spy Kim Philby):

The snow lay thick on the steps and the snowflakes driven by the wind looked black in the headlights of the cars.

a. What is the encryption algorithm?

b. How secure is it?

c. To make the key distribution problem simple, both parties can agree to use the first or last sentence of a book as the key. To change the key, they simply need to agree on a new book. The use of the first sentence would be preferable to the use of the last.

Why?

18.3 In one of his cases, Sherlock Holmes was confronted with the following message.

534 C2 13 127 36 31 4 17 21 41
DOUGLAS 109 293 5 37 BIRLSTONE
26 BIRLSTONE 9 127 171

Although Watson was puzzled, Holmes was able to immediately deduce the type of encryption algorithm. Can you?

18.4 Let π be a permutation of the integers $1, 2, \ldots 2^n - 1$, such that π(m) gives the permuted value of m, $0 \leq m \leq 2^n$. Put another way, π maps the set of n-bit integers into itself and no two integers map into the same integer. DES is such a permutation for 64-bit integers. We say that π has a fixed point at m if $\pi(m) = m$. That is, if π is an encryption mapping, then a fixed point corresponds to a message that encrypts to itself. We are interested in the probability that π has no fixed points. Show the somewhat unexpected result that over 60% of mappings will have at least one fixed point.

18.5 Consider a block encryption algorithm that encrypts blocks of length $N = 2^n$. Say we have t plaintext/ciphertext pairs P_i, $C_i = E_K[P_i]$, where we assume that the key K selects one of the $N!$ possible mappings. Imagine that we wish to find K by exhaustive search. We could generate key K′ and test whether $C_i = E_{K'}(P_i)$ for $1 \leq i \leq t$. If K′ encrypts each P_i to its proper C_i, then we have evidence that K = K′. However, it may be the case that the mappings $E_K(\cdot)$ and $E_{K'(\cdot)}$ exactly agree on the t plaintext/ciphertext pairs P_i, C_i and agree on no other pairs.

a. What is the probability that $E_K(\cdot)$ and $E_{K'}(\cdot)$ are, in fact, distinct mappings?

b. What is the probability that $E_K(\cdot)$ and $E_{K'}(\cdot)$ agree on another t' plaintext/ciphertext pair where $0 \leq t' \leq N\text{–}k$?

18.6 Suppose that someone suggests the following way to confirm that the two of you are both in possession of the same secret key. You create a random bit string that is the length of the key, XOR it with the key, and send the result over the channel. Your partner XORs the incoming block with the key (which should be the same as your key) and sends it back. You check, and if what you receive is your original random string, you have verified that your partner has the same secret key, yet neither of you has ever transmitted the key. Is there a flaw in this scheme?

18.7 Consider the following scheme:

a. Pick an odd number, E.

b. Pick two prime numbers, P and Q, where $(P\text{–}1)(Q\text{–}1)\ –1$ is evenly divisible by E.

c. Multiply P and Q to get N.

d. Calculate $D = \dfrac{(P-1)(Q-1)(E-1)+1}{E}$

Is this scheme equivalent to RSA? Show why or why not.

18.8 Perform encryption and decryption using the RSA algorithm, as in Figure 4.8, for the following:

a. $p = 3; q = 11, d = 7;$ M = 5

b. $p = 5; q = 11, e = 3;$ M = 9

c. $p = 7; q = 11, e = 17;$ M = 8

d. $p = 11; q = 13, e = 11;$ M = 7

e. $p = 17; q = 31, e = 7;$ M = 2. Hint: Decryption is not as hard as you think; use some finesse.

18.9 Construct a 32-bit checksum function as the concatenation of two 16-bit functions: XOR, and RXOR, defined in Figure 18.13 as "two simple hash functions."

a. Will this checksum detect all errors caused by an odd number of error bits? Explain.

b. Will this checksum detect all errors caused by an even number of error bits? If not, characterize the error patterns that will cause the checksum to fail.

c. Comment on the effectiveness of this function for use as a hash function for authentication.

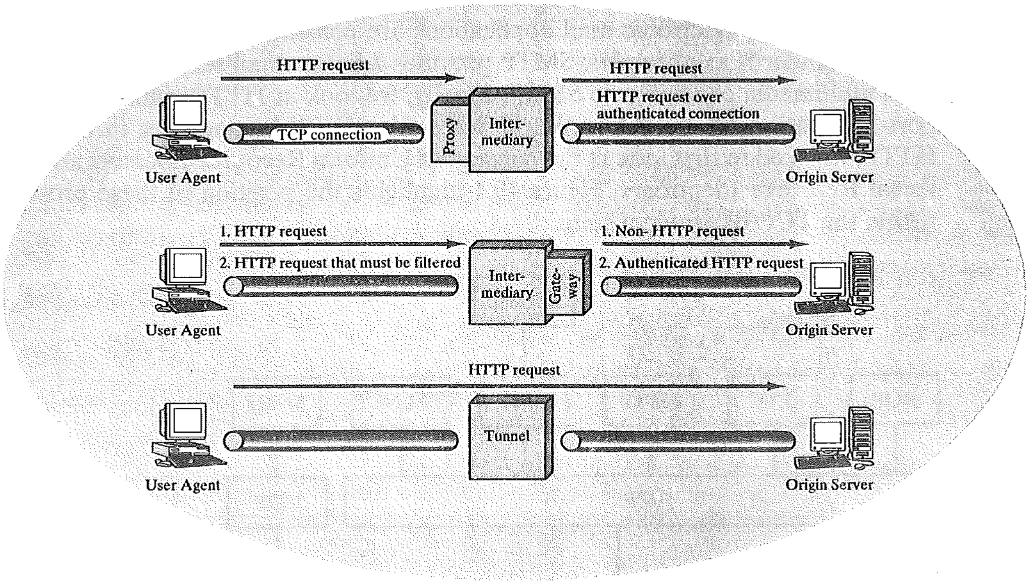
18.10 Consider using an encryption algorithm to construct a one-way hash function. Consider using RSA with a known key. Then, process a message consisting of a sequence of blocks as follows: Encrypt the first block, XOR the result with the second block and encrypt again, etc. Show that this scheme is not secure by solving the following problem. Consider a two-block message B1, B2, and its hash:

$$\text{RSAH(B1, B2)} = \text{RSA (RSA(B1)} \oplus \text{B2)}$$

Given an arbitrary block C1, choose C2 so that RSAH(C1, C2) = RSAH(B1, B2).

CHAPTER 19

DISTRIBUTED APPLICATIONS



All of the protocols and functions described so far in Part Four are geared toward one objective: the support of distributed applications that involve the interaction of multiple independent systems. In the OSI model, such applications occupy the application layer and are directly supported by the presentation layer. In the TCP/IP protocol suite, such applications typically rely on TCP or UDP for support.

We begin this chapter with an introduction to Abstract Syntax Notation One (ASN.1), which has become an important universal language for defining representations of data structures and protocol formats, and which has gained wide use for defining application-level protocols.

Next, we examine a number of quite different applications that give the reader a feel for the range and diversity of applications supported by a communications architecture. The first, *network management*, is itself a support-type application, designed to assure the effective monitoring and control of a distributed system. The specific protocol that is examined is the Simple Network Management Protocol, version 2 (SNMPv2), which is designed to operate in both the TCP/IP and OSI environments. Next, electronic mail applications are considered, with the SMTP and MIME standards as examples; SMTP provides a basic email service, while MIME adds multimedia capability to SMTP. Finally, we look at HTTP, which is the support application on which the World Wide Web (WWW) operates; in discussing HTTP, we need to first look at the concept of Uniform Resource Locators and Universal Resource Identifiers. Figure 19.1 highlights the position of these protocols within the TCP/IP protocol suite.

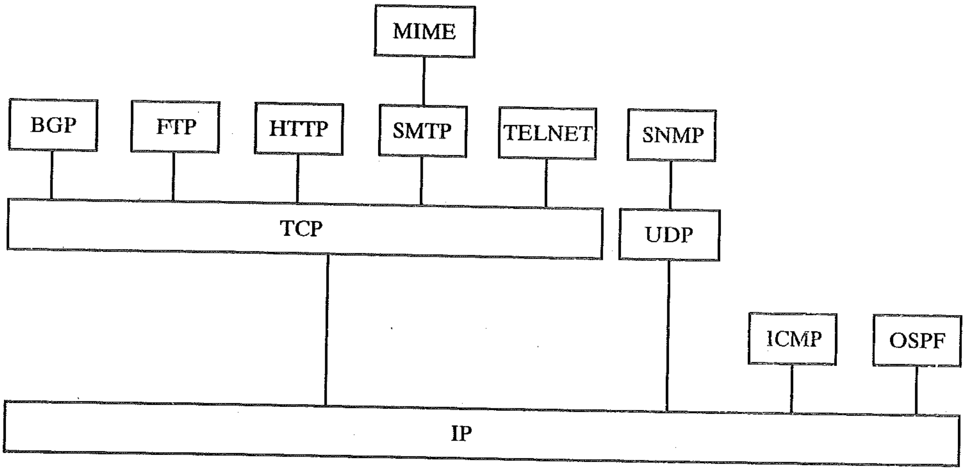


FIGURE 19.1 Application-level protocols discussed in this chapter.

19.1 ABSTRACT SYNTAX NOTATION ONE (ASN.1)

One of the most significant developments in computer communications in recent years is the development of ASN.1, which is now widely used in the development of both OSI-related standards and TCP/IP-related standards. It is used to define the format of protocol data units (PDUs), the representation of distributed information, and operations performed on transmitted data. A basic understanding of ASN.1 is essential for those who wish to study and work in this field.

Before examining the details of ASN.1, we need to introduce the concept of an abstract syntax. Then, we will look at the fundamentals of ASN.1. Next, a special and important facility, the ASN.1 macro facility, is examined.

Abstract Syntax

Table 19.1 defines some key terms that are relevant to a discussion of ASN.1, and Figure 19.1 illustrates the underlying concepts.

For purposes of this discussion, a communications architecture in an end system can be considered to have two major components. The data transfer component is concerned with the mechanisms for the transfer of data between end systems. In the case of the TCP/IP protocol suite, this component would consist of TCP or UDP on down. In the case of the OSI architecture, this component would consist of the session layer on down. The application component is the user of the data transfer component and is concerned with the end user's application. In the case of the TCP/IP protocol suite, this component would consist of an application, such as SNMP, FTP, SMTP, or TELNET. In the case of OSI, this component actually consists of the application layer, which is composed of a number of application service elements, and the presentation layer.

As we cross the boundary from the application to the data transfer component, there is a significant change in the way that data are viewed. For the data transfer component, the data received from an application are specified as the binary value of a sequence of octets. This binary value can be directly assembled into service data units (SDUs) for passing between layers, and into protocol data units (PDUs) for passing between protocol entities within a layer. The application component, however, is concerned with a user's view of data. In general, that view is one of a structured set of information, such as text in a document, a personnel file, an integrated data base, or a visual display of image information. The user is primarily concerned with the semantics of data. The application component must provide a representation of this data that can be converted to binary values; that is, it must be concerned with the syntax of the data.

TABLE 19.1 Terms relevant to ASN.1.

Abstract Syntax	Describes the generic structure of data independent of any encoding technique used to represent the data. The syntax allows data types to be defined and values of those types to be specified.
Data Type	A named set of values. A type may be simple, which is defined by specifying the set of its values, or structured, which is defined in terms of other types.
Encoding	The complete sequence of octets used to represent a data value.
Encoding Rules	A specification of the mapping from one syntax to another. Specifically, encoding rules determine algorithmically, for any set of data values defined in an abstract syntax, the representation of those values in a transfer syntax.
Transfer Syntax	The way in which data are actually represented in terms of bit patterns while in transit between presentation entities.

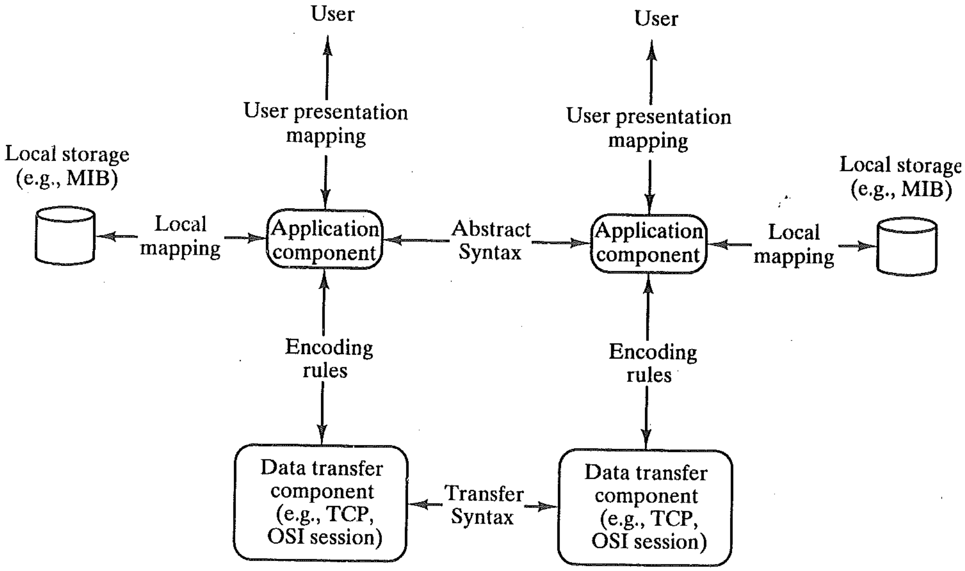

User
User presentation mapping
User
User presentation mapping
Local storage (e.g., MIB)
Local mapping
Application component
Abstract Syntax
Application component
Local mapping
Local storage (e.g., MIB)
Encoding rules
Encoding rules
Data transfer component (e.g., TCP, OSI session)
Transfer Syntax
Data transfer component (e.g., TCP, OSI session)

FIGURE 19.2 The use of abstract and transfer syntaxes.

The approach illustrated in Figure 19.2 to support application data is as follows. For the application component, information is represented in an abstract syntax that deals with data types and data values. The abstract syntax formally specifies data independently from any specific representation. Thus, an abstract syntax has many similarities to the data-type definition aspects of conventional programming languages such as Pascal, C, and Ada, and to grammars such as Backus-Naur Form (BNF). Application protocols describe their PDUs in terms of an abstract syntax.

This abstract syntax is used for the exchange of information between application components in different systems. The exchange consists of application-level PDUs, which contain protocol control information and user data. Within a system, the information represented using an abstract syntax must be mapped into some form for presentation to the human user. Similarly, this abstract syntax must be mapped into some local format for storage. For example, such a mapping is used in the case of network management information. In addition, it is becoming common to use an abstract syntax to define the data elements in local storage. Thus, the abstract syntax notation is employed by a user to define network management information; the application must then convert this definition to a form suitable for local storage.

The component must also translate between the abstract syntax of the application and a transfer syntax that describes the data values in a binary form, suitable for interaction with the data transfer component. For example, an abstract syntax may include a data type of character; the transfer syntax could specify ASCII or EBCDIC encoding.

The transfer syntax thus defines the representation of the data to be exchanged between data-transfer cómponents. The translation from abstract syntax to the transfer syntax is accomplished by means of encoding rules that specify the representation of each data value of each data type.

This approach for the exchange of application data solves the two problems that relate to data representation in a distributed, heterogeneous environment:

- There is a common representation for the exchange of data between differing systems.
- Internal to a system, an application uses some particular representation of data. The abstract/transfer syntax scheme automatically resolves differences in representation between cooperating application entities.

The fundamental requirement for selection of a transfer syntax is that it support the corresponding abstract syntax. In addition, the transfer syntax may have other attributes that are not related to the abstract syntaxes that it can support. For example, an abstract syntax could be supported by any one of four transfer syntaxes, which are the same in all respects except that one provides data compression, one provides encryption, one provides both, and one provides neither. The choice of which transfer syntax to use would depend on cost and security considerations.

ASN.1 Concepts

The basic building block of an ASN.1 specification is the module. We begin this section by looking at the top-level structure of the module. Then, we introduce some lexical conventions used in ASN.1 definitions. Next, the data types defined in ASN.1 are described. Finally, examples of the use of ASN.1 are given.

Module Definition

ASN.1 is a language that can be used to define data structures. A structure definition is in the form of a named module. The name of the module can then be used to reference the structure. For example, the module name can be used as an abstract syntax name; an application can pass this name to the presentation service to specify the abstract syntax of the application PDUs that the application wishes to exchange with a peer application entity.

Modules have the basic form,

```
<modulereference> DEFINITIONS ::=
   BEGIN
        EXPORTS
        IMPORTS
        AssignmentList
   End
```

The modulereference is a module name followed optionally by an object identifier to identify the module. The EXPORTS construct indicates which definitions in this module may be imported by other modules. The IMPORTS construct indicates which type and value definitions from other modules are to be imported into this module. Neither the IMPORTS or EXPORTS constructs may be included unless the object identifier for the module is included. Finally, the assignment list consists of type assignments, value assignments, and macro definitions. Macro definitions are discussed later in this section. Type and value assignments have the form,

```
<name>::=<description>
```

The easiest way to describe the syntax is by example. First, we need to specify some lexical conventions.

Lexical Conventions

ASN.1 structures, types, and values are expressed in a notation similar to that of a programming language. The following lexical conventions are followed:

1. Layout is not significant; multiple spaces and blank lines can be considered as a single space.
2. Comments are delimited by pairs of hyphens (--) at the beginning and end of the comment, or by a pair of hyphens at the beginning of the comment and the end of the line as the end of the comment.
3. Identifiers (names of values and fields), type references (names of types), and module names consist of upper- and lowercase letters, digits, and hyphens.
4. An identifier begins with a lowercase letter.
5. A type reference or a module name begins with an uppercase letter.
6. A built-in type consists of all capital letters and is a commonly used type for which a standard notation is provided.

Abstract Data Types

ASN.1 is a notation for abstract data types and their values. A type can be viewed as a collection of values. The number of values that a type may take on may be infinite. For example, the type INTEGER has an infinite number of values.

We can classify types into four categories:

- **Simple.** These are atomic types, with no components.
- **Structured.** A structured type has components.
- **Tagged.** These are types derived from other types.
- **Other.** This category includes the CHOICE and ANY types, defined later in this section.

Every ASN.1 data type, with the exception of CHOICE and ANY, has an associated tag. The tag consists of a class name and a nonnegative integer tag number. There are four classes of data types, or four classes of tag:

- **Universal.** Generally useful, application-independent types and construction mechanisms; these are defined in the standard and are listed in Table 19.2.
- **Application-wide.** Relevant to a particular application; these are defined in other standards.

TABLE 19.2 Universal class tag assignments.

Tag	Type Name	Set of Values
		Basic Types
UNIVERSAL 1	BOOLEAN	TRUE or FALSE
UNIVERSAL 2	INTEGER	The positive and negative whole numbers, including zero
UNIVERSAL 3	BIT STRING	A sequence of zero or more bits
UNIVERSAL 4	OCTET STRING	A sequence of zero or more octets
UNIVERSAL 9	REAL	Real numbers
UNIVERSAL 10	ENUMERATED	An explicit list of integer values that an instance of a data type may take
		Object Types
UNIVERSAL 6	OBJECT IDENTIFIER	The set of values associated with information objects
UNIVERSAL 7	Object descriptor	Each value is human-readable text providing a brief description of an information object.
		Object Types
UNIVERSAL 18	NumericString	Digits 0 through 9, space
UNIVERSAL 19	PrintableString	Printable characters
UNIVERSAL 20	TeletexString	Character set defined by CCITT Recommendation T.61
UNIVERSAL 21	VideotexString	Set of alphabet and graphical characters defined by CCITT Recommendations T.100 and T.101.
UNIVERSAL 22	IA5String	International alphabet five (equivalent to ASCII)
UNIVERSAL 25	GraphicsString	Character set defined by ISO 8824
UNIVERSAL 26	VisibleString	Character set defined by ISO 646 (equivalent to ASCII)
UNIVERSAL 27	GeneralString	General character string
		Miscellaneous Types
UNIVERSAL 5	NULL	The single value NULL. Commonly used where several alternatives are possible but none of them apply.
UNIVERSAL 8	EXTERNAL	A type defined in some external document. It need not be one of the valid ASN.1 types.
UNIVERSAL 23	UTCTime	Consists of the date, specified with a two-digit year, a two-digit month and a two-digit day, followed by the time, specified in hours, minutes, and optionally seconds, followed by an optional specification of the local time differential from universal time.
UNIVERSAL 24	GeneralizedTime	Consists of the date, specified with a four-digit year, a two-digit month and a two-digit day, followed by the time, specified in hours, minutes, and optionally seconds, followed by an optional specification of the local time differential from universal time.
UNIVERSAL 11–15	Reserves	Reserved for addenda to the ASN.1 standard
UNIVERSAL 28–	Reserved	Reserved for addenda to the ASN.1 standard

(*Continued*)

TABLE 19.2 (*Continued*)

Structured Types		
UNIVERSAL 16	SEQUENCE and SEQUENCE-OF	Sequence: defined by referencing a fixed, ordered list of types; each value is an ordered list of values, one from each component type. Sequence-of: defined by referencing a single existing type; each value is an ordered list of zero or more values of the existing type.
UNIVERSAL 17	SET and SET-OF	Set: defined by referencing a fixed, unordered list of types, some of which may be declared optional; each value is an unordered list of values, one from each component type. Set-of: defined by referencing a single existing type; each value is an unordered list of zero or more values of the existing type.

- **Context-specific.** Also relevant to a particular application, but applicable in a limited context.
- **Private.** Types defined by users and not covered by any standard.

A data type is uniquely identified by its tag. ASN.1 types are the same if and only if their tag numbers are the same. For example, UNIVERSAL 4 refers to OctetString, which is of class UNIVERSAL and has tag number 4 within the class.

A **simple type** is one defined by directly specifying the set of its values. We may think of these as the atomic types; all other types are built up from the simple types. The simple data types in the UNIVERSAL class can be grouped into several categories, as indicated in Table 19.2; these are not "official" categories in the standard but are used here for convenience.

The first group of simple types can be referred to, for want of a better word, as basic types. The Boolean type is straightforward. The Integer type is the set of positive and negative integers and zero. In addition, individual integer values can be assigned names to indicate a specific meaning. The BitString is an ordered set of zero or more bits; individual bits can be assigned names. The actual value of a BitString can be specified as a string of either binary or hexadecimal digits. Similarly, an OctetString can be specified as a string of either binary or hexadecimal digits. The Real data type consists of numbers expressed in scientific notation (mantissa, base, exponent); that is,

$$M \times B^E$$

The mantissa (M) and the exponent (E) may take on any integer values, positive or negative; a base (B) of 2 or 10 may be used.

Finally, the Enumerated type consists of an explicitly enumerated list of integers, together with an associated name for each integer. The same functionality can be achieved with the Integer type by naming some of the integer values; but, because of the utility of this feature, a separate type has been defined. Note, however, that although the values of the enumerated type are integers, they do not have integer semantics. That is, arithmetic operations should not be performed on enumerated values.

Object types are used to name and describe information objects. Examples of information objects are standards documents, abstract and transfer syntaxes, data structures, and managed objects. In general, an information object is a class of information (e.g., a file format) rather than an instance of such a class (e.g., an individual file). The Object identifier is a unique identifier for a particular object. Its value consists of a sequence of integers. The set of defined objects has a tree structure, with the root of the tree being the object referring to the ASN.1 standard. Starting with the root of the object identifier tree, each object-identifier component value identifies an arc in the tree. The Object descriptor is a human-readable description of an information object.

ASN.1 defines a number of character-string types. The values of each of these types consists of a sequence of zero or more characters from a standardized character set.

There are some miscellaneous types that have also been defined in the UNIVERSAL class. The Null type is used in places in a structure where a value may or may not be present. The Null type is simply the alternative of no value being present at that position in the structure. An External type is one whose values are unspecified in the ASN.1 standard; it is defined in some other document or standard and can be defined using any well-specified notation. UTCTime and GeneralizedTime are two different formats for expressing time. In both cases, either a universal or local time may be specified.

Structured types are those consisting of components. ASN.1 provides four structured types for building complex data types from simple data types:

- SEQUENCE
- SEQUENCE-OF
- SET
- SET-OF

The Sequence and Sequence-of types are used to define an ordered list of values of one or more other data types; this is analogous to the record structure found in many programming languages, such as COBOL. A Sequence consists of an ordered list of elements, each specifying a type and, optionally, a name. The notation for defining the sequence type is as follows:

```
SequenceType ::= SEQUENCE {ElementTypeList} | SEQUENCE { }

ElementTypeList ::= ElementType | ElementTypeList, ElementType

ElementType ::=
        NamedType                       |
        NamedType  OPTIONAL             |
        NamedType  DEFAULT Value        |
        COMPONENTS OF Type
```

A NamedType is a type reference with or without a name. Each element definition may be followed by the keyword OPTIONAL or DEFAULT. The

OPTIONAL keyword indicates that the component element need not be present in a sequence value. The DEFAULT keyword indicates that, if the component element is not present, then the value specified by the DEFAULT clause will be assigned. The COMPONENTS OF clause is used to define the inclusion, at this point in the ElementTypeList, of all the ElementType sequences appearing in the referenced type.

A Sequence-of consists of an ordered, variable number of elements, all of one type. A Sequence-of definition has the following form:

SequenceOfType ::= SEQUENCE OF Type | SEQUENCE

The notation SEQUENCE is to be interpreted as SEQUENCE-OF ANY; the type ANY is explained in a later subsection.

A Set is similar to a Sequence, except that the order of the elements is not significant; the elements may be arranged in any order when they are encoded into a specific representation. A Set definition has the following form:

SetType ::= SET {ElementTypeList} | SET { }

Thus, a set may include optional, default, and component-of clauses.

A Set-of is an unordered, variable number of elements, all of one type. A Set-of definition has the following form:

SetOfType ::= SET OF Type | SET

The notation SET is to be interpreted as SET OF ANY; the type ANY is explained in a later subsection.

The term *tagged type* is somewhat of a misnomer, as all data types in ASN.1 have an associated tag. The ASN.1 standard defines a tagged type as follows:

> A type defined by referencing a single existing type and a tag; the new type is isomorphic to the existing type, but is distinct from it. In all encoding schemes, a value of the new type can be distinguished from a value of the old type.

Tagging is useful to distinguish types within an application. It may be desired to have several different type names, such as Employee_name and Customer_name, which are essentially the same type. For some structures, tagging is needed to distinguish component types within the structured type. For example, optional components of a SET or SEQUENCE type are typically given distinct context-specific tags to avoid ambiguity.

There are two categories of tagged types: implicitly tagged types and explicitly tagged types. An implicitly tagged type is derived from another type by *replacing* the tag (old class name, old tag number) of the old type with a new tag (new class name, new tag number). For purposes of encoding, only the new tag is used.

An explicitly tagged type is derived from another type by *adding* a new tag to the underlying type. In effect, an explicitly tagged type is a structured type with one component: the underlying type. For purposes of encoding, both the new and old tags must be reflected in the encoding.

An implicit tag results in shorter encodings, but an explicit tag may be necessary to avoid ambiguity if the tag of the underlying type is indeterminate (e.g., if the underlying type is CHOICE or ANY).

The **CHOICE** and **ANY** types are data types without tags; the reason for this is that when a particular value is assigned to the type, then a particular type must be assigned at the same time. Thus, the type is assigned at "run time."

The CHOICE type is a list of alternative known types. Only one of these types will actually be used to create a value. It was stated earlier that a type can be viewed as a collection of values. The CHOICE type is the union of the sets of values of all of the component types listed in the CHOICE type. This type is useful when the values to be described can be of different types depending on circumstance, and all the possible types are known in advance.

The notation for defining the CHOICE type is as follows:

```
ChoiceType ::= CHOICE {AlternativeTypeList}
AlternativeTypeList ::= NamedType | AlternativeTypeList, NamedType
```

The ANY type describes an arbitrary value of an arbitrary type. The notation is simply

```
AnyType ::= ANY
```

This type is useful when the values to be described can be of different types but the possible types are not known in advance.

Subtypes

A subtype is derived from a parent type by restricting the set of values defined for a parent type. That is, the set of values for the subtype are a subset of the set of values for the parent type. The process of subtyping can extend to more than one level: that is, a subtype may itself be a parent of an even more restricted subtype.

Six different forms of notation for designating the values of a subtype are provided in the standard. Table 19.3 indicates which of these forms can be applied to particular parent types. The remainder of this subsection provides an overview of each form.

A *single-value subtype* is an explicit listing of all of the values that the subtype may take on. For example,

```
SmallPrime  ::= INTEGER ( 2 | 3 | 5 | 7 | 11 | 13 | 17 | 19 | 23 | 29 )
```

In this case, SmallPrime is a subtype of the built-in type INTEGER. As another example,

```
Months ::= ENUMERATED { january (1),
                        february (2),
                        march (3),
                        april (4),
                        may (5),
                        june (6),
                        july (7),
                        august (8),
                        september (9),
                        october (10),
                        november (11),
                        december (12) }
```

TABLE 19.3 Applicability of subtype value sets.

Type (or derived from such a type by tagging)	Single Value	Contained Subtype	Value Range	Size Constraint	Permitted Alphabet	Inner Subtyping
Boolean	✓	✓				
Integer	✓	✓	✓			
Enumerated	✓	✓				
Real	✓	✓	✓			
Object Identifier	✓	✓				
Bit String	✓	✓		✓		
Octet String	✓	✓		✓		
Character String Types	✓	✓		✓	✓	
Sequence	✓	✓				✓
Sequence-of	✓	✓		✓		✓
Set	✓	✓				✓
Set-of	✓	✓		✓		✓
Any	✓	✓				
Choice	✓	✓				✓

First-quarter ::= Months (january | february | march)
Second-quarter ::= Months (april | may | june)

First-quarter and Second-quarter are both subtypes of the enumerated type Months.

A *contained subtype* is used to form new subtypes from existing subtypes. The contained subtype includes all of the values of the subtypes that it contains. For example,

First-half ::= Months (INCLUDES First-quarter | INCLUDES Second-quarter)

A contained subtype may also include listing explicit values:

First-third ::= Months (INCLUDES First-quarter | april)

A *value-range subtype* applies only to INTEGER and REAL types; it is specified by giving the numerical values of the endpoints of the range. The special values PLUS-INFINITY and MINUS-INFINITY may be used. Also, the special values MIN and MAX may be used to indicate the minimum and maximum allowable values in the parent. Each endpoint of the range is either closed or open. When open, the specification of the endpoint includes the less-than symbol (<). The following are equivalent definitions:

PositiveInteger ::= INTEGER (0<..PLUS-INFINITY)
PositiveInteger ::= INTEGER (1..PLUS-INFINITY)
PositiveInteger ::= INTEGER (0<..MAX)
PositiveInteger ::= INTEGER (1..MAX)

The following are equivalent:

```
NegativeInteger ::= INTEGER (MINUS-INFINITY..<0)
NegativeInteger ::= INTEGER (MINUS-INFINITY..–1)
NegativeInteger ::= INTEGER (MIN..<0)
NegativeInteger ::= INTEGER (MIN..–1)
```

The **permitted-alphabet constraint** may only be applied to character string types. A permitted-alphabet type consists of all values (strings) that can be constructed using a subalphabet of the parent type. Examples are

```
TouchToneButtons ::= IA5String ( FROM
                ( "0" | "1" | "2" | "3" | "5" | "6" | "7" | "8" | "9" | "*" | "#" ) )
DigitString ::= IA5String ( FROM
                ( "0" | "1" | "2" | "3" | "5" | "6" | "7" | "8" | "9" ) )
```

A **size constraint** limits the number of items in a type. It can only be applied to the string types (bit string, octet string, character string) and to Sequence-of and Set-of types. The item that is constrained depends on the parent type, as follows:

Type	Unit of Measure
bit string	bit
octet string	octet
character string	character
sequence-of	component value
set-of	component value

As an example of a string type, Recommendation X.121 specifies that international data numbers, which are used for addressing end systems on public data networks, including X.25 networks, should consist of at least 5 digits but not more than 14 digits; this could be specified as follows:

```
ItlDataNumber ::= DigitString ( SIZE ( 5..10 ) )
```

Now consider a parameter list for a message that may include up to 12 parameters:

```
ParameterList ::= SET SIZE ( 0..12 ) OF Parameter
```

An **inner-type constraint** can be applied to the sequence, sequence-of, set, set-of, and choice types. An inner subtype includes in its value set only those values from the parent type that satisfy one or more constraints on the presence and/or values of the components of the parent type. This is a rather complex subtype, and only a few examples are given here.

Consider a protocol data unit (PDU) that may have four different fields, in no particular order:

```
PDU ::= SET { alpha    [0]  INTEGER,
              beta     [1]  IA5String  OPTIONAL,
              gamma    [2]  SEQUENCE OF Parameter,
              delta    [3]  BOOLEAN }
```

To specify a test that requires the Boolean to be false and the integer to be negative,

TestPDU ::= PDU (WITH COMPONENTS { ..., delta (FALSE), alpha (MIN...<0)})

To further specify that the beta parameter is to be present and either 5 or 12 characters in length,

FurtherTestPDU ::= TestPDU (WITH COMPONENTS {..., beta (SIZE (5 | 12) PRESENT})

As another example, consider the use of inner subtyping on a sequence-of construct

Text-block ::= SEQUENCE OF VisibleString

Address ::= Text-block (SIZE (1..6) | WITH COMPONENT (SIZE (1..32)))

The above indicates that the address consists of from 1 to 6 text blocks, and that each text block is from 1 to 32 characters in length.

PDU Example

As an example, consider the ASN.1 specification of the format of the protocol data units for the SNMPv2 protocol (described later in this chapter). The specification from the standard is reproduced in Figure 19.3.

