



**Data Communications
and Networking**

Fourth Edition

Forouzan

Chapter 11

Data Link Control

11-1 FRAMING

*The data link layer needs to pack bits into **frames**, so that each frame is distinguishable from another. Our postal system practices a type of framing. The simple act of inserting a letter into an envelope separates one piece of information from another; the envelope serves as the delimiter.*

Topics discussed in this section:

Fixed-Size Framing

Variable-Size Framing

Figure 11.1 A frame in a character-oriented protocol

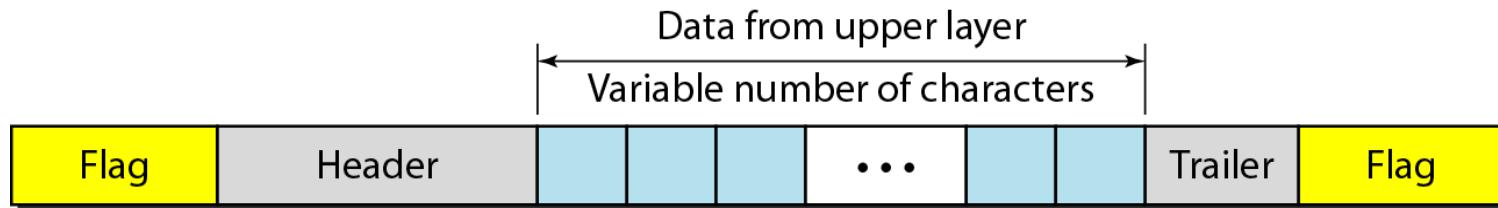
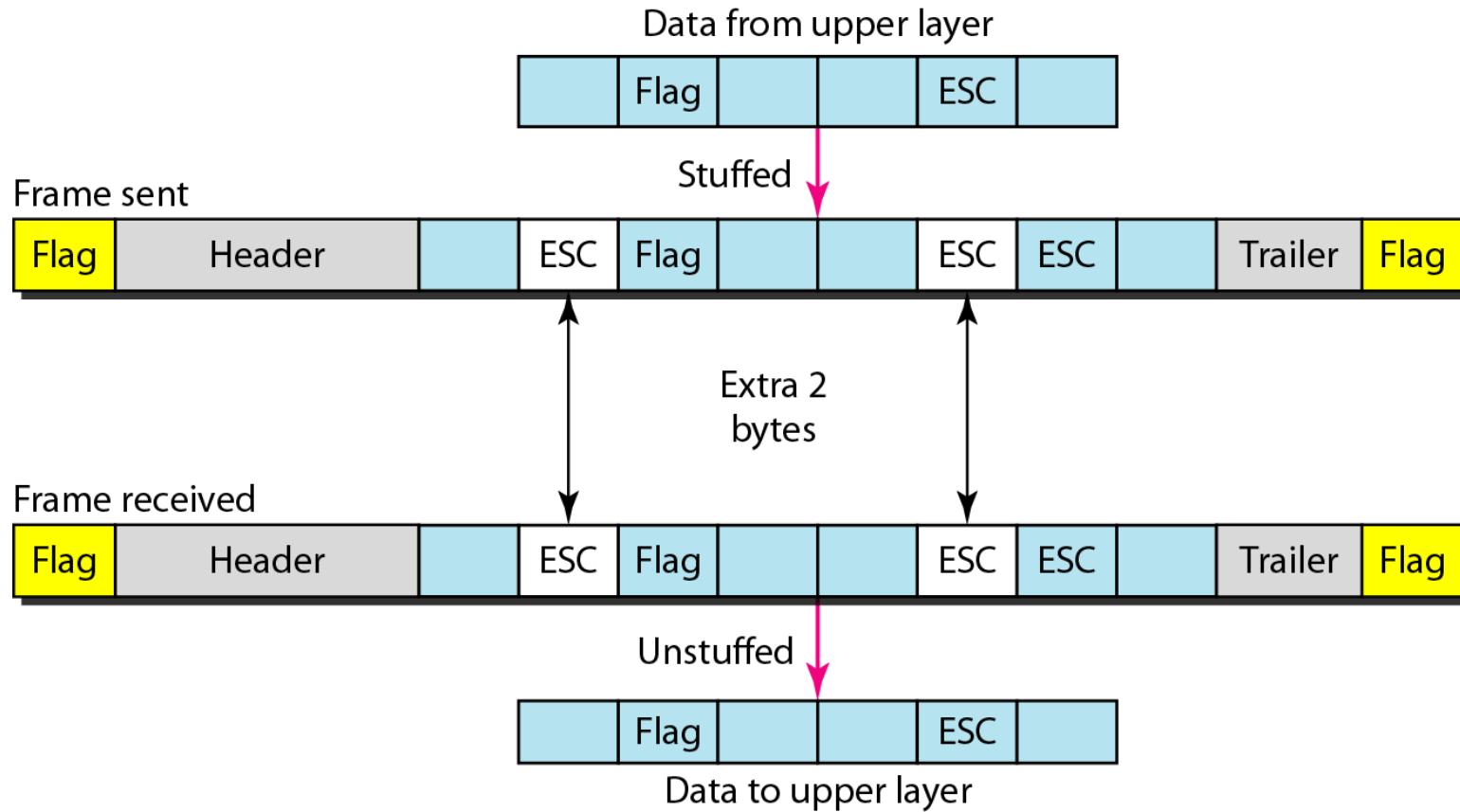
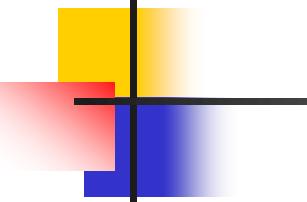


Figure 11.2 Byte stuffing and unstuffing

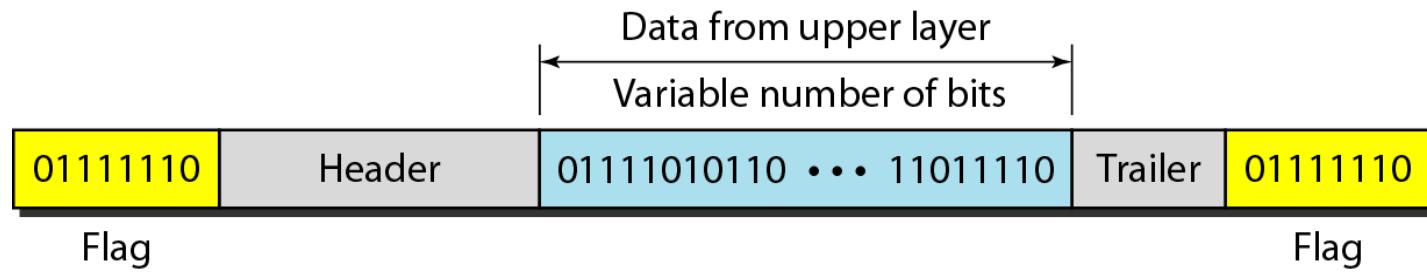


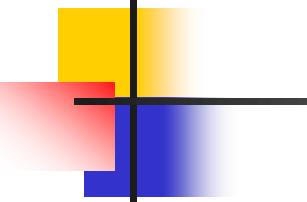


Note

Byte stuffing is the process of adding 1 extra byte whenever there is a flag or escape character in the text.

Figure 11.3 A frame in a bit-oriented protocol

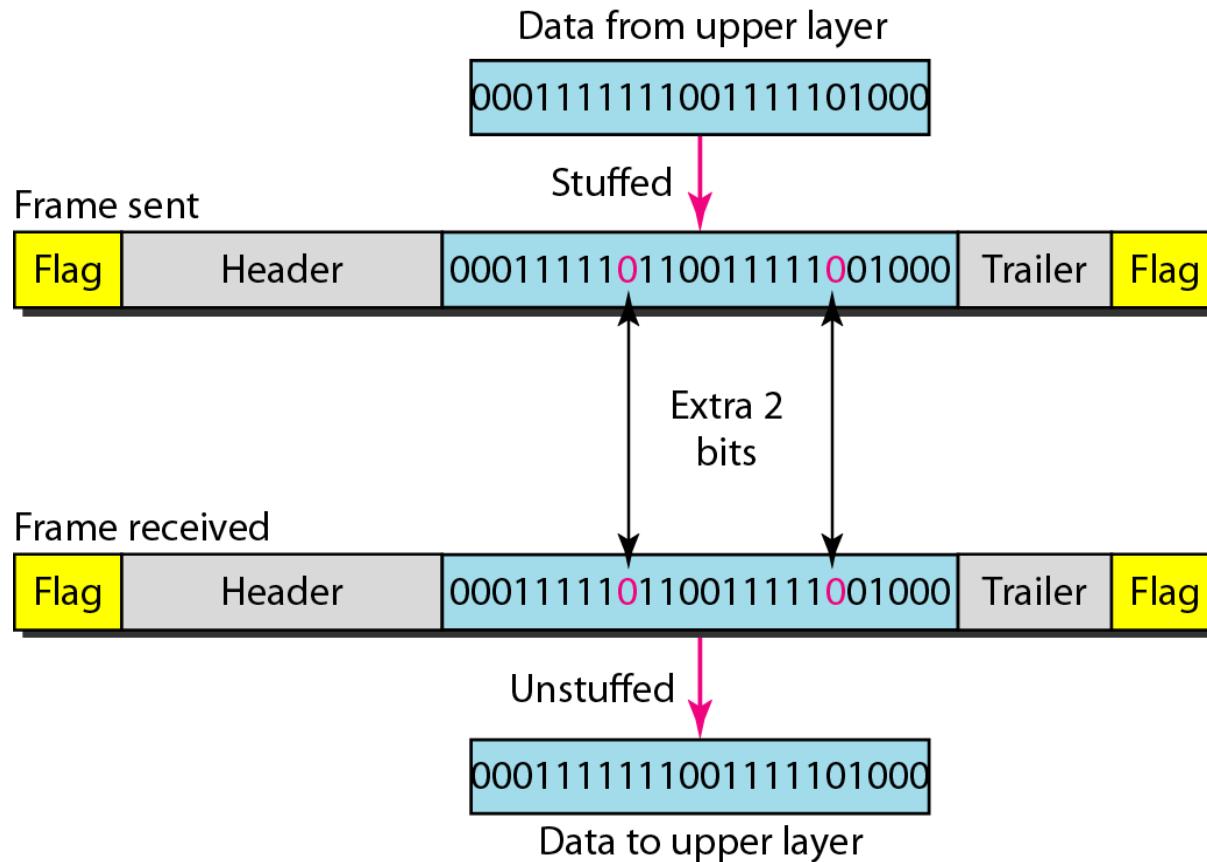




Note

Bit stuffing is the process of adding one extra 0 whenever five consecutive 1s follow a 0 in the data, so that the receiver does not mistake the pattern 0111110 for a flag.

Figure 11.4 Bit stuffing and unstuffing



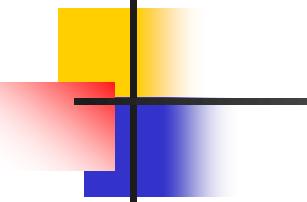
11-2 FLOW AND ERROR CONTROL

*The most important responsibilities of the data link layer are **flow control** and **error control**. Collectively, these functions are known as **data link control**.*

Topics discussed in this section:

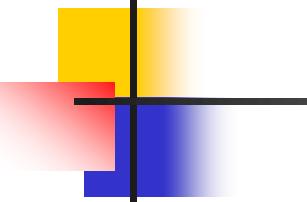
Flow Control

Error Control



Note

Flow control refers to a set of procedures used to restrict the amount of data that the sender can send before waiting for acknowledgment.



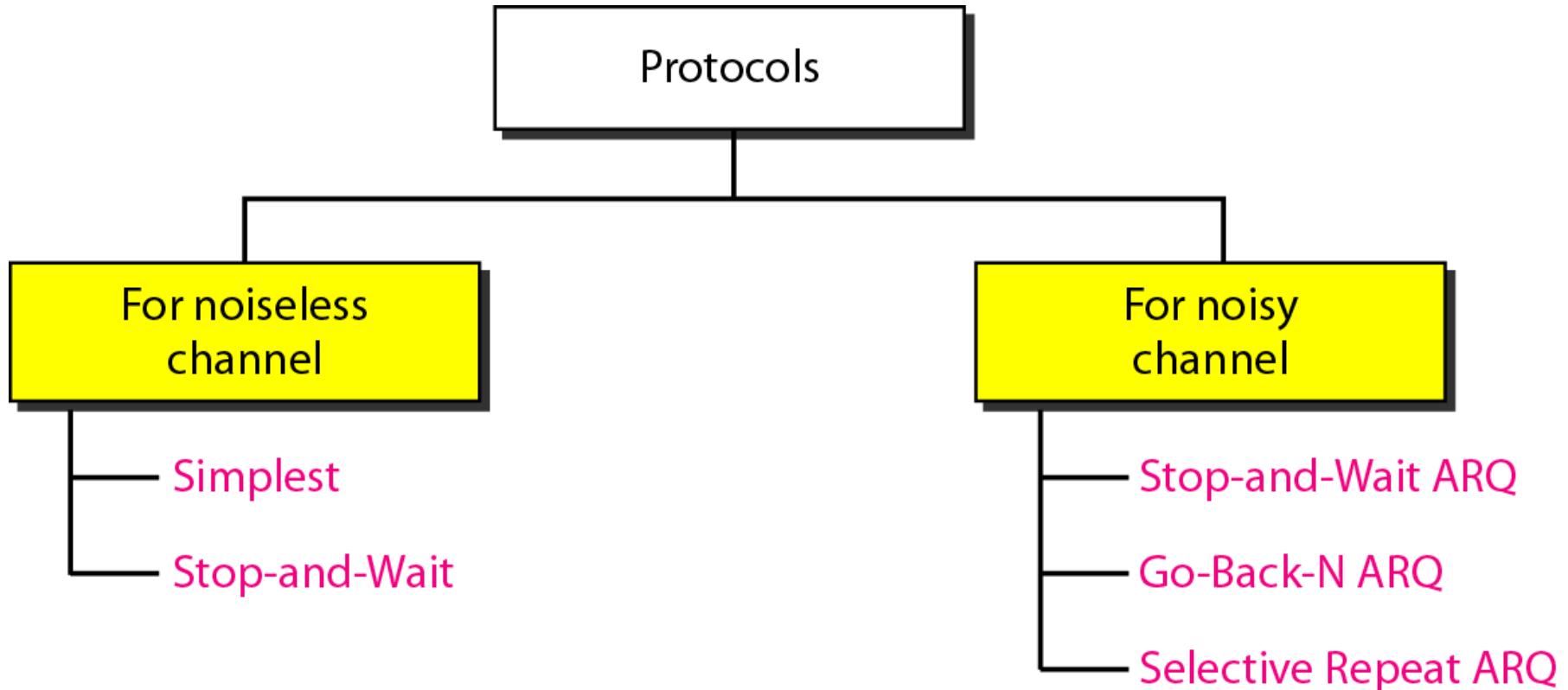
Note

Error control in the data link layer is based on automatic repeat request, which is the retransmission of data.

11-3 PROTOCOLS

Now let us see how the data link layer can combine framing, flow control, and error control to achieve the delivery of data from one node to another. The protocols are normally implemented in software by using one of the common programming languages. To make our discussions language-free, we have written in pseudocode a version of each protocol that concentrates mostly on the procedure instead of delving into the details of language rules.