The top-level construct uses the CHOICE type to describe a variable selected from a collection. Thus, any instance of the type PDUs will be one of eight alternative types. Note that each of the choices is labeled with a name. All of the PDUs

```
SNMPv2-PDU DEFINITIONS ::= BEGIN
PDUs ::= CHOICE {get-request          GetRequest-PDU,
                 get-next-request     GetNextRequest-PDU,
                 get-bulk-request     GetBulkRequest-PDU,
                 response             Response-PDU,
                 set-request          SetRequest-PDU,
                 inform-request       InformRequest-PDU,
                 snmp V2-trap         SNMPv2-Trap-PDU,
                 report               Report-PDU      }
—PDUs

GetRequest-PDU        ::=   [0] IMPLICIT PDU
GetNextRequest-PDU    ::=   [1] IMPLICIT PDU
Response-PDU          ::=   [2] IMPLICIT PDU
SetRequest-PDU        ::=   [3] IMPLICIT PDU
GetBulkRequest-PDU    ::=   [5] IMPLICIT BulkPDU
InformRequest-PDU     ::=   [6] IMPLICIT PDU
SNMPv2-Trap-PDU       ::=   [7] IMPLICIT PDU
Report-PDU            ::=   [8] IMPLICIT PDU

max-bindings INTEGER ::= 2147483647
```

FIGURE 19.3 SNMPv2 PDU format definitions. (*Continued on next page*)

```
PDU ::= SEQUENCE {request-id Integer32,
                  error-status INTEGER {                    —sometimes ignored
                          NoError (0),
                          tooBig (1),
                          noSuchName (2),                   —for proxy compatibility
                          badValue (3),                     —for proxy compatibility
                          readOnly (4),                     —for proxy compatibility
                          genError (5),
                          noAccess (6),
                          wrongType (7),
                          wrongLength (8),
                          wrongEncoding (9),
                          wrongValue (10),
                          noCreation (11),
                          inconsistentValue (12),
                          resourceUnavailable (13),
                          commitFailed (14),
                          undoFailed (15),
                          authorizationError (16),
                          notWritable (17),
                          inconsistentName (18)  },
                  error-Index INTEGER (0..max-bindings),    —sometimes ignored
                  variable-binding VarBindList  }           —values are sometimes ignored
BulkPDU ::= SEQUENCE {                                  —Must be identical in structure to PDU
                   request-id           Integer32,
                   non-repeaters        INTEGER (0..max-bindings),
                   max-repetitions      INTEGER (0..max-bindings),
                   variable-binding     VarBindList  }              --values are ignored

--variable binding

VarBind ::= SEQUENCE {name ObjectName,
    CHOICE {value                          ObjectSyntax,
           unspecified                     NULL,       --in retrieval requests
                                                  --exceptions in responses:
           noSuchObject [0]                IMPLICIT NULL,
           noSuchInstance [1]              IMPLICIT NULL,
           endOfMibView [2]                IMPLICIT NULL   }  }
--variable-binding list

VarBindList ::= SEQUENCE (SIZE (0..max-bindings)) OF VarBind

END
```

FIGURE 19.3 (*Continued*)

defined in this fashion have the same format but different labels, with the exception of GetBulkRequest-PDU. The format consists of a sequence of four elements. The second element, error-status, enumerates 18 possible integer values, each with a label. The last element, variable-binding, is defined as having syntax VarBindList, which is defined later in the same set of definitions.

The BulkPDU definition is also a sequence of four elements, but differs from the other PDUs.

VarBindList is defined as a Sequence-of construct consisting of some number of elements of syntax VarBind, with a size constraint of up to 2147483647, or $2^{31}-1$, elements. Each element, in turn, is a sequence of two values; the first is a name, and the second is a choice among five elements.

ASN.1 Macro Definitions

Included in the ASN.1 specification is the ASN.1 macro notation. This notation allows the user to extent the syntax of ASN.1 to define new types and their values. The subject of ASN.1 macros is a complex one, and this section serves only to introduce the subject.

Let us begin with several observations:

1. There are three levels that must be carefully distinguished:
   - The macro notation, used for defining macros.
   - A macro definition, expressed in the macro notation and used to define a set of macro instances.
   - A macro instance, generated from a macro definition by substituting values for variables.
2. A macro definition functions as a Super Type, generating a class of macro instances that function exactly like a basic ASN.1 type.[1]
3. A macro definition may be viewed as a template that is used to generate a set of related types and values.
4. The macro is used to extend the ASN.1 syntax but does not extend the encoding. Any type defined by means of a macro instance is simply an ASN.1 type, and is encoded in the usual manner.
5. In addition to the convenience of defining a set of related types, the macro definition enables the user to include semantic information with the type.

Macro Definition Format

A macro definition has the following general form:

```
<macroname> MACRO ::=
BEGIN
    TYPE NOTATION ::=  <new-type-syntax>
    VALUE NOTATION ::= <new-value-syntax>
    <supporting-productions>
END
```

The macroname is written in all uppercase letters. A new ASN.1 type is defined by writing the name of the type, which begins with a capital letter, followed by the macroname, followed by a definition of the type dictated by the form of the macro body.

The type and value notations, as well as the supporting productions, are all specified using Backus-Naur Form (BNF). The new-type-syntax describes the new type. The new-value-syntax describes the values of the new type. The supporting-

[1] The ASN.1 standard uses the term *basic ASN.1 type* to refer to any ASN.1 type that is not a macro instance.

productions provide any additional grammar rules for either the type or value syntax; that is, any nonterminals within the new-type-syntax and/or new-value-syntax are expanded in the supporting-productions.

When specific values are substituted for the variables or arguments of a macro definition, a macro instance is formed. This macro instance has two results. First, it generates a representation of a basic ASN.1 type, called the *returned type.* Second, it generates a representation of a generic ASN.1 value—that is, a representation of the set of values that the type may take. This generic value is called the *returned value.*

Macros Versus Defined Types

The ASN.1 macro facility provides tools for

1. Defining new types
2. Representing those types
3. Representing values of those types
4. Encoding specific values of those types

A similar capability already exists within ASN.1, which allows for the construction of defined types, either from built-in types or, recursively, from built-in types and defined types. The macro facility differs from the ASN.1 defined type capability in the following respects:

1. The macro facility allows the definition of a family of types. Each new type generated by a macro definition (a macro instance) is closely related to other types generated from the same macro. In contrast, there is no particular relationship between one basic ASN.1 defined type and other defined types.
2. A defined type is represented in a set way from the strings symbolizing the types from which it is constructed. A macro instance is represented in whatever way the writer of the macro chooses. Thus, the syntax of a type defined via macro instance can be chosen to correspond closely to the notation used within the particular application for which the macro was written. Furthermore, the macro instance may include commentary or semantic narrative. In this way, types defined by a macro may be more readable and more writeable.
3. In basic ASN.1, the representation of a value of a type is derived from the representation of the type in a relatively straightforward manner. The two representations are isomorphic; that is, they have similar or identical structure. This isomorphism is not required with a macro definition. The returned type and the returned generic value may have quite different syntaxes. Again, this allows for more readable and writable values.

SNMPv2 OBJECT-TYPE Macro

An example of a macro is illustrated in Figure 19.4. This macro is defined in the SNMPv2 standard, and is used to define management objects. A management object is an individual variable or item of management information stored in a man-

```
OBJECT-TYPE MACRO ::= BEGIN

TYPE NOTATION ::= "SYNTAX" Syntax
                  UnitsPart
                  "MAX-ACCESS" Access
                  "STATUS" Status
                  "DESCRIPTION" Text
                  ReferPart
                  IndexPart
                  DefValPart

VALUE NOTATION ::= value (VALUE ObjectName)
Syntax ::= type(ObjectSyntax) | "BITS" "{" Kibbles "}"
Kibbles ::= Kibble | Kibbles "," Kibble
Kibble ::= identifier "(" nonNegativeNumber ")"
UnitsPart ::= "UNITS" Text | empty
Access ::= "not-accessible" | "accessible-for-notify" | "read-only" | "read-write" | "read-write" | "read-create"
Status ::= "current" | "deprecated" | "obsolete"
ReferPart ::= "REFERENCE" Text | empty
IndexPart ::= "INDEX" "{" IndexTypes "}" | "AUGMENTS" "{" Entry "}" | empty
IndexTypes ::= IndexType | IndexTypes "," IndexType
IndexTypes ::= "IMPLIED" Index | Index
Index ::= value (indexobject ObjectName)     --use the SYNTAX value of the
                                             --correspondent OBJECT-TYPE invocation
Entry ::= value (entryobject ObjectName)     --use the INDEX value of the
                                             --correspondent OBJECT-TYPE invocation
DefValPart ::= "DEFVAL" "{" value (Defval ObjectSyntax) "}" | empty
—uses the NVT ASCII character set
Text ::= """string"""

END
```

FIGURE 19.4 SNMPv2 macro for object definition.

agement information base. Thus, this is an example of the use of ASN.1 to specify the syntax of locally maintained data that is used by a distributed application.

The type notation portion of this macro consists of eight parts:

- The SYNTAX clause specifies the ASN.1 syntax type of this object. ObjectSyntax defines a subset of ASN.1 types that may be used to define managed objects; its definition is not shown.
- The UNITS clause can be used to specify what units a numerical value has.
- The MAX-ACCESS clause is used to specify the access privileges for this object. Some objects are not accessible by remote users, but are used by the local system for data-structuring purposes. Other objects may be read-only, read-write, or read-create.
- The STATUS clause indicates the current status of this object with respect to whether it is part of the most recent standard.
- The DESCRIPTION clause is used to provide a textual description of the object.
- The REFERENCE clause is used to provide a cross-reference to another portion of the management information base.

- The INDEX clause is used if this object refers to a tabular structure. The clause lists the object or objects that serve as indexes into the table.
- The AUGMENTS clause is used to specify that this object refers to a tabular structure that is to be appended to another tabular structure.
- The DEFVAL clause is used to provide a default value for this object, to be supplied upon creation of the object by the local system.

The value of an object is an identifier, ObjectName, that provides a unique reference for the object.

An example of the use of this macro is the following SNMPv2 object definition:

```
ifTestResult  OBJECT-TYPE
   SYNTAX   INTEGER {
                 none(1),  -- no test yet requested
                 success(2),
                 inProgress(3),
                 notSupported(4),
                 unAbleToRun(5),  -- due to state of system
                 aborted(6),
                 failed(7)
               }
   MAX-ACCESS  read-only
   STATUS   current
   DESCRIPTION
         "This object contains the result of the most recently
         requested test, or the value none(1) if no tests have
         been requested since the last reset. Note that this
         facility provides no provision for saving the results
         of one test when starting another, as could be
         required if used by multiple managers concurrently."
   ::= { ifTestEntry 4 }
```

19.2 NETWORK MANAGEMENT—SNMPv2

Networks and distributed processing systems are of critical and growing importance in business, government, and other organizations. Within a given organization, the trend is toward larger, more complex networks supporting more applications and more users. As these networks grow in scale, two facts become painfully evident:

- The network and its associated resources and distributed applications become indispensable to the organization.
- More things can go wrong, disabling the network or a portion of the network, or degrading performance to an unacceptable level.

A large network cannot be put together and managed by human effort alone. The complexity of such a system dictates the use of automated network management tools. The urgency of the need for such tools is increased, and the difficulty of supplying such tools is also increased, if the network includes equipment from multiple vendors. In response, standards that deal with network management have been developed, covering services, protocols, and management information base.

This section begins with an introduction to the overall concepts of standardized network management, the remainder is devoted to a discussion of SNMPv2, which is an extension of SNMP, the most widely used network management standard.

Network Management Systems

A network management system is a collection of tools for network monitoring and control that is integrated in the following senses:

- A single operator interface with a powerful but user-friendly set of commands for performing most or all network management tasks.
- A minimal amount of separate equipment. That is, most of the hardware and software required for network management are incorporated into the existing user equipment.

A network management system consists of incremental hardware and software additions implemented among existing network components. The software used in accomplishing the network management tasks resides in the host computers and communications processors (e.g., front-end processors, terminal cluster controllers). A network management system is designed to view the entire network as a unified architecture, with addresses and labels assigned to each point and the specific attributes of each element and link known to the system. The active elements of the network provide regular feedback of status information to the network control center.

A network management system incorporates the following key elements:

- Management station, or manager
- Agent
- Management information base
- Network management protocol

The *management station* is typically a stand-alone device, but may be a capability implemented on a shared system. In either case, the management station serves as the interface for the human network manager into the network management system. The management station will have, at minimum,

- A set of management applications for data analysis, fault recovery, and so on.
- An interface by which the network manager may monitor and control the network.

- The capability of translating the network manager's requirements into the actual monitoring and control of remote elements in the network.
- A database of network management information extracted from the databases of all the managed entities in the network.

The other active element in the network management system is the *agent.* Key platforms, such as hosts, bridges, routers, and hubs, may be equipped with agent software so that they may be managed from a management station. The agent responds to requests for information from a management station, responds to requests for actions from the management station, and may asynchronously provide the management station with important but unsolicited information.

The means by which resources in the network may be managed entails representing these resources as objects. Each object is, essentially, a data variable that represents one aspect of the managed agent. The collection of objects is referred to as a *management information base* (MIB). The MIB functions as a collection of access points at the agent for the management station. These objects are standardized across systems of a particular class (e.g., bridges all support the same management objects). A management station performs the monitoring function by retrieving the value of MIB objects. Also, a management station can cause an action to take place at an agent, or it can change the configuration settings of an agent by modifying the value of specific variables.

The management station and agents are linked by a *network management protocol.* The protocol used for the management of TCP/IP networks is the simple network management protocol (SNMP). For OSI-based networks, the common management information protocol (CMIP) is being developed. An enhanced version of SNMP, known as SNMPv2, is intended for both TCP/IP- and OSI-based networks. Each of these protocols includes the following key capabilities:

- **Get:** Enables the management station to retrieve the value of objects at the agent.
- **Set:** Enables the management station to set the value of objects at the agent.
- **Notify:** Enables an agent to notify the management station of significant events.

In a traditional centralized network management scheme, one host in the configuration has the role of a network management station; there may be possibly one or two other management stations in a backup role. The remainder of the devices on the network contain agent software and a MIB, to allow monitoring and control from the management station. As networks grow in size and traffic load, such a centralized system is unworkable. Too much burden is placed on the management station, and there is too much traffic, with reports from every single agent having to wend their way across the entire network to headquarters. In such circumstances, a decentralized, distributed approach works best (e.g., Figure 19.5). In a decentralized network management scheme, there may be multiple top-level management stations, which might be referred to as management servers. Each such server might directly manage a portion of the total pool of agents. However, for many of the

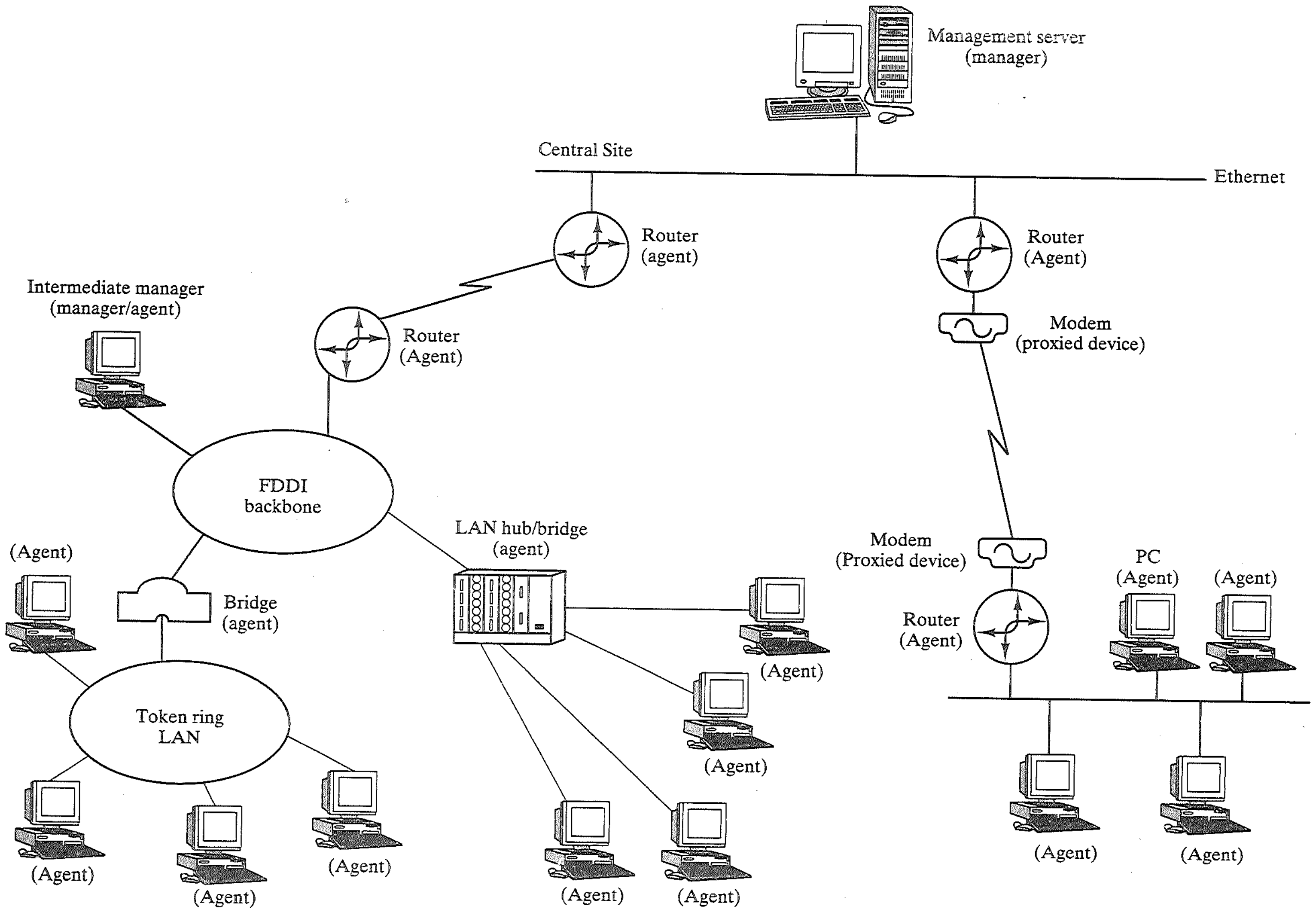


FIGURE 19.5 Example distributed network management configuration.

agents, the management server delegates responsibility to an intermediate manager, which plays the role of manager to monitor and control the agents under its responsibility; it also plays an agent role to provide information and accept control from a higher-level management server. This type of architecture spreads the processing burden and reduces total network traffic.

Simple Network Management Protocol Version 2 (SNMPv2)

In August of 1988, the specification for SNMP was issued and rapidly became the dominant network management standard. A number of vendors offer stand-alone network management workstations based on SNMP, and most vendors of bridges, routers, workstations, and PCs offer SNMP agent packages that allow their products to be managed by an SNMP management station.

As the name suggests, SNMP is a simple tool for network management. It defines a limited, easily implemented management information base (MIB) of scalar variables and two-dimensional tables, and it defines a streamlined protocol to enable a manager to get and set MIB variables and to enable an agent to issue unsolicited notifications, called **traps**. This simplicity is the strength of SNMP; it is easily implemented and consumes modest processor and network resources. Also, the structure of the protocol and the MIB are sufficiently straightforward that it is not difficult to achieve interoperability among management stations and agent software from a mix of vendors.

With its widespread use, the deficiencies of SNMP became increasingly apparent: functional deficiencies and lack of a security facility. As a result, an enhanced version, known as SNMPv2, was issued in 1993, with a revised version issued in 1996 (RFCs). SNMPv2 has quickly gained support, and a number of vendors announced products within months of the issuance of the standard.

The Elements of SNMPv2

Surprisingly, SNMPv2 does not provide network management at all! SNMPv2 instead provides a framework on which network management applications can be built. Those applications, such as fault management, performance monitoring, accounting, and so on, are outside the scope of the standard.

What SNMPv2 does provide is, to use a contemporary term, the infrastructure for network management. Figure 19.6 is an example of a configuration that illustrates that infrastructure.

The essence of SNMPv2 is a protocol that is used to exchange management information. Each "player" in the network management system maintains a local database of information relevant to network management, known as the management information base. The SNMPv2 standard defines the structure of this information and the allowable data types; this definition is known as the structure of management information (SMI); we can think of this as the language for defining management information. The standard also supplies a number of MIBs that are

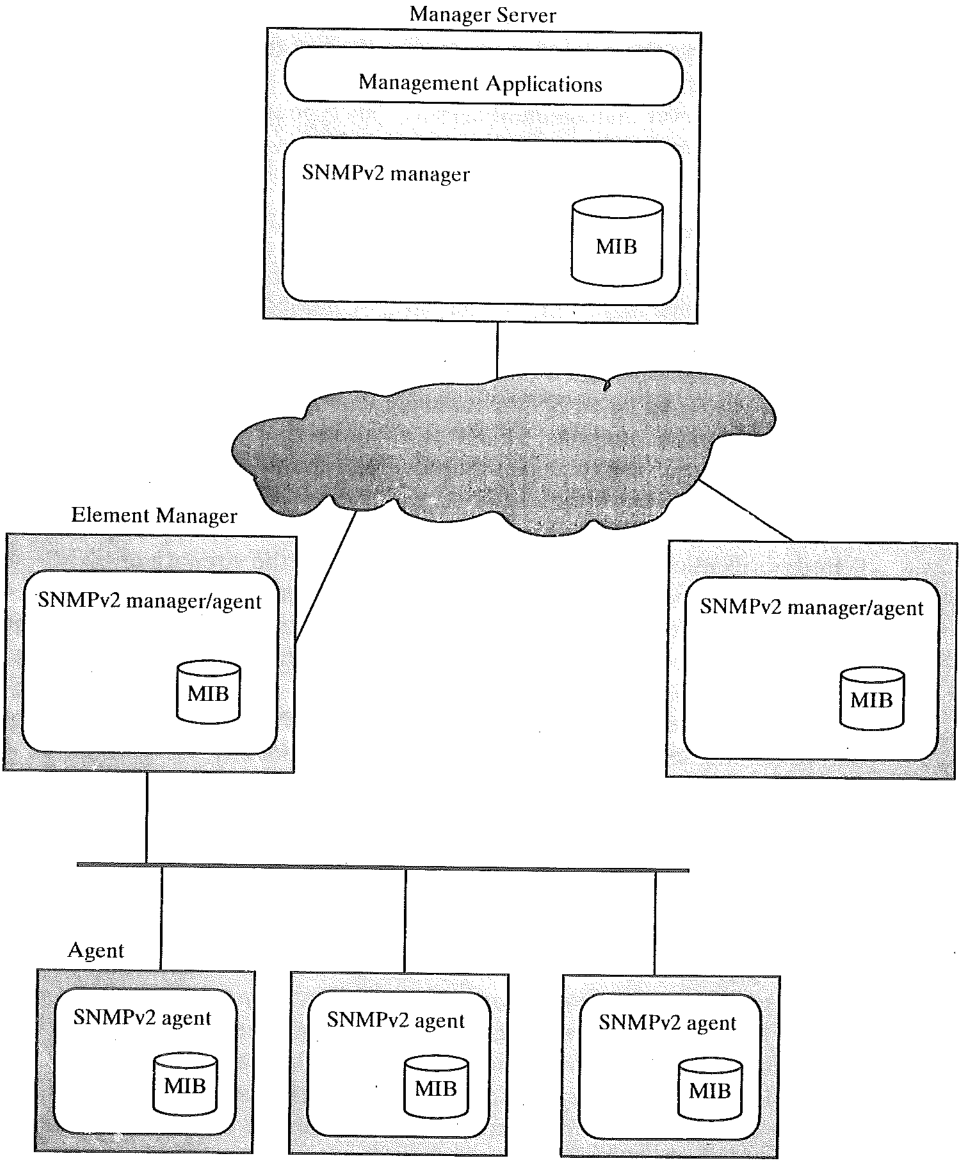


FIGURE 19.6 SNMPv2-managed configuration.

generally useful for network management.[2] In addition, new MIBs may be defined by vendors and user groups.

[2] There is some ambiquity implicit in the term MIB. In its singular form, the term can be used to refer to the entire database of management information at either a manager or an agent. It can also be used in singular or plural form to refer to a specific defined collection of management information that is part of an overall MIB. Thus, the SNMPv2 standard includes the definition of several MIBs and incorporates, by reference, MIBs defined in SNMPv1.

At least one system in the configuration must be responsible for network management. It is here that any network management applications are housed. There may be more than one of these management stations, to provide redundancy or simply to split up the duties in a large network. Most other systems act in the role of agent. An agent collects information locally and stores it for later access by a manager. The information includes data about the system itself and may also include traffic information for the network or networks to which the agent attaches.

SNMPv2 will support either a highly centralized network management strategy or a distributed one. In the latter case, some systems operate both in the role of manager and of agent. In its agent role, such a system will accept commands from a superior management system. Some of those commands relate to the local MIB at the agent. Other commands require the agent to act as a proxy for remote devices. In this case, the proxy agent assumes the role of manager to access information at a remote agent, and then assumes the role of an agent to pass that information on to a superior manager.

All of these exchanges take place using the SNMPv2 protocol, which is of the simple request/response type. Typically, SNMPv2 is implemented on top of the user datagram protocol (UDP), which is part of the TCP/IP protocol suite; it can also be implemented on top of the ISO transport protocol.

Structure of Management Information

The SMI defines the general framework within which an MIB can be defined and constructed. The SMI identifies the data types that can be used in the MIB, and how resources within the MIB are represented and named. The philosophy behind SMI is to encourage simplicity and extensibility within the MIB. Thus, the MIB can store only simple data types: scalars and two-dimensional arrays of scalars, called *tables*. The SMI does not support the creation or retrieval of complex data structures. This philosophy is in contrast to that used with OSI-systems management, which provides for complex data structures and retrieval modes to support greater functionality. SMI avoids complex data types and structures to simplify the task of implementation and to enhance interoperability. MIBs will inevitably contain vendor-created data types and, unless tight restrictions are placed on the definition of such data types, interoperability will suffer.

There are actually three key elements in the SMI specification. At the lowest level, the SMI specifies the data types that may be stored. Then, the SMI specifies a formal technique for defining objects and tables of objects. Finally, the SMI provides a scheme for associating a unique identifier with each actual object in a system, so that data at an agent can be referenced by a manager.

Table 19.4 shows the data types that are allowed by the SMI. This is a fairly restricted set of types. For example, real numbers are not supported; however, it is rich enough to support most network management requirements.

The SNMPv2 specification includes a template, known as an ASN.1 (Abstract Syntax Notation One) macro, which provides the formal model for defining objects. That template was illustrated in Figure 19.4. Figure 19.7 is an example of how this template is used to define objects and tables of objects.

TABLE 19.4 Allowable data types in SNMPv2.

Data Type	Description
INTEGER	Integers in the range of -2^{21} to $2^{31} - 1$.
UInteger32	Integers in the range of 0 to $2^{32} - 1$.
Counter32	A non-negative integer which may be incremented modulo 2^{32}.
Counter64	A non-negative integer which may be incremented modulo 2^{64}.
Gauge 32	A non-negative integer which may increase or decrease, but shall not exceed a maximum value. The maximum value can not be greater than $2^{32} - 1$.
TimeTicks	A non-negative integer which represents the time, modulo 2^{32}, in hundredths of a second.
OCTET STRING	Octet strings for arbitrary binary or texual data; may be limited to 255 octets.
IpAddress	A 32-bit internet address.
Opaque	An arbitrary bit field.
BIT STRING	An enumeration of named bits.
OBJECT IDENTIFIER	Administratively assigned name to object or other standardized element. Value is a sequence of up to 128 non-negative integers.

The first three productions serve to define a table, grokTable, stored at an agent. As with all SNMPv2 tables, grokTable is organized as a sequence of rows, or entries, each of which has the same sequence of objects; in this case, each row consists of four objects. The INDEX clause specifies that the object grokIndex serves as an index into the table; each row of the table will have a unique value of grokIndex.

The access type of grokIPAddress is read-create, which means that the object is read-write and that the object may be assigned a value by a manager at the time that the row containing this object is created by a manager. Each row of the table maintains a counter for the number of grok packets sent to the grokIPAddress specified for that row. The grokCount object is read-only; its value cannot be altered by a manager but is maintained by the agent within which this table resides. The grokStatus object is used in the process of row creation and deletion; for which the algorithm is rather complex. In essence, a RowStatus type of object is used to keep track of the state of a row during the process of creation and deletion.

Each object definition includes a value, which is a unique identifier for that object; for example, the value for grokEntry is {grokTable 1}, which means that the identifier for grokEntry is the concatenation of the identifier for grokTable and 1. The objects in a MIB are organized in a tree structure, and the identifier of an object is found by walking the tree from its root to the position of the object in that tree structure. For scalar objects, this scheme provides a unique identifier for any given object instance. For objects in tables, there is one instance of each object for

```
grokTable OBJECT-TYPE
  SYNTAX   SEQUENCE OF GrokEntry
  MAX-ACCESS     not-accessible
  STATUS         current
  DESCRIPTION
    "The (conceptual) grok table."
  ::= { adhocGroup 2 }

grokEntry OBJECT-TYPE
  SYNTAX         GrokEntry
  MAX-ACCESS     not-accessible
  STATUS         current
  DESCRIPTION
    "An entry (conceptual row) in the
    grok table."
  INDEX          { grokIndex }
  ::= { grokTable }

GrokEntry ::= SEQUENCE {
     grokIndex              INTEGER,
     grokIPAddress          IpAddress,
     grokValue              Counter32,
     grokStatus             RowStatus }

grokIndex OBJECT-TYPE
  SYNTAX         INTEGER
  MAX-ACCESS     not-accessible
  STATUS         current
  DESCRIPTION
    "The auxiliary variable used for
    identify instances of the columnar
    objects in the grok table."
  ::= { grokEntry 1 }

grokIPAddress OBJECT-TYPE
  SYNTAX         IpAddress
  MAX-ACCESS     read-create
  STATUS         current
  DESCRIPTION
    "The Ip address to send grok packets
    to."
  ::= { grokEntry 2 }

grokCount OBJECT-TYPE
  SYNTAX         Counter32
  MAX-ACCESS     read-only
  STATUS         current
  DESCRIPTION
    "The total number of grok packets
    sent so far."
  DEFVAL { 0 }
  ::= { grokEntry 3 }

grokStatus OBJECT-TYPE
  SYNTAX         RowStatus
  MAX-ACCESS     read-create
  STATUS         current
  DESCRIPTION
    "The status object used for creating,
    modifying, and deleting a conceptual
    row instance in the grok table."
  DEFVAL { active }
  ::= { grokEntry 4 }
```

FIGURE 19.7 An example of an SNMPv2 table.

each row of the table, so a further qualification is needed; what is done is to concatenate the value of the INDEX object to the identifier of each object in the table.

Protocol Operation

The heart of the SNMPv2 framework is the protocol itself. The protocol provides a straightforward, basic mechanism for the exchange of management information between manager and agent.

The basic unit of exchange is the message, which consists of an outer message wrapper and an inner protocol data unit (PDU). The outer message header deals with security and is discussed later in this section.

Eight types of PDUs may be carried in an SNMP message. The general formats for these are illustrated informally in Figure 19.8; the formal ASN.1 definition was provided in Figure 19.3. Several fields are common to a number of PDUs. The request-id field is an integer assigned such that each outstanding request can be uniquely identified. This process enables a manager to correlate incoming responses

PDU Type	request-id	0	0	variable-bindings

(a) GetRequest-PDU, GetNextRequest-PDU, SetRequest-PDU, SNMPv2-Trap-PDU, InformRequest-PDU

PDU Type	request-id	error-status	error-index	variable-bindings

(b) Response-PDU

PDU Type	request-id	non-repeaters	max-repetitions	variable-bindings

(c) GetBulkRequest-PDU

name1	value1	name2	value2	. . .	name*n*	value*n*

(d) variable-bindings

FIGURE 19.8 SNMPv2 PDU formats.

with outstanding requests; it also enables an agent to cope with duplicate PDUs generated by an unreliable transport service. The variable-bindings field contains a list of object identifiers; depending on the PDU, the list may also include a value for each object.

The GetRequest-PDU, issued by a manager, includes a list of one or more object names for which values are requested. If the get operation is successful, then the responding agent will send a Response-PDU. The variable-bindings list will contain the identifier and value of all retrieved objects. For any variables that are not in the relevant MIB view, its identifier and an error code are returned in the variable-bindings list. Thus, SNMPv2 permits partial responses to a GetRequest, which is a significant improvement over SNMP. In SNMP, if one or more of the variables in a GetRequest is not supported, the agent returns an error message with a status of noSuchName. In order to cope with such an error, the SNMP manager must either return no values to the requesting application, or it must include an algorithm that responds to an error by removing the missing variables, resending the request, and then sending a partial result to the application.

The GetNextRequest-PDU also is issued by a manager and includes a list of one or more objects. In this case, for each object named in the variable-bindings field, a value is to be returned for the object that is next in lexicographic order, which is equivalent to saying next in the MIB in terms of its position in the tree structure of object identifiers. As with the GetRequest-PDU, the agent will return values for as many variables as possible. One of the strengths of the GetNextRequest-PDU is that it enables a manager entity to discover the structure of an MIB view dynamically—a useful ability if the manager does not know *a priori* the set of objects that are supported by an agent or that are in a particular MIB view.

One of the major enhancements provided in SNMPv2 is the GetBulkRequest PDU. The purpose of this PDU is to minimize the number of protocol exchanges required to retrieve a large amount of management information. The GetBulkRequest PDU allows an SNMPv2 manager to request that the response be as large as possible given the constraints on message size.

The GetBulkRequest operation uses the same selection principle as the GetNextRequest operation; that is, selection is always of the next-object instance in lexicographic order. The difference is that, with GetBulkRequest, it is possible to specify that multiple lexicographic successors be selected.

In essence, the GetBulkRequest operation works in the following way. The GetBulkRequest includes a list of $(N + R)$ variable names in the variable-bindings list. For each of the first N names, retrieval is done in the same fashion as for GetNextRequest. That is, for each variable in the list, the next variable in lexicographic order, plus its value, is returned; if there is no lexicographic successor, then the named variable and a value of endOfMibView are returned. For each of the last R names, multiple lexicographic successors are returned.

The GetBulkRequest PDU has two fields not found in the other PDUs: non-repeaters and max-repetitions. The non-repeaters field specifies the number of variables in the variable-binding list for which a single lexicographic successor is to be returned. The max-repetitions field specifies the number of lexicographic successors to be returned for the remaining variables in the variable binding list. To explain the algorithm, let us define the following:

L = number of variable names in the variable-bindings field of the GetBulkRequest PDU

N = the number of variables, starting with the first variable in the variable-bindings field, for which a single lexicographic successor is requested

R = the number of variables, following the first N variables, for which multiple lexicographic successors are requested

M = the number of lexicographic successors requested for each of the last R variables

The following relationships hold:

$$N = \text{MAX}[\,\text{MIN}(\text{non-repeaters}, L), 0]$$
$$M = \text{MAX}[\,\text{max-repetitions}, 0\,]$$
$$R = L - N$$

The effect of the MAX operator is that if the value of either non-repeaters or max-repetitions is less than zero, a value of 0 is substituted.

If N is greater than 0, then the first N variables are processed as for GetNextRequest. If R is greater than 0 and M is greater than 0, then for each of the last R variables in the variable bindings list, the M lexicographic successors are retrieved. That is, for each variable,

- Obtain the value of the lexicographic successor of the named variable.
- Obtain the value of the lexicographic successor to the object instance retrieved in the previous step.

- Obtain the value of the lexicographic successor to the object instance retrieved in the previous step.
- And so on, until M object instances have been retrieved.

If, at any point in this process, there is no lexicographic successor, then the endOfMibView value is returned, paired with the name of the last lexicographic successor or, if there were no successors, with the name of the variable in the request.

Using these rules, the total number of variable-binding pairs that can be produced is $N + (M \times R)$. The order in which the last $(M \times R)$ of these variable-binding pairs are placed in the Response PDU can be expressed as follows:

```
for $i$ := 1 to $M$ do
    for $r$ := 1 to $R$ do
        retrieve $i$-th successor of $(N + r)$-th variable;
```

The effect of this definition is that the successors to the last R variables are retrieved row by row, rather than retrieving all of the successors to the first variable, followed by all of the successors to the second variable, and so on; this matches with the way in which conceptual tables are lexicographically ordered, so that if the last R values in the GetBulkRequest are columnar objects of the same table, then the Response will return conceptual rows of the table.

The GetBulkRequest operation removes one of the major limitations of SNMP, which is its inability to efficiently retrieve large blocks of data. Moreover, this use of this operator can actually enable reducing the size of management applications that are supported by the management protocol, realizing further efficiencies. There is no need for the management application to concern itself with some of the details of packaging requests. It need not perform a trial-and-error procedure to determine the optimal number of variable bindings to put in a request-PDU. Also, if a request is too big, even for GetBulkRequest, the agent will send back as much data as it can, rather than simply sending a tooBig error message. Thus, the manager simply has to retransmit the request for the missing data; it does not have to figure out how to repackage the original request into a series of smaller requests.

The SetRequest-PDU is issued by a manager to request that the values of one or more objects be altered. The receiving SNMPv2 entity responds with a Response-PDU containing the same request-id. The SetRequest operation is atomic: either all of the variables are updated or none are. If the responding entity is able to set values for all of the variables listed in the incoming variable-bindings list, then the Response-PDU includes the variable-binding field, with a value supplied for each variable. If at least one of the variable values cannot be supplied, then no values are returned, and no values are updated. In the latter case, the error-status code indicates the reason for the failure, and the error-index field indicates the variable in the variable-bindings list that caused the failure.

The SNMPv2-Trap-PDU is generated and transmitted by a SNMPv2 entity acting in an agent role when an unusual event occurs. It is used to provide the management station with an asynchronous notification of some significant event. The variable-bindings list is used to contain the information associated with the trap message. Unlike the GetRequest, GetNextRequest, GetBulkRequest, SetRequest, and InformRequest PDUs, the SNMPv2-Trap-PDU does not elicit a response from the receiving entity; it is an unconfirmed message.

The InformRequest-PDU is sent by an SNMPv2 entity acting in a manager role, on behalf of an application, to another SNMPv2 entity acting in a manager role, to provide management information to an application using the latter entity. As with the SNMPv2-Trap-PDU, the variable-binding field is used to convey the associated information. The manager receiving an InformRequest acknowledges receipt with a Response-PDU.

For both the SNMPv2-Trap and the InformRequest, various conditions can be defined that indicate when the notification is generated; and the information to be sent is also specified.

19.3 ELECTRONIC MAIL—SMTP AND MIME

The most heavily used application in virtually any distributed system is electronic mail. From the start, the Simple Mail Transfer Protocol (SMTP) has been the workhorse of the TCP/IP protocol suite. However, SMTP has traditionally been limited to the delivery of simple text messages. In recent years, there has been a demand for the delivery mail to be able to contain various types of data, including voice, images, and video clips. To satisfy this requirement, a new electronic mail standard, which builds on SMTP, has been defined: the Multi-Purpose Internet Mail Extension (MIME). In this section, we first examine SMTP, then look at MIME.

Simple Mail Transfer Protocol (SMTP)

SMTP is the standard protocol for transferring mail between hosts in the TCP/IP protocol suite; it is defined in RFC 821.

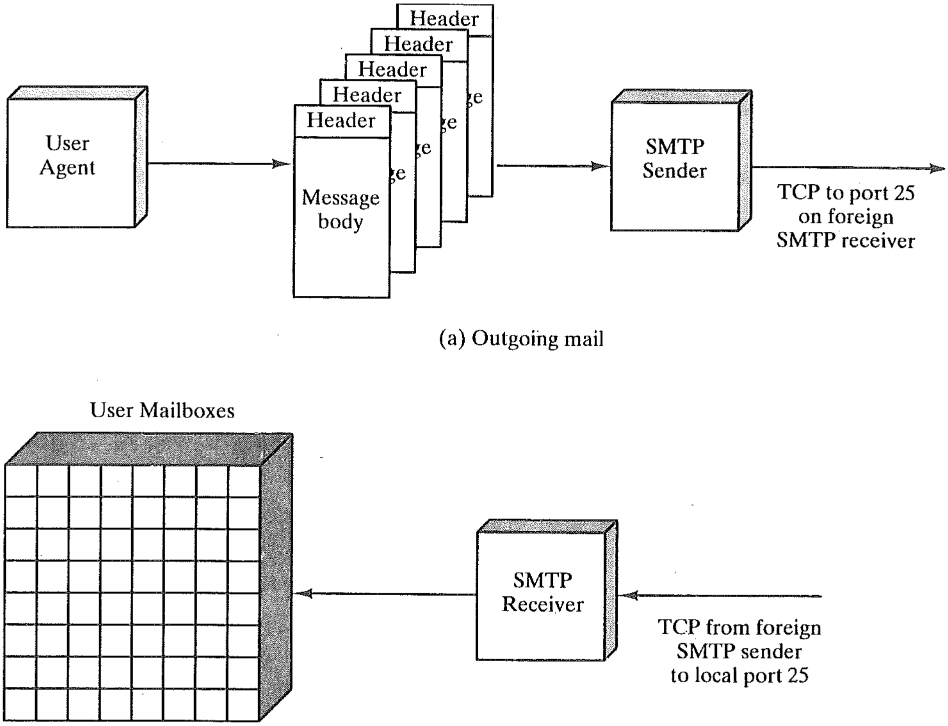
Although messages transferred by SMTP usually follow the format defined in RFC 822, described later, SMTP is not concerned with the format or content of messages themselves, with two exceptions. This concept is often expressed by saying that SMTP uses information written on the **envelope** of the mail (message header), but does not look at the **contents** (message body) of the envelope. The two exceptions are

1. SMTP standardizes the message character set as 7-bit ASCII.
2. SMTP adds log information to the start of the delivered message that indicates the path the message took.

Basic Electronic Mail Operation[3]

Figure 19.9 illustrates the overall flow of mail in a typical system. Although much of this activity is outside the scope of SMTP, the figure illustrates the context within which SMTP typically operates.

[3] The discussion in this section is based partly on a description by Paul Mockapetris, published in [STAL90].



(a) Outgoing mail

FIGURE 19.9 SMTP mail flow.

To begin, mail is created by a user-agent program in response to user input. Each created message consists of a header that includes the recipient's email address and other information, and a body containing the message to be sent. These messages are then queued in some fashion and provided as input to an SMTP Sender program, which is typically an always-present server program on the host.

Although the structure of the outgoing-mail queue will differ depending on the host's operating system, each queued message, conceptually, has two parts:

1. The message text, consisting of
   - The RFC 822 header: the message envelope, which includes an indication of the intended recipient or recipients.
   - The body of the message, composed by the user.
2. A list of mail destinations.

The list of mail destinations for the message is derived by the user agent from the 822 message header. In some cases, the destination, or destinations, are literally specified in the message header. In other cases, the user agent may need to expand mailing list names, remove duplicates, and replace mnemonic names with actual mailbox names. If any blind carbon copies (BCC) are indicated, the user agent needs to prepare messages that conform to this requirement. The basic idea is that the multiple formats and styles preferred by humans in the user interface are replaced by a standardized list suitable for the SMTP send program.

The **SMTP sender** takes messages from the outgoing mail queue and transmits them to the proper destination host via SMTP transactions over one or more TCP connections to port 25 on the target host's. A host may have multiple SMTP senders active simultaneously if it has a large volume of outgoing mail, and it should also have the capability of creating SMTP receivers on demand so that mail from one host cannot delay mail from another.

Whenever the SMTP sender completes delivery of a particular message to one or more users on a specific host, it deletes the corresponding destinations from that message's destination list. When all destinations for a particular message are processed, the message is deleted from the queue. In processing a queue, the SMTP sender can perform a variety of optimization. If a particular message is sent to multiple users on a single host, the message text need be sent only once. If multiple messages are ready to send to the same host, the SMTP sender can open a TCP connection, transfer the multiple messages, and then close the connection, rather than opening and closing a connection for each message.

The SMTP sender must deal with a variety of errors. The destination host may be unreachable, out of operation, or the TCP connection may fail while mail is being transferred. The sender can requeue the mail for later delivery, but give up after some period rather then keep the message in the queue indefinitely. A common error is a faulty destination address, which can occur due to user-input error or because the intended destination user has a new address on a different host. The SMTP sender must either redirect the message, if possible, or return an error notification to the message's originator.

The **SMTP protocol** is used to transfer a message from the SMTP sender to the SMTP receiver over a TCP connection. SMTP attempts to provide reliable operation but does not guarantee to recover from lost messages. No end-to-end acknowledgment is returned to a message's originator that a message is successfully delivered to the message's recipient, and error indications are not guaranteed to be returned either. However, the SMTP-based mail system is generally considered reliable.

The **SMTP receiver** accepts each arriving message and either places it in the appropriate user mailbox or copies it to the local outgoing mail queue if forwarding is required. The SMTP receiver must be able to verify local mail destinations and deal with errors, including transmission errors and lack of disk file capacity.

The SMTP sender is responsible for a message up to the point where the SMTP receiver indicates that the transfer is complete; however, this simply means that the message has arrived at the SMTP receiver, not that the message has been delivered to and retrieved by the intended recipient. The SMTP receiver's error-handling responsibilities are generally limited to giving up on TCP connections that fail or are inactive for very long periods. Thus, the sender has most of the error-recovery responsibility. Errors during completion indication may cause duplicate, but not lost, messages.

In most cases, messages go directly from the mail originator's machine to the destination machine over a single TCP connection. However, mail will occasionally go through intermediate machines via an SMTP forwarding capability, in which case the message must traverse multiple TCP connections between source and destination; one way for this to happen is for the sender to specify a route to the destination in the form of a sequence of servers. A more common event is forwarding required because a user has moved.

It is important to note that the SMTP protocol is limited to the conversation that takes place between the SMTP sender and the SMTP receiver. SMTP's main function is the transfer of messages, although there are some ancillary functions dealing with mail destination verification and handling. The rest of the mail-handling apparatus depicted in Figure 19.9 is beyond the scope of SMTP and may differ from one system to another.

We now turn to a discussion of the main elements of SMTP.

SMTP Overview

The operation of SMTP consists of a series of commands and responses exchanged between the SMTP sender and receiver. The initiative is with the SMTP sender, who establishes the TCP connection. Once the connection is established, the SMTP sender sends commands over the connection to the receiver. Each command generates exactly one reply from the SMTP receiver.

Table 19.5 lists the **SMTP commands**. Each command consists of a single line of text, beginning with a four-letter command code followed in some cases by an argument field. Most replies are a single line, although multiple-line replies are possible. The table indicates those commands that all receivers must be able to recognize. The other commands are optional and may be ignored by the receiver.

TABLE 19.5 SMTP commands

Name	Command Form	Description
HELO	HELO <SP> <domain> <CRLF>	Send identification
MAIL	MAIL <SP> FROM:<reverse-path> <CRLF>	Identifies originator of mail
RCPT	RCPT <SP> TO:<forward-path> <CRLF>	Identifies recipient of mail
DATA	DATA <CRLF>	Transfer message text
RSET	RSET <CRLF>	Abort current mail transaction
NOOP	NOOP <CRLF>	No operation
QUIT	QUIT <CRLF>	Close TCP connection
SEND	SEND <SP> FROM:<reverse-path> <CRLF>	Send mail to terminal
SOML	SOML <SP> FROM:<reverse-path> <CRLF>	Send mail to terminal if possible; otherwise to mailbox
SAML	SAML <SP> FROM:<reverse-path> <CRLF>	Send mail to terminal and mailbox
VRFY	VRFY <SP> <string> <CRLF>	Confirm user name
EXPN	EXPN <SP> <string> <CRLF>	Return membership of mailing list
HELP	HELP [<SP> <string>] <CRLF>	Send system-specific documentation
TURN	TURN <CRLF>	Reverse role of sender and receiver

<CRLF> = carriage return, line feed
<SP> = space
square brackets denote optional elements
shaded commands are optional in a conformant SMTP implementation

TABLE 19.6 SMTP replies

Code	Description
	Positive Completion Reply
211	System status, or system help reply
214	Help message (Information on how to use the receiver or the meaning of a particular non-standard command; this reply is useful only to the human user)
220	<domain> Service ready
221	<domain> Service closing transmission channel
250	Requested mail action okay, completed
251	User not local; will forward to <forward-path>
	Positive Intermediate Reply
354	Start mail input; end with <CRLF>.<CRLF>
	Transient Negative Completion Reply
421	<domain> Service not available, losing transmission channel (This may be a reply to any command if the service knows it must shut down)
450	Requested mail action not taken: mailbox unavailable (e.g., mailbox busy)
451	Requested action aborted: local error in processing
452	Requested action not taken: insufficient system storage
	Permanent Negative Completion Reply
500	Syntax error, command unrecognized (This may include errors such as command line too long)
501	Syntax error in parameters or arguments
502	Command not implemented
503	Bad sequence of commands
504	Command parameter not implemented
550	Requested action not taken: mailbox unavailable (e.g., mailbox not found, no access)
551	User not local; please try <forward-path>
552	Requested mail action aborted: exceeded storage allocation
553	Requested action not taken: mailbox name not allowed (e.g., mailbox syntax incorrect)
554	Transaction failed

SMTP replies are listed in Table 19.6. Each reply begins with a three-digit code and may be followed by additional information. The leading digit indicates the category of the reply:

- **Positive Completion reply.** The requested action has been successfully completed. A new request may be initiated.
- **Positive Intermediate reply.** The command has been accepted, but the requested action is being held in abeyance, pending receipt of further information. The sender-SMTP should send another command specifying this information. This reply is used in command sequence groups.
- **Transient Negative Completion reply.** The command was not accepted, and the requested action did not occur. However, the error condition is temporary and the action may be requested again.

- **Permanent Negative Completion reply.** The command was not accepted and the requested action did not occur.

Basic SMTP operation occurs in three phases: connection setup, exchange of one or more command-response pairs, and connection termination. We examine each phase in turn.

Connection Setup

An SMTP sender will attempt to set up a TCP connection with a target host when it has one or more mail messages to deliver to that host. The sequence is quite simple:

1. The sender opens a TCP connection with the receiver.
2. Once the connection is established, the receiver identifies itself with "220 Service Ready".
3. The sender identifies itself with the HELO command.
4. The receiver accepts the sender's identification with "250 OK".

If the mail service on the destination is unavailable, the destination host returns a "421 Service Not Available" reply in step 2, and the process is terminated.

Mail Transfer

Once a connection has been established, the SMTP sender may send one or more messages to the SMTP receiver. There are three logical phases to the transfer of a message:

1. A MAIL command identifies the originator of the message.
2. One or more RCPT commands identify the recipients for this message.
3. A DATA command transfers the message text.

The **MAIL command** gives the reverse-path, which can be used to report errors. If the receiver is prepared to accept messages from this originator, it returns a "250 OK" reply. Otherwise, the receiver returns a reply indicating failure to execute the command (codes 451, 452, 552), or an error in the command (codes 421, 500, 501).

The **RCPT command** identifies an individual recipient of the mail data; multiple recipients are specified by multiple use of this command. A separate reply is returned for each RCPT command, with one of the following possibilities:

1. The receiver accepts the destination with a 250 reply; this indicates that the designated mailbox is on the receiver's system.
2. The destination will require forwarding, and the receiver will forward (251).
3. The destination requires forwarding, but the receiver will not forward; the sender must resend to the forwarding address (551).
4. A mailbox does not exist for this recipient at this host (550).

5. The destination is rejected due to some other failure to execute (codes 450, 451, 452, 552, 553), or an error in the command (codes 421, 500, 501, 503).

The advantage of using a separate RCPT phase is that the sender will not send the message until it is assured that the receiver is prepared to receive the message for at least one recipient, thereby avoiding the overhead of sending an entire message only to learn that the destination is unknown. Once the SMTP receiver has agreed to receive the mail message for at least one recipient, the SMTP sender uses the **DATA command** to initiate the transfer of the message. If the SMTP receiver is still prepared to receive the message, it returns a 354 message; otherwise, the receiver returns a reply indicating failure to execute the command (codes 451, 554), or an error in the command (codes 421, 500, 501, 503). If the 354 reply is returned, the SMTP sender proceeds to send the message over the TCP connection as a sequence of ASCII lines. The end of the message is indicated by a line containing only a period. The SMTP receiver responds with a "250 OK" reply if the message is accepted, or with the appropriate error code (451, 452, 552, 554).

An example, taken from RFC 821, illustrates the process:

```
S: MAIL FROM:<Smith@Alpha.ARPA>
R: 250 OK

S: RCPT TO:<Jones@Beta.ARPA>
R: 250 OK

S: RCPT TO:<Green@Beta.ARPA>
R: 550 No such user here

S: RCPT TO:<Brown@Beta.ARPA>
R: 250 OK

S: DATA
R: 354 Start mail input; end with <CRLF>.<CRLF>
S: Blah blah blah...
S: ...etc. etc. etc.
S: <CRLF>.<CRLF>
R: 250 OK
```

The SMTP sender is transmitting mail that originates with the user Smith@Alpha.ARPA. The message is addressed to three users on machine Beta.ARPA, namely, Jones, Green, and Brown. The SMTP receiver indicates that it has mailboxes for Jones and Brown but does not have information on Green. Because at least one of the intended recipients has been verified, the sender proceeds to send the text message.

Connection Closing

The SMTP sender closes the connection in two steps. First, the sender sends a QUIT command and waits for a reply. The second step is to initiate a TCP close operation for the TCP connection. The receiver initiates its TCP close after sending its reply to the QUIT command.

RFC 822

RFC 822 defines a format for text messages that are sent using electronic mail. The SMTP standard adopts RFC 822 as the format for use in constructing messages for transmission via SMTP. In the RFC 822 context, messages are viewed as having an envelope and contents. The envelope contains whatever information is needed to accomplish transmission and delivery. The contents compose the object to be delivered to the recipient. The RFC 822 standard applies only to the contents. However, the content standard includes a set of header fields that may be used by the mail system to create the envelope, and the standard is intended to facilitate the acquisition of such information by programs.

An RFC 822 message consists of a sequence of lines of text, and uses a general "memo" framework. That is, a message consists of some number of header lines, which follow a rigid format, followed by a body portion consisting of arbitrary text.

A header line usually consists of a keyword, followed by a colon, followed by the keyword's arguments; the format allows a long line to be broken up into several lines. The most frequently used keywords are From, To, Subject, and Date. Here is an example message:

```
Date: Tue, 16 Jan 1996 10:37:17 (EST)
From: "William Stallings" <ws@host.com>
Subject: The Syntax in RFC 822
To: Smith@Other-host.com
Cc: Jones@Yet-Another-Host.com

Hello. This section begins the actual message body, which is
delimited from the message heading by a blank line.
```

Another field that is commonly found in RFC 822 headers is Message-ID. This field contains a unique identifier associated with this message.

Multipurpose Internet Mail Extensions (MIME)

MIME is an extension to the RFC 822 framework that is intended to address some of the problems and limitations of the use of SMTP and RFC 822 for electronic mail. [MURP95] lists the following limitations of the SMTP/822 scheme:

1. SMTP cannot transmit executable files or other binary objects. A number of schemes are in use for converting binary files into a text form that can be used by SMTP mail systems, including the popular UNIX UUencode/UUdecode scheme. However, none of these is a standard or even a de facto standard.
2. SMTP cannot transmit text data that includes national language characters, as these are represented by 8-bit codes with values of 128 decimal or higher, and SMTP is limited to 7-bit ASCII.
3. SMTP servers may reject mail message over a certain size.
4. SMTP gateways that translate between ASCII and the character code EBCDIC do not use a consistent set of mappings, resulting in translation problems.

5. SMTP gateways to X.400 electronic mail networks cannot handle non-textual data included in X.400 messages.
6. Some SMTP implementations do not adhere completely to the SMTP standards defined in RFC 821. Common problems include the following:
   - Deletion, addition, or reording of carriage return and linefeed.
   - Truncating or wrapping lines longer than 76 characters.
   - Removal of trailing white space (tab and space characters).
   - Padding of lines in a message to the same length.
   - Conversion of tab characters into multiple-space characters.

MIME is intended to resolve these problems in a manner that is compatible with existing RFC 822 implementations. The specification is provided in RFC 1521 and 1522.

Overview

The MIME specification includes the following elements:

1. Five new message header fields are defined, which may be included in an RFC 822 header. These fields provide information about the body of the message.
2. A number of content formats are defined, thus standardizing representations that support multimedia electronic mail.
3. Transfer encodings are defined that enable the conversion of any content format into a form that is protected from alteration by the mail system.

In this subsection, we introduce the five message header fields. The next two subsections address content formats and transfer encodings.

The five header fields defined in MIME are

- **MIME-version.** Must have the parameter value 1.0. This field indicates that the message conforms to RFC 1521 and 1522.
- **Content-type.** Describes the data contained in the body with sufficient detail that the receiving user agent can pick an appropriate agent or mechanism to represent the data to the user or otherwise handle the data in an appropriate manner.
- **Content-transfer-encoding.** Indicates the type of transformation that has been used to represent the body of the message in a way that is acceptable for mail transport.
- **Content-id.** Used to uniquely identify MIME entities in multiple contexts.
- **Content-description.** A plain-text description of the object with the body; this is useful when the object is not readable (e.g., audio data).

Any or all of these fields may appear in a normal RFC 822 header. A compliant implementation must support the MIME-Version, Content-Type, and Content-Transfer-Encoding fields; the Content-ID and Content-Description fields are optional and may be ignored by the recipient implementation.

MIME Content Types

The bulk of the MIME specification is concerned with the definition of a variety of content types; this reflects the need to provide standardized ways of dealing with a wide variety of information representations in a multimedia environment.

Table 19.7 lists the content types specified in RFC 1521. There are seven different major types of content and a total of 14 subtypes. In general, a content type declares the general type of data, and the subtype specifies a particular format for that type of data.

For the *text type* of body, no special software is required to get the full meaning of the text, aside from support of the indicated character set. RFC 1521 defines only one subtype: plain text, which is simply a string of ASCII characters or ISO 8859 characters. An earlier version of the MIME specification included a *richtext* subtype, that allows greater formatting flexibility. It is expected that this subtype will reappear in a later RFC.

The *multipart type* indicates that the body contains multiple, independent parts. The Content-Type header field includes a parameter, called a boundary, that defines the delimiter between body parts. This boundary should not appear in any parts of the message. Each boundary starts on a new line and consists of two hyphens followed by the boundary value. The final boundary, which indicates the

TABLE 19.7 MIME content types.

Type	Subtype	Description
Text	Plain	Unformatted text; may be ASCII or ISO 8859.
Multipart	Mixed	The different parts are independent but are to be transmitted together. They should be presented to the receiver in the order that they appear in the mail message.
	Parallel	Differs from Mixed only in that no order is defined for delivering the parts to the receiver.
	Alternative	The different parts are alternative versions of the same information. They are ordered in increasing faithfulness to the original and the recipient's mail system should display the "best" version to the user.
	Digest	Similar to Mixed, but the default type/subtype of each part is message/rfc822
Message	rfc822	The body is itself an encapsulated message that conforms to RFC 822.
	Partial	Used to allow fragmentation of large mail items, in a way that is transparent to the recipient.
	External-body	Contains a pointer to an object that exists elsewhere.
Image	jpeg	The image is in JPEG format, JFIF encoding.
	gif	The image is in GIF format.
Video	mpeg	MPEG format.
Audio	Basic	Single-channel 8-bit ISDN mu-law encoding at a sample rate of 8 kHz.
Application	PostScript	Adobe Postscript
	octet-stream	General binary data consisting of 8-bit bytes.

end of the last part, also has a suffix of two hyphens. Within each part, there may be an optional, ordinary MIME header.

Here is a simple example of a multipart message, containing two parts, both consisting of simple text (taken from RFC 1521):

```
From: Nathaniel Borenstein <nsb@bellcore.com>
To: Ned Freed <ned@innosoft.com>
Subject: Sample message
MIME-Version: 1.0
Content-type: multipart/mixed; boundary="simple boundary"

This is the preamble. It is to be ignored, though it is a handy place for mail
composers to include an explanatory note to non–MIME-conformant readers.
--simple boundary

This is implicitly-typed plain ASCII text. It does NOT end with a linebreak.
--simple boundary
Content-type: text/plain; charset=us-ascii

This is explicitly-typed plain ASCII text. It DOES end with a linebreak.

--simple boundary--
This is the epilogue. It is also to be ignored.
```

There are four subtypes of the multipart type, all of which have the same overall syntax. The *multipart/mixed subtype* is used when there are multiple independent body parts that need to be bundled in a particular order. For the *multipart/parallel subtype*, the order of the parts is not significant. If the recipient's system is appropriate, the multiple parts can be presented in parallel. For example, a picture or text part could be accompanied by a voice commentary that is played while the picture or text is displayed.

For the *multipart/alternative subtype*, the various parts are different representations of the same information. The following is an example:

```
From: Nathaniel Borenstein <nsb@bellcore.com>
To: Ned Freed <ned@innosoft.com>
Subject: Formatted text mail
MIME-Version: 1.0
Content-Type: multipart/alternative; boundary=boundary42

--boundary42

Content-Type: text/plain; charset=us-ascii

   ...plain-text version of message goes here....
--boundary42
Content-Type: text/richtext

   .... RFC 1341 richtext version of same message goes here ...
--boundary42--
```

In this subtype, the body parts are ordered in terms of increasing preference. For this example, if the recipient system is capable of displaying the message in the richtext format, this is done; otherwise, the plain-text format is used.

The *multipart/digest subtype* is used when each of the body parts is interpreted as an RFC 822 message with headers. This subtype enables the construction of a message whose parts are individual messages. For example, the moderator of a group might collect email messages from participants, bundle these messages, and send them out in one encapsulating MIME message.

The *message type* provides a number of important capabilities in MIME. The *message/rfc822 subtype* indicates that the body is an entire message, including header and body. Despite the name of this subtype, the encapsulated message may be not only a simple RFC 822 message, but any MIME message.

The *message/partial subtype* enables fragmentation of a large message into a number of parts, which must be reassembled at the destination. For this = subtype, three parameters are specified in the Content-Type: Message/Partial field:

- **Id.** A value that is common to each fragment of the same message, so that the fragments can be identified at the recipient for reassembly, but which is unique across different messages.
- **Number.** A sequence number that indicates the position of this fragment in the original message. The first fragment is numbered 1, the second 2, and so on.
- **Total.** The total number of parts. The last fragment is identified by having the same value for the number and total parameters.

The rules for fragmenting a message are as follows:

1. Divide the body of the original message into *N* parts.
2. The first fragment begins with a header that has no Content-Transfer-Encoding field; the default of 7-bit ASCII is used. The header has a Content-Type of Message/Partial, with a unique id, number = 1, and total = *N*. The remaining fields of the header are copied from the original message header.
3. The body of the first fragment is an encapsulated MIME message that has the Content-Type and Content-Transfer-Encoding of the original message body. The Message-ID field of the encapsulated header must differ from that of the enclosing header.
4. The remaining fragments include header fields from the outer header of the first fragment. The Message-ID field must be unique. The Content-Type field has the same id and total values as the outer header of the first fragment, as well as the appropriate number value. There is no Content-Transfer-Encoding field.

The rules for reassembly are as follows:

1. The fields for the header of the reassembled message are taken from the outer header of the first fragment, with the following exceptions. The Content-Type,

Content-Transfer-Encoding, and Message-ID fields are taken from the inner header of the first fragment.

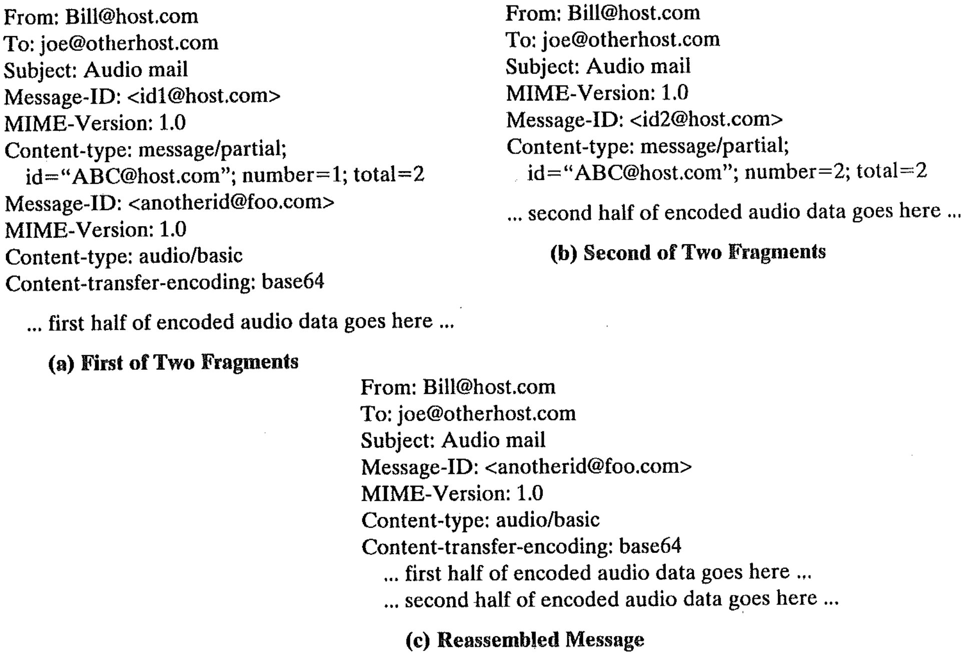
2. All of the header fields from the second and any subsequent fragments are ignored.
3. The body parts of the messages, not including the inner header of the first part, are reassembled in order to form the body of the reassembled message.

Figure 19.10 illustrates a message that is transferred in two fragments.

The *message/external-body subtype* indicates that the actual data to be conveyed in this message are not contained in the body. Instead, the body contains the information needed to access the data. As with the other message types, the message/external-body subtype has an outer header and an encapsulated message with its own header. The only necessary field in the outer header is the Content-Type field, which identifies this as a message/external-body subtype. The inner header is the message header for the encapsulated message.

The Content-Type field in the outer header must include an access-type parameter, which has one of the following values:

- **FTP.** The message body is accessible as a file using the file transfer protocol (FTP). For this access type, the following additional parameters are mandatory: *name*, indicating the name of the file; and *site*, indicating the domain name of the host where the file resides. Optional parameters are *directory*, the directory in which the file is located; and *mode*, which indicates how FTP



```
From: Bill@host.com
To: joe@otherhost.com
Subject: Audio mail
Message-ID: <id1@host.com>
MIME-Version: 1.0
Content-type: message/partial;
  id="ABC@host.com"; number=1; total=2
Message-ID: <anotherid@foo.com>
MIME-Version: 1.0
Content-type: audio/basic
Content-transfer-encoding: base64

  ... first half of encoded audio data goes here ...
```

(a) First of Two Fragments

```
From: Bill@host.com
To: joe@otherhost.com
Subject: Audio mail
MIME-Version: 1.0
Message-ID: <id2@host.com>
Content-type: message/partial;
  id="ABC@host.com"; number=2; total=2

... second half of encoded audio data goes here ...
```

(b) Second of Two Fragments

```
From: Bill@host.com
To: joe@otherhost.com
Subject: Audio mail
Message-ID: <anotherid@foo.com>
MIME-Version: 1.0
Content-type: audio/basic
Content-transfer-encoding: base64
  ... first half of encoded audio data goes here ...
  ... second half of encoded audio data goes here ...
```

(c) Reassembled Message

FIGURE 19.10 Message fragmentation and reassembly.

should retrieve the file (e.g., ASCII, image). Before the file transfer can take place, the user will need to provide a user id and password; these are not transmitted with the message for security reasons.

- **TFPT.** The message body is accessible as a file using the trivial file transfer protocol (TFTP). The same parameters as for FTP are used, and the user id and password must also be supplied.
- **Anon-FTP.** Identical to FTP, except that the user is not asked to supply a user id and password. The parameter name supplies the name of the file.
- **Local-File.** The message body is accessible as a file on the recipient's machine.
- **AFS.** The message body is accessible as a file via the global AFS (Andrew File System). The parameter name supplies the name of the file.
- **Mail-Server.** The message body is accessible by sending an email message to a mail server. A server parameter must be included that gives the email address of the server. The body of the original message, known as the phantom body, should contain the exact command to be sent to the mail server.

The *image type* indicates that the body contains a displayable image. The subtype, jpeg or gif, specifies the image format. In the future, more subtypes will be added to this list.

The *video type* indicates that the body contains a time-varying picture image, possibly with color and coordinated sound. The only subtype so far specified is mpeg.

The *audio type* indicates that the body contains audio data. The only subtype, basic, conforms to an ISDN service known as "64-kbps, 8-kHz Structured, Usable for Speech Information," with a digitized speech algorithm referred to as μ-law PCM (pulse-code modulation). This general type is the typical way of transmitting speech signals over a digital network. The term μ-law refers to the specific encoding technique; it is the standard technique used in North America and Japan. A competing system, known as A-law, is standard in Europe.

The *application type* refers to other kinds of data, typically either uninterpreted binary data or information to be processed by a mail-based application. The *application/octet-stream subtype* indicates general binary data in a sequence of octets. RFC 1521 recommends that the receiving implementation should offer to put the data in a file or use it as input to a program.

The *application/postscript subtype* indicates the use of Adobe Postscript.

MIME Transfer Encodings

The other major component of the MIME specification, in addition to content-type specification, is a definition of transfer encodings for message bodies. The objective is to provide reliable delivery across the largest range of environments.

The MIME standard defines two methods of encoding data. The Content-Transfer-Encoding field can actually take on six values, as listed in Table 19.8. However, three of these values (7bit, 8bit, and binary) indicate that no encoding has been done, but they do provide some information about the nature of the data. For SMTP transfer, it is safe to use the 7bit form. The 8bit and binary forms may be

TABLE 19.8 MIME transfer encodings.

7bit	The data are all represented by short lines of ASCII characters.
8bit	The lines are short, but there may be non-ASCII characters (octets with the high-order bit set).
binary	Not only may non-ASCII characters be present but the lines are not necessarily short enough for SMTP transport.
quoted-printable	Encodes the data in such a way that if the data being encoded are mostly ASCII text, the encoded form of the data remains largely recognizable by humans.
base64	Encodes data by mapping 6-bit blocks of input to 8-bit blocks of output, all of which are printable ASCII characters.
x-token	A named non-standard encoding.

usable in other mail-transport contexts. Another Content-Transfer-Encoding value is *x*-token, which indicates that some other encoding scheme is used, for which a name is to be supplied; this could be a vendor-specific or application-specific scheme. The two actual encoding schemes defined are quoted-printable and base64. Two schemes are defined to provide a choice between a transfer technique that is essentially human-readable and one that is safe for all types of data in a way that is reasonably compact.