Figure 11.5 *Taxonomy of protocols discussed in this chapter*



11-4 NOISELESS CHANNELS

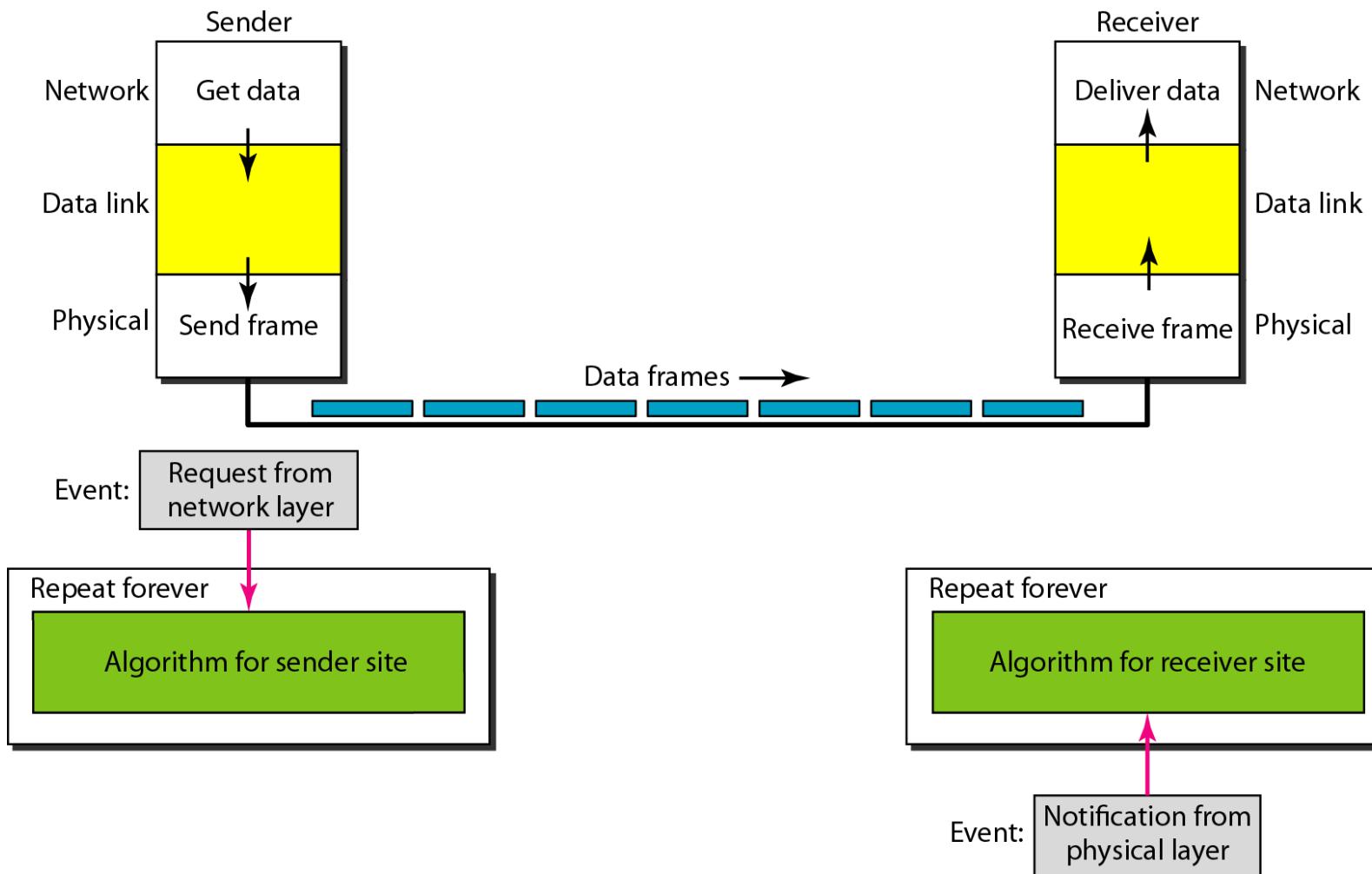
Let us first assume we have an ideal channel in which no frames are lost, duplicated, or corrupted. We introduce two protocols for this type of channel.

Topics discussed in this section:

Simplest Protocol

Stop-and-Wait Protocol

Figure 11.6 The design of the simplest protocol with no flow or error control



Algorithm 11.1 *Sender-site algorithm for the simplest protocol*

```
1 while(true)          // Repeat forever
2 {
3     WaitForEvent();    // Sleep until an event occurs
4     if(Event(RequestToSend)) //There is a packet to send
5     {
6         GetData();
7         MakeFrame();
8         SendFrame();      //Send the frame
9     }
10 }
```

Algorithm 11.2 *Receiver-site algorithm for the simplest protocol*

```
1 while(true)                                // Repeat forever
2 {
3     WaitForEvent();                         // Sleep until an event occurs
4     if(Event(ArrivalNotification)) //Data frame arrived
5     {
6         ReceiveFrame();
7         ExtractData();
8         DeliverData();                  //Deliver data to network layer
9     }
10 }
```

Example 11.1

Figure 11.7 shows an example of communication using this protocol. It is very simple. The sender sends a sequence of frames without even thinking about the receiver. To send three frames, three events occur at the sender site and three events at the receiver site. Note that the data frames are shown by tilted boxes; the height of the box defines the transmission time difference between the first bit and the last bit in the frame.

Figure 11.7 Flow diagram for Example 11.1

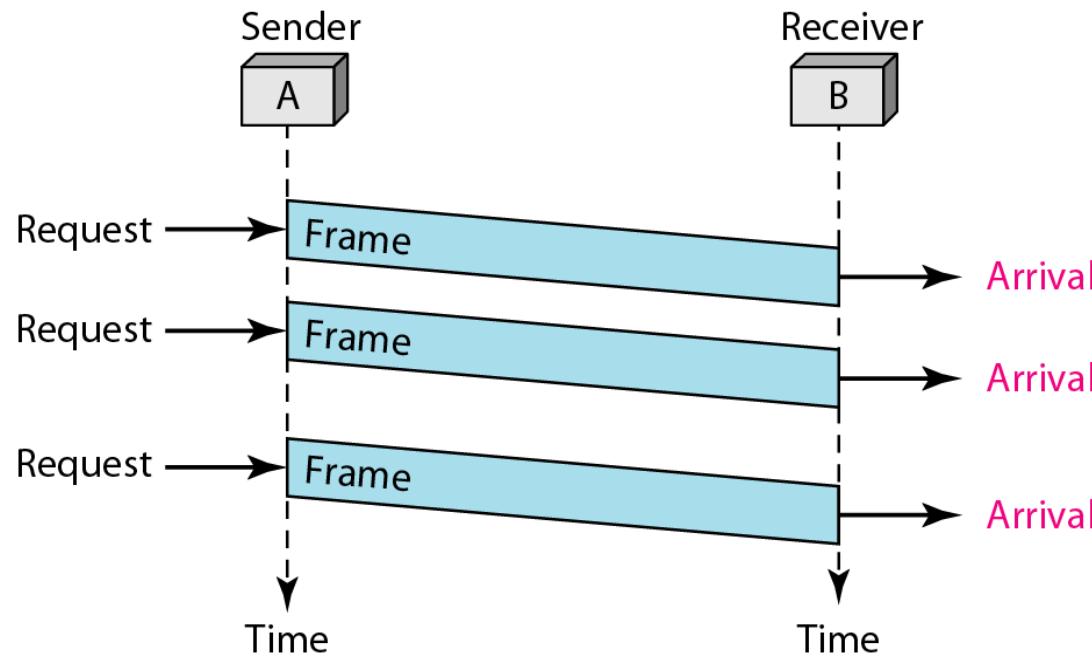
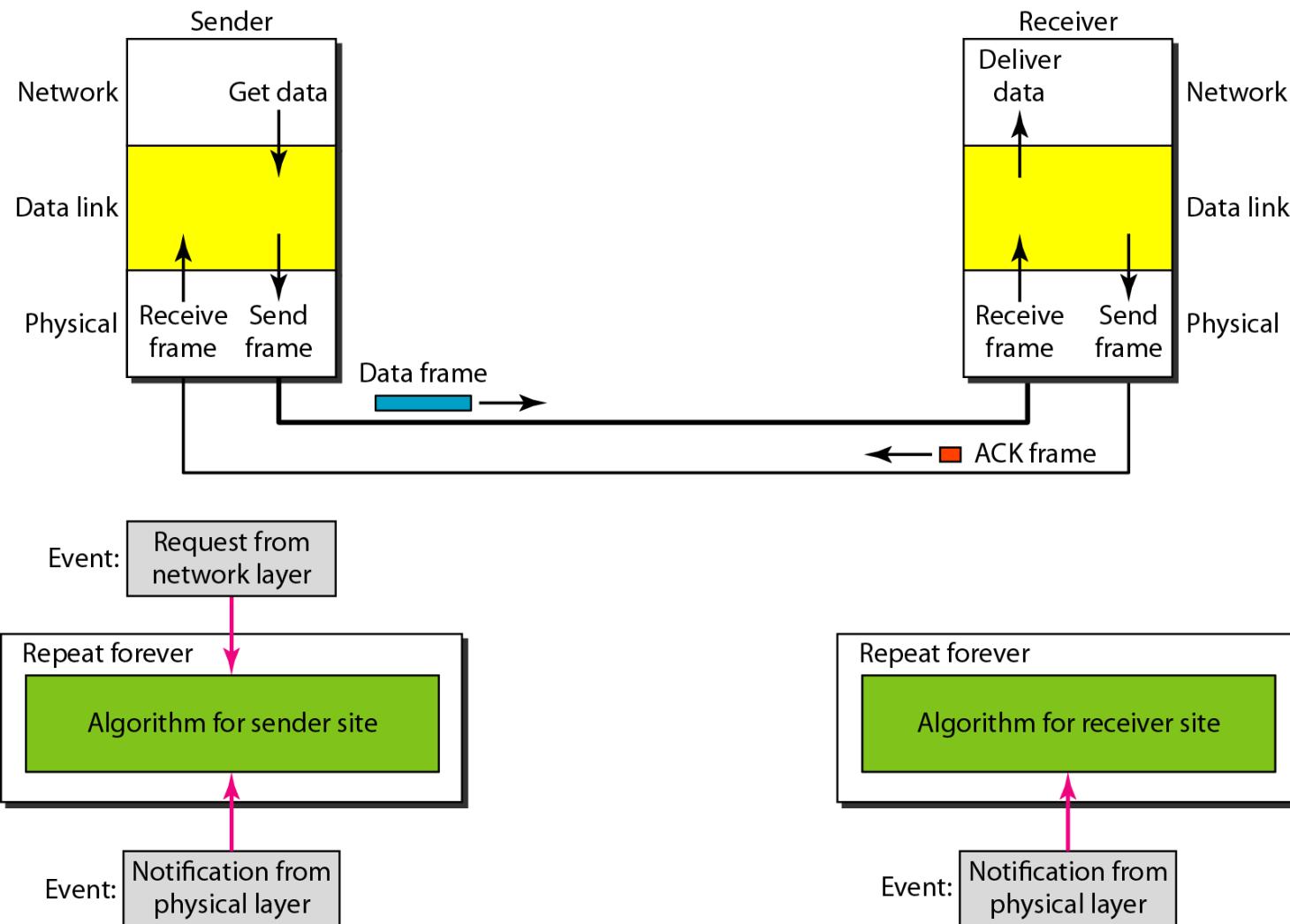


Figure 11.8 Design of Stop-and-Wait Protocol

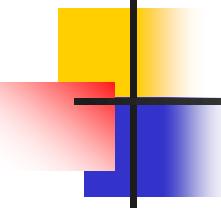


Algorithm 11.3 Sender-site algorithm for Stop-and-Wait Protocol

```
1 while(true)                                //Repeat forever
2 canSend = true                            //Allow the first frame to go
3 {
4     WaitForEvent();                      // Sleep until an event occurs
5     if(Event(RequestToSend) AND canSend)
6     {
7         GetData();
8         MakeFrame();
9         SendFrame();                     //Send the data frame
10        canSend = false;                //Cannot send until ACK arrives
11    }
12    WaitForEvent();                      // Sleep until an event occurs
13    if(Event(ArrivalNotification) // An ACK has arrived
14    {
15        ReceiveFrame();                //Receive the ACK frame
16        canSend = true;
17    }
18 }
```

Algorithm 11.4 *Receiver-site algorithm for Stop-and-Wait Protocol*

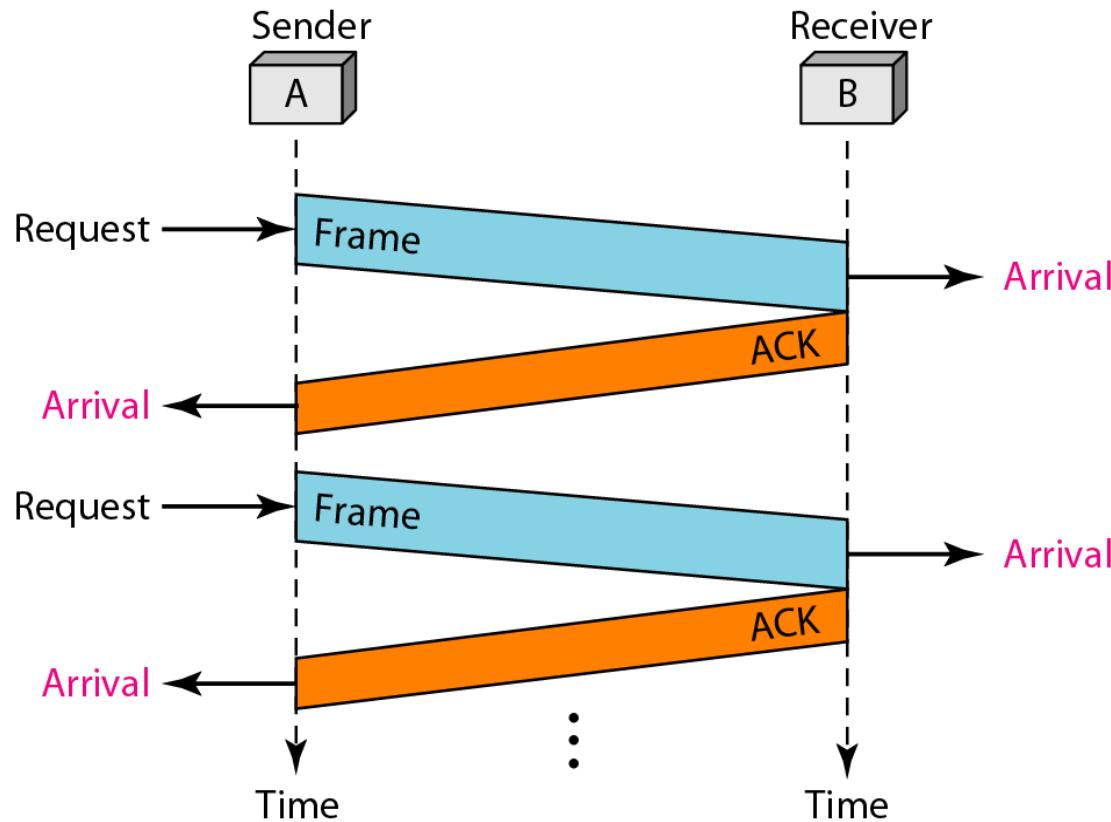
```
1 while(true)                                //Repeat forever
2 {
3     WaitForEvent();                         // Sleep until an event occurs
4     if(Event(ArrivalNotification)) //Data frame arrives
5     {
6         ReceiveFrame();
7         ExtractData();
8         Deliver(data);                  //Deliver data to network layer
9         SendFrame();                  //Send an ACK frame
10    }
11 }
```



Example 11.2

Figure 11.9 shows an example of communication using this protocol. It is still very simple. The sender sends one frame and waits for feedback from the receiver. When the ACK arrives, the sender sends the next frame. Note that sending two frames in the protocol involves the sender in four events and the receiver in two events.