The *quoted-printable* transfer encoding is useful when the data consist largely of octets that correspond to printable ASCII characters (see Table 2.1). In essence, it represents non-safe characters by the hexadecimal representation of their code and introduces reversible (soft) line breaks to limit message lines to 76 characters. The encoding rules are as follows:

1. General 8-bit representation: This rule is to be used when none of the other rules apply. Any character is represented by an equal sign, followed by a two-digit hexadecimal representation of the octet's value. For example, the ASCII form-feed, which has an 8-bit value of decimal 12, is represented by "=0C".
2. Literal representation: Any character in the range decimal 33 ("!") through decimal 126 ("~"), except decimal 61, ("=") is represented as that ASCII character.
3. White space: Octets with the values 9 and 32 may be represented as ASCII tab and space characters, respectively, except at the end of a line. Any white space (tab or blank) at the end of a line must be represented by rule 1. On decoding, any trailing white space on a line is deleted; this eliminates any white space added by intermediate transport agents.
4. Line breaks: Any line break, regardless of its initial representation, is represented by the RFC 822 line break, which is a carriage-return/line-feed combination.
5. Soft line breaks: If an encoded line would be longer than 76 characters (excluding <CRLF>), a soft line break must be inserted at or before character position 75. A soft line break consists of the hexadecimal sequence 3D0D0A, which is the ASCII code for an equal sign followed by carriage return line feed.

The *base64 transfer encoding*, also known as radix-64 encoding, is a common one for encoding arbitrary binary data in such a way as to be invulnerable to the processing by mail-transport programs. For example, both PGP (Pretty Good Privacy) and PEM (Privacy Enhanced Mail) secure electronic-mail schemes make use of base64; this technique maps arbitrary binary input into printable character output. The form of encoding has the following relevant characteristics:

1. The range of the function is a character set that is universally representable at all sites, not a specific binary encoding of that character set. Thus, the characters themselves can be encoded into whatever form is needed by a specific system. For example, the character "E" is represented in an ASCII-based system as hexadecimal 45 and in an EBCDIC-based system as hexadecimal C5.
2. The character set consists of 65 printable characters, one of which is used for padding. With $2^6 = 64$ available characters, each character can be used to represent 6 bits of input.
3. No control characters are included in the set. Thus, a message encoded in radix 64 can traverse mail-handling systems that scan the data stream for control characters.
4. The hyphen character ("-") is not used. This character has significance in the RFC 822 format and should therefore be avoided.

Table 19.9 shows the mapping of 6-bit input values to characters. The character set consists of the alphanumeric characters plus "+" and "/". The "=" character is used as the padding character.

Figure 19.11 illustrates the simple mapping scheme. Binary input is processed in blocks of 3 octets, or 24 bits. Each set of 6 bits in the 24-bit block is mapped into

TABLE 19.9 Radix-64 encoding.

6-bit value	character encoding	6-bit value	character encoding	6-bit value	character encoding	6-bit value	character encoding
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
1	K	26	a	42	q	58	6
11	L	27	b	42	r	59	7
12	M	28	c	44	d	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/
						(pad)	=

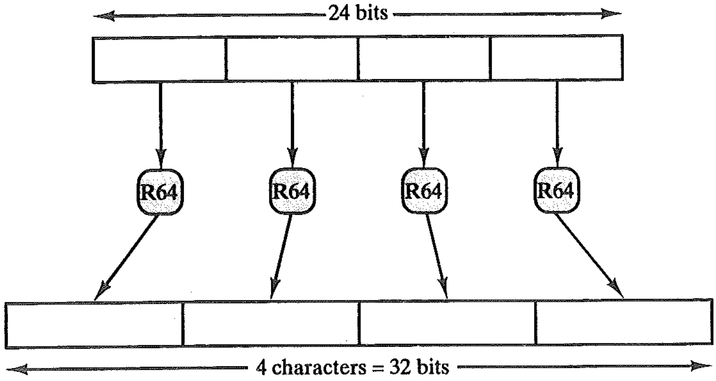


FIGURE 19.11 Printable encoding of binary data into radix-64 format.

a character. In the figure, the characters are shown encoded as 8-bit quantities. In this typical case, each 24-bit input is expanded to 32 bits of output.

One important feature of this mapping is that the least significant 6 bits of the representation of these 65 characters is the same in all commonly used character sets. For example, as was mentioned, "E" in 7-bit ASCII is 0100 0101 and in 8-bit EBCDIC is 1100 0101. The rightmost 6 bits are the same in both cases. Thus, the reverse mapping from radix 64 to binary is simply a matter of extracting the least significant 6 bits of each character.

For example, the sequence "H52Q" in ASCII is

1001000 0110101 0110010 1010001

The extracted 6-bit values are 8, 53, 50, 17:

001000 110101 110010 010001

The resulting 24-bit value can be expressed in hexadecimal as 235CA1.

A Multipart Example

Figure 19.12, taken from RFC 1521, is the outline of a complex multipart message. The message has five parts to be displayed serially: two introductory plain text parts, an embedded multipart message, a richtext part, and a closing encapsulated text message in a non-ASCII character set. The embedded multipart message has two parts to be displayed in parallel: a picture and an audio fragment.

19.4 UNIFORM RESOURCE LOCATORS (URL) AND UNIVERSAL RESOURCE IDENTIFIERS (URI)

Before turning to a description of the Hypertext Transfer Protocol (HTTP), we need to examine two important concepts: the Uniform Resource Locator (URL) and the Universal Resource Identifier (URI).

```
MIME-Version: 1.0
From: Nathaniel Borenstein <nsb@bellcore.com>
To: Ned Freed <ned@innosoft.com>
Subject: A multipart example
Content-Type: multipart/mixed;
   boundary=unique-boundary-1

This is the preamble area of a multipart message. Mail readers that understand multipart format
should ignore this preamble. If you are reading this text, you might want to consider changing to a mail
reader that understands how to properly display multipart messages. --unique-boundary-1

   ... Some text appears here ...
[Note that the preceding blank line means no header fields were given and this is text, with charset US
ASCII. It could have been done with explicit typing as in the next part.]

--unique-boundary-1
Content-type: text/plain; charset=US-ASCII

This could have been part of the previous part, but illustrates explicit versus implicit typing of body
parts.

--unique-boundary-1
Content-Type: multipart/parallel;
   boundary=unique-boundary-2

--unique-boundary-2
Content-Type: audio/basic
Content-Transfer-Encoding: base64

   ... base64-encoded 8000 Hz single-channel
      mu-law-format audio data goes here ...

--unique-boundary-2
Content-Type: image/gif
Content-Transfer-Encoding: base64

   ... base64-encoded image data goes here ...

--unique-boundary-2—

--unique-boundary-1
Content-type: text/richtext

This is <>bold><italic>richtext<>/italic></bold><smaller>as defined in RFC
1341</smaller><nl><snl>Isn't it <bigger><bigger>cool?</bigger></bigger>

--unique-boundary-1
Content-Type: message/rfc822

From: (mailbox in US-ASCII)
To: (address in US-ASCII)
Subject: (subject in US-ASCII)
Content-Type: Text/plain; charset=ISO-8859-1
Content-Transfer-Encoding: Quoted-printable

   ... Additional text in ISO-8859-1 goes here ...

--unique-boundary-1--
```

FIGURE 19.12 Example MIME message structure.

Uniform Resource Locator

A key concept in the operation of the World-Wide Web (WWW) is that of Uniform Resource Locator (URL). In the defining documents (RFC 1738, 1808), the URL is characterized as follows:

> A Uniform Resource Locator (URL) is a compact representation of the location and access method for a resource available via the Internet. URLs are used to locate resources by providing an abstract identification of the resource location. Having located a resource, a system may perform a variety of operations on the resource, as might be characterized by such words as access, update, replace, and find attributes. In general, only the access method needs to be specified for any URL scheme.

A *resource* is any object that can be accessed by the Internet, and includes file directories, files, documents, images, audio or video clips, and any other data that may be stored on an Internet-connected computer. The term *resource* in this context also includes electronic mail addresses, the results of a finger or archie command, USENET newsgroups, and individual messages in a USENET newsgroup.

With the exception of certain dynamic URLs, such as the email address, we can think of a URL as a networked extension of a filename. The URL provides a pointer to any object that is accessible on any machine connected to the Internet. Furthermore, because different objects are accessible in different ways (e.g., via Web, FTP, Gopher, etc., the URL also indicates the access method that must be used to retrieve the object.

The general form of a URL is as follows:

<scheme>:<scheme-specific-part>

The URL consists of the name of the access scheme being used, followed by a colon, and then by an identifier of a resource whose format is specific to the scheme being used.

Although the scheme-specific formats differ, they have a number of points in common, as we will see. In particular, many of the access schemes support the use of hierarchical structures, similar to the hierarchical directory and file structures common to file systems such as UNIX. For the URL, the components of the hierarchy are separated by a "/", similar to the UNIX approach.

RFC 1738 defines URL formats for the following access schemes:

ftp	File Transfer Protocol
http	Hypertext Transfer Protocol
gopher	The Gopher Protocol
mailto	Electronic mail address
news	USENET news
nntp	USENET news using NNTP access
telnet	Reference to interactive sessions
wais	Wide-Area Information Servers
file	Host-specific file names
prospero	Prospero Directory Service

Table 19.10 shows the general format of each of these schemes.

TABLE 19.10 Uniform resource locator (URL) schemes.

Scheme	Default Port	Syntax
ftp	21	ftp://<user>:<password>@<host>:<port>/<cwd1>/<cwd2>/.../<cwdN>/<name>; type=<typecode>
http	80	http://<host>:<port>/<path>?<searchpart>
gopher	70	gopher://<host>:<port>/<selector> or gopher://<host>:<port>/<selector>%09<search> or gopher://<host>:<port>/<selector>%09<search>%09<gopher+_string> or
mailto	—	mailto:<rfc822-addr-spec>
news	—	news:<newsgroup-name> or news:<message-id>
nntp	119	nntp://<host>:<port>/<newsgroup-name>/<article-number>
telnet	23	telnet://<user>:<password>@<host>:<port>
wais	210	wais://<host>:<port>/<database> or wais://<host>:<port>/<database>?<search> or wais://<host>:<port>/<database>/<wtype>/<wpath> or
file	—	file://<host>/<path>
prospero	1525	prospero://<host>:<port>/<hsoname>;<field>=<value>

File Transfer Protocol (FTP)

The FTP URL scheme designates files and directories accessible using the FTP protocol. In its simplest form, an FTP URL has the following format:

ftp://<host>/<directoryname>/<filename>

where <host> is the name on an Internet host or a dotted decimal IP address of the host (e.g., acm.org).

As an example, consider the document named Index.README on the anonymous FTP server rtfm.mit.edu in directory pub. The URL for this file is

ftp://rtfm.mit.edu/pub/Index.README

The URL for the directory is

ftp://rtfm.mit.edu/pub/

And the URL for the ftp site itself is simply

ftp://rtfm.mit.edu

The most general form of the ftp URL is

ftp://<user>:<password>@<host>:<port>/<cwd1>/<cwd2>/.../<cwdN>/
<name>;type=<typecode>

Some FTP sites require that the user provide a user id and a password; these are provided in the form <user>:<password> and the @ symbol, preceding the <host> value. The next new item in the format is <port>. Most access schemes, including ftp, designate protocols that have a default port number; for ftp, it is 21. Another port number may be optionally used and, if so, is supplied by a colon and the port number following the <host> value.

After the specification of the host, with an optional user-ID and password, and a port number, a slash indicates the beginning of the file designation. Each of the <cwd> elements is a directory name, or, more precisely, an argument to a CWD (change working directory) command, such as is used in UNIX. The <name> value, if present, is the name of a file. Finally, the <typecode> value can be used to designate a particular type of file; otherwise, the type defaults in an implementation-dependent way.

Hypertext Transfer Protocol (HTTP)

The HTTP URL scheme designates accessible Internet resources, using the HTTP protocol, and, in particular, designates web sites. In its simplest form, an HTTP URL has the following format:

http://<host>:<port>/<path>

The default port number for HTTP is 80. If the <path> portion is omitted, then the URL points to the top level resource, such as a home page. For example,

http://www.shore.net

points to the home page of the Internet site "www.shore.net." A more complex path points to hierarchically subordinate pages. For example,

http://www.shore.net/~ws

points to the author's home page on the shore.net computer. An HTTP URL can also point to a document available via the web, such as

http://www.w3.org/pub/WWW/Addressing/rfc1738.txt

This is the URL for RFC 1738, available through the web site of the WWW consortium.

The ?<searchpart> portion of an HTTP URL is optional. When present, it designates a query that will be invoked when the resource is accessed.

The Gopher Protocol

The FTP URL scheme designates files and directories accessible using the FTP protocol. A Gopher URL takes the form:

gopher://<host>:<port>/<gopher-path>

where <gopher-path> is one of the following:

<gophertype><selector>
<gophertype><selector>%09<search>
<gophertype><selector>%09<search>%09<gopher+_string>

The default port is 70. The first form selects a Gopher site, or a directory or file at the Gopher site. For example,

gopher://mitdir.mit.edu:105/2

Selects the Gopher-accessible telephone directory at M.I.T. This directory is searchable by keyword. A user who accesses this directory can then interactively enter a key word to initiate a search. Alternatively, this can be part of the URL; for example

gopher://mitdir.mit.edu:105/2?chomsky

will access the directory and search it for the word "chomsky."

A URL can also be used to access a more-general Gopher search engine and initiate a search by supplying the string %09<search>, where %09 refers to the tab. URLs for Gopher+ have a second tab and a Gopher+ string. Gopher+ is a set of extensions to the original Gopher protocol.

Electronic Mail Address

The mailto URL scheme designates the Internet mailing address of an individual or service. When invoked by a web client, it triggers the creation of an email message to be sent by Internet electronic mail. For example,

mailto:webmaster@w3.org

designates the email address of the webmaster for the WWW consortium web site.

USENET News

The news URL scheme designates either a news group or the individual articles of USENET news. For example,

news:comp.dcom.cell-relay

designates the ATM newsgroup, and

news:4bs62n$10le@usenetw1.news.prodigy.com

refers to a message in that newsgroup

USENET News Using NNTP Access

The NNTP URL scheme is an alternative way of designating news articles, useful for specifying articles from NNTP servers. The general form is

nntp://<host>:<port>/<newsgroup-name>/<article-number>

This technique designates a particular location for the news server, whereas the news:// scheme does not. Because most NNTP servers are configured to allow access only from local clients, the news form of the URL is preferred.

Reference to Interactive Sessions (TELNET)

The TELNET URL scheme designates interactive services accessible by the TELNET protocol. Thus, this URL does not designate a data object but a service.

Wide Area Information Servers (WAIS)

The WAIS URL scheme designates WAIS databases, searches, or individual documents available from a WAIS database. A WAIS takes one of the following forms:

wais://<host>:<port>/<database>
wais://<host>:<port>/<database>?<search>
wais://<host>:<port>/<database>/<wtype>/<wpath>

The first form designates a WAIS database. The second form designates a search submitted to a database. The third form designates a particular document within a database, where <wtype> is the WAIS designation of the document type.

Host-Specific File Names

The file URL scheme differs from other URL schemes in that it does not designate an Internet-accessible object or service. It provides a way of uniquely identifying a directory or file on an Internet-addressable host, but does not designate an access protocol. Thus, it has limited utility in a network context.

Prospero Directory Service

The Prospero URL scheme designates resources that are accessed via the Prosperó Directory Service. A prospero URL takes the form

prospero://<host>:<port>/<hsoname>;<field>=<value>

where <hsoname> is the host-specific object name in the Prospero protocol. The optional clause <field>=<value> serves to identify a particular target entry.

Universal Resource Identifier

Universal Resource Identifier (URI) is a term for a generic WWW identifier. The URI specification (RFC 1630) defines a syntax for encoding arbitrary naming or addressing schemes, and provides a list of such schemes. The concept of a URI, and in particular its details, are still evolving. The URL is a type of URI, in which an access protocol is designated and a specific Internet address is provided.

The potential advantage of the URI is that it decouples the name of a resource from its location and even from its access method. With the URL, a specific instance of a resource at a specific location is designated. If there are multiple instances, and that specific instance is unavailable at the time of a request, then a requester must determine an alternative URL and try that. In principle, with a URI, this process could be automated. In practice, documents such as the HTTP specification refer to the use of URIs, but are currently implemented using only URLs.

19.5 HYPERTEXT TRANSFER PROTOCOL (HTTP)

The Hypertext Transfer Protocol (HTTP) is the foundation protocol of the world-wide web (WWW) and can be used in any client-server application involving hypertext. The name is somewhat misleading in that HTTP is not a protocol for transferring hypertext; rather, it is a protocol for transmitting information with the efficiency necessary for making hypertext jumps. The data transferred by the protocol can be plain text, hypertext, audio, images, or any Internet-accessible information.

We begin with an overview of HTTP concepts and operation and then look at some of the details.[4] A number of important terms defined in the HTTP specification are summarized in Table 19.11; these will be introduced as the discussion proceeds.

HTTP Overview

HTTP is a transaction-oriented client/server protocol. The most typical use of HTTP is between a web browser and a web server. To provide reliability, HTTP

TABLE 19.11 Key terms related to HTTP.

Cache

A program's local store of response messages and the subsystem that controls its message storage, retrieval, and deletion. A cache stores cacheable responses in order to reduce the response time and network bandwidth consumption on future, equivalent requests. Any client or server may include a cache, though a cache cannot be used by a server while it is acting as a tunnel.

Client

An application program that establishes connections for the purpose of sending requests.

Connection

A transport layer virtual circuit established between two application programs for the purposes of communication.

Entity

A particular representation or rendition of a data resource, or reply from a service resource, that may be enclosed within a request or response message. An entity consists of entity headers and an entity body.

Gateway

A server that acts as an intermediary for some other server. Unlike a proxy, a gateway receives requests as if it were the original server for the requested resource; the requesting client may not be aware that it is communicating with a gateway. Gateways are often used as server-side portals through network firewalls and as protocol translators for access to resources stored on non-HTTP systems.

Message

The basic unit of HTTP communication, consisting of a structured sequence of octets transmitted via the connection.

Origin Server

The server on which a given resource resides or is to be created.

Proxy

An intermediary program that acts as both a server and a client for the purpose of making requests on behalf of other clients. Requests are serviced internally or by passing them, with possible translation, on to other servers. A proxy must interpret and, if necessary, rewrite a request message before forwarding it. Proxies are often used as client-side portals through network firewalls and as helper applications for handling requests via protocols not implemented by the user agent.

Resource

A network data object or service that can be identified by a URI.

Server

An application program that accepts connections in order to service requests by sending back responses.

Tunnel

A tunnel is an intermediary program that is acting as a blind relay between two connections. Once active, a tunnel is not considered a party to the HTTP communication, though the tunnel may have been initiated by an HTTP request. A tunnel ceases to exist when both ends of the relayed connections are closed. Tunnels are used when a portal is necessary and the intermediary cannot, or should not, interpret the relayed communication.

User Agent

The client that initiates a request. These are often browsers, editors, spiders, or other end-user tools.

[4] This section is based on the most recent (at the time of this writing) specification, HTTP 1.1, which is the first version to be put on the IETF standards track.

makes use of TCP. Nevertheless, HTTP is a "stateless" protocol: Each transaction is treated independently. Accordingly, a typical implementation will create a new TCP connection between client and server for each transaction and then terminate the connection as soon as the transaction completes, although the specification does not dictate this one-to-one relationship between transaction and connection lifetimes.

The stateless nature of HTTP is well-suited to its typical application. A normal session of a user with a web browser involves retrieving a sequence of web pages and documents. The sequence is, ideally, performed rapidly, and the locations of the various pages and documents may be a number of widely distributed servers.

Another important feature of HTTP is that it is flexible in the formats that it can handle. When a client issues a request to a server, it may include a prioritized list of formats that it can handle, and the server replies with the appropriate format. For example, a Lynx browser cannot handle images, so a web server need not transmit any images on web pages. This arrangement prevents the transmission of unnecessary information and provides the basis for extending the set of formats with new standardized and proprietary specifications.

Figure 19.13 illustrates three examples of HTTP operation. The simplest case is one in which a *user agent* establishes a direct connection with an *origin server*. The *user agent* is the client that initiates the request, such as a web browser being run on behalf of an end user. The *origin server* is the server on which a resource of interest resides; an example is a web server at which a desired web home page resides. For this case, the client opens a TCP connection that is end-to-end between the client and the server. The client then issues an HTTP request. The request consists of a specific command, referred to as a method, a URL, and a MIME-like message containing request parameters, information about the client, and perhaps some additional content information.

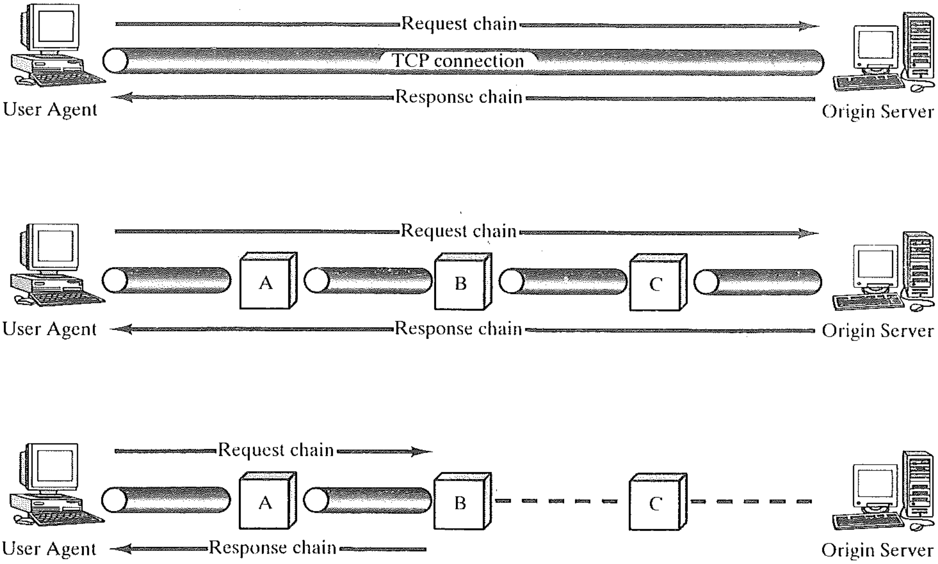


FIGURE 19.13 Examples of HTTP operation.

When the server receives the request, it attempts to perform the requested action and then returns an HTTP response. The response includes status information, a success/error code, and a MIME-like message containing information about the server, information about the response itself, and possible body content. The TCP connection is then closed.

The middle part of Figure 19.13 shows a case in which there is not an end-to-end TCP connection between the user agent and the origin server. Instead, there are one or more intermediate systems with TCP connections between logically adjacent systems. Each intermediate system acts as a relay, so that a request initiated by the client is relayed through the intermediate systems to the server, and the response from the server is relayed back to the client.

Three forms of intermediate systems are defined in the HTTP specification: *proxy*, *gateway*, and *tunnel*, all of which are illustrated in Figure 19.14.

Proxy

A proxy acts on behalf of other clients and presents requests from other clients to a server. The proxy acts as a server in interacting with a client, and as a client in interacting with a server. There are several scenarios that call for the use of a proxy:

1. **Security intermediary.** The client and server may be separated by a security intermediary such as a firewall, with the proxy on the client side of the firewall. Typically, the client is part of a network secured by a firewall, and the server is external to the secured network. In this case, the server must authenticate itself to the firewall to set up a connection with the proxy. The proxy accepts responses after they have passed through the firewall.

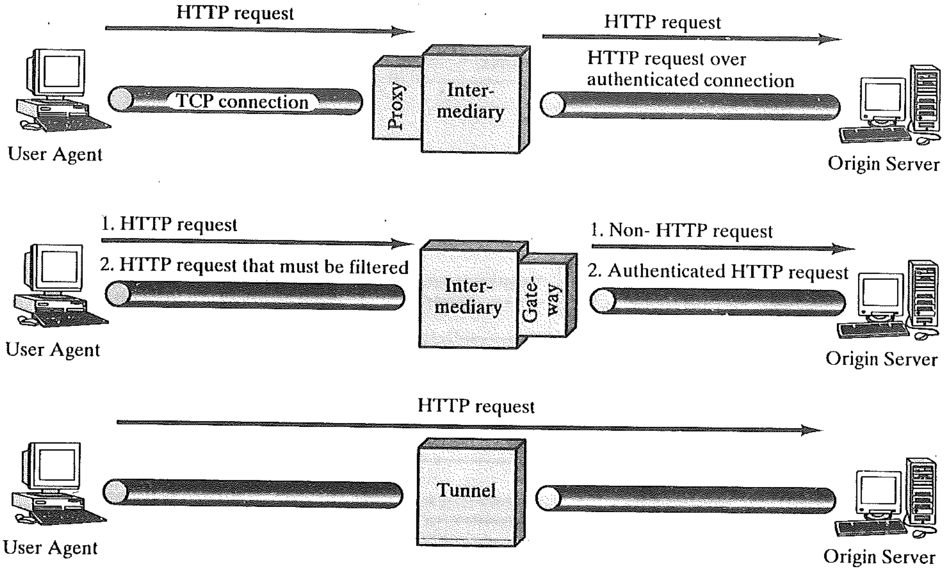


FIGURE 19.14 Intermediate HTTP systems.

2. **Different versions of HTTP.** If the client and server are running different versions of HTTP, then the proxy can implement both versions and perform the required mapping.

In summary, a proxy is a forwarding agent, receiving a request for a URL object, modifying the request, and forwarding that request toward the server identified in the URL.

Gateway

A gateway is a server that appears to the client as if it were an origin server. It acts on behalf of other servers that may not be able to communicate directly with a client. There are several scenarios in which servers can be used:

1. **Security intermediary.** The client and server may be separated by a security intermediary such as a firewall, with the gateway on the server side of the firewall. Typically, the server is connected to a network protected by a firewall, with the client external to the network. In this case, the client must authenticate itself to the proxy, which can then pass the request on to the server.
2. **Non-HTTP server.** Web browsers have built into them the capability to contact servers for protocols other than HTTP, such as FTP and Gopher servers. This capability can also be provided by a gateway. The client makes an HTTP request to a gateway server. The gateway server then contacts the relevant FTP or Gopher server to obtain the desired result. This result is then converted into a form suitable for HTTP and transmitted back to the client.

Tunnel

Unlike the proxy and the gateway, the tunnel performs no operations on HTTP requests and responses. Instead, a tunnel is simply a relay point between two TCP connections, and the HTTP messages are passed unchanged as if there were a single HTTP connection between user agent and origin server. Tunnels are used when there must be an intermediary system between client and server, but it is not necessary for that system to understand the contents of messages. An example is a firewall in which a client or server external to a protected network can establish an authenticated connection, and which can then maintain that connection for purposes of HTTP transactions.

Cache

Returning to Figure 19.13, the lowest portion of the figure shows an example of a cache. A cache is a facility that may store previous requests and responses for handling new requests. If a new request arrives that is the same as a stored request, then the cache can supply the stored response rather than accessing the resource indicated in the URL. The cache can operate on a client or server, or on an intermediate system other than a tunnel. In the figure, intermediary B has cached a request/response transaction, so that a corresponding new request from the client need not travel the entire chain to the origin server, but is handled by B.

Not all transactions can be cached, and a client or server can dictate that a certain transaction may be cached only for a given time limit.

Messages

The best way to describe the functionality of HTTP is to describe the individual elements of the HTTP message. HTTP consists of two types of messages: requests from clients to servers, and responses from servers to clients. The general structure of such messages is shown in Figure 19.15. More formally, using enhanced BNF (Backus-Naur Form) notation (Table 19.12), we have

```
     HTTP-Message = Simple-Request | Simple-Response | Full-Request | Full-
                    Response
     Full-Request = Request-Line
                    *( General-Header | Request-Header | Entity-Header )
                    CRLF
                    [ Entity-Body ]
    Full-Response = Status-Line
                    *( General-Header | Response-Header | Entity-Header )
                    CRLF
                    [ Entity-Body ]
   Simple-Request = "GET" SP Request-URI CRLF
  Simple-Response = [ Entity-Body ]
```

The Simple-Request and Simple-Response messages were defined in HTTP/0.9. The request is a simple GET command with the requested URI; the response is simply a block containing the information identified in the URI. In

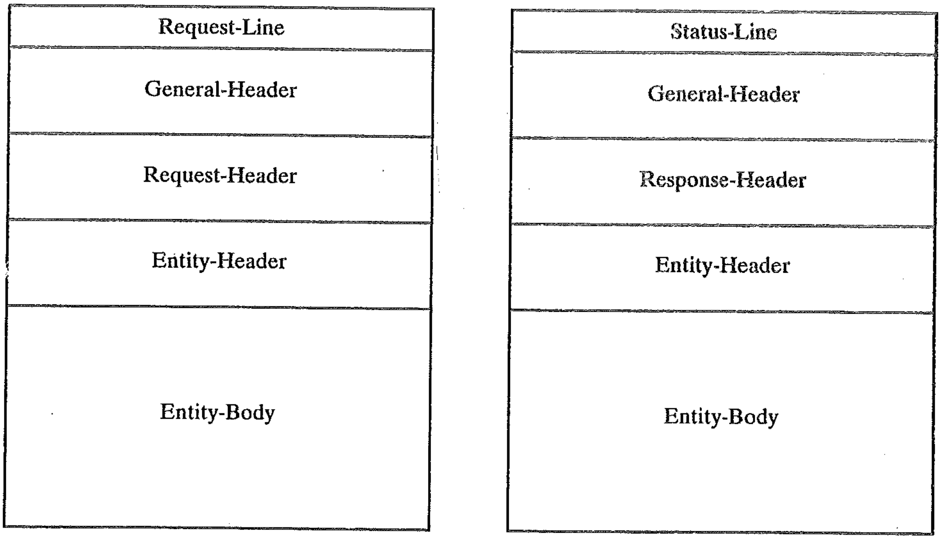


FIGURE 19.15 General structure of HTTP messages.

TABLE 19.12 Augmented BNF notation used in URL and HTTP specifications.

- Words in lower case represent variables or names of rules.
- A rule has the form

  name = definition

- DIGIT is any decimal digit; CRLF is carriage return, line feed; SP is one or more spaces.
- Quotation marks enclose literal text.
- Angle brackets, "<" ">", may be used within a definition to enclose a rule name when their presence will facilitate clarity.
- Elements separated by bar ("|") are alternatives.
- Ordinary parentheses are used simply for grouping.
- The character "*" preceding an element indicates repetition. The full form is

  <I>*<J>element

  indicating at least I and at most J occurrences of the element. *element allows any number, including 0; 1*element requires at least one element; and 1*2element allows 1 or 2 elements; <N>element means exactly *N* elements.
- Square brackets, "[" "]", enclose optional elements.
- The construct "#" is used to define, with the following form,

  <I>#<J>element

  indicating at least *I* and at most *J* elements, each separated by a comma and optional linear white space.
- A semicolon at the right of a rule starts a comment that continues to the end of the line.

HTTP/1.1, the use of these simple forms is discouraged because it prevents the client from using content negotiation and the server from identifying the media type of the returned entity.

With full requests and responses, the following fields are used:

- **Request-line.** Identifies the message type and the requested resource.
- **Response-line.** Provides status information about this response.
- **General-header.** Contains fields that are applicable to both request and response messages, but which do not apply to the entity being transferred.
- **Request-header.** Contains information about the request and the client.
- **Response-header.** Contains information about the response.
- **Entity-header.** Contains information about the resource identified by the request and information about the entity body.
- **Entity-body.** The body of the message.

All of the HTTP headers consist of a sequence of fields, following the same generic format as RFC 822 (described in Section 19.3). Each field begins on a new line and consists of the field name followed by a colon and the field value.

Although the basic transaction mechanism is simple, there are a large number of fields and parameters defined in HTTP; these are listed in Table 19.13. In the remainder of this section, we look at the general header fields. Succeeding sections describe request headers, response headers, and entities.

TABLE 19.13 HTTP elements.

ALL MESSAGES			
GENERAL HEADER FIELDS		ENTITY HEADER FIELDS	
Cache-Control	Keep-Alive	Allow	Derived-From
Connection	MIME-Version	Content-Encoding	Expires
Data	Pragma	Content-Language	Last-Modified
Forwarded	Upgrade	Content-Length	Link
		Content-MD5	Title
		Content-Range	Transfer-Encoding
		Content-Type	URI-Header
		Content-Version	extention-header

REQUEST MESSAGES			
REQUEST METHODS		REQUEST HEADER FIELDS	
OPTIONS	MOVE	Accept	If-Modified-Since
GET	DELETE	Accept-Charset	Proxy-Authorization
HEAD	LINK	Accept-Encoding	Range
POST	UNLINK	Accept-Language	Referer
PUT	TRACE	Authorization	Unless
PATCH	WRAPPED	From	User-Agent
COPY	extension-method	Host	

RESPONSE MESSAGES			
RESPONSE STATUS CODES			RESPONSE HEADER FIELDS
Continue	Moved Temporarily	Request Timeout	Location
Switching Protocols	See Other	Conflict	Proxy-Authenticate
OK	Not Modified	Gone	Public
Created	Use Proxy	Length Required	Retry-After
Accepted	Bad Request	Unless True	Server
Non-Authoritative Information	Unauthorized	Internal Server Error	WWW-Authenticate
No Content	Payment Required	Not Implemented	
Reset Content	Forbidden	Bad Gateway	
Partial Content	Not Found	Service Unavailable	
Multiple Choices	Method Not Allowed	Gateway Timeout	
Moved Permanently	None Acceptable	extension code	
	Proxy Authentication Required		

General Header Fields

General header fields can be used in both request and response messages. These fields are applicable in both types of messages and contain information that does not directly apply to the entity being transferred. The fields are the following:

- **Cache-Control.** Specifies directives that must be obeyed by any caching mechanisms along the request/response chain; the purpose is to prevent a cache from adversely interfering with this particular request or response.

- **Connection.** Contains a list of keywords and header-field names that only apply to this TCP connection between the sender and the nearest non-tunnel recipient.
- **Data.** Data and time at which the message originated.
- **Forwarded.** Used by gateways and proxies to indicate intermediate steps along a request or response chain. Each gateway or proxy that handles a message may attach a Forwarded field that gives its URI.
- **Keep-Alive.** May be present if the Keep-Alive keyword is present in an incoming Connection field, to provide information to the requester of the persistent connection. This field may indicate a maximum time that the sender will keep the connection open while waiting for the next request or the maximum number of additional requests that will be allowed on the current persistent connection.
- **MIME-Version.** Indicates that the message complies with the indicated version of MIME.
- **Pragma.** Contains implementation-specific directives that may apply to any recipient along the request/response chain.
- **Upgrade.** Used in a request to specify what additional protocols the client supports and would like to use; used in a response to indicate which protocol will be used.

Two of these fields warrant further elaboration: Cache-Control and Connection.

Cache-Control

A Cache-Control field can be attached to either a request or a response. Any caching mechanisms that receive a message with this header must follow the directives in the header, which may mean deviating from the default caching action. This field has the following format:

```
Cache-Control    = “Cache-Control” “:” 1#cache-directive
cache-directive  = “cachable”
                   | “max-age” “=” delta-seconds
                   | “private” [ “=” <”> 1#field-name <”> ]
                   | “no-cache” [ “=” <”> 1#field-name <”> ]
```

That is, this field consists of the phrase “Cache-Control:” followed by one or more directives.

A *cachable* directive is included in a response to indicate that the server generating the response declares it to be cachable. Any caching mechanism that forwards this response may cache it for future use.

A *max-age* directive is used in a request to inform any caching mechanism en route that it may use a cached response to this message only if it has a cached response that is no older than the age specified. A server may include this directive in a response to inform any caching mechanism en route that it may cache this response for future requests up to the max-age time limit.

A *private* directive in a response indicates that parts of the response message are intended for a single user and must not be cached except within a non-shared cache controlled by the user agent. If no field names are listed, the entire message is private.

A *no-cache* directive in a request forces that request to be forwarded to the origin server and not answered by an intermediate cache. This directive allows a client to request an authoritative response or to refresh a suspect cache. The list of field names is not used in a request message. In a response, the no-cache directive indicates that part or all of the message must not be cached for future use.

Connection

A Connection field can be attached to either a request or a response. It is used to communicate from one end point of a TCP connection to the other end point. Thus, this field is not end-to-end at the HTTP level. When an intermediary system receives and forwards a message containing this field, that system must remove the field prior to forwarding.

The body of this field may include one or more field names for fields included in this message. These fields are to be processed by the recipient and not forwarded with the rest of the message. Alternatively, the body may consist of one or more keywords. At present, only the Keep-Alive keyword is defined in version 1.1 of HTTP; this indicates that the sender would like a persistent TCP connection (one that remains open beyond the current transaction).

Request Messages

A full-request message consists of a status line followed by one or more general, request, and entity headers, followed by an optional entity body.

Request Methods

A full request message always begins with a Request-Line, which has the following format:

Request-Line = Method SP Request-URI SP HTTP-Version CRLF

The Method parameter indicates the actual request command, called a *method* in HTTP. Request-URI is the URI of the requested resource, and HTTP-Version is the version number of HTTP used by the sender.

The following request methods are defined in HTTP/1.1:

- **OPTIONS.** A request for information about the options available for the request/response chain identified by this URI.
- **GET.** A request to retrieve the information identified in the URI and return it in an entity body. A GET is conditional if the If-Modified-Since header field is included, and is partial if a Range header field is included.
- **HEAD.** This request is identical to a GET, except that the server's response must not include an entity body; all of the header fields in the response are the same as if the entity body were present; this enables a client to get information about a resource without transferring the entity body.

- **POST.** A request to accept the attached entity as a new subordinate to the identified URI. The posted entity is subordinate to that URI in the same way that a file is subordinate to a directory containing it, a news article is subordinate to a newsgroup to which it is posted, or a record is subordinate to a database.
- **PUT.** A request to accept the attached entity and store it under the supplied URI. This may be a new resource with a new URI, or a replacement of the contents of an existing resource with an existing URI.
- **PATCH.** Similar to a PUT, except that the entity contains a list of differences from the content of the original resource identified in the URI.
- **COPY.** Requests that a copy of the resource identified by the URI in the Request-Line be copied to the location(s) given in the URI-Header field in the Entity-Header of this message.
- **MOVE.** Requests that the resource identified by the URI in the Request-Line be moved to the location(s) given in the URI-Header field in the Entity-Header of this message; equivalent to a COPY followed by a DELETE.
- **DELETE.** Requests that the origin server delete the resource identified by the URI in the Request-Line.
- **LINK.** Establishes one or more link relationships from the resource identified in the Request-Line. The links are defined in the Link field in the Entity-Header.
- **UNLINK.** Removes one or more link relationships from the resource identified in the Request-Line. The links are defined in the Link field in the Entity-Header.
- **TRACE.** Requests that the server return whatever is received as the entity body of the response; this can be used for testing and diagnostic purposes.
- **WRAPPED.** Allows a client to send one or more encapsulated requests. The requests may be encrypted or otherwise processed. The server must unwrap the requests and process accordingly.
- **Extension-method.** Allows additional methods to be defined without changing the protocol, but these methods cannot be assumed to be recognizable by the recipient.

Request Header Fields

Request header fields function as request modifiers, providing additional information and parameters related to the request. The following fields are defined in HTTP/1.1:

- **Accept.** A list of media types and ranges that are acceptable as a response to this request.
- **Accept-charset.** A list of character sets acceptable for the response.
- **Accept-encoding.** List of acceptable content encodings for the entity body. Content encodings are primarily used to allow a document to be compressed or encrypted. Typically, the resource is stored in this encoding and only decoded before actual use.

- **Accept-language.** Restricts the set of natural languages that are preferred for the response.
- **Authorization.** Contains a field value, referred to as *credentials*, used by the client to authenticate itself to the server.
- **From.** The Internet e-mail address for the human user who controls the requesting user agent.
- **Host.** Specifies the Internet host of the resource being requested.
- **If-modified-since.** Used with the GET method. This header includes a date/time parameter; the resource is to be transferred only if it has been modified since the date/time specified. This feature allows for efficient cache update. A caching mechanism can periodically issue GET messages to an origin server, and will receive only a small response message unless an update is needed.
- **Proxy-authorization.** Allows the client to identify itself to a proxy that requires authentication.
- **Range.** For future study. The intent is that, in a GET message, a client can request only a portion of the identified resource.
- **Referer.** The URI of the resource from which the Request-URI was obtained. This enables a server to generate lists of back-links.
- **Unless.** Similar in function to the If-Modified-Since field, with two differences: (1) It is not restricted to the GET method, and (2) comparison is based on any Entity-Header field value rather than a date/time value.
- **User-agent.** Contains information about the user agent originating this request. This is used for statistical purposes, the tracing of protocol violations, and automated recognition of user agents for the sake of tailoring responses to avoid particular user agent limitations.

Response Messages

A full-response message consists of a status line followed by one or more general, response, and entity headers, followed by an optional entity body.

Status Codes

A full-response message always begins with a Status-Line, which has the following format:

Status-Line = HTTP-Version SP Status-Code SP Reason-Phrase CRLF

The HTTP-Version value is the version number of HTTP used by the sender. The Status-Code is a 3-digit integer that indicates the response to a received request, and the Reason-Phrase provides a short textual explanation of the status code.

There are a rather large number of status codes defined in HTTP/1.1; these are listed in Table 19.14, together with a brief definition. The codes are organized into the following categories:

TABLE 19.14 HTTP status codes.

Informational	
Continue	Initial part of request received; client may continue with request.
Switching Protocols	Server will switch to requested new application protocol.
Successful	
OK	Request has succeeded and the appropriate response information is included.
Created	Request fulfilled and a new resource has been created; the URI(s) are included.
Accepted	Request accepted but processing not completed. The request may or may not eventually be acted upon.
Non-Authoritative Information	Returned contents of entity header is not the definitive set available from origin server, but is gathered from a local or third-party copy.
No Content	Server has fulfilled request but there is no information to send back.
Reset Content	Request has succeeded and the user agent should reset the document view that caused the request to be generated.
Partial Content	Server has fulfilled the partial GET request and the corresponding information is included.
Redirection	
Multiple Choices	Requested resource is available at multiple locations and a preferred location could not be determined.
Moved Permanently	Requested resource has been assigned a new permanent URI; future reference should use this URI
Moved Temporarily	Requested resource resides temporarily under a different URI.
See Other	Response to the request can be found under a different URI and should be retrieved using a GET on that resource.
Not Modified	The client has performed a conditional GET, access is allowed, and the document has not been modified since the date/time specified in the request.
Use Proxy	Requested resource must be accessed through the proxy indicated in the Location field.
Client Error	
Bad Request	Malformed syntax in request.
Unauthorized	Request requires user authentication.
Payment Required	Reserved for future use.
Forbidden	Server refuses to fulfill request; used when server does not wish to reveal why the request was refused.
Not Found	Requested URI not found.
Method Not Allowed	Method (command) not allowed for the requested resource.
None Acceptable	Resource found that matches requested URI, but does not satisfy conditions specified in the request.
Proxy Authentication Required	Client must first authenticate itself with the proxy.
Request Timeout	Client did not produce a request within the time that the server was prepared to wait.
Conflict	Request could not be completed due to a conflict with the current state of the resource.
Gone	Requested resource no longer available at the server and no forwarding address is known.
Length Required	Server refuses to accept request without a defined content length.
Unless True	Condition given in the Unless field was true when tested on server.

(Continued)

TABLE 19.14 (*Continued*)

Server Error	
Internal Server Error	Server encountered an unexpected condition that prevented it from fulfilling the request.
Not Implemented	Server does not support the functionality required to fulfill the request.
Bad Gateway	Server, while acting as a gateway or proxy, received an invalid response from the upstream server it accessed to fulfill the request.
Service Unavailable	Server unable to handle request due to temporary overloading or maintenance of the server.
Gateway Timeout	Server, while acting as a gateway or proxy, did not receive a timely response from the upstream server it accessed to fulfill the request.

- **Informational.** The request has been received and processing continues. No entity body accompanies this response.
- **Successful.** The request was successfully received, understood, and accepted. The information returned in the response message depends on the request method, as follows:

  —GET: The contents of the entity-body corresponds to the requested resource.

  —HEAD: No entity body is returned.

  —POST: The entity describes or contains the result of the action.

  —TRACE: The entity contains the request message.

  —Other methods: The entity describes the result of the action.
- **Redirection.** Further action is required to complete the request.
- **Client error.** The request contains a syntax error or the request cannot be fulfilled.
- **Server error.** The server failed to fulfill an apparently valid request.

Response Header Fields

Response header fields providing additional information related to the response that cannot be placed in the Status-Line. The following fields are defined in HTTP/1.1:

- **Location.** Defines the exact location of the resource identified by the Request-URI.
- **Proxy-authenticate.** Included with a response that has a status code of Proxy Authentication Required. This field contains a "challenge" that indicates the authentication scheme and parameters required.
- **Public.** Lists the non-standard methods supported by this server.
- **Retry-after.** Included with a response that has a status code of Service Unavailable, and indicates how long the service is expected to be unavailable.
- **Server.** Identifies the software product used by the origin server to handle the request.

- **WWW-authenticate.** Included with a response that has a status code of Unauthorized. This field contains a challenge that indicates the authentication scheme and parameters required.

Entities

An entity consists of an entity header and an entity body in a request or response message. An entity may represent a data resource, or it may constitute other information supplied with a request or response.

Entity Header Fields

Entity header fields provide optional information about the entity body or, if no body is present, about the resource identified by the request. The following fields are defined in HTTP/1.1:

- **Allow.** Lists methods supported by the resource identified in the Request-URI. This field must be included with a response that has a status code of Method Not Allowed and may be included in other responses.
- **Content-encoding.** Indicates what content encodings have been applied to the resource. The only encoding currently defined is zip compression.
- **Content-language.** Identifies the natural language(s) of the intended audience of the enclosed entity.
- **Content-length.** The size of the entity body in octets.
- **Content-MD5.** For future study. MD5 refers to the MD5 hash code function, described in Chapter 18.
- **Content-range.** For future study. The intent is that this designation will indicate a portion of the identified resource that is included in this response.
- **Content-type.** Indicates the media type of the entity body.
- **Content-version.** A version tag associated with an evolving entity.
- **Derived-from.** Indicates the version tag of the resource from which this entity was derived before modifications were made by the sender. This field and the Content-Version field can be used to manage multiple updates by a group of users.
- **Expires.** Date/time after which the entity should be considered stale.
- **Last-modified.** Date/time that the sender believes the resource was last modified.
- **Link.** Defines links to other resources.
- **Title.** A textual title for the entity.
- **Transfer-encoding.** Indicates what type of transformation has been applied to the message body to safely transfer it between the sender and the recipient. The only encoding defined in the standard is *chunked*. The chunked option defines a procedure for breaking an entity body into labeled chunks that are transmitted separately.
- **URI-header.** Informs the recipient of other URIs by which the resource can be identified.

- **Extension-header.** Allows additional fields to be defined without changing the protocol, but these fields cannot be assumed to be recognizable by the recipient.

Entity Body

An entity body consists of an arbitrary sequence of octets. HTTP is designed to be able to transfer any type of content, including text, binary data, audio, images, and video. When an entity body is present in a message, the interpretation of the octets in the body is determined by the entity header fields Content-Encoding, Content-Type, and Transfer-Encoding. These define a three-layer, ordered encoding model:

entity-body := Transfer-Encoding(Content-Encoding(Content-Type
(data)))

The data are the contents of a resource identified by a URI. The Content-Type field determines the way in which the data are interpreted. A Content-Encoding may be applied to the data and stored at the URI instead of the data. Finally, on transfer, a Transfer-Encoding may be applied to form the entity body of the message.

Access Authentication

HTTP/1.1 defines a simple challenge-response technique for authentication. This definition does not restrict HTTP clients and servers from using other forms of authentication, but the current standard only covers this simple form.

Two authentication exchanges are defined: one between a client and a server, and one between a client and a proxy. Both types of exchange use a challenge-response mechanism. The challenge, issued by a server or proxy, is of the form

challenge = auth-scheme 1*SP realm *("," auth-param)
auth-scheme = token
auth-param = token "=" quoted-string
realm = "realm" "=" realm-value
realm-value = quoted-string

Auth-scheme is the name of a particular authentication scheme. The realm defines a particular *protection space*, which is simply a conceptual partition of the resource, with its own authentication scheme and authorization database. For example, a resource may define several realms, one for end users and one for network managers. The latter realm may have more privileges and requires a more powerful authentication scheme.

In response to an authentication challenge, a client must provide credentials. These are of the form

credentials = basic-credentials | auth-scheme *("," auth-param)

Basic credentials are covered below. In the general case, the user would return the name of the authentication scheme and a set of parameters required to authenticate itself.

Client-Server Authentication

A user agent that wishes to authenticate itself with a server may do so by including an Authorization field in the request header; an agent may do this when initially sending the request. An alternative, which may be more common, is that a client sends a Request message without an Authorization field and is then required to return an authorization by the server. Figure 19.16 illustrates this scenario, which involves three steps:

1. The client sends a request, such as a GET request to the server, with no Authorization field in the request header.
2. The server returns a response with a status code in the Status line of Unauthorized and a WWW-Authenticate field in the response header. The WWW-Authenticate field consists of a challenge that indicates the type of authentication required and may include other parameters. No entity body is returned.
3. The client repeats the request but includes an Authorization field that contains the authorization data needed by the server.

If authentication succeeds, the server returns a response with some other status code and without a WWW-Authenticate field. If authentication fails, the server can initiate a new authentication sequence by returning a response with a status of Unauthorized and a WWW-Authenticate field containing the (possibly new) challenge. The entity body should explain the reason for the refusal.

In client-server authentication, any proxy or gateway must be transparent, as far as authentication is concerned. That is, the WWW-Authenticate and Authorization fields must be forwarded unmodified, and the response to a request containing an Authorization field must not be cached. This latter requirement dictates that the authentication always takes place between client and server and does not simply replay the server's prior acceptance of authentication.

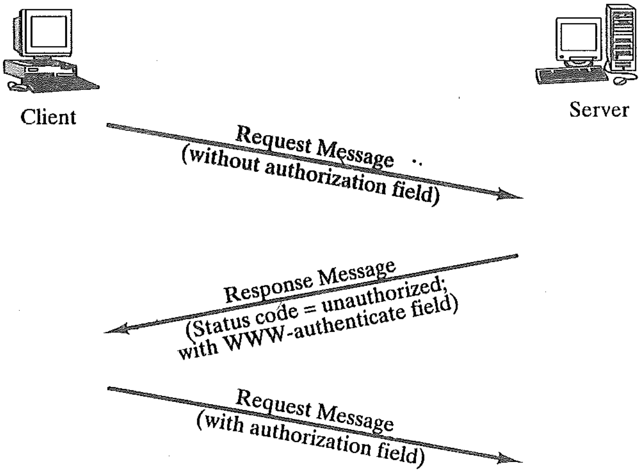

Client
Server
Request Message
(without authorization field)
Response Message
(Status code = unauthorized;
with WWW-authenticate field)
Request Message
(with authorization field)

FIGURE 19.16 HTTP access-authentication scenario.

Proxy Authentication

A proxy may be configured so that a client must first authenticate itself to the proxy before being granted access to an origin server. The sequence is similar to that described for client-server authentication. In this case, the authentication information is carried in the Proxy-Authorization field in the request header. A client may authenticate itself when first issuing a Request message. Alternatively, a scenario similar to Figure 19.16 occurs:

1. The client sends a request, such as a GET request to the server, with no Proxy-Authorization field in the request header.
2. The proxy does not forward the request, but returns a response with a status code in the Status line of Proxy-Authentication Required and a Proxy-Authenticate field in the response header.
3. The client repeats the request but includes a Proxy-Authorization field that contains the authorization data needed by the proxy.

If the request is authenticated, then the proxy may forward the request to a server, but will omit the Proxy-Authorization field. The proxy could also return a cached response.

Basic Authentication Scheme

For the basic authentication scheme, a user agent authenticates itself within a particular realm by supplying a user ID and a password. This is the simplest form of authentication, comparable to logging on to a system. Within HTTP, there is no provision for protecting the user ID or password with encryption, so this method provides minimal security. The form of the credentials for basic authentication are

```
basic-credentials   = " Basic" SP basic-cookie
basic-cookie = <base64 [7] encoding of userid-password, except not limited to
               76 char/line>
userid-password     = [ token ] ":" *TEXT
```

The combination of user ID and password is an arbitrary string of textual characters, transmitted in base-64 encoding. For example, consider a server that has an end-user realm and that receives a Request message without an Authorization field. The server sends a Response message with a status code of Unauthorized and this WWW-Authenticate field:

WWW-Authenticate: Basic-realm="UserSpace"

where "UserSpace" is the string assigned by the server to identify the protection space of the Request-URI. The client then sends a Request message that includes an Authorization field that base-64 encodes the user ID of "Aladdin" and the password "open sesame." The field looks like this:

Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==

19.6 RECOMMENDED READING

A thorough presentation of ASN.1 is to be found in [STEE90]. Two quite useful documents are [KALI91] and [GAUD89]. [STAL96] provides a comprehensive and detailed examination of SNMP and SNMPv2; the book also provides an overview of network-management technology. One of the few textbooks on the subject of network management is [TERP92]. [ROSE93] provides a book-length treatment of electronic mail, including some coverage of SMTP and MIME.

GAUD89 Gaudette, P. *A Tutorial on ASN.1.* Technical Report NCSL/SNA-89/12. Gaithersburg, MD: National Institute of Standards and Technology, 1989.

KALI91 Kaliski, B. *A Layman's Guide to a Subset of ASN.1, BER, and DER. Report SEC-SIG-91-17.* Redwood City, CA: RSA Data Security Inc. 1991.

ROSE93 Rose, M. *The Internet Message: Closing the Book with Electronic Mail.* Englewood Cliffs, NJ: Prentice Hall, 1993.

STAL96 Stallings, W. *SNMP, SNMPv2, and RMON: Practical Network Management,* Reading, MA: Addison-Wesley, 1996.

STEE90 Steedman, D. *ASN.1: The Tutorial and Reference.* London: Technology Appraisals, 1990.

TERP92 Terplan, K. *Communication Networks Management.* Englewood Cliffs, NJ: Prentice Hall, 1992.

Recommended Web Sites

- http://snmp.cs.utwente.nl: Exhaustive source of information on SNMP.
- http://www.w3.org/pub/WWW: Web site of the World Wide Web Consortium, containing up-to-date information on HTTP.
- http://www.inria.fr/rodeo/personnel/hoschka/asn1.html: ASN.1 web site, contains tutorial information, links to software tools, ASN.1-based applications and products, and standards information.

19.7 PROBLEMS

19.1 a. Consider the following definitions:

```
ExpeditedDataAcknowledgement ::= SET {
destRef                              [0]     Reference,
yr-tu-nr                             [1]     TPDUnumber,
checkSum                             [2]     CheckSum OPTIONAL }
NormalEA ::= ExpeditedDataAcknowledgement
( WITH COMPONENTS {
                        destRef,
                        yr-tu-nr (0..127) }
```

Find an equivalent expression for defining NormalEA.

b. Consider the following definition

```
ExtendedEA ::= ExpeditedDataAcknowledgement
( WITH COMPONENTS {..., checkSum PRESENT } )
```

Find an equivalent expression for defining ExtendedEA.

c. Define a new type, EA, based on ExpeditedDataAcknowledgement, with the restriction that either the checksum is absent, in which case yr-tu-nr must be in the range of 0 to 127, or the checksum is present, in which case there is no additional constraint on yr-tu-nr.

19.2 Given the following definition,

```
TypH ::= SEQUENCE {
r                          INTEGER,
s                          BOOLEAN,
t                          INTEGER OPTIONAL }
```

Which of the following values are valid?

```
valH1 TypH ::= { r -5, s TRUE, t 0}
valH2 TypH ::= {10, FALSE}
valH3 TypH ::= { t 1, r 2, s TRUE}
```

19.3 Given the following definitions,

```
T1 ::= SEQUENCE { X1,
                     b BOOLEAN }
X1 ::= CHOICE {y INTEGER, z REAL }
```

Would the identifiers for T1 be: y, z, b?

19.4 Define an IA5String-based type that contains only either "A" or "B" characters and is restricted to a maximum length of 10 chars.

19.5 The original (version 1) specification of SNMP has the following definition of a new type:

```
Gauge ::= [APPLICATION 2] IMPLICIT INTEGER (0..4294967295)
```

The stanoard includes the following explanation of the semantics of this type:

> This application-wide type represents a non-negative integer, which may increase or decrease, but which latches at a maximum value. This standard specifies a maximum value of 2^{32}–1 (4294967295 decimal) for Gauges.

Unfortunately, the word latch is not defined, and has resulted in two different interpretations. The SNMPv2 standard cleared up the ambiguity with the following definition:

> The value of a Gauge has its maximum value whenever the information being modeled is greater than or equal to that maximum value; if the information being modeled subsequently decreases below the maximum value, the Gauge also decreases.

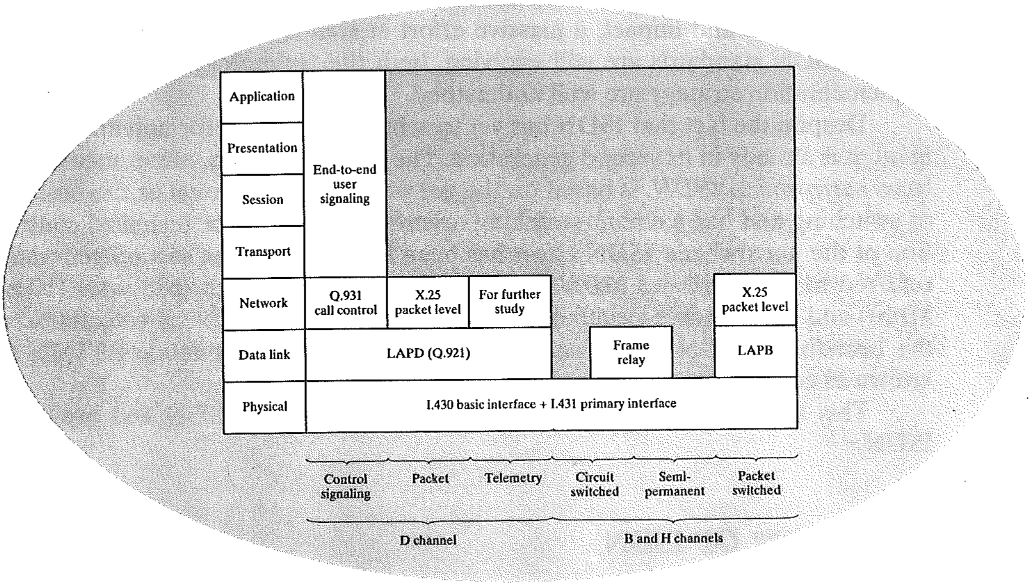
a. What is the alternative interpretation?

b. Discuss the pros and cons of the two interpretations.

19.6 Electronic mail systems differ in the manner in which multiple recipients are handled. In some systems, the originating user agent or mail sender makes all the necessary copies, and these are sent out independently. An alternative approach is to determine the route for each destination first. Then a single message is sent out on a common portion of the route, and copies are only made when the routes diverge; this process is referred to as mail-bagging. Discuss the relative advantages and disadvantages of the two methods.

APPENDIX A

ISDN AND BROADBAND ISDN



Rapid advances in computer and communication technologies have resulted in the increasing merging of these two fields. The lines have blurred among computing, switching, and digital transmission equipment, and the same digital techniques are being used for data, voice, and image transmission. Merging and evolving technologies, coupled with increasing demands for efficient and timely collection, processing, and dissemination of information, are leading to the development of integrated systems that transmit and process all types of data. The ultimate goal of this evolution is the integrated services digital network (ISDN).

The ISDN is intended to be a worldwide public telecommunications network to replace existing public telecommunications networks and deliver a wide variety of services. The ISDN is defined by the standardization of user interfaces and is implemented as a set of digital switches and paths supporting a broad range of traffic types and providing value-added processing services. In practice, there are multiple networks, implemented within national boundaries, but from the user's point of view, there will be a single, uniformly accessible, worldwide network.

The impact of ISDN on both users and vendors will be profound. To control ISDN evolution and impact, a massive effort at standardization is underway. Although ISDN standards are still evolving, both the technology and the emerging implementation strategy are well understood.

Despite the fact that ISDN has yet to achieve the hoped for universal deployment, it is already in its second generation. The first generation, sometimes referred to as *narrowband ISDN*, is based on the use of a 64-kbps channel as the basic unit of switching and has a circuit-switching orientation. The major technical contribution of the narrowband ISDN effort has been frame relay. The second generation, referred to as *broadband ISDN* (B-ISDN), supports very high data rates (100s of Mbps) and has a packet-switching orientation. The major technical contribution of the broadband ISDN effort has been asynchronous transfer mode (ATM), also known as cell relay.

This appendix provides an overview of narrowband ISDN and broadband ISDN.

A.1 OVERVIEW OF ISDN

ISDN Concept

The concept of ISDN is best introduced by considering it from several different viewpoints:

- Principles of ISDN
- The user interface
- Objectives
- Services

Principles of ISDN

Standards for ISDN have been defined by ITU-T (formerly CCITT), a topic that we explore later in this section. Table A.1, which is the complete text of one of the

TABLE A.1 Recommendation I.120 (1988).

1 Principles of ISDN

1.1 The main feature of the ISDN concept is the support of a wide range of voice and nonvoice applications in the same network. A key element of service integration for an ISDN is the provision of a range of services (see Part II of the I-Series in this Fascicle) using a limited set of connection types and multipurpose user-network interface arrangements (see Parts III and IV of the I-Series in Fascicle III.8).

1.2 ISDNs support a variety of applications including both switched and non-switched connections. Switched connections in an ISDN include both circuit-switched and packet-switched connections and their concatenations.

1.3 As far as practicable, new services introduced into an ISDN should be arranged to be compatible with 64 kbit/s switched digital connections.

1.4 An ISDN will contain intelligence for the purpose of providing service features, maintenance and network management functions. This intelligence may not be sufficient for some new services and may have to be supplemented by either additional intelligence within the network, or possibly compatible intelligence in the user terminals.

1.5 A layered protocol structure should be used for the specification of the access to an ISDN. Access from a user to ISDN resources may vary depending upon the service required and upon the status of implementation of national ISDNs.

1.6 It is recognized that ISDNs may be implemented in a variety of configurations according to specific national situations.

2 Evolution of ISDNs

2.1 ISDNs will be based on the concepts for telephone IDNs and may evolve by progressively incorporating additional functions and network features including those of any other dedicated networks such as circuit-switching and packet-switching for data so as to provide for existing and new services.

2.2 The transition from an existing network to a comprehensive ISDN may require a period of time extending over one or more decades. During this period arrangements must be developed for the networking of services on ISDNs and services on other networks (see Part V).

2.3 In the evolution towards an ISDN, digital end-to-end connectivity will be obtained via plant and equipment used in existing networks, such as digital transmission, time-division multiplex switching and/or space-division multiplex switching. Existing relevant recommendations for these constituent elements of an ISDN are contained in the appropriate series of recommendations of CCITT and of CCIR.

2.4 In the early stages of the evolution of ISDNs, some interim user-network arrangements may need to be adopted in certain countries to facilitate early penetration of digital service capabilities. Arrangements corresponding to national variants may comply partly or wholly with I-Series Recommendations. However, the intention is that they not be specifically included in the I-Series.

2.5 An evolving ISDN may also include at later stages switched connections at bit rates higher and lower than 64 kbit/s.

ISDN-related standards, states the principles of ISDN from the point of view of CCITT. Let us look at each of these points in turn:

1. *Support of voice and nonvoice applications using a limited set of standardized facilities.* This principle defines both the purpose of ISDN and the means of achieving it. The ISDN supports a variety of services related to voice communications (telephone calls) and nonvoice communications (digital data exchange). These services are to be provided in conformance with standards

(ITU-T recommendations) that specify a small number of interfaces and data transmission facilities.

2. *Support for switched and nonswitched applications.* ISDN supports both circuit switching and packet switching. In addition, ISDN supports nonswitched services in the form of dedicated lines.
3. *Reliance on 64-kbps connections.* ISDN provides circuit-switched and packet-switched connections at 64 kbps; this is the fundamental building block of ISDN. This rate was chosen because, at the time, it was the standard rate for digitized voice, and, hence, was being introduced into the evolving integrated digital networks (IDNs). Although this data rate is useful, it is unfortunately restrictive to rely solely on it. Future developments in ISDN will permit greater flexibility.
4. *Intelligence in the network.* An ISDN is expected to be able to provide sophisticated services beyond the simple setup of a circuit-switched call.
5. *Layered protocol architecture.* The protocols for user access to ISDN exhibit a layered architecture and can be mapped into the OSI model. This procedure has a number of advantages:
    - Standards already developed for OSI-related applications may be used on ISDN. An example is X.25 level 3 for access to packet-switching services in ISDN.
    - New ISDN-related standards can be based on existing standards, reducing the cost of new implementations. An example is LAPD, which is based on LAPB.
    - Standards can be developed and implemented independently for various layers and functions within a layer; this allows for the gradual implementation of ISDN services at a pace appropriate for a given provider or a given customer base.
6. *Variety of configurations.* More than one physical configuration is possible for implementing ISDN; this allows for differences in national policy (single-source versus competition), in the states of technology, and in the needs and existing equipment of the customer base.

The User Interface

Figure A.1 is a conceptual view of the ISDN from a user, or customer, point of view. The user has access to the ISDN by means of a local interface to a "digital pipe" of a certain bit rate. Pipes of various sizes are available to satisfy differing needs. For example, a residential customer may require only sufficient capacity to handle a telephone and a videotex terminal. An office will undoubtedly wish to connect to the ISDN via an on-premise digital PBX, and will require a much higher capacity pipe.

At any given point in time, the pipe to the user's premises has a fixed capacity, but the traffic on the pipe may be a variable mix up to the capacity limit. Thus, a user may access circuit-switched and packet-switched services, as well as other services, in a dynamic mix of signal types and bit rates. To provide these services, the ISDN requires rather complex control signals to instruct it how to sort out the time-

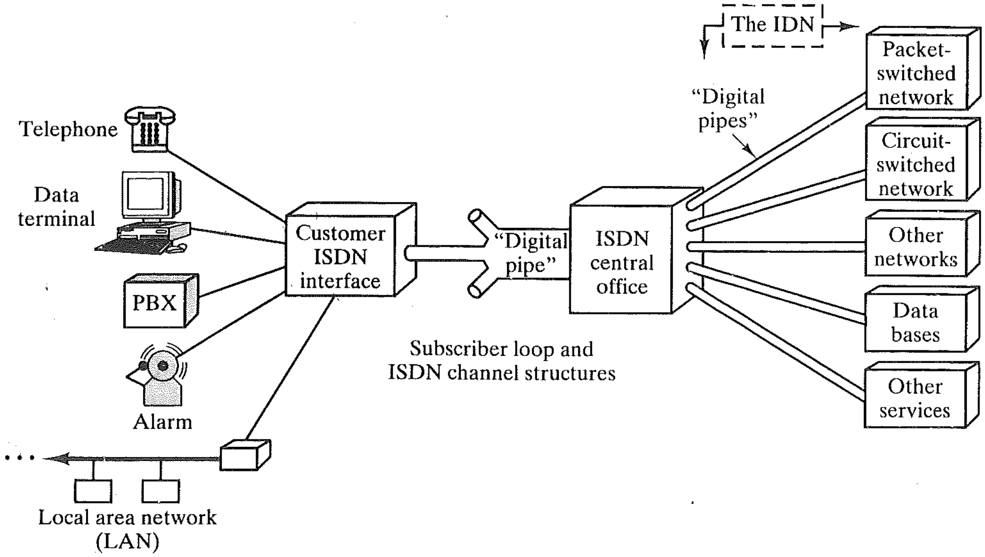


FIGURE A.1 Conceptual view of ISDN connection features.

multiplexed data and provide the required services. These control signals are also multiplexed onto the same digital pipe.

An important aspect of the interface is that the user may, at any time, employ less than the maximum capacity of the pipe, and will be charged according to the capacity used rather than "connect time." This characteristic significantly diminishes the value of current user design efforts that are geared to optimize circuit utilization by use of concentrators, multiplexers, packet switches, and other line-sharing arrangements.

Objectives

Activities currently under way are leading to the development of a worldwide ISDN. This effort involves national governments, data processing and communications companies, standards organizations, and other agencies. Certain common objectives are, by and large, shared by this disparate group. We list here the key objectives:

- **Standardization.** It is essential that a single set of ISDN standards be provided to permit universal access and to permit the development of cost-effective equipment.
- **Transparency.** The most important service to be provided is a transparent transmission service, thereby permitting users to develop applications and protocols with the confidence that they will not be affected by the underlying ISDN.
- **Separation of competitive functions.** It must be possible to separate out functions that could be provided competitively as opposed to those that are fun-

damentally part of the ISDN. In many countries, a single, government-owned entity provides all services. Some countries desire (in the case of the United States, require) that certain enhanced services be offered competitively (e.g., videotex, electronic mail).

- **Leased and switched services.** The ISDN should provide dedicated point-to-point services as well as switched services, thereby allowing the user to optimize implementation of switching and routing techniques.
- **Cost-related tariffs.** The price for ISDN service should be related to cost, and should be independent of the type of data being carried. One type of service should not be in the position of subsidizing others.
- **Smooth migration.** The conversion to ISDN will be gradual, and the evolving network must coexist with existing equipment and services. Thus, ISDN interfaces should evolve from current interfaces, and provide a migration path for users.
- **Multiplexed support.** In addition to providing low-capacity support to individual users, multiplexed support must be provided to accommodate user-owned PBX and local network equipment.