Figure 11.9 Flow diagram for Example 11.2



11-5 NOISY CHANNELS

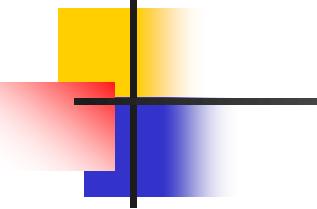
Although the Stop-and-Wait Protocol gives us an idea of how to add flow control to its predecessor, noiseless channels are nonexistent. We discuss three protocols in this section that use error control.

Topics discussed in this section:

Stop-and-Wait Automatic Repeat Request

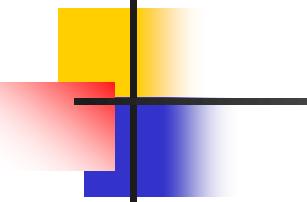
Go-Back-N Automatic Repeat Request

Selective Repeat Automatic Repeat Request



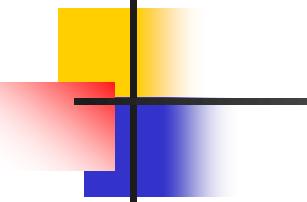
Note

Error correction in Stop-and-Wait ARQ is done by keeping a copy of the sent frame and retransmitting of the frame when the timer expires.



Note

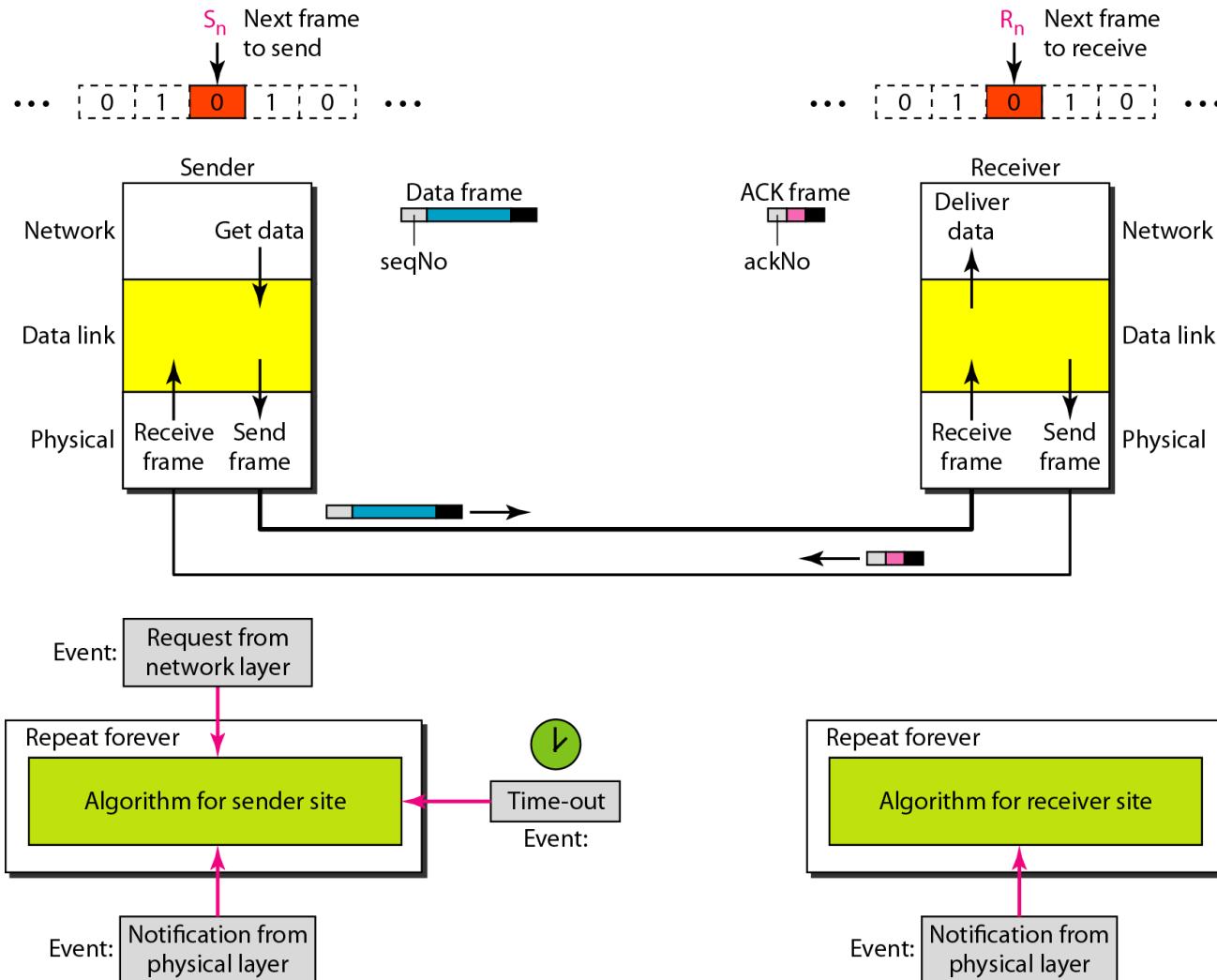
**In Stop-and-Wait ARQ, we use sequence numbers to number the frames.
The sequence numbers are based on modulo-2 arithmetic.**



Note

In Stop-and-Wait ARQ, the acknowledgment number always announces in modulo-2 arithmetic the sequence number of the next frame expected.

Figure 11.10 Design of the Stop-and-Wait ARQ Protocol



Algorithm 11.5 Sender-site algorithm for Stop-and-Wait ARQ

```
1 Sn = 0;                                // Frame 0 should be sent first
2 canSend = true;                           // Allow the first request to go
3 while(true)                               // Repeat forever
4 {
5   WaitForEvent();                         // Sleep until an event occurs
6   if(Event(RequestToSend) AND canSend)
7   {
8     GetData();
9     MakeFrame(Sn);                   //The seqNo is Sn
10    StoreFrame(Sn);                  //Keep copy
11    SendFrame(Sn);
12    StartTimer();
13    Sn = Sn + 1;
14    canSend = false;
15  }
16  WaitForEvent();                         // Sleep
```

(continued)

Algorithm 11.5 Sender-site algorithm for Stop-and-Wait ARQ (continued)

```
17     if(Event(ArrivalNotification))          // An ACK has arrived
18     {
19         ReceiveFrame(ackNo);           //Receive the ACK frame
20         if(not corrupted AND ackNo == Sn) //Valid ACK
21         {
22             StopTimer();
23             PurgeFrame(Sn-1);        //Copy is not needed
24             canSend = true;
25         }
26     }
27
28     if(Event(TimeOut))                  // The timer expired
29     {
30         StartTimer();
31         ResendFrame(Sn-1);        //Resend a copy check
32     }
33 }
```

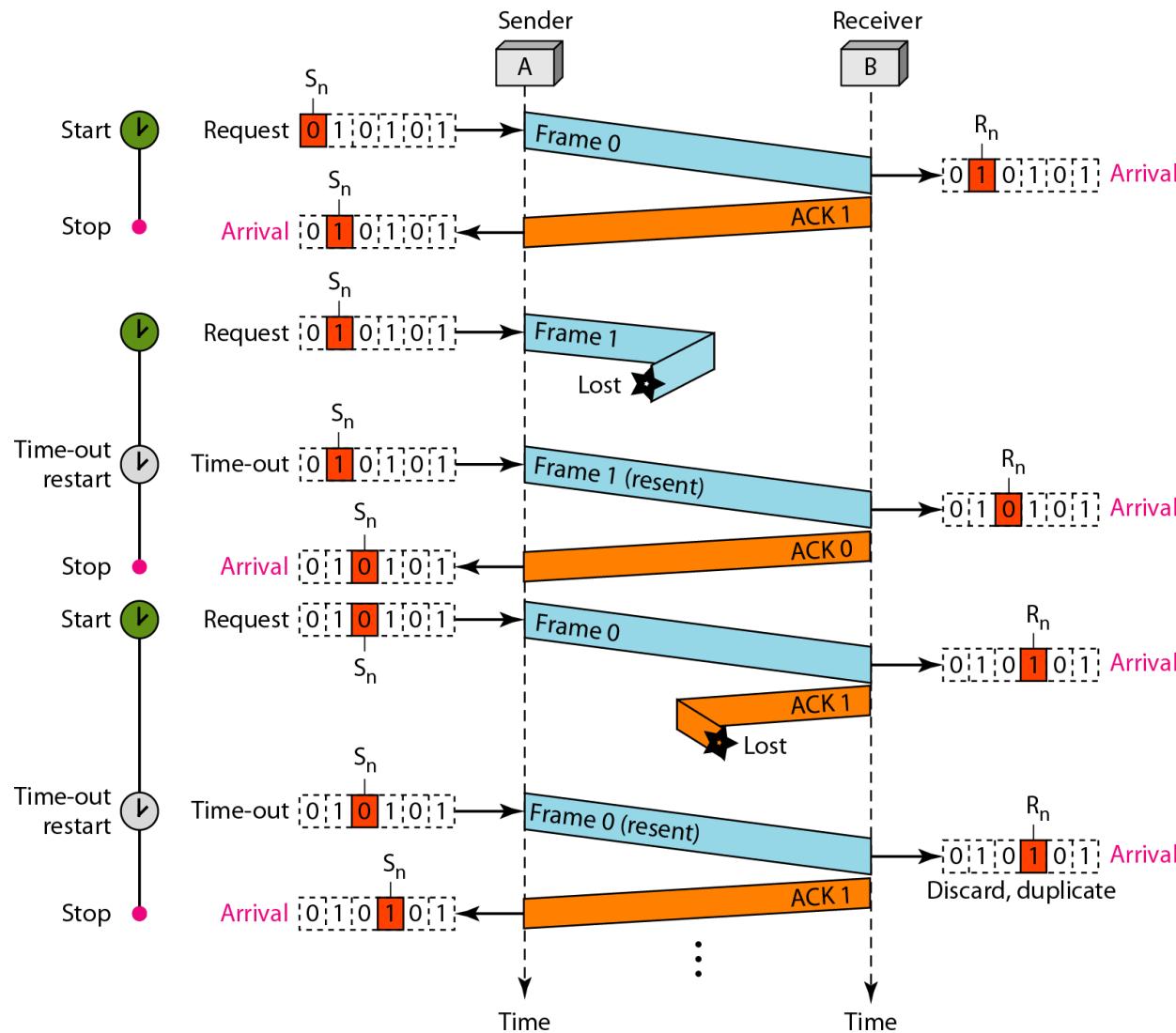
Algorithm 11.6 Receiver-site algorithm for Stop-and-Wait ARQ Protocol

```
1 Rn = 0;                                // Frame 0 expected to arrive first
2 while(true)
3 {
4     WaitForEvent();                      // Sleep until an event occurs
5     if(Event(ArrivalNotification))      //Data frame arrives
6     {
7         ReceiveFrame();
8         if(corrupted(frame));
9             sleep();
10        if(seqNo == Rn)              //Valid data frame
11        {
12            ExtractData();
13            DeliverData();           //Deliver data
14            Rn = Rn + 1;
15        }
16        SendFrame(Rn);           //Send an ACK
17    }
18 }
```

Example 11.3

Figure 11.11 shows an example of Stop-and-Wait ARQ. Frame 0 is sent and acknowledged. Frame 1 is lost and resent after the time-out. The resent frame 1 is acknowledged and the timer stops. Frame 0 is sent and acknowledged, but the acknowledgment is lost. The sender has no idea if the frame or the acknowledgment is lost, so after the time-out, it resends frame 0, which is acknowledged.

Figure 11.11 Flow diagram for Example 11.3



Example 11.4

Assume that, in a Stop-and-Wait ARQ system, the bandwidth of the line is 1 Mbps, and 1 bit takes 20 ms to make a round trip. What is the bandwidth-delay product? If the system data frames are 1000 bits in length, what is the utilization percentage of the link?

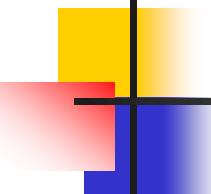
Solution

The bandwidth-delay product is

$$(1 \times 10^6) \times (20 \times 10^{-3}) = 20,000 \text{ bits}$$

Example 11.4 (continued)

The system can send 20,000 bits during the time it takes for the data to go from the sender to the receiver and then back again. However, the system sends only 1000 bits. We can say that the link utilization is only $1000/20,000$, or 5 percent. For this reason, for a link with a high bandwidth or long delay, the use of Stop-and-Wait ARQ wastes the capacity of the link.

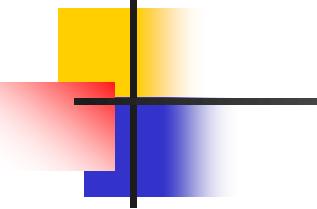


Example 11.5

What is the utilization percentage of the link in Example 11.4 if we have a protocol that can send up to 15 frames before stopping and worrying about the acknowledgments?

Solution

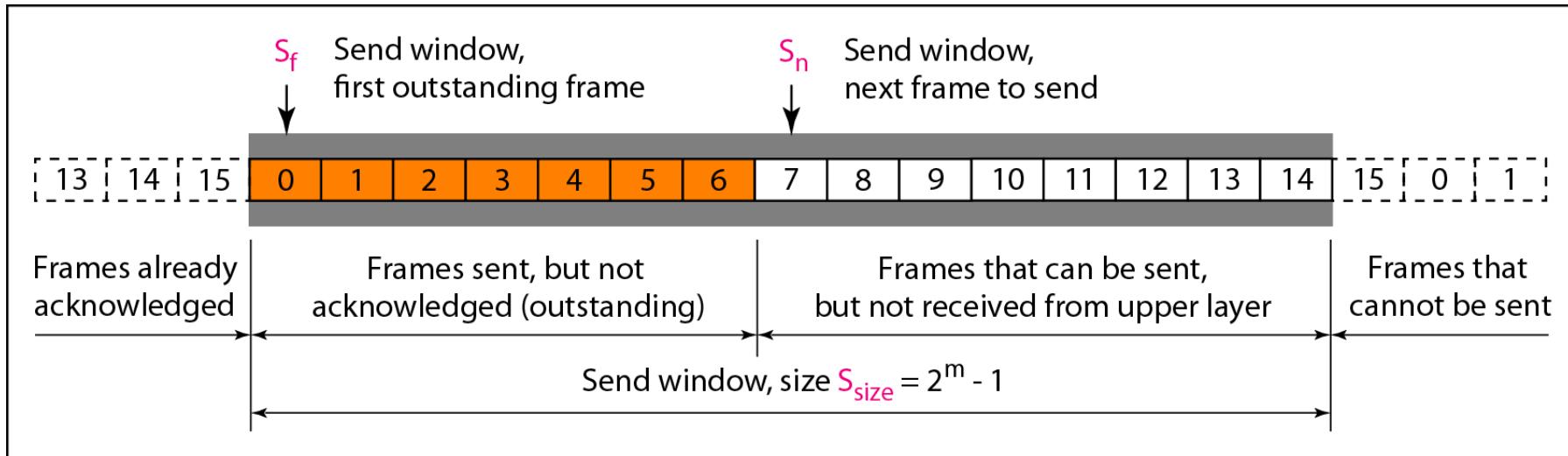
The bandwidth-delay product is still 20,000 bits. The system can send up to 15 frames or 15,000 bits during a round trip. This means the utilization is 15,000/20,000, or 75 percent. Of course, if there are damaged frames, the utilization percentage is much less because frames have to be resent.