There are, of course, other objectives that could be named. Those listed above are certainly among the most important and widely accepted, and each helps to define the character of the ISDN.

Architecture

Figure A.2 is a block diagram of ISDN. ISDN supports a new physical connecter for users, a digital subscriber loop (link from end user to central or end office), and modifications to all central office equipment.

The area to which most attention has been paid by standards organizations is that of user access. A common physical interface has been defined to provide, in essence, a DTE-DCE connection. The same interface should be usable for telephone, computer terminal, and videotex terminal. Protocols are needed for the exchange of control information between user device and the network. Provision must be made for high-speed interfaces to, for example, a digital PBX or a LAN.

The subscriber loop portion of today's telephone network consists of twisted pair links between the subscriber and the central office, carrying 4-kHz analog signals. Under the ISDN, one or two twisted pairs are used to provide a basic full-duplex digital communications link.

The digital central office connects the numerous ISDN subscriber loop signals to the IDN. In addition to providing access to the circuit-switched network, the central office provides subscriber access to dedicated lines, packet-switched networks, and time-shared, transaction-oriented computer services. Multiplexed access via digital PBX and LAN must also be accommodated.

Standards

The development of ISDN is governed by a set of recommendations issued by ISDN, called the I-series Recommendations. These Recommendations, or stan-

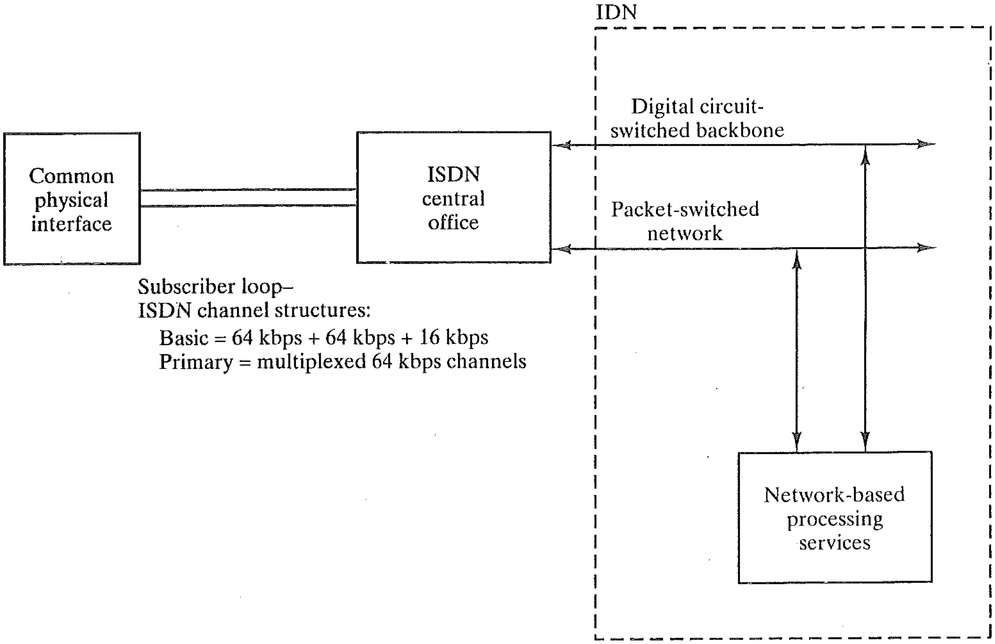

IDN
Digital circuit-switched backbone
Packet-switched network
Common physical interface
ISDN central office
Subscriber loop–
ISDN channel structures:
Basic = 64 kbps + 64 kbps + 16 kbps
Primary = multiplexed 64 kbps channels
Network-based processing services

FIGURE A.2 Block diagram of ISDN functions.

dards, were first issued in 1984. A more complete set was issued in 1988. Most of the Recommendations have been updated, at irregular intervals, since that time. The bulk of the description of ISDN is contained in the I-series Recommendations, with some related topics covered in other Recommendations. The characterization of ISDN contained in these Recommendations is centered on three main areas:

1. The standardization of services offered to users, so as to enable services to be internationally compatible.
2. The standardization of user-network interfaces, so as to enable terminal equipment to be portable, and to assist in (1).
3. The standardization of ISDN capabilities to the degree necessary to allow user-network and network-network interworking, and thus achieve (1) and (2).

The I-series Recommendations are broken up into six main groupings, labeled I.100 through I.600.

I.100 Series—General Concepts

The I.100 series serves as a general introduction to ISDN. The general structure of the ISDN recommendations is presented as well as a glossary of terms. I.120 provides an overall description of ISDN and the expected evolution of ISDNs. I.130 introduces terminology and concepts that are used in the I.200 series to specify services.

I.200 Series—Service Capabilities

The I.200 series is in a sense the most important part of the ITU-T ISDN recommendations. Here, the services to be provided to users are specified. We may look on this as a set of requirements that the ISDN must satisfy. In the ISDN glossary (I.112), the term *service* is defined as

> That which is offered by an Administration or recognized private operating agency (RPOA) to its customers in order to satisfy a specific telecommunication requirement.

Although this is a very general definition, the term "service" has come to have a very specific meaning in ITU-T, a meaning that is somewhat different from the use of that term in an OSI context. For ITU-T, a standardized service is characterized by

- Complete, guaranteed end-to-end compatibility
- ITU-T–standardized terminals, including procedures
- Listing of the service subscribers in an international directory
- ITU-T–standardized testing and maintenance procedures
- Charging and accounting rules

There are three fully standardized ITU-T services: telegraphy, telephony, and data. There are four additional *telematic* services in the process of being standardized: teletex, facsimile, videotex, and message handling. The goal with all of these services is to ensure high-quality international telecommunications for the end user, regardless of the make of the terminal equipment and the type of network used nationally to support the service.

I.300 Series—Network Aspects

Whereas the I.200 series focuses on the user, in terms of the services provided, the I.300 series focuses on the network, in terms of how the network goes about providing those services. A protocol reference model is presented that, while based on the 7-layer OSI model, attempts to account for the complexity of a connection that may involve two or more users (e.g., a conference call) plus a related common-channel signaling dialogue. Issues such as numbering and addressing are covered. There is also a discussion of ISDN connection types.

I.400 Series—User-Network Interfaces

The I.400 series deals with the interface between the user and the network. Three major topics are addressed:

- **Physical configurations.** The issue of how ISDN functions are configured into equipment. The standards specify functional groupings and define reference points between those groupings.
- **Transmission rates.** The data rates and combinations of data rates to be offered to the user.
- **Protocol specifications.** The protocols at OSI layers 1 through 3 that specify the user-network interaction.

I.500 Series—Internetwork Interfaces

ISDN supports services that are also provided on older circuit-switched and packet-switched networks. Thus, it is necessary to provide interworking between an ISDN and other types of networks to allow communications between terminals belonging to equivalent services offered through different networks. The I.500 series deals with the various network issues that arise in attempting to define interfaces between ISDN and other types of networks.

I.600 Series—Maintenance Principles

This series provides guidance for maintenance of the ISDN subscriber installation, the network portion of the ISDN basic access, primary access, and higher data-rate services. Maintenance principles and functions are related to the reference configuration and general architecture of ISDN. A key function that is identified in the series is loopback. In general, loopback testing is used for failure localization and verification.

A.2 ISDN CHANNELS

The digital pipe between the central office and the ISDN user is used to carry a number of communication channels. The capacity of the pipe, and therefore the number of channels carried, may vary from user to user. The transmission structure of any access link is constructed from the following types of channels:

- B channel: 64 kbps
- D channel: 16 or 64 kbps
- H channel: 384(H0), 1536(H11), and 1920 (H12) kbps

The *B channel* is the basic user channel. It can be used to carry digital data, PCM-encoded digital voice, or a mixture of lower-rate traffic, including digital data and digitized voice encoded at a fraction of 64 kbps. In the case of mixed traffic, all traffic must be destined for the same endpoint. Four kinds of connections can be set up over a B channel:

- **Circuit-switched.** This is equivalent to switched digital service available today. The user places a call, and a circuit-switched connection is established with another network user. An interesting feature is that call-establishment dialogue does not take place over the B channel, but is done over the D, as explained below.
- **Packet-switched.** The user is connected to a packet-switching node, and data are exchanged with other users via X.25.
- **Frame mode.** The user is connected to a frame relay node, and data are exchanged with other users via LAPF.

- **Semipermanent.** This is a connection to another user set up by prior arrangement, and not requiring a call-establishment protocol; this is equivalent to a leased line.

The designation of 64 kbps as the standard user channel rate highlights the fundamental contradiction in standards activities. This rate was chosen as the most effective for digitized voice, yet the technology has progressed to the point at which 32 kbps, or even less, produces equally satisfactory voice reproduction. To be effective, a standard must freeze the technology at some defined point. Yet by the time the standard is approved, it may already be obsolete.

The *D channel* serves two purposes. First, it carries signaling information to control circuit-switched calls on associated B channels at the user interface. In addition, the D channel may be used for packet-switching or low-speed (e.g., 100 bps) telemetry at times when no signaling information is waiting. Table A.2 summarizes the types of data traffic to be supported on B and D channels.

H channels are provided for user information at higher bit rates. The user may employ such a channel as a high-speed trunk, or the channel may be subdivided according to the user's own TDM scheme. Examples of applications include fast facsimile, video, high-speed data, high-quality audio, and multiple information streams at lower data rates.

These channel types are grouped into transmission structures that are offered as a package to the user. The best-defined structures at this time are the basic channel structure (basic access) and the primary channel structure (primary access), which are depicted in Figure A.3.

Basic access consists of two full-duplex 64-kbps B channels and a full-duplex 16-kbps D channel. The total bit rate, by simple arithmetic, is 144 kbps. However, framing, synchronization, and other overhead bits bring the total bit rate on a basic access link to 192 kbps. The frame structure for basic access was shown in Figure 7.10. Each frame of 48 bits includes 16 bits from each of the B channels and 4 bits from the D channel.

The basic service is intended to meet the needs of most individual users, including residential and very small offices. It allows the simultaneous use of voice and several data applications, such as packet-switched access, a link to a central

TABLE A.2 ISDN Channel functions.

B Channel (64 kbps)	D Channel (16 kbps)
Digital voice	Signaling
64 kbps PCM	Basic
Low bit rate (32 kbps)	Enhanced
High-speed data	Low-speed data
Circuit-switched	Videotex
Packet-switched	Teletex
Other	Terminal
Facsimile	Telemetry
Slow-scan video	Emergency services
	Energy management

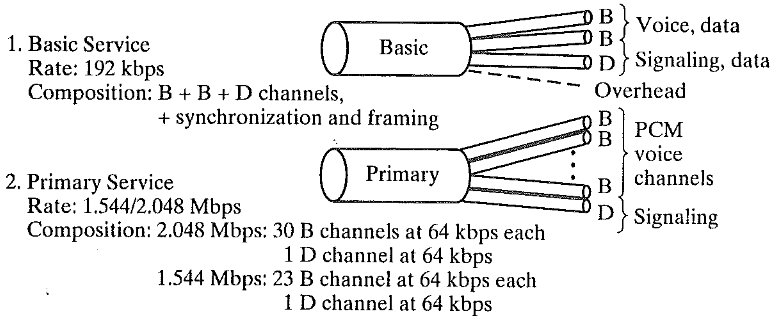


FIGURE A.3 ISDN channel structures.

alarm service, facsimile, videotex, and so on. These services could be accessed through a single multifunction terminal or several separate terminals. In either case, a single physical interface is provided. Most existing two-wire local loops can support this interface.

In some cases, one or both of the B channels remain unused; this results in a B+D or D interface, rather than the 2B+D interface. However, to simplify the network implementation, the data rate at the interface remains at 192 kbps. Nevertheless, for those subscribers with more modest transmission requirements, there may be a cost savings in using a reduced basic interface.

Primary access is intended for users with greater capacity requirements, such as offices with a digital PBX or a local network. Because of differences in the digital transmission hierarchies used in different countries, it was not possible to get agreement on a single data rate. The United States, Canada, and Japan make use of a transmission structure based on 1.544 Mbps; this corresponds to the T1 transmission facility using the DS-1 transmission format. In Europe, 2.048 Mbps is the standard rate. Both of these data rates are provided as a primary interface service. Typically, the channel structure for the 1.544-Mbps rate is 23 B channels plus one 64-kbps D channel and, for the 2.048-Mbps rate, 30 B channels plus one 64-kbps D channel. Again, it is possible for a customer with lower requirements to employ fewer B channels, in which case the channel structure is *n*B+D, where *n* ranges from 1 to 23, or 1 to 30 for the two primary services. Also, a customer with high data-rate demands may be provided with more than one primary physical interface. In this case, a single D channel on one of the interfaces may suffice for all signaling needs, and the other interfaces may consist solely of B channels (24B or 31B). The frame structure for primary access was shown in Figure 7.11.

The primary interface may also be used to support H channels. Some of these structures include a 64-kbps D channel for control signaling. When no D channel is present, it is assumed that a D channel on another primary interface at the same subscriber location will provide any required signaling. The following structures are recognized:

- **Primary rate interface H0 channel structures.** This interface supports multiple 384-kbps H0 channels. The structures are 3H0 + D and 4H0 for the 1.544-Mbps interface, and 5H0 + D for the 2.048-Mbps interface.

- **Primary rate interface H1 channel structures.** The H11 channel structure consists of one 1536-kbps H11 channel. The H12 channel structure consists of one 1920-kbps H12 channel and one D channel.
- **Primary rate interface structures for mixtures of B and H0 channels.** This interface consists of 0 or 1 D channels plus any possible combination of B and H0 channels, up to the capacity of the physical interface (e.g., 3H0 + 5B + D and 3H0 + 6B).

A.3 USER ACCESS

To define the requirements for ISDN user access, an understanding of the anticipated configuration of user premises equipment and of the necessary standard interfaces is critical. The first step is to group functions that may exist on the user's premises. Figure A.4 shows the CCITT approach to this task, using

- **Functional groupings.** Certain finite arrangements of physical equipment or combinations of equipment.
- **Reference points.** Conceptual points used to separate groups of functions.

The architecture on the subscriber's premises is broken up functionally into groupings separated by reference points. This separation permits interface standards to be developed at each reference point; this effectively organizes the standards work and provides guidance to the equipment providers. Once stable interface standards exist, technical improvements on either side of an interface can be

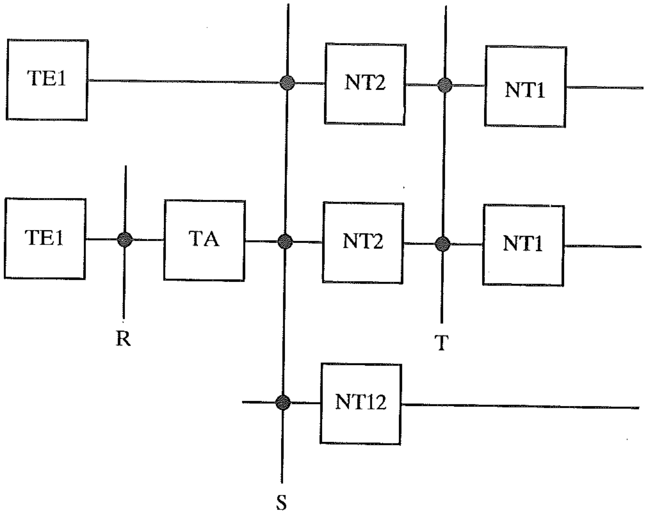


FIGURE A.4 ISDN reference points and functional groupings.

made without impacting adjacent functional groupings. Finally, with stable interfaces, the subscriber is free to procure equipment from different suppliers for the various functional groupings, so long as the equipment conforms to the relevant interface standards.

Network termination 1 (NT1) includes functions associated with the physical and electrical termination of the ISDN on the user's premises; these correspond to OSI layer 1. The NT1 may be controlled by the ISDN provider and forms a boundary to the network. This boundary isolates the user from the transmission technology of the subscriber loop and presents a physical connector interface for user device attachment. In addition, the NT1 performs line maintenance functions such as loopback testing and performance monitoring. The NT1 supports multiple channels (e.g., 2B + D); at the physical level, the bit streams of these channels are multiplexed together, using synchronous time-division multiplexing. Finally, the NT1 interface might support multiple devices in a multidrop arrangement. For example, a residential interface might include a telephone, personal computer, and alarm system, all attached to a single NT1 interface via a multidrop line.

Network termination 2 (NT2) is an intelligent device that can perform switching and concentration functions; it may include functionality up through layer 3 of the OSI model. Examples of NT2 are a digital PBX, a terminal controller, and a LAN. An example of a switching function is the construction of a private network using semipermanent circuits among a number of sites, each of which could include a PBX that acts as a circuit switch, or a host computer that acts as a packet switch. The concentration function simply means that multiple devices, attached to a digital PBX, LAN, or terminal controller, may transmit data across an ISDN.

Network termination 1, 2 (NT12) is a single piece of equipment that contains the combined functions of NT1 and NT2; this points out one of the regulatory issues associated with ISDN interface development. In many countries, the ISDN provider will own the NT12 and provide full service to the user. In the United States, there is a need for a network termination with a limited number of functions to permit competitive provision of user premises equipment. Hence, the user premises network functions are split into NT1 and NT2.

Terminal equipment refers to subscriber equipment that makes use of ISDN; two types are defined. *Terminal equipment type 1* (TE1) refers to devices that support the standard ISDN interface. Examples are digital telephones, integrated voice/data terminals, and digital facsimile equipment. Terminal equipment type 2 (TE2) encompasses existing non-ISDN equipment. Examples are terminals with a physical interface, such as EIA-232-E, and host computers with an X.25 interface. Such equipment requires a terminal adapter (TA) to plug into an ISDN interface.

The definitions of the functional groupings also define, by implication, the reference points. *Reference point T* (terminal) corresponds to a minimal ISDN network termination at the customer's premises; it separates the network provider's equipment from the user's equipment. *Reference point S* (system) corresponds to the interface of individual ISDN terminals and separates user terminal equipment from network-related communications functions. *Reference point R* (rate) provides a non-ISDN interface between user equipment that is not ISDN-compatible and adapter equipment. Typically, this interface will comply with an older interface standard, such as EIA-232-E.

A.4 ISDN PROTOCOLS

ISDN Protocol Architecture

Figure A.5 illustrates, in the context of the OSI model, the protocols defined or referenced in the ISDN documents. As a network, ISDN is essentially unconcerned with user layers 4–7. These are end-to-end layers employed by the user for the exchange of information. Network access is concerned only with layers 1–3. Layer 1, defined in I.430 and I.431, specifies the physical interface for both basic and primary access. Because B and D channels are multiplexed over the same physical interface, these standards apply to both types of channels. Above this layer, the protocol structure differs for the two channels.

For the D channel, a new data link layer standard, LAPD (Link Access Protocol, D channel) has been defined. This standard is based on HDLC, modified to meet ISDN requirements. All transmission on the D channel is in the form of LAPD frames that are exchanged between the subscriber equipment and an ISDN switching element. Three applications are supported: control signaling, packet-switching, and telemetry. For **control signaling**, a call control protocol has been

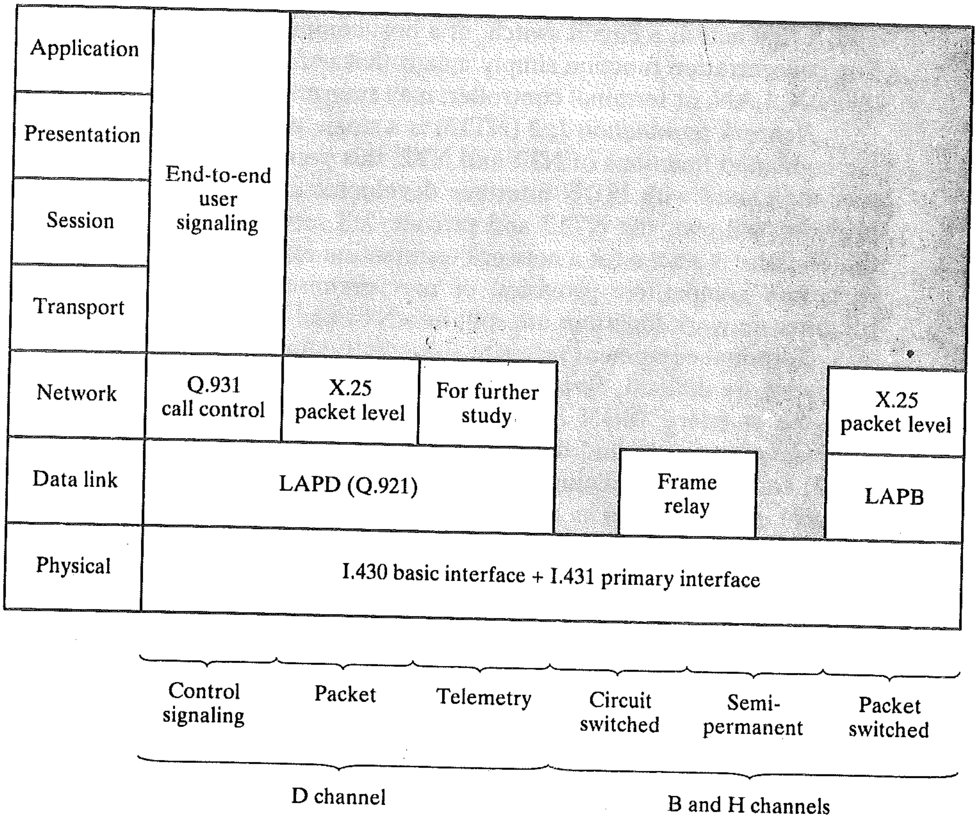


FIGURE A.5 ISDN protocols at the user-network interface.

defined (I.451/Q.931). This protocol is used to establish, maintain, and terminate connections on B channels; thus, it is a protocol between the user and the network. Above layer 3, there is the possibility for higher-layer functions associated with user-to-user control signaling. These functions are a subject for further study. The D channel can also be used to provide *packet-switching* services to the subscriber. In this case, the X.25 level-3 protocol is used, and X.25 packets are transmitted in LAPD frames. The X.25 level-3 protocol is used to establish virtual circuits on the D channel to other users and to exchange packetized data. The final application area, *telemetry*, is a subject for further study.

The B channel can be used for circuit switching, semipermanent circuits, and packet-switching. For *circuit switching*, a circuit is set up on a B channel, on demand. The D-channel call control protocol is used for this purpose. Once the circuit is set up, it may be used for data transfer between the users. A *semipermanent circuit* is a B-channel circuit that is set up by prior agreement between the connected users and the network. As with a circuit-switched connection, it provides a transparent data path between end systems.

With either a circuit-switched connection or a semipermanent circuit, it appears to the connected stations that they have a direct, full-duplex link with each other. They are free to use their own formats, protocols, and frame synchronization. Hence, from the point of view of ISDN, layers 2 through 7 are not visible or specified.

In the case of *packet-switching*, a circuit-switched connection is set up on a B channel between the user and a packet-switched node using the D-channel control protocol. Once the circuit is set up on the B channel, the user may employ X.25 layers 2 and 3 to establish a virtual circuit to another user over that channel and to exchange packetized data. As an alternative, the frame relay service may be used. Frame relay can also be used over H channels and over the D channel.

Some of the protocols shown in Figure A.5 are summarized in the remainder of this section. First, we look at the way in which packet-switched and circuit-switched connections are set up. Next, we examine the control-signaling protocol and then LAPD. Finally, the physical-layer specifications are reviewed.

ISDN Connections

ISDN provides four types of service for end-to-end communication:

- Circuit-switched calls over a B channel.
- Semipermanent connections over a B channel.
- Packet-switched calls over a B channel.
- Packet-switched calls over the D channel.

Circuit-Switched Calls

The network configuration and protocols for circuit switching involve both the B and D channels. The B channel is used for the transparent exchange of user data. The communicating users may employ any protocols they wish for end-to-end communication. The D channel is used to exchange control information between the user and the network for call establishment and termination, as well as to gain access to network facilities.

The B channel is serviced by an NT1 or NT2 using only layer-1 functions. On the D channel, a three-layer network access protocol is used and is explained below. Finally, the process of establishing a circuit through ISDN involves the cooperation of switches internal to ISDN to set up the connection. These switches interact by using an internal protocol: Signaling-System Number 7.

Semipermanent Connections

A semipermanent connection between agreed points may be provided for an indefinite period of time after subscription, for a fixed period, or for agreed-upon periods during a day, a week, or some other interval. As with circuit-switched connections, only Layer-1 functionality is provided by the network interface. The call-control protocol is not needed because the connection already exists.

Packet-Switched Calls over a B Channel

The ISDN must also permit user access to packet-switched services for data traffic (e.g., interactive) that is best serviced by packet switching. There are two possibilities for implementing this service: Either the packet-switching capability is furnished by a separate network, referred to as a packet-switched public data network (PSPDN), or the packet-switching capability is integrated into ISDN. In the former case, the service is provided over a B channel. In the latter case, the service may be provided over a B or D channel. We first examine the use of a B channel for packet-switching.

When the packet-switching service is provided by a separate PSPDN, the access to that service is via a B channel. Both the user and the PSPDN must therefore be connected as subscribers to the ISDN. In the case of the PSPDN, one or more of the packet-switching network nodes, referred to as packet handlers, are connected to ISDN. We can think of each such node as a traditional X.25 DCE supplemented by the logic needed to access ISDN. That is, the ISDN subscriber assumes the role of an X.25 DTE, the node in the PSPDN to which it is connected functions as an X.25 DCE, and the ISDN simply provides the connection from DTE to DCE. Any ISDN subscriber can then communicate, via X.25, with any user connected to the PSPDN, including

- Users with a direct, permanent connection to the PSPDN.
- Users of the ISDN that currently enjoy a connection, through the ISDN, to the PSPDN.

The connection between the user (via a B channel) and the packet handler with which it communicates may be either semipermanent or circuit-switched. In the former case, the connection is always there and the user may freely invoke X.25 to set up a virtual circuit to another user. In the latter case, the D channel is involved, and the following sequence of steps occurs (Figure A.6):

1. The user requests, via the D-channel call-control protocol (I.451/Q.931), a circuit-switched connection on a B channel to a packet handler.
2. The connection is set up by ISDN, and the user is notified via the D channel call-control protocol.

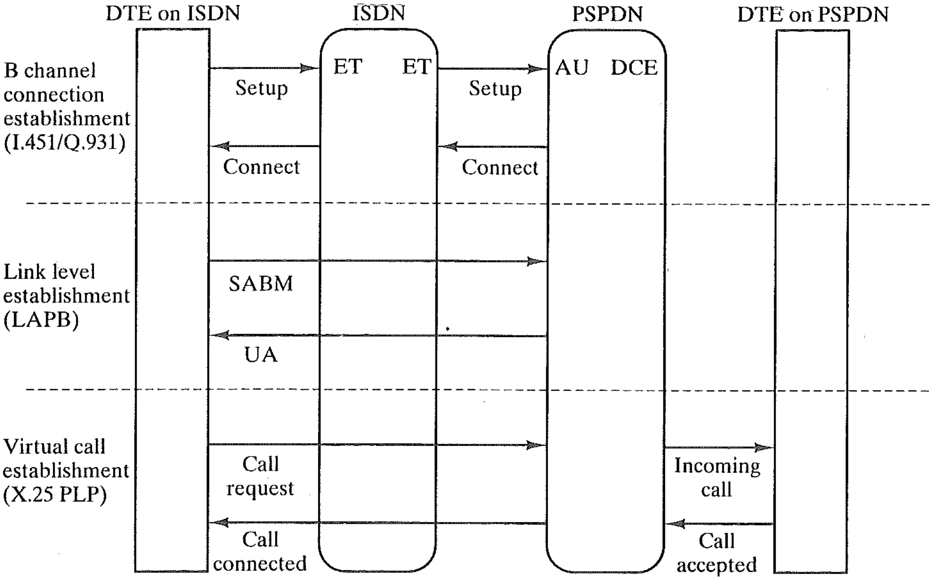
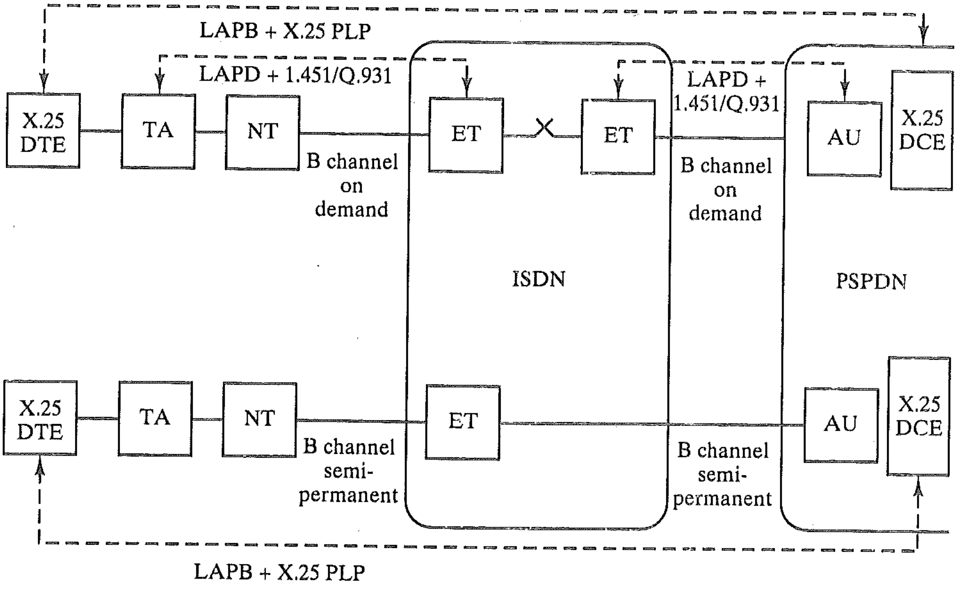


FIGURE A.6 Virtual call setup.

3. The user sets up a virtual circuit to another user via the X.25 call establishment procedure on the B channel (described in Section 3.2). This step requires first that a data link connection, using LAPB, must be set up between the user and the packet handler.
4. The user terminates the virtual circuit, using X.25 on the B channel.
5. After one or more virtual calls on the B channel, the user is done and signals via the D channel to terminate the circuit-switched connection to the packet-switching node.
6. The connection is terminated by ISDN.

Figure A.7 shows the configuration involved in providing this service. In the figure, the user is shown to employ a DTE device that expects an interface to an X.25 DCE. Hence, a terminal adapter is required. Alternatively, the X.25 capability can be an integrated function of an ISDN TE1 device, dispensing with the need for a separate TA.

When the packet-switching service is provided by ISDN, the packet-handling function is provided within the ISDN, either by separate equipment or as part of the exchange equipment. The user may connect to a packet handler either by a B channel or the D channel. On a B channel, the connection to the packet handler may be either switched or semipermanent, and the same procedures described above apply for switched connections. In this case, rather than establish a B-channel connection to another ISDN subscriber that is a PSPDN packet handler, the connection is to an internal element of ISDN that is a packet handler.



LEGEND

AU = ISDN access unit
TA = Terminal adapter
NT = Network termination 2 and/or 1

ET = Exchange termination
PLP = Packet-level procedure
PSPDN = Packet-switched public data network

FIGURE A.7 Access to PSPDN for packet-mode service.

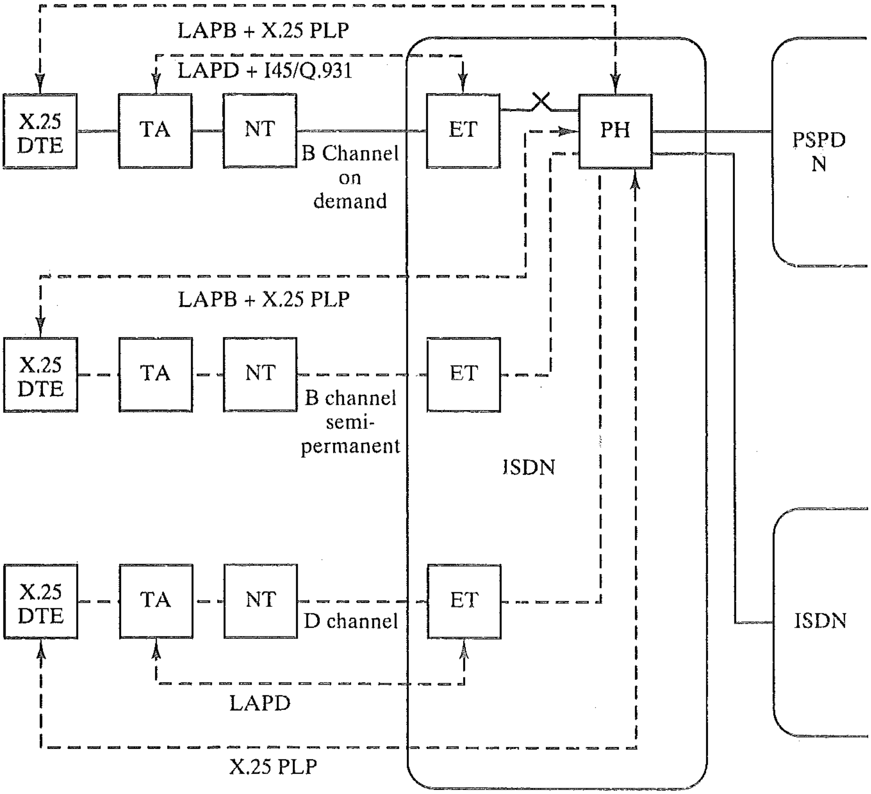
Packet-Switched Calls over a D Channel

When the packet-switching service is provided internal to the ISDN, it can also be accessed on the D channel. For this access, ISDN provides a semipermanent connection to a packet-switching node within the ISDN. The user employs the X.25 level-3 protocol, as is done in the case of a B-channel virtual call. Here, the level-3 protocol is carried by LAPD frames. Because the D channel is also used for control signaling, some means is needed to distinguish between X.25 packet traffic and ISDN control traffic; this is accomplished by means of the link-layer addressing scheme, as explained below.

Figure A.8 shows the configuration for providing packet-switching within ISDN. The packet-switching service provided internal to the ISDN over the B and D channels is logically provided by a single packet-switching network. Thus, virtual calls can be set up between two D-channel users, two B-channel users, and between a B and D channel user. In addition, it will be typical to also provide access to X.25 users on other ISDNs and PSPDNs by appropriate interworking procedures.

Common Channel Signaling at the ISDN User-Network Interface

ITU-T has developed a standard, I.451, for common channel signaling. The primary application of this standard is for the Integrated Services Digital Network. In OSI terms, I.451 is a layer-3, or network-layer, protocol. As Figure A.9 indicates, I.451



LEGEND

TA = Terminal adapter
NT = Network termination 2 and/or 1
ET = Exchange termination

PLP = Packet-level procedure
PSPDN = Packet-switched public data network
PH = Packet handling function

FIGURE A.8 Access to ISDN for packet-mode service.

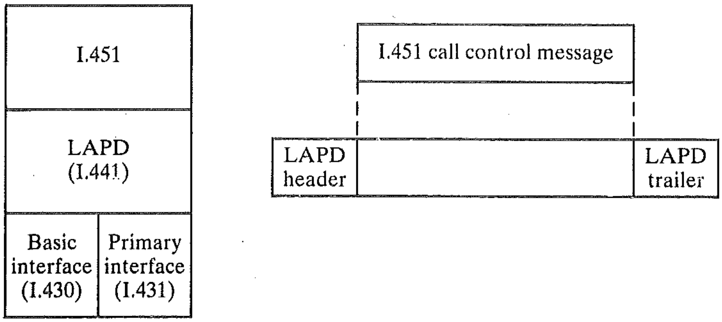
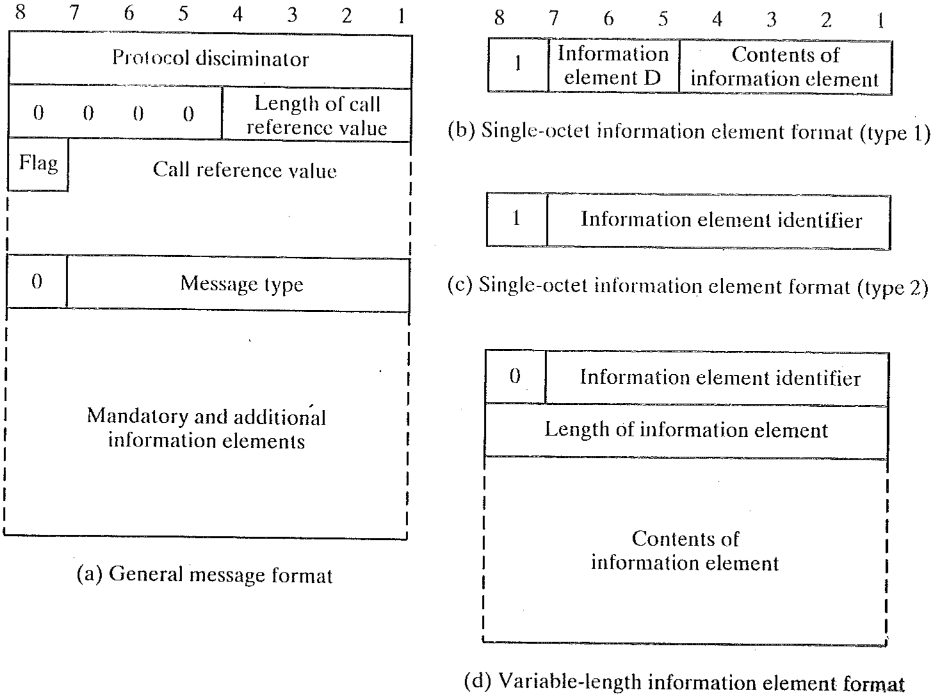


FIGURE A.9 Call control communications architecture.

relies on a link layer protocol to transmit messages over the D channel. I.451 specifies procedures for establishing connections on the B channels that share the same physical interface to ISDN as the D channel. It also provides user-to-user control signaling over the D channel.

The process of establishing, controlling, and terminating a call occurs as a result of control-signaling messages exchanged between the user and the network over a D channel. A common format is used for all messages defined in I.451, illustrated in Figure A.10a. Three fields are common to all messages:

- **Protocol discriminator.** Used to distinguish messages for user-network call control from other message types. Other sorts of protocols may share the common signaling channel.
- **Call reference.** Identifies the user-channel call to which this message refers. As with X.25 virtual circuit numbers, it has only local significance. The call reference field comprises three subfields. The length subfield specifies the length of the remainder of the field in octets. This length is one octet for a basic-rate interface, and two octets for a primary-rate interface. The flag indicates which end of the LAPD logical connection initiated the call.
- **Message type.** Identifies which I.451 message is being sent. The contents of the remainder of the message depend on the message type.


8 7 6 5 4 3 2 1
Protocol discriminator
0 0 0 0
Length of call reference value
Flag
Call reference value
0
Message type
Mandatory and additional information elements
8 7 6 5 4 3 2 1
1
Information element D
Contents of information element
1
Information element identifier
0
Information element identifier
Length of information element
Contents of information element

(a) General message format

(b) Single-octet information element format (type 1)

(c) Single-octet information element format (type 2)

(d) Variable-length information element format

FIGURE A.10 I.451 formats.

Following these three common fields, the remainder of the message consists of a sequence of zero or more information elements, or parameters. These contain additional information to be conveyed with the message. Thus, the message type specifies a command or response, and the details are provided by the information elements. Some information elements must always be included with a given message (mandatory), and others are optional (additional). Three formats for information elements are used, as indicated in Figure A.10b through d.

Table A.3 lists the I.451 messages. The messages can be grouped along two dimensions. Messages apply to one of four applications: circuit-mode control, packet-mode access connection control, user-to-user signaling not associated with circuit-switched calls, and messages used with the global call reference. In addition, messages perform functions in one of four categories: call establishment, call information, call clearing, and miscellaneous.

Circuit-mode control refers to the functions needed to set up, maintain, and clear a circuit-switched connection on a user channel. This function corresponds to call control in existing circuit-switching telecommunications networks. *Packet-mode access connection control* refers to the functions needed to set up a circuit-switched connection (called an access connection in this context) to an ISDN packet-switching node; this connects the user to the packet-switching network provided by the ISDN provider. *User-to-user signaling messages* allow two users to communicate without setting up a circuit-switched connection. A temporary signaling connection is established and cleared in a manner similar to the control of a circuit-switched connection. Signaling takes place over the signaling channel and thus does not consume user-channel resources. Finally, global call reference refers to the functions that enable either user or network to return one or more channels to an idle condition.

Call establishment messages are used to initially set up a call. This group includes messages between the calling terminal and the network, and between the network and the called terminal. These messages support the following services:

- Set up a user-channel call in response to user request.
- Provide particular network facilities for this call.
- Inform calling user of the progress of the call establishment process.

Once a call has been set up, but prior to the disestablishment (termination) phase, *call-information* phase messages are sent between user and network. One of the messages in this group allows the network to relay, without modification, information between the two users of the call. The nature of this information is beyond the scope of the standard, but it is assumed that it is control signaling information that can't or should not be sent directly over the user-channel circuit. The remainder of the messages allow users to request both the suspension and later resumption of a call. When a call is suspended, the network remembers the identity of the called parties and the network facilities supporting the call, but it deactivates the call so that no additional charges are incurred and so that the corresponding user channel is freed up. Presumably, the resumption of a call is quicker and cheaper than the origination of a new call.

TABLE A.3 I.451/Q.931 messages for circuit-mode connection control.

Message	Significance	Direction	Function
Call establishment messages			
ALERTING	global	both	Indicates that user alerting has begun
CALL PROCEEDING	local	both	Indicates that call establishment has been initiated
CONNECT	global	both	Indicates call acceptance by called TE
CONNECT ACKNOWLEDGE	local	both	Indicates that user has been awarded the call
PROGRESS	global	both	Reports progress of a call
SETUP	global	both	Initiates call establishment
SETUP ACKNOWLEDGE	local	both	Indicates that call establishment has been initiated but requests more information
Call information phase messages			
RESUME	local	u → n	Requests resumption of previously suspended call
RESUME ACKNOWLEDGE	local	n → u	Indicates requested call has been reestablished
RESUME REJECT	local	n → u	Indicates failure to resume suspended call
SUSPEND	local	u → n	Requests suspension of a call
SUSPEND ACKNOWLEDGE	local	n → u	Call has been suspended
SUSPEND REJECT	local	n → u	Indicates failure of requested call suspension
USER INFORMATION	access	both	Transfers information from one user to another
Call clearing messages			
DISCONNECT	global	both	Sent by user to request connection clearing; sent by network to indicate connection clearing
RELEASE	local	both	Indicates intent to release channel and call reference.
RELEASE COMPLETE	local	both	Indicates release of channel and call reference
Miscellaneous messages			
CONGESTION CONTROL	local	both	Sets or releases flow control on USER INFORMATION messages
FACILITY	local	both	Requests or acknowledges a supplementary service
INFORMATION	local	both	Provides additional information
NOTIFY	access	both	Indicates information pertaining to a call
STATUS	local	both	Sent in response to a STATUS ENQUIRY or at any time to report an error
STATUS ENQUIRY	local	both	Solicits STATUS message

Call-clearing messages are sent between user and network in order to terminate a call. Finally, there are some *miscellaneous messages* that may be sent between user and network at various stages of the call. Some may be sent during call setup; others may be sent even though no calls exist. The primary function of these messages is to negotiate network features (supplementary services).

LAPD

All traffic over the D channel employs a link-layer protocol known as LAPD (Link Access Protocol-D Channel).

LAPD Services

The LAPD standard provides two forms of service to LAPD users: the unacknowledged information-transfer service and the acknowledged information-transfer service. The *unacknowledged information-transfer service* simply provides for the transfer of frames containing user data with no acknowledgment. The service does not guarantee that data presented by one user will be delivered to another user, nor does it inform the sender if the delivery attempt fails. The service does not provide any flow control or error-control mechanism. This service supports both point-to-point (deliver to one user) or broadcast (deliver to a number of users); it allows for fast data transfer and is useful for management procedures such as alarm messages and messages that need to be broadcast to multiple users.

The *acknowledged information-transfer* is the more common service, and is similar to that offered by LAP-B and HDLC. With this service, a logical connection is established between two LAPD users prior to the exchange of data.

LAPD Protocol

The LAPD protocol is based on HDLC. Both user information and protocol-control information and parameters are transmitted in frames. Corresponding to the two types of service offered by LAPD, there are two types of operation:

- **Unacknowledged operation.** Layer-3 information is transferred in unnumbered frames. Error detection is used to discard damaged frames, but there is no error control or flow control.
- **Acknowledged operation.** Layer-3 information is transferred in frames that include acknowledged sequence numbers. Error control and flow control procedures are included in the protocol. This type is also referred to in the standard as multiple-frame operation.

These two types of operation may coexist on a single D channel, and both make use of the frame format illustrated in Figure A.11. This format is identical to that of HDLC (Figure 6.10), with the exception of the address field.

To explain the address field, we need to consider that LAPD has to deal with two levels of multiplexing. First, at the subscriber site, there may be multiple user devices sharing the same physical interface. Second, within each user device, there may be multiple types of traffic: specifically, packet-switched data and control signaling. To accommodate these levels of multiplexing, LAPD employs a two-part

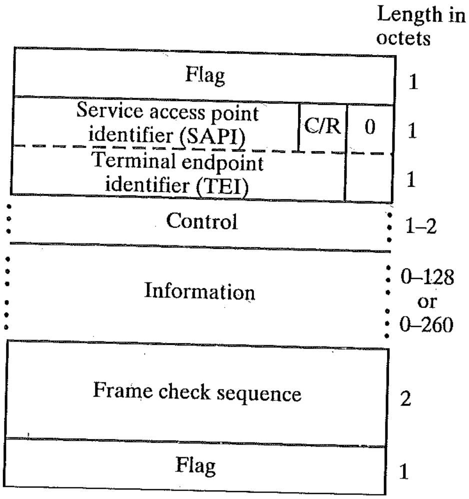


FIGURE A.11 LAP-D format.

address, consisting of a terminal endpoint identifier (TEI) and a service access point identifier (SAPI).

Typically, each user device is given a unique *terminal endpoint identifier* (TEI). It is also possible for a single device to be assigned more than one TEI; this might be the case for a terminal concentrator. TEI assignment occurs either automatically when the equipment first connects to the interface, or manually by the user. In the latter case, care must be taken that multiple equipment attached to the same interface do not have the same TEI. The advantage of the automatic procedure is that it allows the user to change, add, or delete equipment at will without prior notification to the network administration. Without this feature, the network would be obliged to manage a data base for each subscriber that would need to be updated manually. Table A.4a shows the assignment of TEI numbers.

The *service access point identifier* (SAPI) identifies a layer-3 user of LAPD, and thus corresponds to a layer-3 protocol entity within a user device. Four specific values have been assigned, as shown in Table A.4b. A SAPI of 0 is used for call-control procedures for managing B-channel circuits; the value 16 is reserved for packet-mode communication on the D channel using X.25 level 3; and a value of 63 is used for the exchange of layer-2 management information. Finally, values in the range 32 to 62 are reserved to support frame-relay connections.

For acknowledged operation, LAPD follows essentially the same procedures described for HDLC in Chapter 6. For unacknowledged operation, the user information (UI) frame is used to transmit user data. When a LAPD user wishes to send data, it passes the data to its LAPD entity, which passes the data in the information field of a UI frame. When this frame is received, the information field is passed up to the destination user. There is no acknowledgment returned to the other side. However, error detection is performed, and frames in error are discarded.

TABLE A.4 SAPI and TEI assignments.

(a) TEI Assignments

TE1 value	User type
0–63	Nonautomatic TEI assignment user equipment
64–126	Automatic TEI assignment user equipment
127	Used during automatic TEI assignment

(b) SAPI Assignments

SAPI value	Related protocol or management entity
0	Call control procedures
16	Packet communication conforming to X.25 level 3
32–61	Frame relay communication
63	Layer 2 management procedures
All others	Reserved for future standardization

Physical Layer

The ISDN physical layer is presented to the user at either reference point S or T (Figure A.4). The mechanical interface was described in Chapter 5.

The electrical specification depends on the specific interface. For the basic-access interface, pseudoternary coding is used (Figure 4.2). Recall that with pseudoternary, the line signal may take one of three levels; this is not as efficient as a two-level code, but it is reasonably simple and inexpensive. At the relatively modest data rate of the basic-access interface, this is a suitable code.

For the higher-speed, primary-access interface, a more efficient coding scheme is needed. For the 1.544-Mbps data rate, the B8ZS code is used, while for the 2.048-Mbps data rate, the HDB3 code is used (Figure 4.6). There is no particular advantage of one over the other; the specification reflects historical usage.

The functional specification for the physical layer includes the following functions:

- Full-duplex transmission of B-channel data
- Full-duplex transmission of D-channel data
- Full-duplex transmission of timing signals
- Activation and deactivation of physical circuit
- Power feeding from the network termination to the terminal
- Terminal identification
- Faulty-terminal isolation
- D-channel–contention access

The final function is required when multiple TE1 terminals share a single physical interface (i.e., a multipoint line). In that case, no additional functionality is needed to control access in the B channels, as each channel is dedicated to a particular circuit at any given time. However, the D channel is available for use by all the devices for both control signaling and for packet transmission. For incoming data, the LAPD addressing scheme is sufficient to sort out the proper destination for each data unit. For outgoing data, some sort of contention resolution protocol is needed to assure that only one device at a time attempts to transmit. The D-channel, contention-resolution algorithm was described in Chapter 7.

A.5 BROADBAND ISDN

In 1988, as part of its I-series of recommendations on ISDN, CCITT issued the first two recommendations relating to broadband (B-ISDN): I.113, Vocabulary of Terms for Broadband Aspects of ISDN; and I.121, Broadband Aspects of ISDN. These documents provided a preliminary description and a basis for future standardization and development work, and from those documents a rich set of recommendations has been developed. Some of the important notions developed in these documents are presented in Table A.5.

TABLE A.5 Noteworthy statements in I.113 and I.121.

Broadband: A service or a system requiring transmission channels capable of supporting rates greater than the primary rate.
The term B-ISDN is used for convenience in order to refer to and emphasize the broadband aspects of ISDN. The intent, however, is that there be one comprehensive notion of an ISDN which provides broadband and other ISDN services.
Asynchronous transfer mode (ATM) is the transfer mode for implementing B-ISDN and is independent of the means of transport at the Physical Layer.
B-ISDN will be based on the concepts developed for ISDN and may evolve by progressively incorporating directly into the network additional B-ISDN functions enabling new and advanced services.
Since the B-ISDN is based on overall ISDN concepts, the ISDN access reference configuration is also the basis for the B-ISDN reference configuration.

CCITT modestly defines B-ISDN as "a service requiring transmission channels capable of supporting rates greater than the primary rate." Behind this innocuous statement lie plans for a network and a set of services that will have far more impact on business and residential customers than ISDN. With B-ISDN, services, especially video services, requiring data rates in excess of those that can be delivered by ISDN will become available. To contrast this new network and these new services to the original concept of ISDN, that original concept is now being referred to as narrowband ISDN.

Broadband ISDN Architecture

B-ISDN differs from a narrowband ISDN in a number of ways. To meet the requirement for high-resolution video, an upper channel rate of approximately

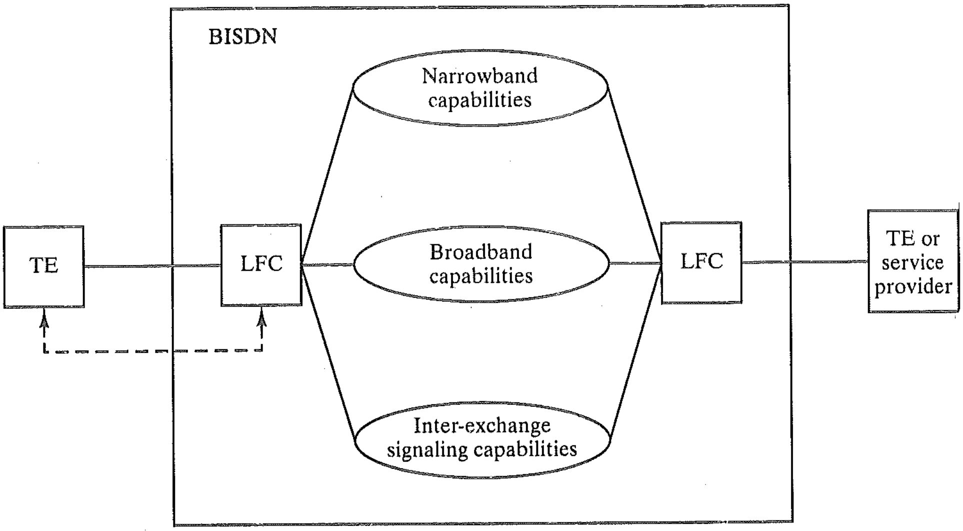
150 Mbps is needed. To simultaneously support one or more interactive and distributive services, a total subscriber line rate of about 600 Mbps is needed. In terms of today's installed telephone plant, this is a stupendous data rate to sustain. The only appropriate technology for widespread support of such data rates is optical fiber. Hence, the introduction of B-ISDN depends on the pace of introduction of fiber subscriber loops.

Internal to the network, there is the issue of the switching technique to be used. The switching facility has to be capable of handling a wide range of different bit rates and traffic parameters (e.g., burstiness). Despite the increasing power of digital circuit-switching hardware and the increasing use of optical fiber trunking, it is difficult to handle the large and diverse requirements of B-ISDN with circuit-switching technology. For this reason, there is increasing interest in some type of fast packet-switching as the basic switching technique for B-ISDN. This form of switching readily supports ATM at the user-network interface.

Functional Architecture

Figure A.12 depicts the functional architecture of B-ISDN. As with narrowband ISDN, control of B-ISDN is based on common-channel signaling. Within the network, an SS7, enhanced to support the expanded capabilities of a higher-speed network, is used. Similarly, the user-network control-signaling protocol is an enhanced version of I.451/Q.931.

B-ISDN must, of course, support all of the 64-kbps transmission services, both circuit-switching and packet-switching, that are supported by narrowband ISDN;



LEGEND
LFC = local function capabilities TE = terminal equipment

FIGURE A.12 BISDN architecture.

this protects the user's investment and facilitates migration from narrowband to broadband ISDN. In addition, broadband capabilities are provided for higher data-rate transmission services. At the user-network interface, these capabilities will be provided with the connection-oriented asynchronous transfer mode (ATM) facility.

User-Network Interface

The reference configuration defined for narrowband ISDN is considered general enough to be used for B-ISDN. Figure A.13, which is almost identical to Figure A.4, shows the reference configuration for B-ISDN. In order to clearly illustrate the broadband aspects, the notations for reference points and functional groupings are appended with the letter B (e.g., B-NT1, TB). The broadband functional groups are equivalent to the functional groups defined for narrowband ISDN, and are discussed below. Interfaces at the R reference point may or may not have broadband capabilities.

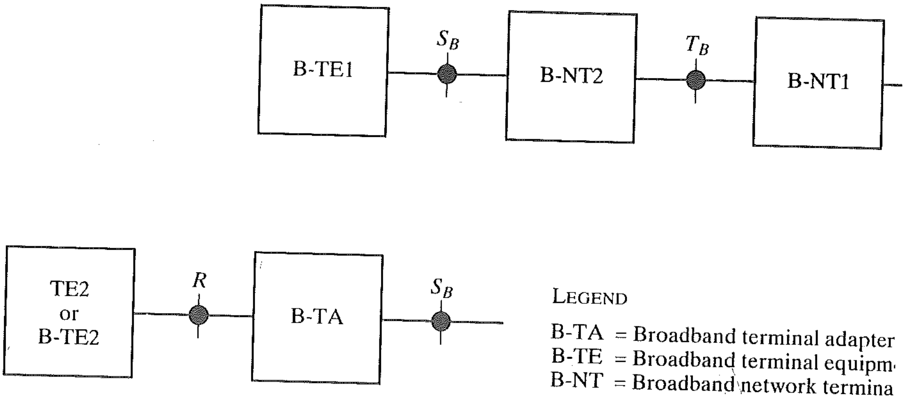

B-TE1
S_B
B-NT2
T_B
B-NT1
TE2
or
B-TE2
R
B-TA
S_B
LEGEND
B-TA = Broadband terminal adapter
B-TE = Broadband terminal equipm
B-NT = Broadband network termina

FIGURE A.13 B ISDN reference configurations.

Transmission Structure

In terms of data rates available to B-ISDN subscribers, three new transmission services are defined. The first of these consists of a full-duplex 155.52-Mbps service. The second service defined is asymmetrical, providing transmission from the subscriber to the network at 155.52 Mbps, and in the other direction at 622.08 Mbps; and the highest-capacity service yet defined is a full-duplex, 622.08-Mbps service.

A data rate of 155.52 Mbps can certainly support all of the narrowband ISDN services. That is, such a rate readily supports one or more basic- or primary-rate interfaces; in addition, it can support most of the B-ISDN services. At that rate, one or several video channels can be supported, depending on the video resolution and the coding technique used. Thus, the full-duplex 155.52-Mbps service will probably be the most common B-ISDN service.

The higher data rate of 622.08 Mbps is needed to handle multiple video distribution, such as might be required when a business conducts multiple simultaneous videoconferences. This data rate makes sense in the network-to-subscriber direction. The typical subscriber will not initiate distribution services and thus would still be able to use the lower, 155.52-Mbps service. The full-duplex, 622.08-Mbps service would be appropriate for a video-distribution provider.

Broadband ISDN Protocols

The protocol architecture for B-ISDN introduces some new elements not found in the ISDN architecture, as depicted in Figure A.14. For B-ISDN, it is assumed that the transfer of information across the user-network interface will use ATM.

The decision to use ATM for B-ISDN is a remarkable one; it implies that B-ISDN will be a packet-based network, certainly at the interface, and almost certainly in terms of its internal switching. Although the recommendation also states that B-ISDN will support circuit-mode applications, this will be done over a packet-based transport mechanism. Thus, ISDN, which began as an evolution from the circuit-switching telephone network, will transform itself into a packet-switching network as it takes on broadband services.

The protocol reference model makes reference to three separate planes:

- **User Plane.** Provides for user-information transfer, along with associated controls (e.g., flow control, error control).
- **Control Plane.** Performs call-control and connection-control functions.
- **Management Plane.** Includes plane management, which performs management functions related to a system as a whole and provides coordination between all the planes, and layer management, which performs management functions relating to resources and parameters residing in its protocol entities.

Table A.6 highlights the functions to be performed at each sublayer.

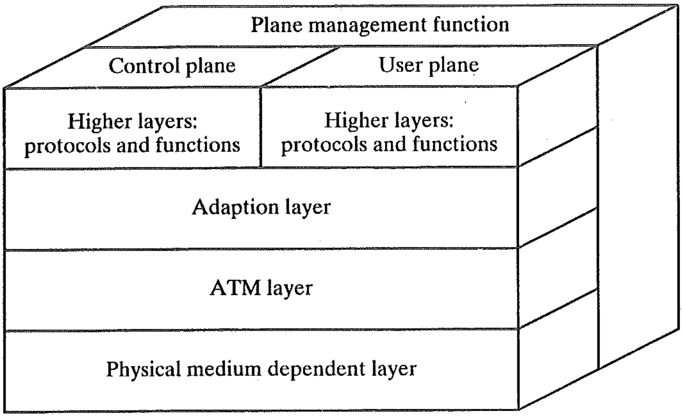

Plane management function
Control plane
User plane
Higher layers: protocols and functions
Higher layers: protocols and functions
Adaption layer
ATM layer
Physical medium dependent layer

FIGURE A.14 B-ISDN protocol model for ATM.

TABLE A.6 Functions of the B-ISDN layers.

Layer Management	Higher-layer functions	Higher layers	
	Convergence	CS	AAL
	Segmentation and reassembly	SAR	
	Generic flow control Cell header generation/extraction Cell VPI/VCI translation Cell multiplex and demultiplex	ATM	
	Cell rate decoupling HEC header sequence generation/verification Cell delineation Transmission frame adaptation Transmission frame generation/recovery	TC	Physical layer
	Bit timing Physical medium	PM	

CS = Covergence sublayer
SAR = Segmentation and reassembly sublayer
AAL = ATM adaptation layer
ATM = Asynchronous transfer mode
TC = Transmission control sublayer
PM = Physical medium sublayer

A.6 RECOMMENDED READING

A detailed technical treatment of ISDN and broadband ISDN can be found in [STAL95a]. Other book-length treatments include [KESS93] and [HELG91].

HELG91 Helgert, H. *Integrated Services Digital Networks: Architectures, Protocols, and Standards*. Reading, MA: Addison-Wesley, 1991.

KESS93 Kessler, G. *ISDN: Concepts, Facilities, and Services*. New York: McGraw-Hill, 1993.

STAL95a Stallings, W. *ISDN and Broadband ISDN, with Frame Relay and ATM*. Upper Saddle River, NJ: Prentice Hall, 1995.

Recommended Web Site

- http:/alumni.caltech.edu/~dank/isdn: Information on ISDN tariffs, standards status, and links to vendors.

A.7 PROBLEMS

A.1 It was mentioned that user-implemented multidrop lines and multiplexers may disappear. Explain why.

A.2 An ISDN customer has offices at a number of sites. A typical office is served by two 1.544-Mbps digital pipes. One provides circuit-switched access to the ISDN; the other is a leased line connecting to another user site. The on-premises equipment consists of a PBX aligned with packet-switching node logic. The user has three requirements:

- Telephone service
- A private packet-switched network for data
- Video teleconferencing at 1.544 Mbps

How might the user allocate capacity optimally to meet these requirements?

A.3 An ISDN basic access frame has 32 B bits and 4 D bits. Suppose that more bits were used—say 160 B bits and 20 D bits per frame. Would this reduce the percentage overhead and therefore the basic-access bit rate? If so, discuss any potential disadvantages.

A.4 Under what circumstances would user layer 3 on the B channel not be null?

A.5 Compare the addressing schemes in HDLC, LLC, and LAPD:

a. Are the SAPI of LAP-D and the SAP of LLC the same thing?

b. Are the TEI of LAP-D and the MAC-level address of IEEE 802 the same thing?

c. Why are two levels of addressing needed for LAP-D and LLC, but only one level for HDLC?

d. Why does LLC need a source address, but LAP-D and HDLC do not?

A.6 What is the percentage overhead on the basic channel structure?

A.7 From Figure A.5, it would appear that layers 4–7 of the OSI model are little affected by ISDN. Would you expect this? Why or why not?

A.8 X.25 and most other layer-3 protocols provide techniques for flow control and error control. Why are such features not provided in I.451?

Appendix B

RFCs Cited in This Book

RFC Number	Title	Date
768	User Datagram Protocol (UDP)	August 1980
791	Internet Protocol (IP)	September 1981
792	Internet Control Message Protocol (ICMP)	September 1981
793	Transmission Control Protocol (TCP)	September 1981
821	Simple Mail-Transfer Protocol (SMTP)	August 1982
822	Standard for the Format of ARPA Internet Text Messages	August 1982
1321	The MD5 Message-Digest Algorithm	April 1992
1521	MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies	September 1993
1522	MIME (Multipurpose Internet Mail Extensions) Part Two: Message Header Extensions for Non-ASCII Text	September 1993
1583	OSPF Version 2	March 1994
1630	Universal Resource Identifiers in WWW	June 1994
1738	Uniform Resource Locators (URL)	December 1994
1752	The Recommendation for the IP Next-Generation Protocol	January 1995
1771	A Border Gateway Protocol 4 (BGP-4)	March 1995
1808	Relative Uniform Resource Locators	June 1995
1809	Using the Flow Label in IPv6	June 1995
1825	Security Architecture for the Internet Protocol	August 1995
1826	IP Authentication Header	August 1995
1827	IP Encapsulating Security Payload (ESP)	August 1995
1828	IP Authentication Using Keyed MD5	August 1995
1829	The ESP DES-CBC Transform	August 1995
1883	Internet Protocol, Version 6 Specification	December 1995
1884	IP Version-6 Addressing Architecture	December 1995
1885	Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version-6 (IPv6) Specification	December 1995
1886	DNS Extensions to Support IP Version 6	December 1995
1887	An Architecture for IPv6 Unicast Address Allocation	December 1995
1901	Introduction to Community-Based SNMPv2	January 1996
1902	Structure of Management Information for SNMPv2	January 1996
1903	Textual Conventions for SNMPv2	January 1996
1904	Conformance Statements for SNMPv2	January 1996
1905	Protocol Operations for SNMPv2	January 1996
1906	Transport Mappings for SNMPv2	January 1996
1907	Management Information Base for SNMPv2	January 1996
1908	Coexistence Between Version 1 and Version 2 of the Internet-Standard Network Management Framework	January 1996

GLOSSARY

Some of the definitions in this glossary are from the *American National Standard Dictionary of Information Technology*, ANSI Standard X3.172, 1995; these are marked with an asterisk.

Abstract Syntax Notation One (ASN.1) A formal language used to define syntax. In the case of SNMP, ASN.1 notation is used to define the format of SNMP protocol data units and of objects.

Aloha A medium access control technique for multiple-access transmission media. A station transmits whenever it has data to send. Unacknowledged transmissions are repeated.

Amplitude The size or magnitude of a voltage or current waveform.

Amplitude modulation* A form of modulation in which the amplitude of a carrier wave is varied in accordance with some characteristic of the modulating signal.

Amplitude-shift keying Modulation in which the two binary values are represented by two different amplitudes of the carrier frequency.

Analog data* Data represented by a physical quantity that is considered to be continuously variable and whose magnitude is made directly proportional to the data or to a suitable function of the data.

Analog signal A continuously varying electromagnetic wave that may be propagated over a variety of media.

Analog transmission The transmission of analog signals without regard to content. The signal may be amplified, but there is no intermediate attempt to recover the data from the signal.

Angle modulation* Modulation in which the angle of a sine wave carrier is varied. Phase and frequency modulation are particular forms of angle modulation.

Application layer Layer 7 of the OSI model. This layer determines the interface of the system with the user.

Asymmetric encryption A form of cryptosystem in which encryption and decryption are performed using two different keys, one of which is referred to as the public key and one of which is referred to as the private key. Also known as public-key encryption.

Asynchronous transfer mode (ATM) A form of packet transmission using fixed-size packets, called cells. ATM is the data transfer interface for B-ISDN. Unlike X.25, ATM does not provide error control and flow control mechanisms.

Asynchronous transmission Transmission in which each information character is individually synchronized (usually by the use of start elements and stop elements).

Attenuation A decrease in magnitude of current, voltage, or power of a signal in transmission between points.

ATM adaptation layer (AAL) The layer that maps information transfer protocols onto ATM.

Authentication* A process used to verify the integrity of transmitted data, especially a message.

Automatic repeat request A feature that automatically initiates a request for retransmission when an error in transmission is detected.

Balanced transmission A transmission mode in which signals are transmitted as a current that travels down one conductor and returns on the other. For digital signals, this technique is known as differential signaling, with the binary value depending on the voltage difference.

Bandlimited signal A signal, all of whose energy is contained within a finite frequency range.

Bandwidth* The difference between the limiting frequencies of a continuous frequency spectrum.

Baseband Transmission of signals without modulation. In a baseband local network, digital signals (1s and 0s) are inserted directly onto the cable as voltage pulses. The entire spectrum of the cable is consumed by the signal. This scheme does not allow frequency-division multiplexing.

Bit stuffing The insertion of extra bits into a data stream to avoid the appearance of unintended control sequences.

Bridge* A functional unit that interconnects two local area networks (LANs) that use the same logical link control protocol but may use different medium access control protocols.

Broadband The use of coaxial cable for providing data transfer by means of analog (radio-frequency) signals. Digital signals are passed through a modem and transmitted over one of the frequency bands of the cable.

Broadcast The simultaneous transmission of data to a number of stations.

Broadcast address An address that designates all entities within a domain (e.g., network, internet).

Broadcast communication network A communication network in which a transmission from one station is broadcast to and received by all other stations.

Bus* One or more conductors that serve as a common connection for a related group of devices.

Carrier A continuous frequency capable of being modulated or impressed with a second (information carrying) signal.

CATV Community Antenna Television. CATV cable is used for broadband/local networks, and broadcast TV distribution.

Cell relay The packet-switching mechanism used for the fixed-size packets called cells. ATM is based on cell relay technology.

Checksum An error-detecting code based on a summation operation performed on the bits to be checked.

Ciphertext The output of an encryption algorithm; the encrypted form of a message or data.

Circuit switching A method of communicating in which a dedicated communications path is established between two devices through one or more intermediate switching nodes. Unlike packet switching, digital data are sent as a continuous stream of bits. Bandwidth is guaranteed, and delay is essentially limited to propagation time. The telephone system uses circuit switching.

Coaxial cable A cable consisting of one conductor, usually a small copper tube or wire, within and insulated from another conductor of larger diameter, usually copper tubing or copper braid.

Codec (Coder-decoder) Transforms analog data into a digital bit stream (coder), and digital signals into analog data (decoder).

Collision A condition in which two packets are being transmitted over a medium at the same time. Their interference makes both unintelligible.

Common carrier In the United States, companies that furnish communication services to the public. The usual connotation is for long-distance telecommunications services. Common carriers are subject to regulation by federal and state regulatory commissions.

Common channel signaling Technique in which network control signals (e.g., call request) are separated from the associated voice or data path by placing the signaling from a group of voice or data paths on a separate channel dedicated to signaling only.