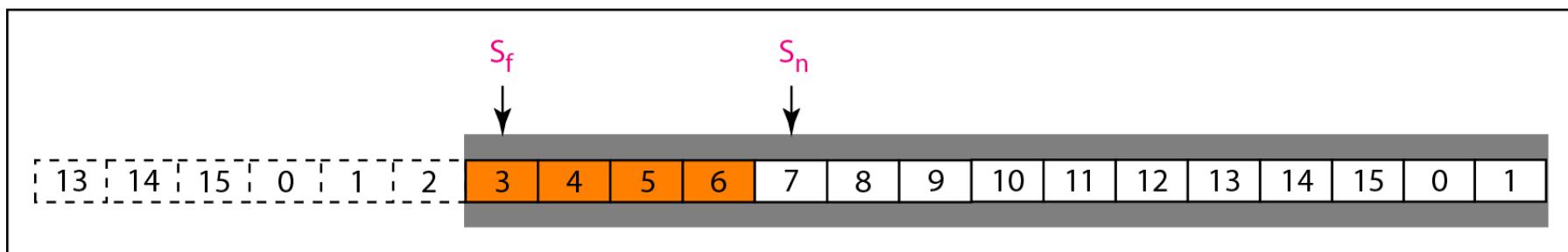
Note

In the Go-Back-N Protocol, the sequence numbers are modulo 2^m , where m is the size of the sequence number field in bits.

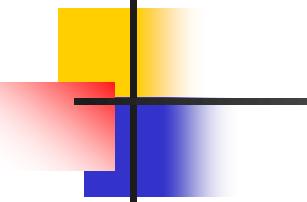
Figure 11.12 Send window for Go-Back-N ARQ



a. Send window before sliding

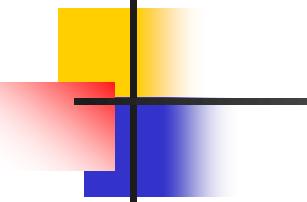


b. Send window after sliding



Note

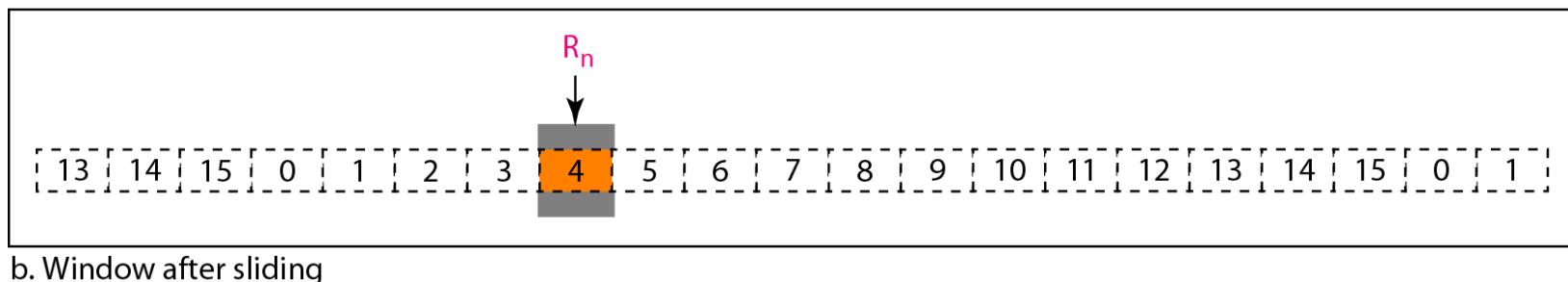
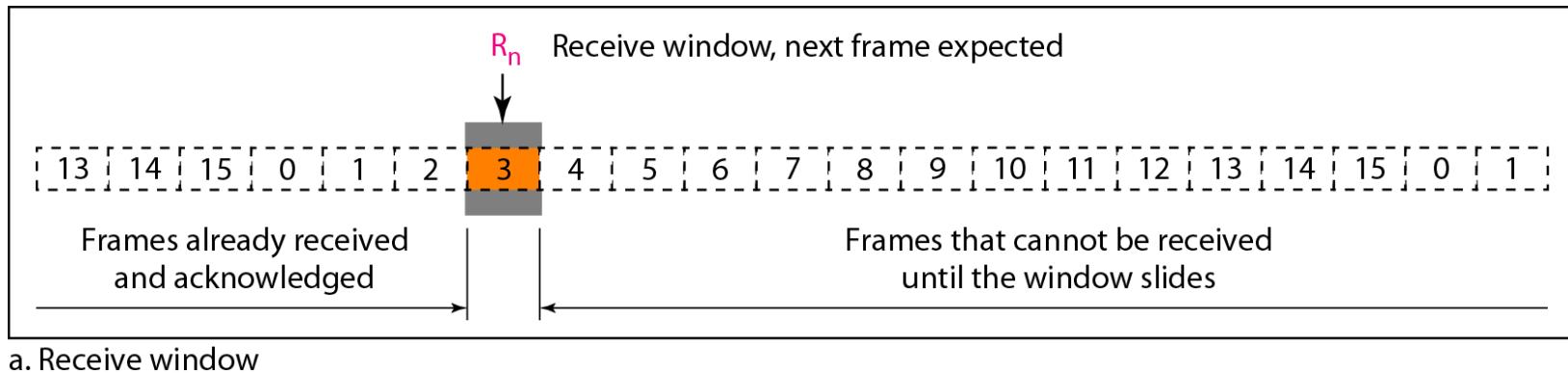
The send window is an abstract concept defining an imaginary box of size $2^m - 1$ with three variables: S_f , S_n , and S_{size} .

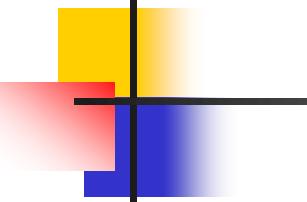


Note

The send window can slide one or more slots when a valid acknowledgment arrives.

Figure 11.13 Receive window for Go-Back-N ARQ





Note

The receive window is an abstract concept defining an imaginary box of size 1 with one single variable R_n .

The window slides when a correct frame has arrived; sliding occurs one slot at a time.

Figure 11.14 Design of Go-Back-N ARQ

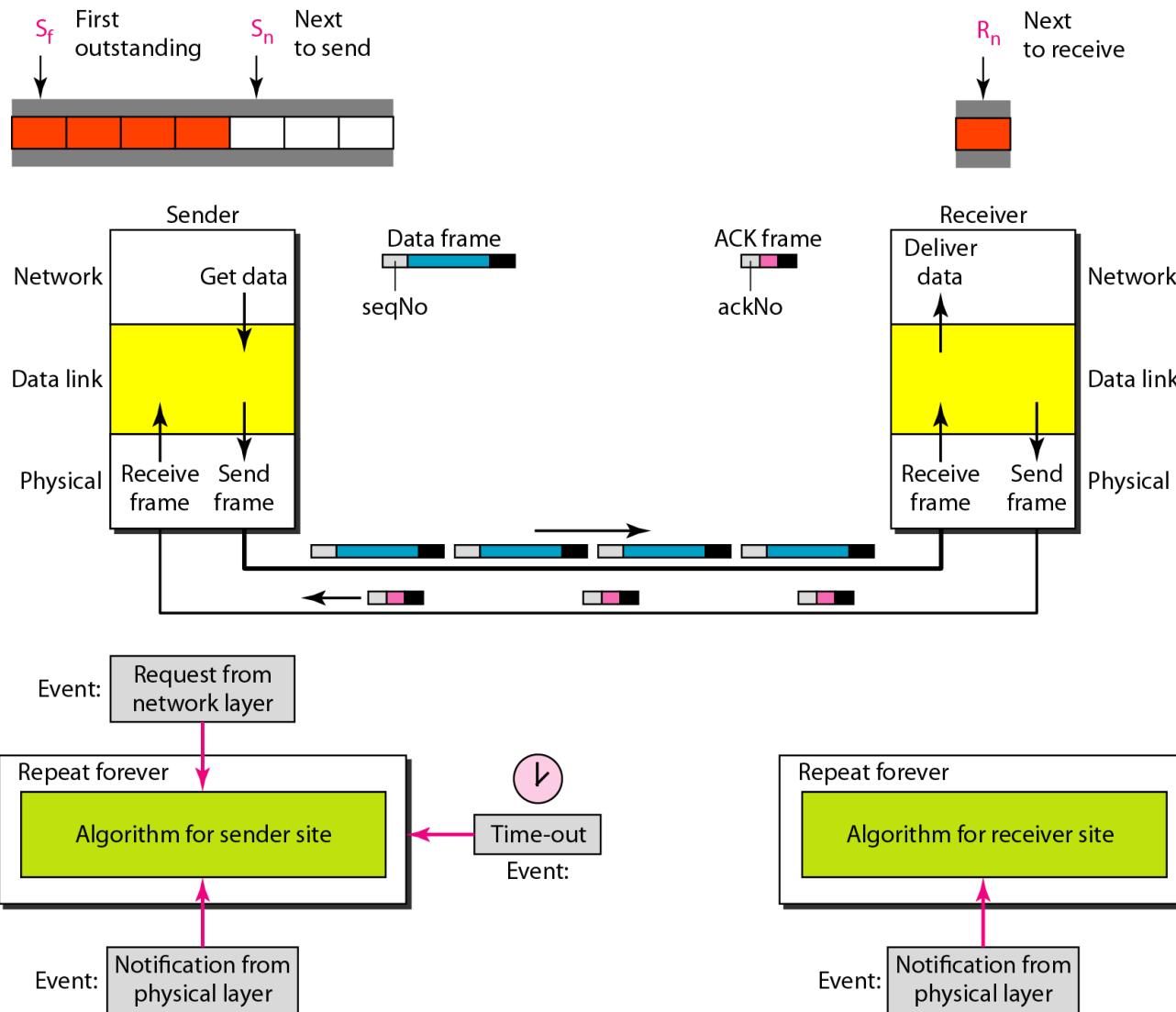
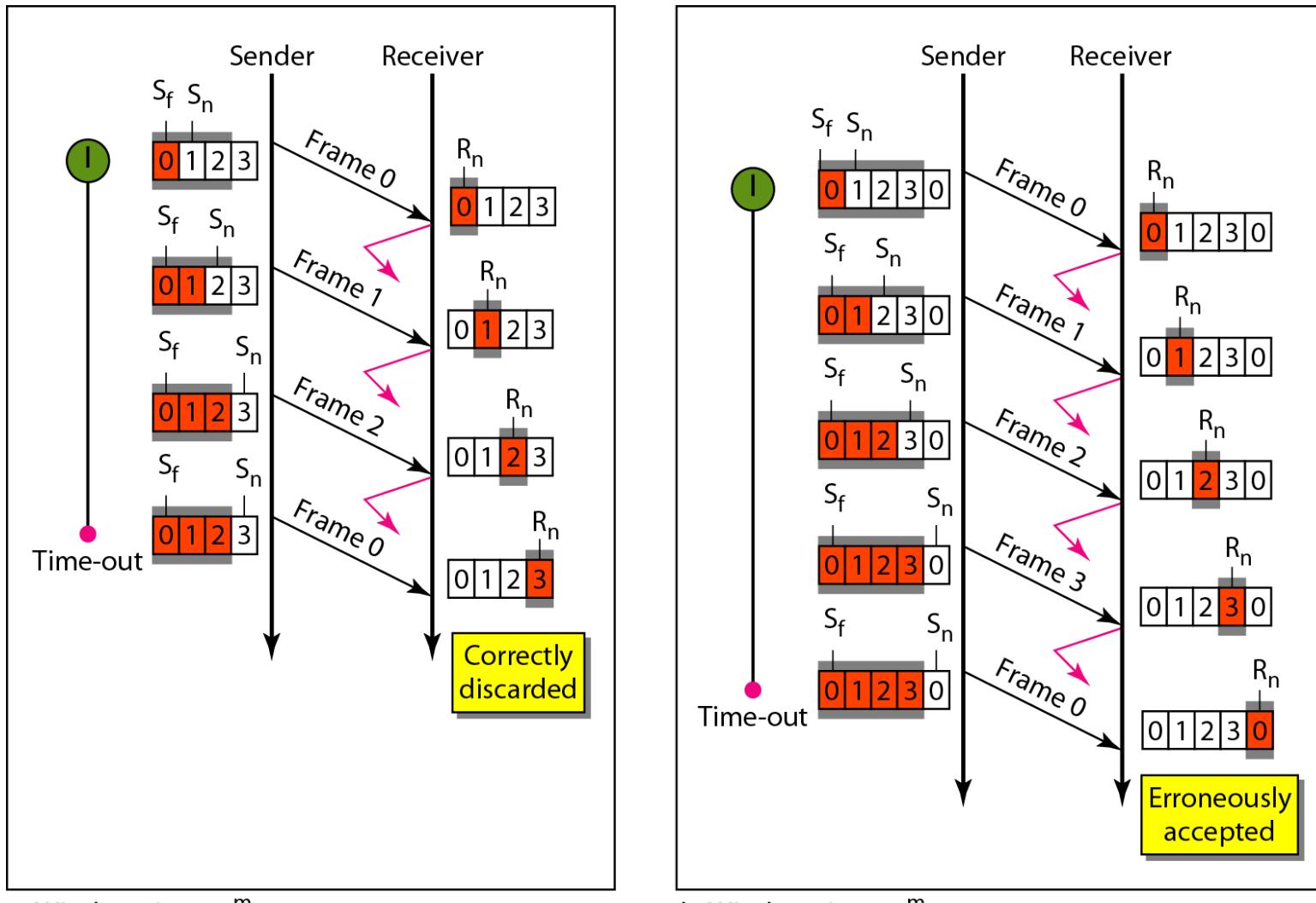
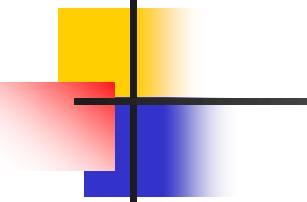


Figure 11.15 Window size for Go-Back-N ARQ



a. Window size < 2^m

b. Window size = 2^m



Note

In Go-Back-N ARQ, the size of the send window must be less than 2^m ; the size of the receiver window is always 1.