Communications architecture The hardware and software structure that implements the communications function.

Communication network A collection of interconnected functional units that provides a data communications service among stations attached to the network.

Connectionless data transfer A protocol for exchanging data in an unplanned fashion and without prior coordination (e.g., datagram).

Connection-oriented data transfer A protocol for exchanging data in which a logical connection is established between the endpoints (e.g., virtual circuit).

Contention The condition when two or more stations attempt to use the same channel at the same time.

Conventional encryption Symmetric encryption.

Crosstalk* The phenomenon in which a signal transmitted on one circuit or channel of a transmission system creates an undesired effect in another circuit or channel.

CSMA (Carrier Sense Multiple Access) A medium access control technique for multiple-access transmission media. A station wishing to transmit first senses the medium and transmits only if the medium is idle.

CSMA/CD (Carrier Sense Multiple Access with collision detection) A refinement of CSMA in which a station ceases transmission if it detects a collision.

Current-mode transmission A transmission mode in which the transmitter alternately applies current to each of two conductors in a twisted pair to represent logic 1 or 0. The total current is constant and always in the same direction.

Cyclic redundancy check An error detecting code in which the code is the remainder resulting from dividing the bits to be checked by a predetermined binary number.

Data circuit-terminating equipment (DCE) In a data station, the equipment that provides the signal conversion and coding between the data terminal equipment (DTE) and the line. The DCE may be separate equipment, or an integral part of the DTE, or of intermediate equipment. The DCE may perform other functions that are normally performed at the network end of the line.

Datagram* In packet switching, a packet, independent of other packets, that carries information sufficient for routing from the originating data terminal equipment (DTE) to the destination DTE without the necessity of establishing a connection between the DTEs and the network.

Data link layer* In OSI, the layer that provides service to transfer data between network layer entities, usually in adjacent nodes. The data link layer detects and possibly corrects errors that may occur in the physical layer.

Data terminal equipment (DTE)* Equipment consisting of digital end instruments that convert the user information into data signals for transmission, or reconvert the received data signals into user information.

Decibel A measure of the relative strength of two signals. The number of decibels is 10 times the log of the ratio of the power of two signals, or 20 times the log of the ratio of the voltage of two signals.

Decryption The translation of encrypted text or data (called ciphertext) into original text or data (called plaintext). Also called deciphering.

Delay distortion Distortion of a signal occurring when the propagation delay for the transmission medium is not constant over the frequency range of the signal.

Demand-assignment multiple access A technique for allocating satellite capacity, based on either FDM or TDM, in which capacity is granted on demand.

Differential encoding A means of encoding digital data on a digital signal such that the binary value is determined by a signal change rather than a signal level.

Digital data Data consisting of a sequence of discrete elements.

Digital signal A discrete or discontinuous signal, such as voltage pulses.

Digital signature An authentication mechanism that enables the creator of a message to attach a code that acts as a signature. The signature guarantees the source and integrity of the message.

Digital switch A star-topology local network. Usually refers to a system that handles only data, but not voice.

Digital transmission The transmission of digital data, using either an analog or digital signal, in which the digital data are recovered and repeated at intermediate points to reduce the effects of noise.

Digitize* To convert an analog signal to a digital signal.

Encapsulation The addition of control information by a protocol entity to data obtained from a protocol user.

Encrypt* To convert plain text or data into unintelligible form by the use of a code in such a manner that reconversion to the original form is possible.

Error detecting code* A code in which each expression conforms to specific rules of construction, so that if certain errors occur in an expression, the resulting expression will not conform to the rules of construction, and thus the presence of the errors is detected.

Error rate* The ratio of the number of data units in error to the total number of data units.

Flow control The function performed by a receiving entity to limit the amount or rate of data that is sent by a transmitting entity.

Frame A group of bits that includes data plus one or more addresses, and other protocol control information. Generally refers to a link layer (OSI layer 2) protocol data unit.

Frame check sequence An error-detecting code inserted as a field in a block of data to be transmitted. The code serves to check for errors upon reception of the data.

Frame relay A form of packet switching based on the use of variable-length, link-layer frames. There is no network layer, and many of the basic functions have been streamlined or eliminated to provide for greater throughput.

Frequency Rate of signal oscillation in hertz.

Frequency-division multiplexing The division of a transmission facility into two or more channels by splitting the frequency band transmitted by the facility into narrower bands, each of which is used to constitute a distinct channel.

Frequency modulation Modulation in which the frequency of an alternating current is the characteristic varied.

Frequency-shift keying Modulation in which the two binary values are represented by two different frequencies near the carrier frequency.

Full-duplex transmission Data transmission in both directions at the same time.

Half-duplex transmission Data transmission in either direction, one direction at a time.

Hash function A function that maps a variable-length data block or message into a fixed-length value called a hash code. The function is designed in such a way that, when protected, it provides an authenticator to the data or message. Also referred to as a message digest.

HDLC (high-level data link control) A very common, bit-oriented data link protocol (OSI layer 2) issued by ISO. Similar protocols are LAPB, LAPD, and LLC.

Header System-defined control information that precedes user data.

Impulse noise A high-amplitude, short-duration noise pulse.

Integrated services digital network A planned worldwide telecommunication service that will use digital transmission and switching technology to support voice and digital data communication.

Intermediate system (IS) A device attached to two or more subnetworks in an internet and that performs routing and relaying of data between end systems. Examples of intermediate systems are bridges and routers.

Intermodulation noise Noise due to the nonlinear combination of signals of different frequencies.

Internetwork A collection of packet-switching and broadcast networks that are connected together via routers.

Internet protocol An internetworking protocol that provides connectionless service across multiple packet-switching networks.

Internetworking Communication among devices across multiple networks.

Layer* A group of services, functions, and protocols that is complete from a conceptual point of view, that is one out of a set of hierarchically arranged groups, and that extends across all systems that conform to the network architecture.

Local area network A communication network that provides interconnection of a variety of data communicating devices within a small area.

Local loop Transmission path, generally twisted pair, between the individual subscriber and the nearest switching center of the public telecommunications network.

Longitudinal redundancy check The use of a set of parity bits for a block of characters, such that there is a parity bit for each bit position in the characters.

Manchester encoding A digital signaling technique in which there is a transition in the middle of each bit time. A 1 is encoded with a high level during the first half of the bit time; a 0 is encoded with a low level during the first half of the bit time.

Medium access control (MAC) For broadcast networks, the method of determining which device has access to the transmission medium at any time. CSMA/CD and token are common access methods.

Microwave Electromagnetic waves in the frequency range of about 2 to 40 GHz.

Modem (Modulator/Demodulator) Transforms a digital bit stream into an analog signal (modulator), and vice versa (demodulator).

Modulation* The process, or result of the process, of varying certain characteristics of a signal, called a carrier, in accordance with a message signal.

Multicast address An address that designates a group of entities within a domain (e.g., network, internet).

Multiplexing In data transmission, a function that permits two or more data sources to share a common transmission medium such that each data source has its own channel.

Multipoint A configuration in which more than two stations share a transmission path.

Network layer Layer 3 of the OSI model. Responsible for routing data through a communication network.

Network terminating equipment Grouping of ISDN functions at the boundary between the ISDN and the subscriber.

Noise Unwanted signals that combine with and, hence, distort the signal intended for transmission and reception.

Nonreturn to zero A digital signaling technique in which the signal is at a constant level for the duration of a bit time.

Octet A group of eight bits, usually operated upon as an entity.

Open systems interconnection (OSI) reference model A model of communications between cooperating devices. It defines a seven-layer architecture of communication functions.

Optical fiber A thin filament of glass or other transparent material, through which a signal-encoded light beam may be transmitted by means of total internal reflection.

Packet A group of bits that include data plus control information. Generally refers to a network layer (OSI layer 3) protocol data unit.

Packet switching A method of transmitting messages through a communication network, in which long messages are subdivided into short packets. The packets are then transmitted as in message switching.

Parity bit* A check bit appended to an array of binary digits to make the sum of all the binary digits, including the check bit, always odd or always even.

PBX Private branch exchange. A telephone exchange on the user's premises. Provides a switching facility for telephones on extension lines within the building as well as access to the public telephone network.

Phase The relative position in time within a single period of a signal.

Phase modulation Modulation in which the phase angle of a carrier is the characteristic varied.

Phase-shift keying Modulation in which the phase of the carrier signal is shifted to represent digital data.

Physical layer Layer 1 of the OSI model. Concerned with the electrical, mechanical, and timing aspects of signal transmission over a medium.

Piggybacking The inclusion of an acknowledgment to a previously received packet in an outgoing data packet.

Plaintext The input to an encryption function or the output of a decryption function.

Point-to-point A configuration in which two stations share a transmission path.

Poll and select The process by which a primary station invites secondary stations, one at a time, to transmit (poll), and by which a primary station requests that a secondary receive data (select).

Presentation layer* Layer 6 of the OSI model. Provides for the selection of a common syntax for representing data and for transformation of application data into and from the common syntax.

Private key One of the two keys used in an asymmetric encryption system. For secure communication, the private key should only be known to its creator.

Propagation delay The delay between the time a signal enters a channel and the time it is received.

Protocol A set of rules that govern the operation of functional units to achieve communication.

Protocol control information* Information exchanged between entities of a given layer, via the service provided by the next lower layer, to coordinate their joint operation.

Protocol data unit (PDU)* A set of data specified in a protocol of a given layer and consisting of protocol control information of that layer, and possibly user data of that layer.

Public data network A government-controlled or national-monopoly packet-switched network. This service is publicly available to data processing users.

Public key One of the two keys used in an asymmetric encryption system. The public key is made public, to be used in conjunction with a corresponding private key.

Public-key encryption Asymmetric encryption.

Pulse code modulation A process in which a signal is sampled, and the magnitude of each sample with respect to a fixed reference is quantized and converted by coding to a digital signal.

Residual error rate The error rate remaining after attempts at correction are made.

Ring A local-network topology in which stations are attached to repeaters connected in a closed loop. Data are transmitted in one direction around the ring, and can be read by all attached stations.

Router An internetworking device that connects two computer networks. It makes use of an internet protocol and assumes that all of the attached devices on the networks use the same communications architecture and protocols. A router operates at OSI layer 3.

Routing The determination of a path that a data unit (frame, packet, message) will traverse from source to destination.

Service access point A means of identifying a user of the services of a protocol entity. A protocol entity provides one or more SAPs for use by higher-level entities.

Session layer Layer 5 of the OSI model. Manages a logical connection (session) between two communicating processes or applications.

Simplex transmission Data transmission in one preassigned direction only.

Sliding-window technique A method of flow control in which a transmitting station may send numbered packets within a window of numbers. The window changes dynamically to allow additional packets to be sent.

Space-division switching A circuit-switching technique in which each connection through the switch takes a physically separate and dedicated path.

Spectrum Refers to an absolute range of frequencies. For example, the spectrum of CATV cable is now about 5 to 400 MHz.

Star A topology in which all stations are connected to a central switch. Two stations communicate via circuit switching.

Statistical time-division multiplexing A method of TDM in which time slots on a shared transmission line are allocated to I/O channels on demand.

Stop-and-wait A flow control protocol in which the sender transmits a block of data and then awaits an acknowledgment before transmitting the next block.

Subnetwork Refers to a constituent network of an internet; this avoids ambiguity because the entire internet, from a user's point of view, is a single network.

Switched communication network A communication network consisting of a network of nodes connected by point-to-point links. Data are transmitted from source to destination through intermediate nodes.

Symmetric encryption A form of cryptosystem in which encryption and decryption are performed using the same key. Also known as conventional encryption.

Synchronous time-division multiplexing A method of TDM in which time slots on a shared transmission line are assigned to I/O channels on a fixed, predetermined basis.

Synchronous transmission Data transmission in which the time of occurrence of each signal representing a bit is related to a fixed time frame.

Telematics User-oriented information transmission services. Includes Teletex, Videotex, and facsimile.

Thermal noise Statistically uniform noise due to the temperature of the transmission medium.

Time-division multiplexing The division of a transmission facility into two or more channels by allotting the facility to several different information channels, one at a time.

Time-division switching A circuit-switching technique in which time slots in a time-multiplexed stream of data are manipulated to pass data from an input to an output.

Token bus A medium access control technique for bus/tree. Stations form a logical ring, around which a token is passed. A station receiving the token may transmit data and then must pass the token on to the next station in the ring.

Token ring A medium access control technique for rings. A token circulates around the ring. A station may transmit by seizing the token, inserting a packet onto the ring, and then retransmitting the token.

Topology The structure, consisting of paths and switches, that provides the communications interconnection among nodes of a network.

Transmission medium The physical path between transmitters and receivers in a communications system.

Transport layer Layer 4 of the OSI model. Provides reliable, transparent transfer of data between endpoints.

Tree A local network topology in which stations are attached to a shared transmission medium. The transmission medium is a branching cable emanating from a headend, with no closed circuits. Transmissions propagate throughout all branches of the tree, and are received by all stations.

Twisted pair A transmission medium consisting of two insulated wires arranged in a regular spiral pattern.

Unbalanced transmission A transmission mode in which signals are transmitted on a single conductor. Transmitter and receiver share a common ground.

Value-added network A privately-owned, packet-switching network whose services are sold to the public.

Virtual circuit A packet-switching service in which a connection (virtual circuit) is established between two stations at the start of transmission. All packets follow the same route; they need not carry a complete address, and they arrive in sequence.

References

ABRA95 Abrams, M., Jajodia, S., and Podell, H., eds. *Information Security*. Los Alamitos, CA: IEEE Computer Society Press, 1995.

ABSX92 Apple Computer, Bellcore, Sun Microsystems, and Xerox. *Network Compatible ATM for Local Network Applications, Version 1.01*. October 19, 1992 (available at parcftp.xerox.com/pub/latm).

ANSI85 American National Standards Institute. *Financial Institution Key Management (Wholesale)*. ANS X9.17, 1985.

ARMI93 Armitage, G. and Adams, K. "Packet Reassembly During Cell Loss." *IEEE Network*, September 1993.

ASH90 Ash, G. "Design and Control of Networks with Dynamic Nonhierarchical Routing." *IEEE Communications Magazine*, October 1990.

ATM95 ATM Forum. *LAN Emulation over ATM Specification—Version 1.0*. 1995.

BANT94 Bantz, D. and Bauchot, F. "Wireless LAN Design Alternatives." *IEEE Network*, March/April, 1994.

BELL90 Bellcore (Bell Communications Research). *Telecommunications Transmission Engineering*. Three volumes. 1990.

BELL91 Bellamy, J. *Digital Telephony*. New York: Wiley, 1991.

BENE87 Benedetto, S., Biglieri, E., and Castellani, V. *Digital Transmission Theory*. Englewood Cliffs, NJ: Prentice Hall, 1987.

BERG91 Bergman, W. "Narrowband Frame Relay Congestion Control." *Proceedings of the Tenth Annual Phoenix Conference of Computers and Communications*, March 1991.

BERT92 Bertsekas, D. and Gallager, R. *Data Networks*. Englewood Cliffs, NJ: Prentice Hall, 1992.

BIRD94 Bird, D. *Token Ring Network Design*. Reading, MA: Addison-Wesley, 1994.

BLAC93 Black, U. *Data Link Protocols*. Englewood Cliffs, NJ: Prentice Hall, 1993.

BLAC94 Black, U. *Frame Relay Networks: Specifications and Implementations*. New York: McGraw-Hill, 1994.

BLAC95a Black, U. *Physical Level Interfaces and Protocols*. Los Alamitos, CA: IEEE Computer Society Press, 1995.

BLAC95b Black, U. *The V Series Recommendations: Standards for Data Communications Over the Telephone Network*. New York: McGraw-Hill, 1995.

BOUD92 Boudec, J. "The Asynchronous Transfer Mode: A Tutorial." *Computer Networks and ISDN Systems*, May 1992.

BRAD96 Bradner, S. and Mankin, A. *IPng: Internet Protocol Next Generation*. Reading, MA: Addison-Wesley, 1996.

BURG91 Burg, J. and Dorman, D. "Broadband ISDN Resource Management: The Role of Virtual Paths." *IEEE Communications Magazine*, September 1991.

CHEN89 Chen, K., Ho, K., and Saksena, V. "Analysis and Design of a Highly Reliable Transport Architecture for ISDN Frame-Relay Networks." *IEEE Journal on Selected Areas in Communications*, October 1989.

COME94a Comer, D. and Stevens, D. *Internetworking with TCP/IP, Volume II: Design Implementation, and Internals*. Englewood Cliffs, NJ: Prentice Hall, 1994.

COME94b Comer, D. and Stevens, D. *Internetworking with TCP/IP, Volume III: Client-Server Programming and Applications*. Englewood Cliffs, NJ: Prentice Hall, 1994.

COME95 Comer, D. *Internetworking with TCP/IP, Volume I: Principles, Protocols, and Architecture*. Englewood Cliffs, NJ: Prentice Hall, 1995.

COUC95 Couch, L. *Modern Communication Systems: Principles and Applications*. Englewood Cliffs, NJ: Prentice Hall, 1995.

DAVI89 Davies, D. and Price, W. *Security for Computer Networks*. New York: Wiley, 1989.

DAVI95 Davis, P. and McGuffin, C. *Wireless Local Area Networks*. New York: McGraw-Hill, 1995.

DIFF76 Diffie, W. and Hellman, M. "New Directions in Cryptography." *IEEE Transactions on Information Theory*, November 1976.

DIJK59 Dijkstra, E. "A Note on Two Problems in Connection with Graphs." *Numerical Mathematics*, October 1959.

DIXO94 Dixon, R. *Spread Spectrum Systems with Commercial Applications*. New York: Wiley, 1994.

DOSH88 Doshi, B. and Nguyen, H. "Congestion Control in ISDN Frame-Relay Networks." *AT&T Technical Journal*, November/December 1988.

DOTY95 Doty, T. "Towards Real Internet Security." *Connexions*, December 1995.

FCA94 Fibre Channel Association. *Fibre Channel: Connection to the Future*. Austin, TX: Fibre Channel Association, 1994.

FORD62 Ford, L. and Fulkerson, D. *Flows in Networks*. Princeton, NJ: Princeton University Press, 1962.

FREE91 Freeman, R. *Telecommunication Transmission Handbook*. New York: Wiley, 1991.

FREE94 Freeman, R. *Reference Manual for Telecommunications Engineering*. New York: Wiley, 1994.

FREE96 Freeman, R. *Telecommunication System Engineering*. New York: Wiley, 1996.

GAUD89 Gaudette, P. *A Tutorial on ASN.1*. Technical Report NCSL/SNA-89/12. Gaithersburg, MD: National Institute of Standards and Technology, 1989.

GERS91 Gersht, A. and Lee, K. "A Congestion Control Framework for ATM Networks." IEEE Journal on Selected Areas in Communications, September 1991.

GILL95 Gilligan, R. and Callon, R. "IPv6 Transition Mechanisms Overview." *Connexions*, October 1995.

GIRA90 Girard, A. *Routing and Dimensioning in Circuit-Switched Networks*. Reading, MA: Addison-Wesley, 1990.

GORA95 Goralski, W. Introduction to ATM Networking. New York: McGraw-Hill, 1995.

GREE93 Green, P. *Fiber Optic Networks*. Englewood Cliffs, NJ: Prentice Hall, 1993.

HAMM86 Hammond, J. and O'Reilly, P. *Performance Analysis of Local Computer Networks*. Reading, MA: Addison-Wesley, 1986.

HAND94 Handel, R., Huber, N., and Schroder, S. *ATM Networks: Concepts, Protocols, Applications*. Reading, MA: Addison-Wesley, 1994.

HALS96 Halsall, F. *Data Communications, Computer Networks, and Open Systems*. Reading, MA: Addison-Wesley, 1996.

HARB92 Harbison, R. "Frame Relay: Technology for Our Time." *LAN Technology*, December 1992.

HAYK94 Haykin, S. *Communication Systems*. New York: Wiley, 1994.

HEGE93 Hegering, H. and Lapple, A. *Ethernet: Building a Communications Infrastructure*. Reading, MA: Addison-Wesley, 1993.

HELG91 Helgert, H. *Integrated Services Digital Networks: Architectures, Protocols, and Standards*. Reading, MA: Addison-Wesley, 1991.

HIND83 Hinden, R., Haverty, J., and Sheltzer, A. "The DARPA Internet: Interconnecting Heterogeneous Computer Networks with Gateways." *Computer*, September 1983.

HIND95 Hinden, R. "IP Next Generation Overview." *Connexions*, March 1995.

HUIT95 Huitema, C. *Routing in the Internet*. Englewood Cliffs, NJ: Prentice Hall, 1995.

JAIN93 Jain, B. and Agrawala, A. *Open Systems Interconnection*. New York: McGraw-Hill, 1993.

JEFF94 Jeffries, R. "ATM LAN Emulation: The Inside Story." *Data Communications*, September 21, 1994.

KALI91 Kaliski, B. *A Layman's Guide to a Subset of ASN.1, BER, and DER*. Report SEC-SIG-91-17, Redwood City, CA: RSA Data Security Inc., 1991.

KAVA95 Kavak, N. "Data Communication in ATM Networks." *IEEE Network*, May/June 1995.

KESS92 Kessler, G. and Train, D. *Metropolitan Area Networks: Concepts, Standards, and Services*. New York: McGraw-Hill, 1992.

KESS93 Kessler, G. *ISDN: Concepts, Facilities, and Services*. New York: McGraw-Hill, 1993.

KHAN89 Khanna, A. and Zinky, J. "The Revised ARPANET Routing Metric." *Proceedings, SIGCOMM '89 Sypmosium*, 1989.

KLEI76 Kleinrock, L. *Queueing Systems, Volume II: Computer Applications*. New York: Wiley, 1976.

LANG95 Lang, L. "Using Multilayer Switches to Connect Legacy LANs and the ATM Backbone." *Telecommunications*, March 1995.

LEUT94 Leutwyler, K. "Superhack." *Scientific American*, July 1994.

MADR94 Madron, T. *Local Area Networks: New Technologies, Emerging Standards*. New York: Wiley, 1994.

MART90 Martin, J. *Telecommunications and the Computer*. Englewood Cliffs, NJ: Prentice Hall, 1990.

MART94 Martin, J., Chapman, K., and Leben, J. *Local Area Networks: Architectures and Implementations*. Englewood Cliffs, NJ: Prentice Hall, 1994.

MCDY95 McDysan, D. and Spohn, D. *ATM: Theory and Application*. New York: McGraw-Hill, 1995.

MCQU80 McQuillan, J., Richer, I., and Rosen, E. "The New Routing Algorithm for the ARPANET." *IEEE Transactions on Communications*, May 1980.

MEEK90 Meeks, F. "The Sound of Lamarr." *Forbes*, May 1990.

MILL95 Mills, A. *Understanding FDDI*. Englewood Cliffs, NJ: Prentice Hall, 1995.

MURP95 Murphy, E., Hayes, S., and Enders, M. *TCP/IP: Tutorial and Technical Overview*. Englewood Cliffs, NJ: Prentice Hall, 1995.

NECH92 Nechvatal, J. "Public Key Cryptography." [SIMM92].

NIST94 National Institute of Standards and Technology. *Data Encryption Standard.* FIPS-PUB 46-2, June 1994.

NEWM94 Newman, P. "ATM Local Area Networks." *IEEE Communications Magazine*, March 1994.

ONVU94 Onvural, R. *Asynchronous Transfer Mode Networks: Performance Issues.* Boston: Artech House, 1994.

PAHL95a Pahlavan, K., Probert, T., and Chase, M. "Trends in Local Wireless Networks." *IEEE Communications Magazine*, March 1995.

PAHL95b Pahlavan, K. and Levesque, A. *Wireless Information Networks.* New York: Wiley, 1995.

PARK92 Park, Y. et al. "2.488 Gb/s-318 km Repeaterless Transmission Using Erbium-Doped Fiber Amplifiers in a Direct-Detection System." *IEEE Photonics Technology Letters*, February 1992.

PART88 Partridge, C. *Innovations in Internetworking.* Norwood, MA: Artech House, 1988.

PEAR92 Pearson, J. *Basic Communication Theory.* Englewood Cliffs, NJ: Prentice Hall, 1992.

PEEB87 Peebles, P. *Digital Communication Systems.* Englewood Cliffs, NJ: Prentice Hall, 1987.

PERL92 Perlman, R. *Interconnections: Bridges and Routers.* Reading, MA: Addison-Wesley, 1992.

PETE61 Peterson, W. and Brown, D. "Cyclic Codes for Error Detection." *Proceedings of the IRE*, January 1961.

PETE95 Peterson, R., Ziemer, R., and Borth, D. *Introduction to Spread Spectrum Communications.* Englewood Cliffs, NJ: Prentice Hall, 1995.

POWE90 Powers, J. and Stair, H. *Megabit Data Communications.* Englewood Cliffs, NJ: Prentice Hall, 1990.

PROA94 Proakis, J. and Salehi, M. *Communication Systems Engineering.* Englewood Cliffs, NJ: Prentice Hall, 1994.

PRYC93 Prycker, M. *Asynchronous Transfer Mode: Solutions for Broadband ISDN.* New York: Ellis Horwood, 1993.

REEV95 Reeve, W. *Subscriber Loop Signaling and Transmission Handbook.* Piscataway, NJ: IEEE Press, 1995.

REGN90 Regnier, J. and Cameron, W. "State-Dependent Dynamic Traffic Management for Telepone Networks." *IEEE Communications Magazine*, October 1990.

RIVE78 Rivest, R., Shamir, A., and Adleman, L. "A Method for Obtaining Digital Signatures and Public Key Cryptosystems." *Communications of the ACM*, February 1978.

ROSE93 Rose, M. *The Internet Message: Closing the Book with Electronic Mail.* Englewood Cliffs, NJ: Prentice Hall, 1993.

SADI95 Sadiku, M. *Metropolitan Area Networks.* Boca Raton, FL: CRC Press, 1995.

SANT94 Santamaria, A. and Lopez-Hernandez, F., eds. *Wireless LAN Systems.* Boston MA: Artech House, 1994.

SATO90 Sato, K., Ohta, S., and Tokizawa, I. "Broad-band ATM Network Architecture Based on Virtual Paths." *IEEE Transactions on Communications*, August 1990.

SATO91 Sato, K., Ueda, H., and Yoshikai, M. "The Role of Virtual Path Crossconnection." *IEEE LTS*, August 1991.

SCHN91 Schneier, B. "One-Way Hash Functions." *Dr. Dobb's Journal*, September 1991.

SCHN96 Schneier, B. *Applied Cryptography*. New York: Wiley, 1996.

SCHW77 Schwartz, M. *Computer-Communication Network Design and Analysis*. Englewood Cliffs, NJ: Prentice Hall, 1977.

SHAH94 Shah, A. and Ramakrishnan, G. *FDDI: A High-Speed Network*. Englewood Cliffs, NJ: Prentice Hall, 1994.

SEYE91 Seyer, M. *RS-232 Made Easy: Connecting Computers, Printers, Terminals, and Modems*. Englewood Cliffs, NJ: Prentice Hall, 1991.

SIMM92 Simmons, G., ed. *Contemporary Cryptology: The Science of Information Integrity*. Piscataway, NJ: IEEE Press, 1992.

SKLA88 Sklar, B. *Digital Communications: Fundamentals and Applications*. Englewood Cliffs, NJ: Prentice Hall, 1988.

SMIT93 Smith, P. *Frame Relay: Principles and Applications*. Reading, MA: Addison-Wesley, 1993.

SPOH93 Spohn, D. *Data Network Design*. New York: McGraw-Hill, 1994.

SPRA91 Spragins, J., Hammond, J., and Pawlikowski, K. *Telecommunications Protocols and Design*. Reading, MA: Addison-Wesley, 1991.

SPUR95 Spurgeon, C. *Quick Reference Guide to Ethernet*. Austin, TX: Harris Park Press, 1995.

STAL90 Stallings, W. *Handbook of Computer Communications Standards, Volume 3: The TCP/IP Protocol Suite, Second Edition*. Englewood Cliffs, NJ: Prentice Hall, 1990.

STAL95a Stallings, W. *ISDN and Broadband ISDN, with Frame Relay and ATM, Third Edition*. Englewood Cliffs, NJ: Prentice Hall, 1995.

STAL95b Stallings, W. *Network and Internetwork Security: Principles and Practice*. Englewood Cliffs, NJ: Prentice Hall, 1995.

STAL96 Stallings, W. *Computer Organization and Architecture, Fourth Edition*. Upper Saddle River, NJ: PrenticeHall, 1996.

STAL97 Stallings, W. *Local and Metropolitan Area Networks, Fifth Edition*. Upper Saddle River, NJ: Prentice Hall, 1997.

STEE90 Steedman, D. *ASN.1: The Tutorial and Reference*. London: Technology Appraisals, 1990.

STEE95 Steenstrup, M. *Routing in Communications Networks*. Englewood Cliffs, NJ: Prentice-Hall, 1995.

STEV94 Stevens, W. *TCP/IP Illustrated, Volume 1: The Protocols*. Reading, MA: Addison-Wesley, 1994.

STEV96 Stevens, W. *TCP/IP Illustrated, Volume 3: TCP for Transactions, HTTP, NNTP, and the UNIX(R) Domain Protocol*. Reading, MA: Addison-Wesley, 1996.

STER93 Sterling, D. *Technician's Guide to Fiber Optics*. Albany, NY: Delmar Publications, 1993.

STUC85 Stuck, B. and Arthurs, E. *A Computer Communications Network Performance Analysis Primer*. Englewood Cliffs, NJ: Prentice Hall, 1985.

SUZU94 Suzuki, T. "ATM Adaptation Layer Protocol." *IEEE Communications Magazine*, April 1994.

TANE88 Tanenbaum, A. *Computer Networks*. Englewood Cliffs, NJ: Prentice Hall, 1988.

TERP92 Terplan, K. *Communication Networks Management*. Englewood Cliffs, NJ: Prentice Hall, 1992.

TRUO95 Truong, H. et al. "LAN Emulation on an ATM Network." *IEEE Communications Magazine*, May 1995.

TSUD92 Tsudik, G. "Message Authentication with One-Way Hash Functions." *Computer Communications Review*, October, 1992

TUCH79 Tuchman, W. *Hellman Presents No-Shortcut Solutions to DES. IEEE Spectrum*, July 1979.

VALK93 Valkenburg, M. ed. *Reference Data for Engineers: Radio, Electronics, Computer, and Communications*. Carmel, IN: Prentice Hall/SAMS, 1993.

VALE92 Valenzano, A., DeMartini, C., and Ciminiera, L. *MAP and TOP Communications: Standards and Applications*. Reading, MA: Addison-Wesley, 1992.

WIDM83 Widmer, A. and Franaszek, P. "A DC-Balanced, Partitioned, 8B/10B Transmission Code." *IBM Journal of Research and Development*, September 1983

WIEN93 Wiener, M. *Efficient DES Key Search. Proceedings, Crypto '93*. Springer-Verlag, 1993.

WRIG95 Wright, G. and Stevens, W. *TCP/IP Illustrated, Volume 2: The Implementation*. Reading, MA: Addison-Wesley, 1995.

YEN83 Yen, C. and Crawford, R. "Distribution and Equalization of Signal on Coaxial Cables Used in 10-Mbits Baseband Local Area Networks." *IEEE Transactions on Communications*, October 1983.

INDEX

ACRONYMS

AAL	ATM Adaptation Layer
AM	Amplitude Modulation
AMI	Alternate Mark Inversion
ANS	American National Standard
ANSI	American National Standard Institute
ARQ	Automatic Repeat Request
ASCII	American Standard Code for Information Interchange
ASK	Amplitude-Shift Keying
ATM	Asynchronous Transfer Mode
B-ISDN	Broadband ISDN
BOC	Bell Operating Company
CBR	Constant Bit Rate
CCITT	International Consultative Committee on Telegraphy and Telephony
CIR	Committed Information Rate
CRC	Cyclic Redundancy Check
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
DCE	Data Circuit-Terminating Equipment
DES	Data Encryption Standard
DTE	Data Terminal Equipment
FCC	Federal Communications Commission
FCS	Frame Check Sequence
FDDI	Fiber Distributed Data Interface
FDM	Frequency-Division Multiplexing
FSK	Frequency-Shift Keying
FTP	File Transfer Protocol
FM	Frequency Modulation
HDLC	High-Level Data Link Control
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
IDN	Integrated Digital Network
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPng	Internet Protocol - Next Generation
ISDN	Integrated Services Digital Network
ISO	International Organization for Standardization
ITU	International Telecommunication Union
ITU-T	ITU Telecommunication Standardization Sector

LAN	Local Area Network
LAPB	Link Access Procedure–Balanced
LAPD	Link Access Procedure on the D Channel
LAPF	Link Access Procedure for Frame Mode Bearer Services
LLC	Logical Link Control
MAC	Medium Access Control
MAN	Metropolitan Area Network
MIME	Multi-Purpose Internet Mail Extension
NRZI	Nonreturn to Zero, Inverted
NRZL	Nonreturn to Zero, Level
NT	Network Termination
OSI	Open Systems Interconnection
PBX	Private Branch Exchange
PCM	Pulse-Code Modulation
PDU	Protocol Data Unit
PSK	Phase-Shift Keying
PTT	Postal, Telegraph, and Telephone
PM	Phase Modulation
QOS	Quality of Service
QPSK	Quadrature Phase Shift Keying
RBOC	Regional Bell Operating Company
RF	Radio Frequency
RSA	Rivest, Shamir, Adleman Algorithm
SAP	Service Access Point
SDH	Synchronous Digital Hierarchy
SDU	Service Data Unit
SMTP	Simple Mail Transfer Protocol
SONET	Synchronous Optical Network
TCP	Transmission Control Protocol
TDM	Time-Division Multiplexing
TE	Terminal Equipment
UNI	User-Network Interface
URI	Universal Resource Identifier
URL	Uniform Resource Locator
VAN	Value-Added Network
VBR	Variable Bit Rate
VCC	Virtual Channel Connection
VPC	Virtual Path Connection
WWW	World Wide Web

DATA AND COMPUTER COMMUNICATIONS

FIFTH EDITION

by

WILLIAM STALLINGS

This is the only book that covers the full range of data and computer communications, from the physical layer details of transmission media and signaling up to the most popular Internet application protocols, including electronic mail (SMTP/MIME) and the world wide web (HTTP). No previous background in data communications is assumed for this book.

FEATURES

- ❑ Expanded coverage of WANs, including ATM, frame relay, packet switching, and circuit switching.
- ❑ Increased coverage of LANs, including Fast Ethernet, 100VG-AnyLAN, ATM LANs, Fiber Channel, and FDDI.
- ❑ Routine algorithms and protocols, including BGP and OSPF are covered thoroughly.
- ❑ Indepth coverage of TCP/IP, IPv6, ICMPv6, SNMPv2, SMTP, MIME, HTTP, and URLs.
- ❑ An entire chapter devoted to network security, including the new Internet security standards.