Algorithm 11.7 Go-Back-N sender algorithm

```
1 Sw = 2m - 1;  
2 Sf = 0;  
3 Sn = 0;  
4  
5 while (true) //Repeat forever  
6 {  
7   WaitForEvent();  
8   if(Event(RequestToSend)) //A packet to send  
9   {  
10     if(Sn-Sf >= Sw) //If window is full  
11       Sleep();  
12     GetData();  
13     MakeFrame(Sn);  
14     StoreFrame(Sn);  
15     SendFrame(Sn);  
16     Sn = Sn + 1;  
17     if(timer not running)  
18       StartTimer();  
19   }  
20 }
```

(continued)

Algorithm 11.7 Go-Back-N sender algorithm

(continued)

```
21  if(Event(ArrivalNotification)) //ACK arrives
22  {
23      Receive(ACK);
24      if(corrupted(ACK))
25          Sleep();
26      if((ackNo>Sf)&&(ackNo<=Sn)) //If a valid ACK
27      While(Sf <= ackNo)
28      {
29          PurgeFrame(Sf);
30          Sf = Sf + 1;
31      }
32      StopTimer();
33  }

34

35  if(Event(TimeOut)) //The timer expires
36  {
37      StartTimer();
38      Temp = Sf;
39      while(Temp < Sn);
40      {
41          SendFrame(Sf);
42          Sf = Sf + 1;
43      }
44  }
45 }
```

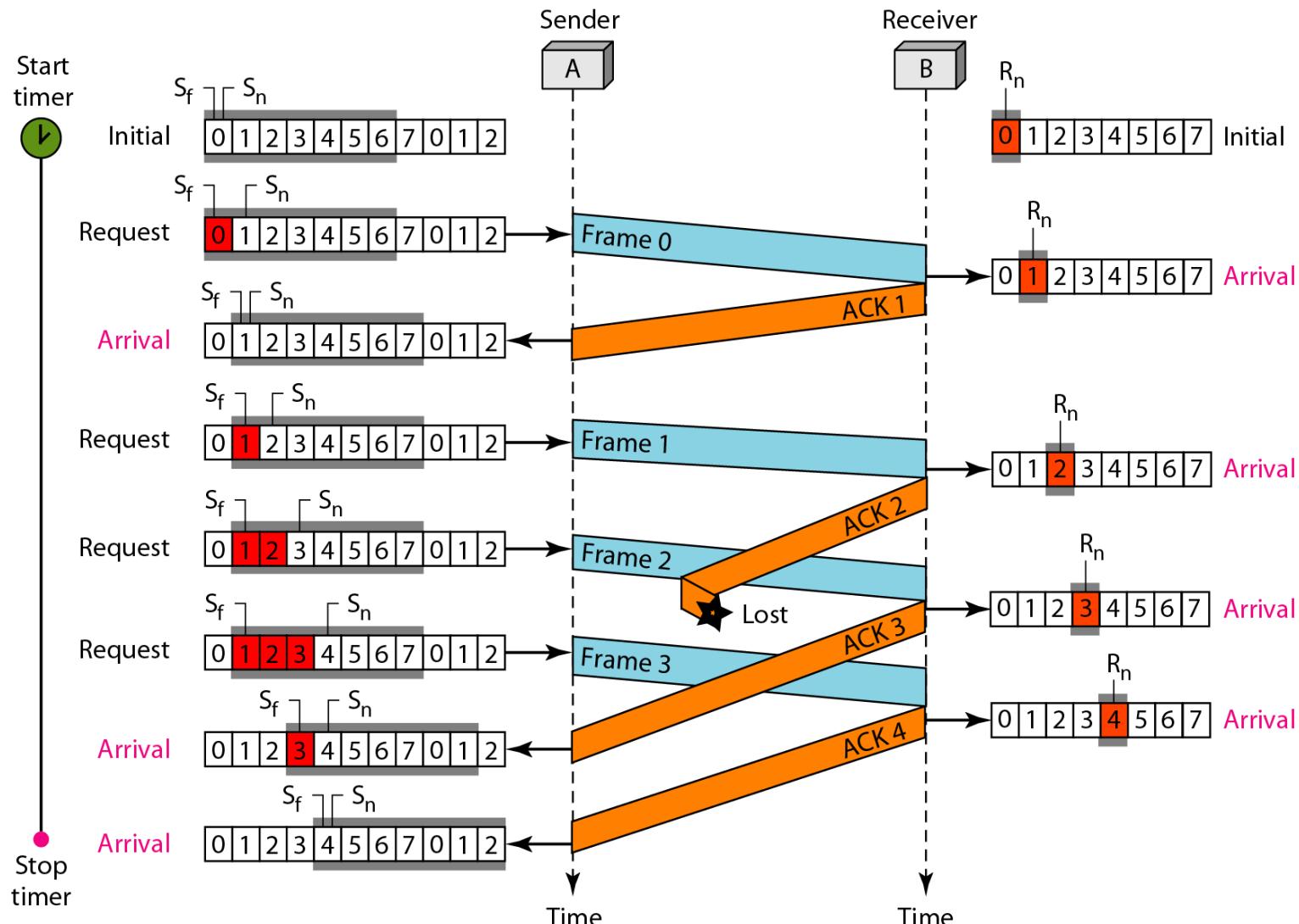
Algorithm 11.8 Go-Back-N receiver algorithm

```
1 Rn = 0;  
2  
3 while (true) //Repeat forever  
4 {  
5     WaitForEvent();  
6  
7     if(Event(ArrivalNotification)) /Data frame arrives  
8     {  
9         Receive(Frame);  
10        if(corrupted(Frame))  
11            Sleep();  
12        if(seqNo == Rn) //If expected frame  
13        {  
14            DeliverData(); //Deliver data  
15            Rn = Rn + 1; //Slide window  
16            SendACK(Rn);  
17        }  
18    }  
19}
```

Example 11.6

Figure 11.16 shows an example of Go-Back-N. This is an example of a case where the forward channel is reliable, but the reverse is not. No data frames are lost, but some ACKs are delayed and one is lost. The example also shows how cumulative acknowledgments can help if acknowledgments are delayed or lost. After initialization, there are seven sender events. Request events are triggered by data from the network layer; arrival events are triggered by acknowledgments from the physical layer. There is no time-out event here because all outstanding frames are acknowledged before the timer expires. Note that although ACK 2 is lost, ACK 3 serves as both ACK 2 and ACK 3.

Figure 11.16 Flow diagram for Example 11.6



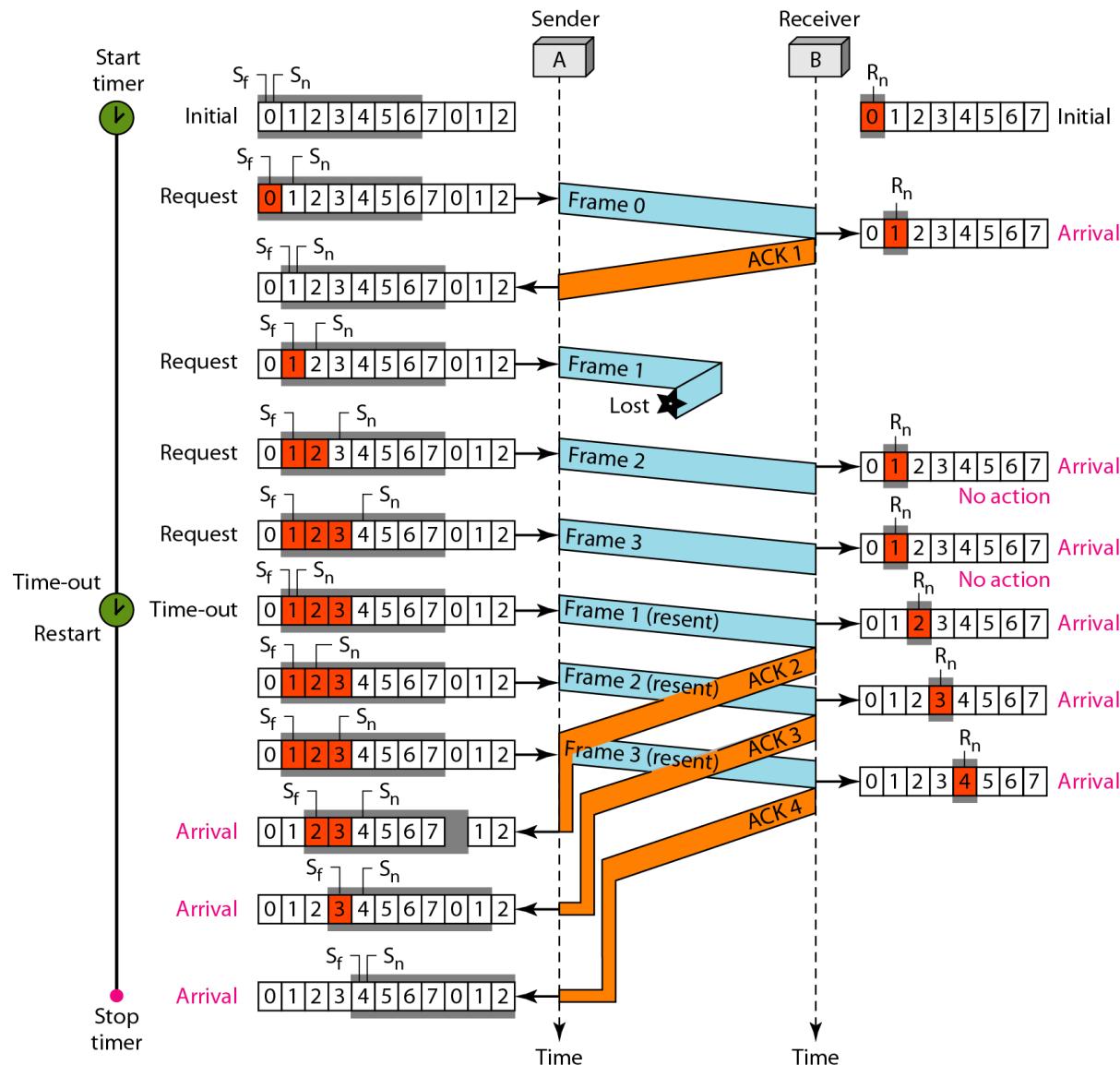
Example 11.7

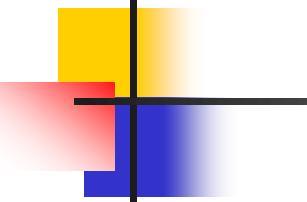
Figure 11.17 shows what happens when a frame is lost. Frames 0, 1, 2, and 3 are sent. However, frame 1 is lost. The receiver receives frames 2 and 3, but they are discarded because they are received out of order. The sender receives no acknowledgment about frames 1, 2, or 3. Its timer finally expires. The sender sends all outstanding frames (1, 2, and 3) because it does not know what is wrong. Note that the resending of frames 1, 2, and 3 is the response to one single event. When the sender is responding to this event, it cannot accept the triggering of other events. This means that when ACK 2 arrives, the sender is still busy with sending frame 3.

Example 11.7 (continued)

The physical layer must wait until this event is completed and the data link layer goes back to its sleeping state. We have shown a vertical line to indicate the delay. It is the same story with ACK 3; but when ACK 3 arrives, the sender is busy responding to ACK 2. It happens again when ACK 4 arrives. Note that before the second timer expires, all outstanding frames have been sent and the timer is stopped.

Figure 11.17 Flow diagram for Example 11.7





Note

Stop-and-Wait ARQ is a special case of Go-Back-N ARQ in which the size of the send window is 1.

Figure 11.18 Send window for Selective Repeat ARQ

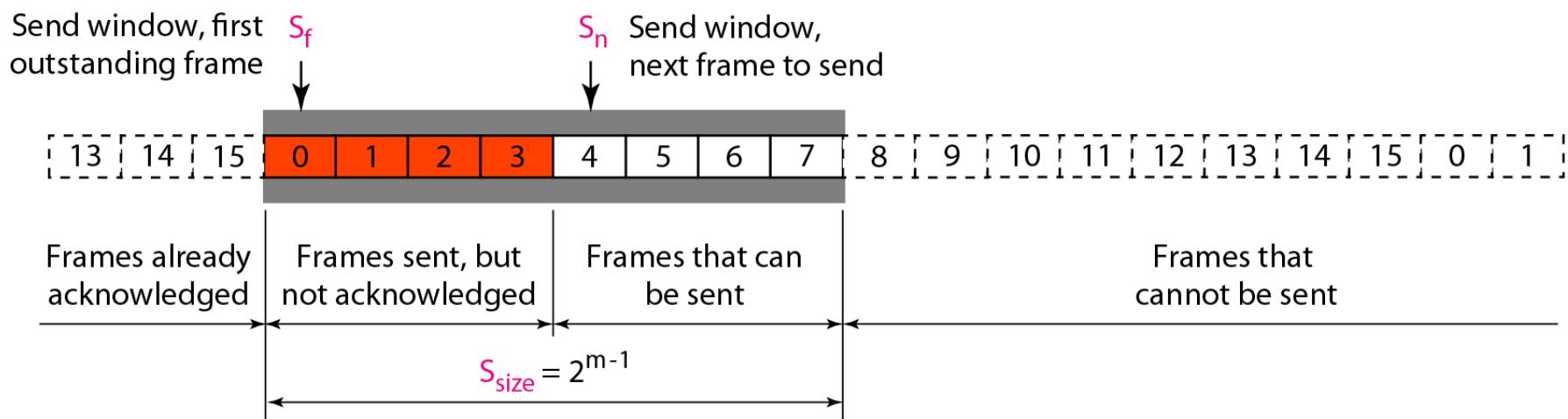


Figure 11.19 *Receive window for Selective Repeat ARQ*

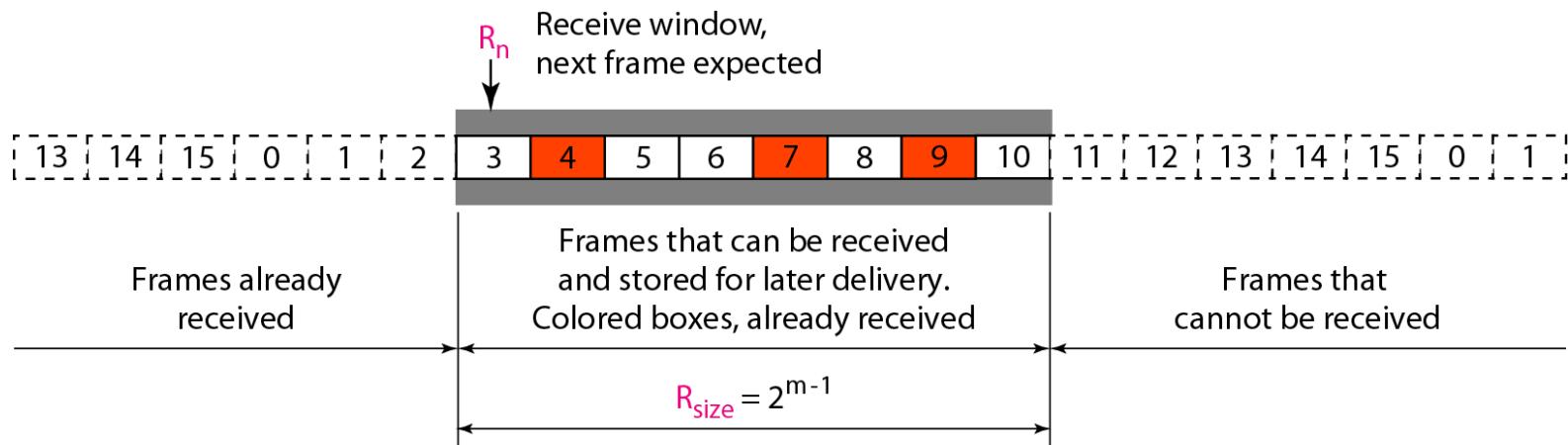


Figure 11.20 Design of Selective Repeat ARQ

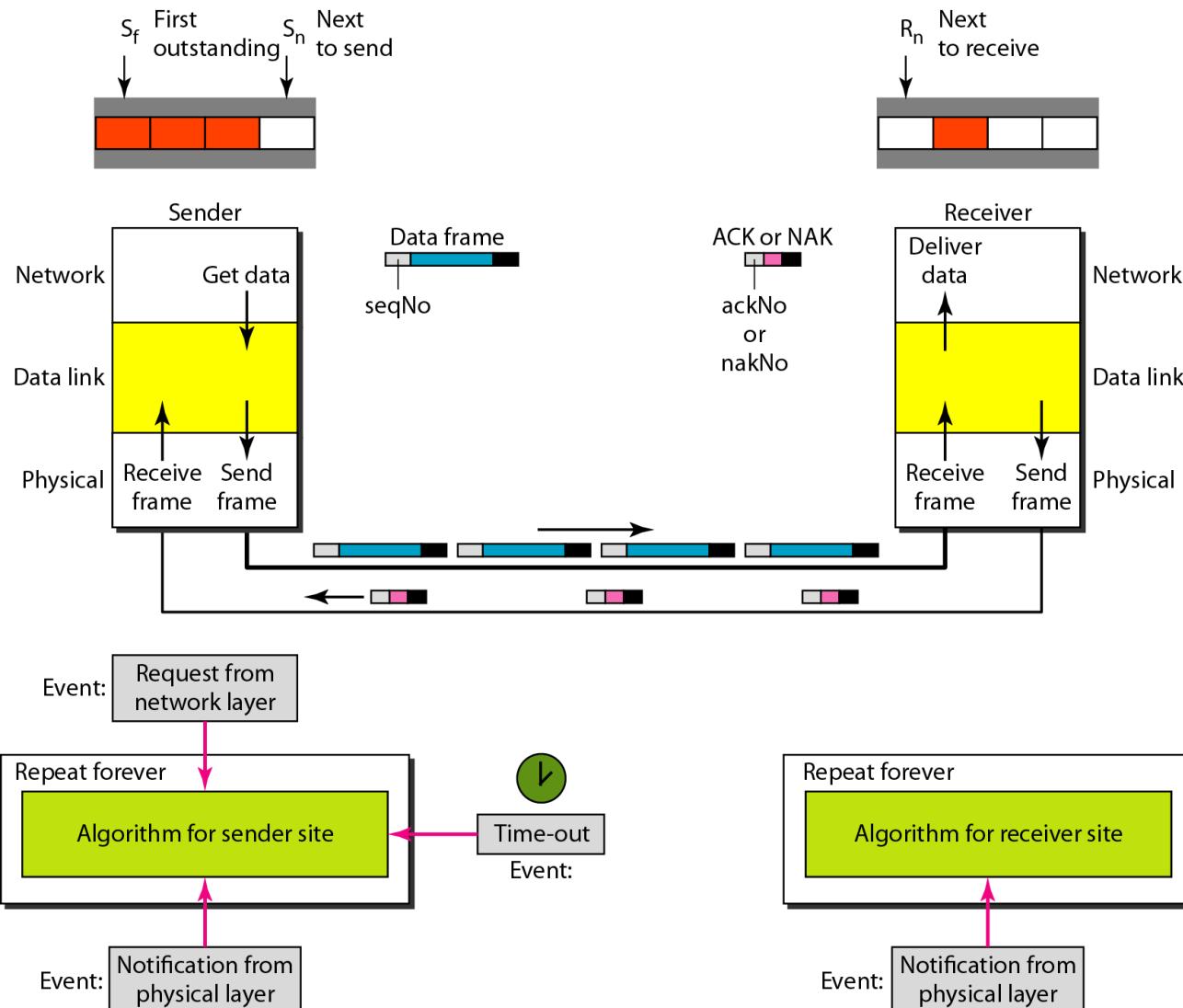
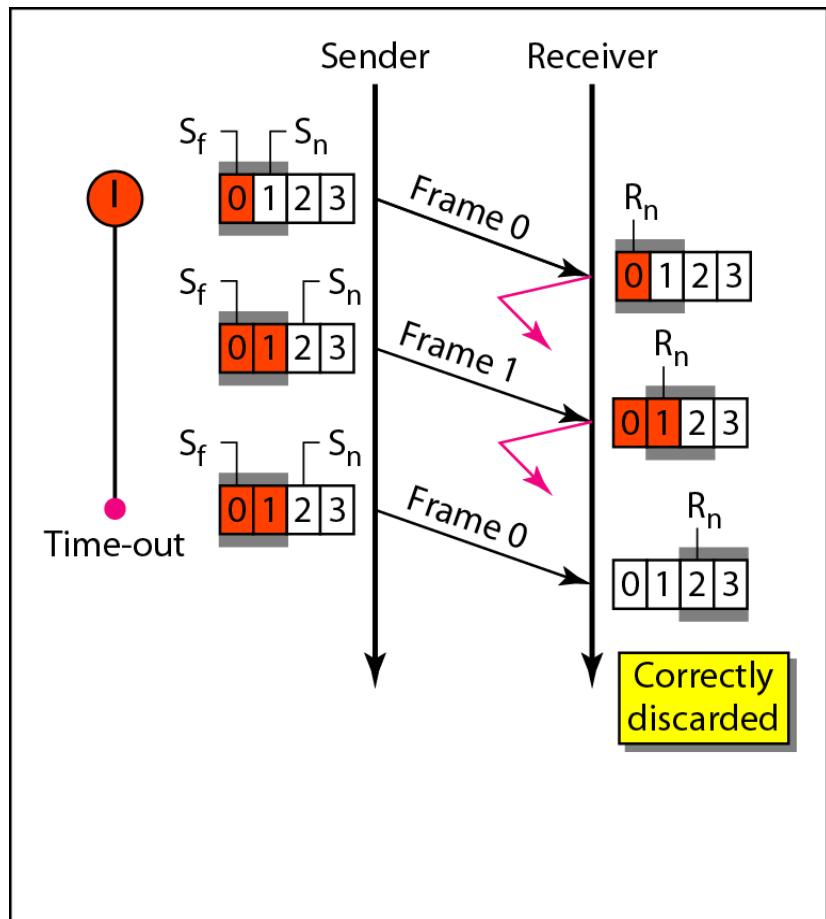
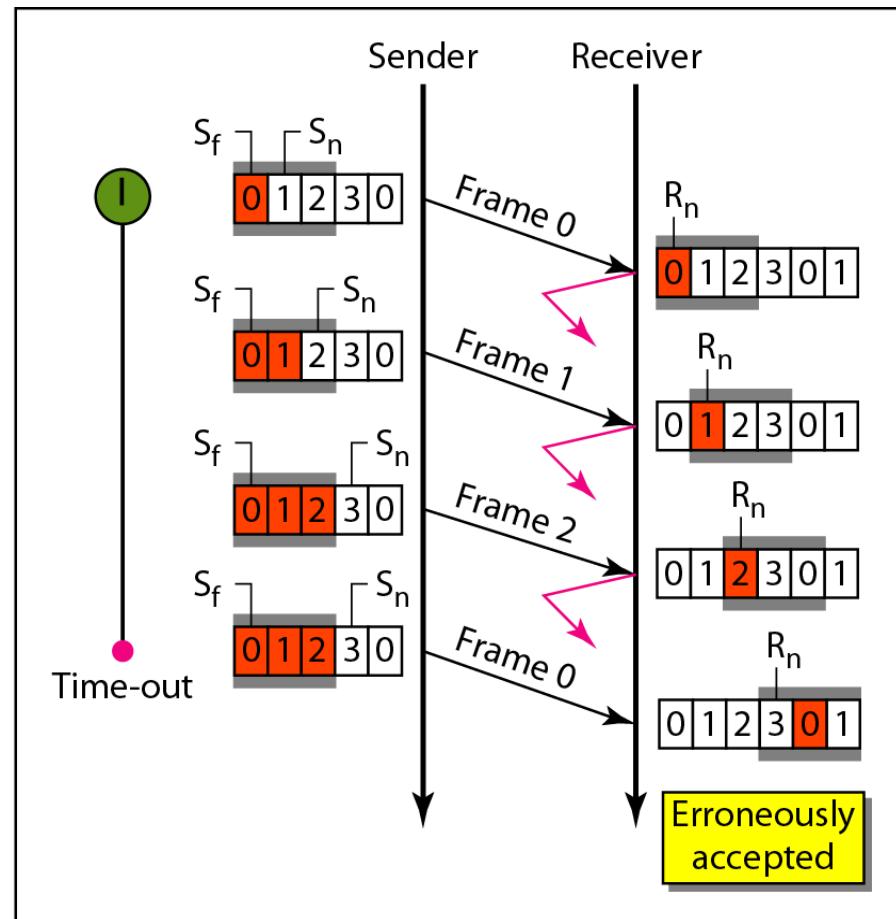


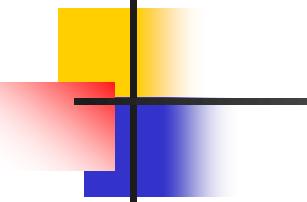
Figure 11.21 Selective Repeat ARQ, window size



a. Window size = 2^{m-1}



b. Window size > 2^{m-1}



Note

In Selective Repeat ARQ, the size of the sender and receiver window must be at most one-half of 2^m .

Algorithm 11.9 Sender-site Selective Repeat algorithm

```
1 Sw = 2m-1 ;
2 Sf = 0 ;
3 Sn = 0 ;
4
5 while (true)                                //Repeat forever
6 {
7     WaitForEvent() ;
8     if(Event(RequestToSend))                  //There is a packet to send
9     {
10         if(Sn-Sf >= Sw)                //If window is full
11             Sleep() ;
12         GetData() ;
13         MakeFrame(Sn) ;
14         StoreFrame(Sn) ;
15         SendFrame(Sn) ;
16         Sn = Sn + 1 ;
17         StartTimer(Sn) ;
18     }
19 }
```

(continued)

Algorithm 11.9 *Sender-site Selective Repeat algorithm*

(continued)

```
20  if(Event(ArrivalNotification)) //ACK arrives
21  {
22      Receive(frame);           //Receive ACK or NAK
23      if(corrupted(frame))
24          Sleep();
25      if (FrameType == NAK)
26          if (nakNo between Sf and Sn)
27          {
28              resend(nakNo);
29              StartTimer(nakNo);
30          }
31      if (FrameType == ACK)
32          if (ackNo between Sf and Sn)
33          {
34              while(sf < ackNo)
35              {
36                  Purge(sf);
37                  StopTimer(sf);
38                  Sf = Sf + 1;
39              }
40          }
41 }
```

(continued)

Algorithm 11.9 *Sender-site Selective Repeat algorithm*

(continued)

```
42  
43     if(Event(TimeOut(t)))          //The timer expires  
44     {  
45         StartTimer(t);  
46         SendFrame(t);  
47     }  
48 }
```

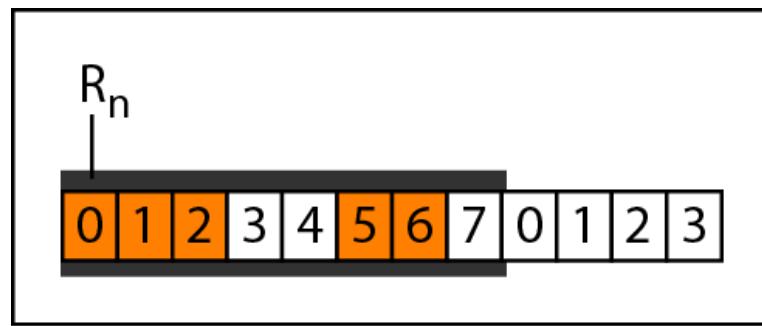
Algorithm 11.10 *Receiver-site Selective Repeat algorithm*

```
1 Rn = 0;
2 NakSent = false;
3 AckNeeded = false;
4 Repeat(for all slots)
5     Marked(slot) = false;
6
7 while (true)                                //Repeat forever
8 {
9     WaitForEvent();
10
11    if(Event(ArrivalNotification))           /Data frame arrives
12    {
13        Receive(Frame);
14        if(corrupted(Frame))&& (NOT NakSent)
15        {
16            SendNAK(Rn);
17            NakSent = true;
18            Sleep();
19        }
20        if(seqNo <> Rn)&& (NOT NakSent)
21        {
22            SendNAK(Rn);
```

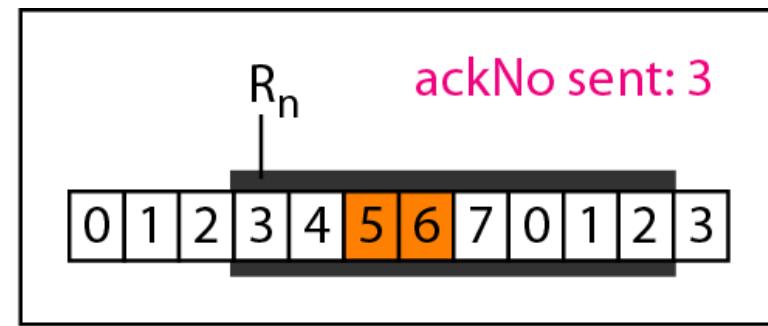
Algorithm 11.10 *Receiver-site Selective Repeat algorithm*

```
23     NakSent = true;
24     if ((seqNo in window) && (!Marked(seqNo)))
25     {
26         StoreFrame(seqNo)
27         Marked(seqNo)= true;
28         while(Marked(Rn))
29         {
30             DeliverData(Rn);
31             Purge(Rn);
32             Rn = Rn + 1;
33             AckNeeded = true;
34         }
35         if(AckNeeded);
36         {
37             SendAck(Rn);
38             AckNeeded = false;
39             NakSent = false;
40         }
41     }
42 }
43 }
44 }
```

Figure 11.22 *Delivery of data in Selective Repeat ARQ*



a. Before delivery



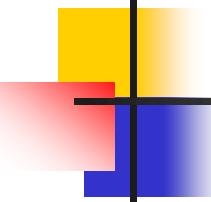
b. After delivery

Example 11.8

This example is similar to Example 11.3 in which frame 1 is lost. We show how Selective Repeat behaves in this case. Figure 11.23 shows the situation. One main difference is the number of timers. Here, each frame sent or resent needs a timer, which means that the timers need to be numbered (0, 1, 2, and 3). The timer for frame 0 starts at the first request, but stops when the ACK for this frame arrives. The timer for frame 1 starts at the second request, restarts when a NAK arrives, and finally stops when the last ACK arrives. The other two timers start when the corresponding frames are sent and stop at the last arrival event.

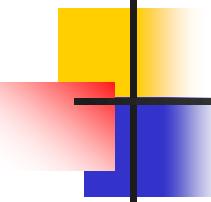
Example 11.8 (continued)

At the receiver site we need to distinguish between the acceptance of a frame and its delivery to the network layer. At the second arrival, frame 2 arrives and is stored and marked, but it cannot be delivered because frame 1 is missing. At the next arrival, frame 3 arrives and is marked and stored, but still none of the frames can be delivered. Only at the last arrival, when finally a copy of frame 1 arrives, can frames 1, 2, and 3 be delivered to the network layer. There are two conditions for the delivery of frames to the network layer: First, a set of consecutive frames must have arrived. Second, the set starts from the beginning of the window.



Example 11.8 (continued)

Another important point is that a NAK is sent after the second arrival, but not after the third, although both situations look the same. The reason is that the protocol does not want to crowd the network with unnecessary NAKs and unnecessary resent frames. The second NAK would still be NAK1 to inform the sender to resend frame 1 again; this has already been done. The first NAK sent is remembered (using the nakSent variable) and is not sent again until the frame slides. A NAK is sent once for each window position and defines the first slot in the window.



Example 11.8 (continued)

The next point is about the ACKs. Notice that only two ACKs are sent here. The first one acknowledges only the first frame; the second one acknowledges three frames. In Selective Repeat, ACKs are sent when data are delivered to the network layer. If the data belonging to n frames are delivered in one shot, only one ACK is sent for all of them.

Figure 11.23 Flow diagram for Example 11.8

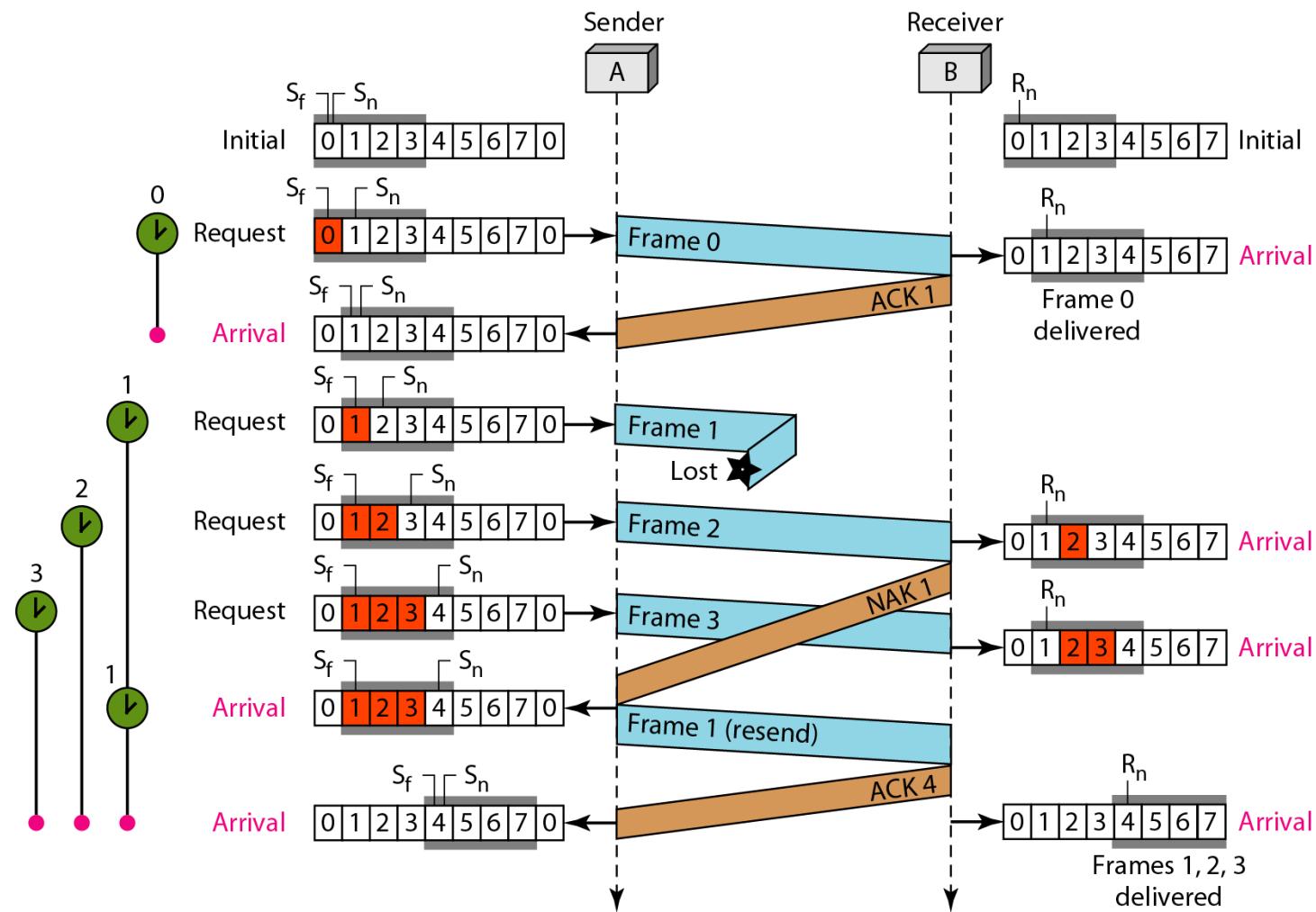
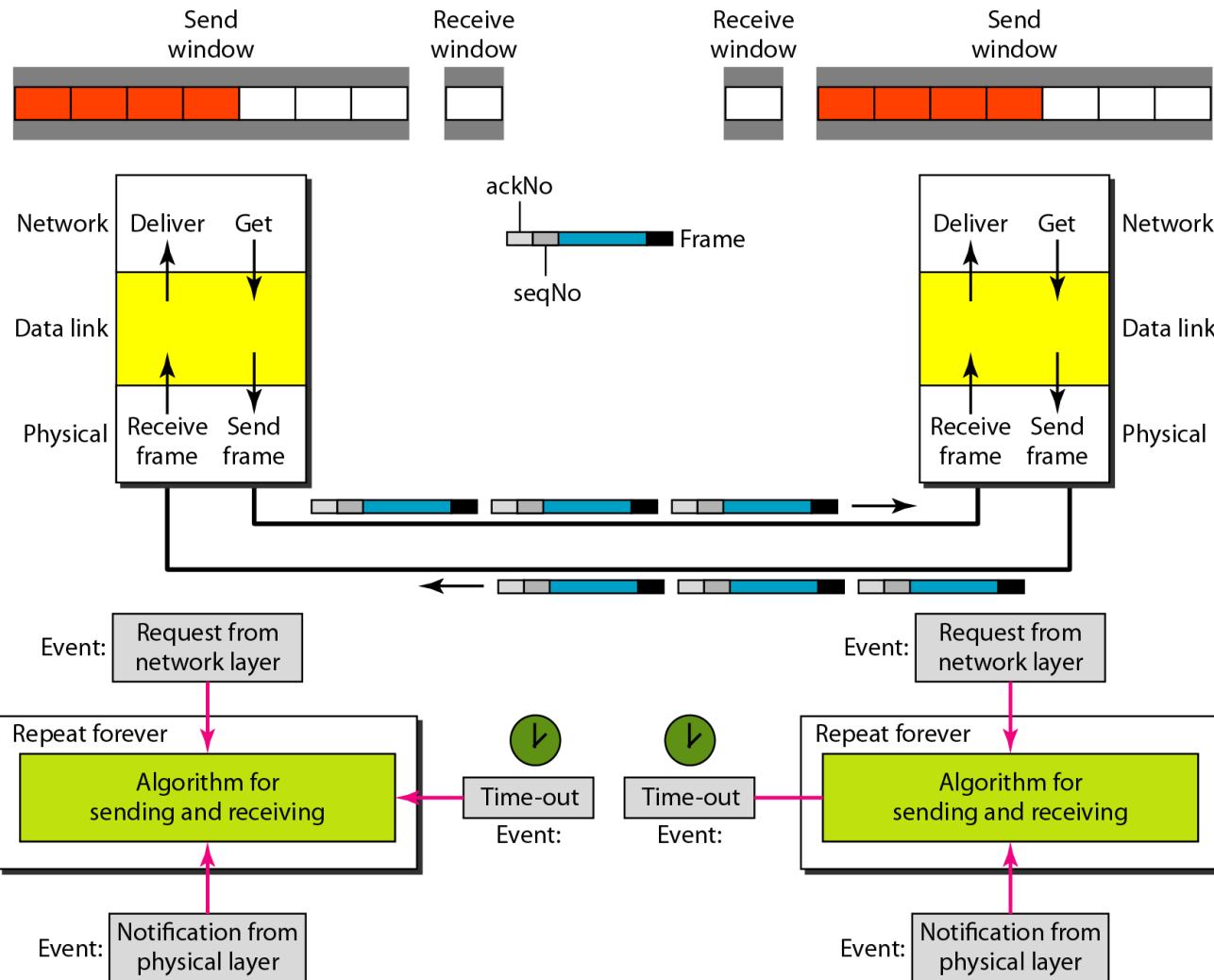


Figure 11.24 Design of piggybacking in Go-Back-N ARQ



11-6 HDLC

High-level Data Link Control (HDLC) is a bit-oriented protocol for communication over point-to-point and multipoint links. It implements the ARQ mechanisms we discussed in this chapter.

Topics discussed in this section:

Configurations and Transfer Modes

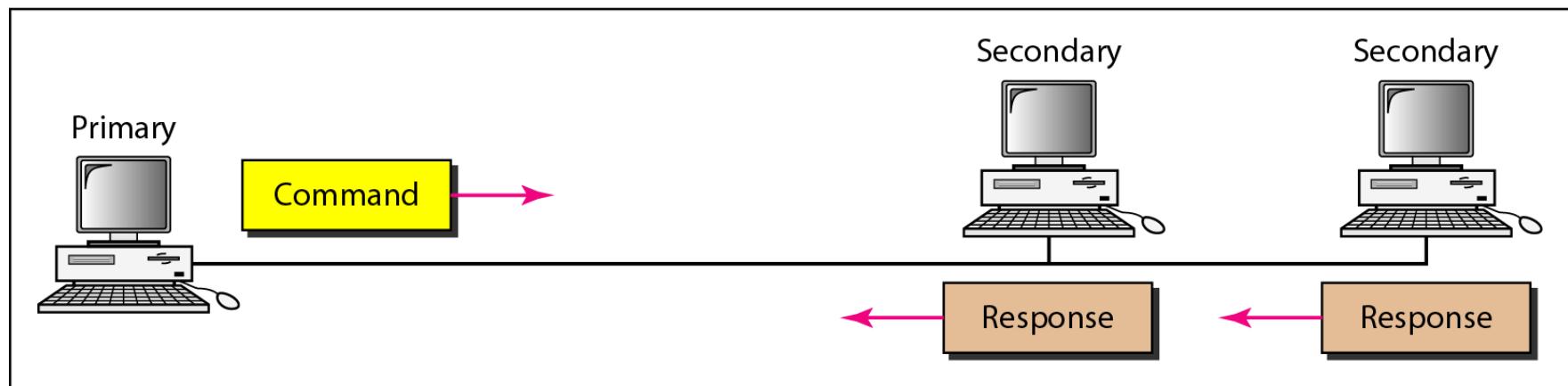
Frames

Control Field

Figure 11.25 *Normal response mode*



a. Point-to-point



b. Multipoint

Figure 11.26 *Asynchronous balanced mode*

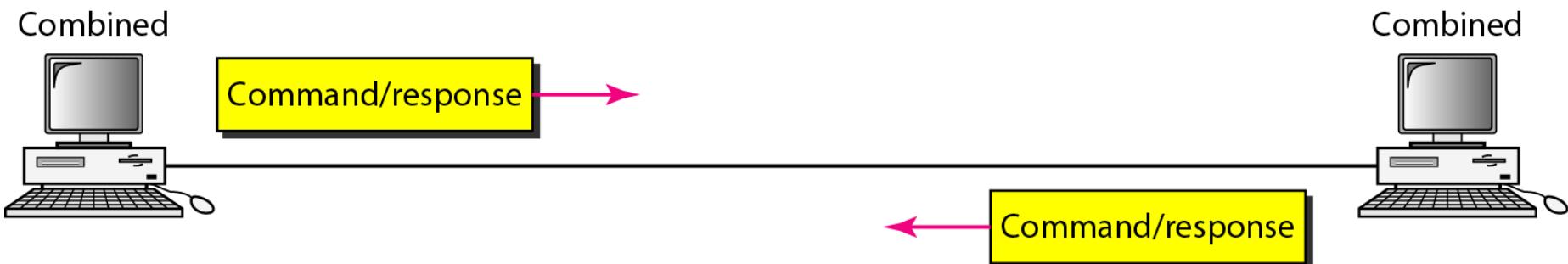


Figure 11.27 HDLC frames

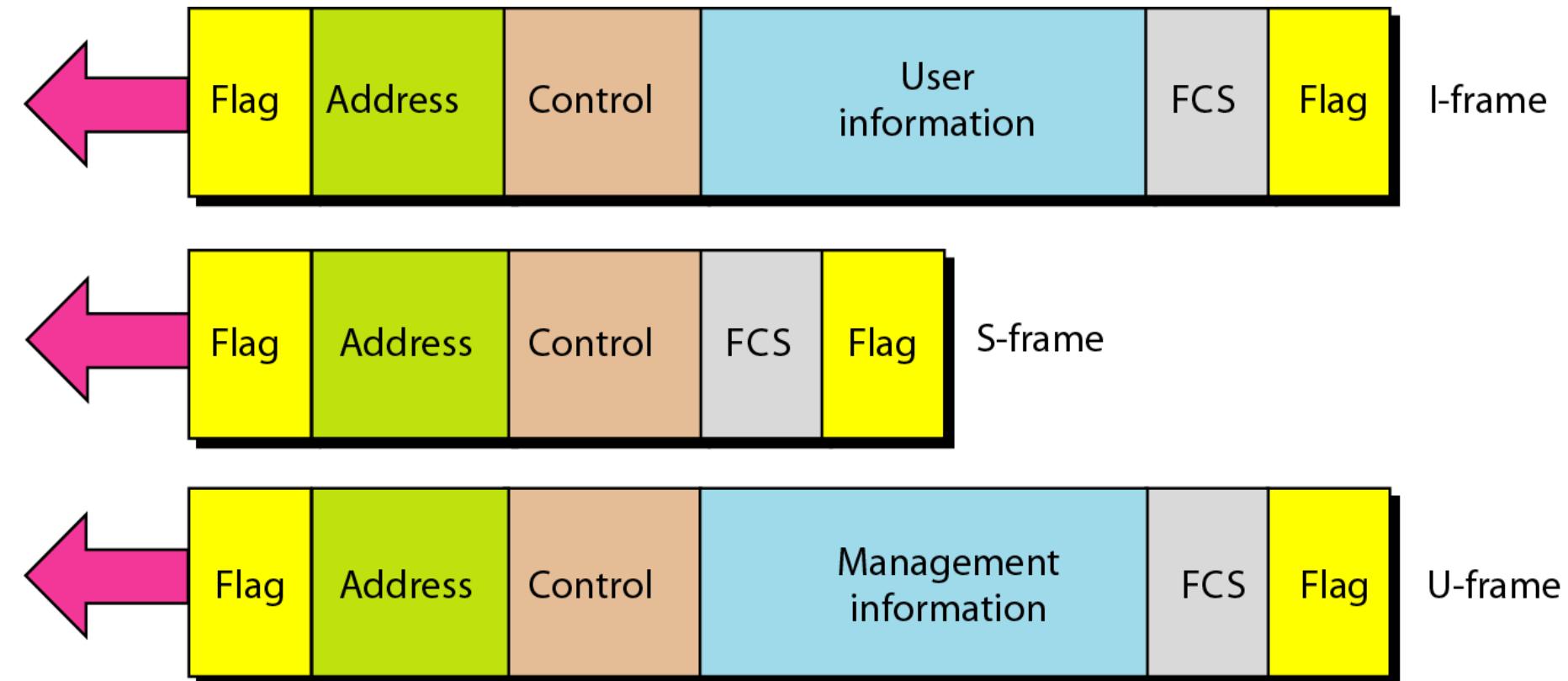


Figure 11.28 Control field format for the different frame types

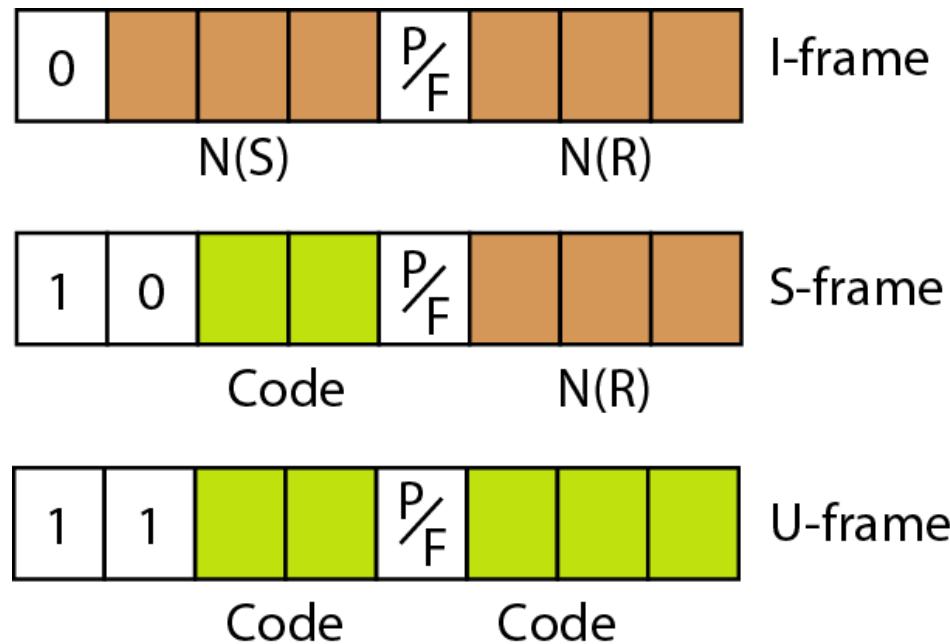


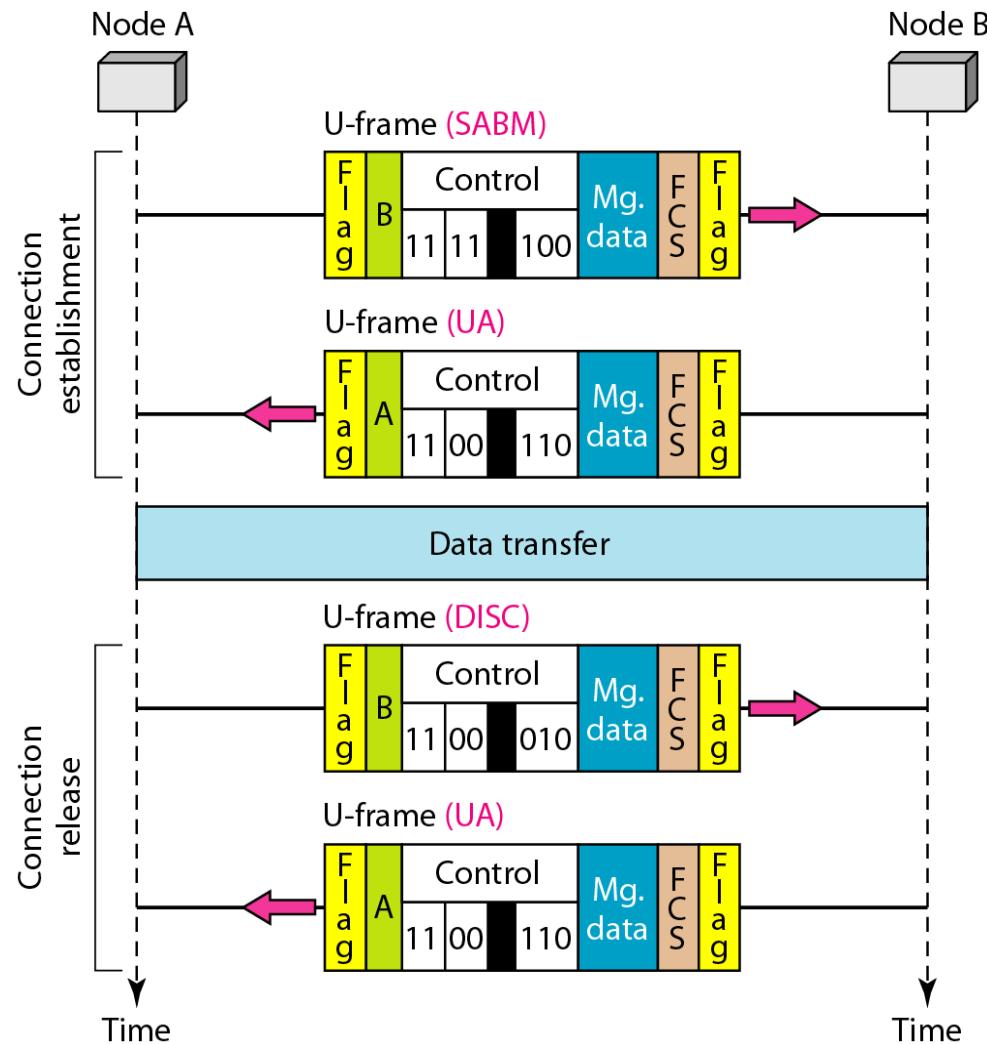
Table 11.1 *U-frame control command and response*

<i>Code</i>	<i>Command</i>	<i>Response</i>	<i>Meaning</i>
00 001	SNRM		Set normal response mode
11 011	SNRME		Set normal response mode, extended
11 100	SABM	DM	Set asynchronous balanced mode or disconnect mode
11 110	SABME		Set asynchronous balanced mode, extended
00 000	UI	UI	Unnumbered information
00 110		UA	Unnumbered acknowledgment
00 010	DISC	RD	Disconnect or request disconnect
10 000	SIM	RIM	Set initialization mode or request information mode
00 100	UP		Unnumbered poll
11 001	RSET		Reset
11 101	XID	XID	Exchange ID
10 001	FRMR	FRMR	Frame reject

Example 11.9

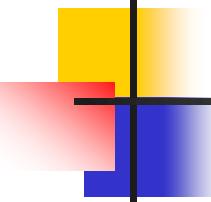
*Figure 11.29 shows how **U-frames** can be used for connection establishment and connection release. Node A asks for a connection with a set asynchronous balanced mode (SABM) frame; node B gives a positive response with an unnumbered acknowledgment (UA) frame. After these two exchanges, data can be transferred between the two nodes (not shown in the figure). After data transfer, node A sends a DISC (disconnect) frame to release the connection; it is confirmed by node B responding with a UA (unnumbered acknowledgment).*

Figure 11.29 Example of connection and disconnection



Example 11.10

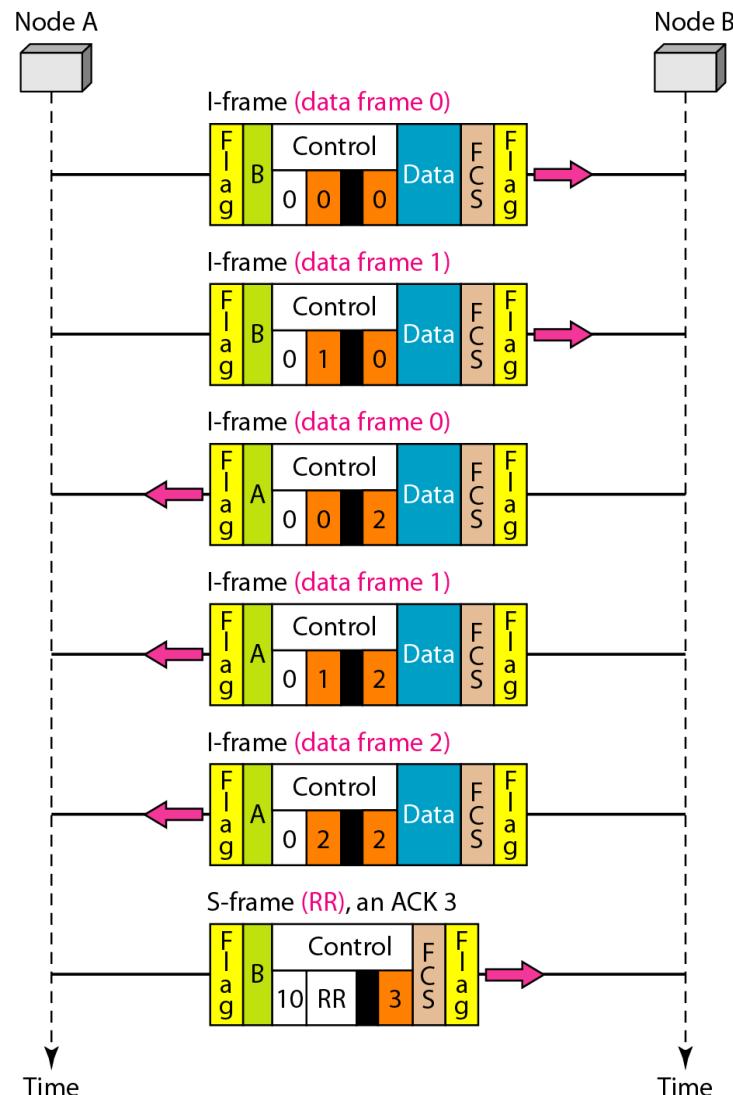
Figure 11.30 shows an exchange using piggybacking. Node A begins the exchange of information with an I-frame numbered 0 followed by another I-frame numbered 1. Node B piggybacks its acknowledgment of both frames onto an I-frame of its own. Node B's first I-frame is also numbered 0 [N(S) field] and contains a 2 in its N(R) field, acknowledging the receipt of A's frames 1 and 0 and indicating that it expects frame 2 to arrive next. Node B transmits its second and third I-frames (numbered 1 and 2) before accepting further frames from node A.



Example 11.10 (continued)

Its $N(R)$ information, therefore, has not changed: B frames 1 and 2 indicate that node B is still expecting A's frame 2 to arrive next. Node A has sent all its data. Therefore, it cannot piggyback an acknowledgment onto an I-frame and sends an S-frame instead. The RR code indicates that A is still ready to receive. The number 3 in the $N(R)$ field tells B that frames 0, 1, and 2 have all been accepted and that A is now expecting frame number 3.

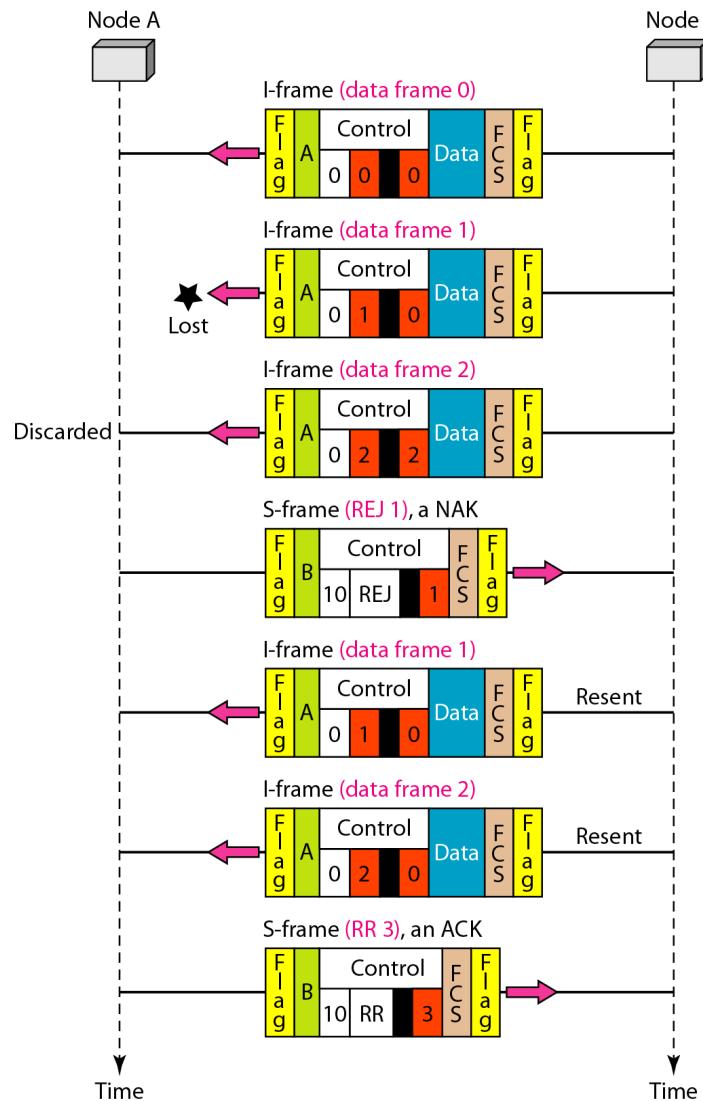
Figure 11.30 Example of piggybacking without error



Example 11.11

Figure 11.31 shows an exchange in which a frame is lost. Node B sends three data frames (0, 1, and 2), but frame 1 is lost. When node A receives frame 2, it discards it and sends a REJ frame for frame 1. Note that the protocol being used is Go-Back-N with the special use of an REJ frame as a NAK frame. The NAK frame does two things here: It confirms the receipt of frame 0 and declares that frame 1 and any following frames must be resent. Node B, after receiving the REJ frame, resends frames 1 and 2. Node A acknowledges the receipt by sending an RR frame (ACK) with acknowledgment number 3.

Figure 11.31 Example of piggybacking with error





**Data Communications
and Networking**

Fourth Edition

Forouzan

Chapter 13

Wired LANs: Ethernet

13-1 IEEE STANDARDS

In 1985, the Computer Society of the IEEE started a project, called Project 802, to set standards to enable intercommunication among equipment from a variety of manufacturers. Project 802 is a way of specifying functions of the physical layer and the data link layer of major LAN protocols.

Topics discussed in this section:

Data Link Layer
Physical Layer

Figure 13.1 IEEE standard for LANs

LLC: Logical link control

MAC: Media access control

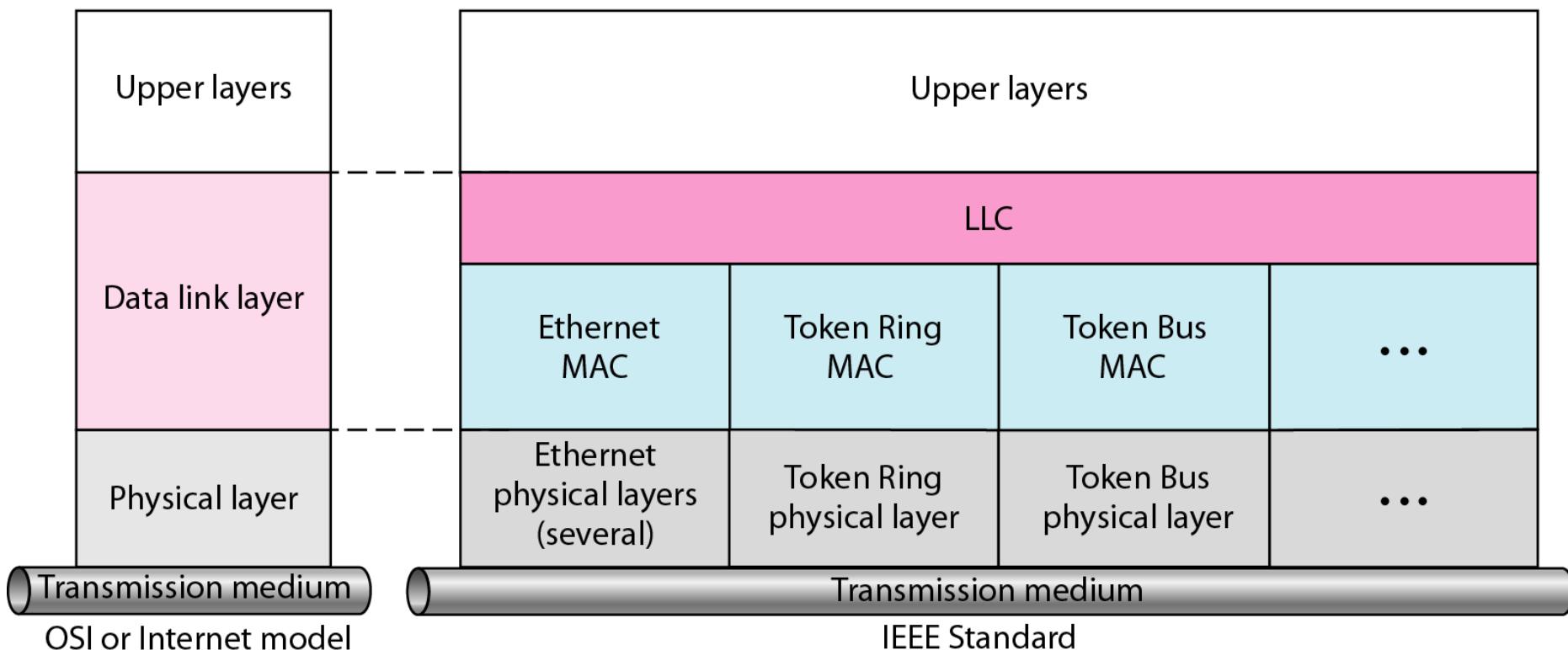
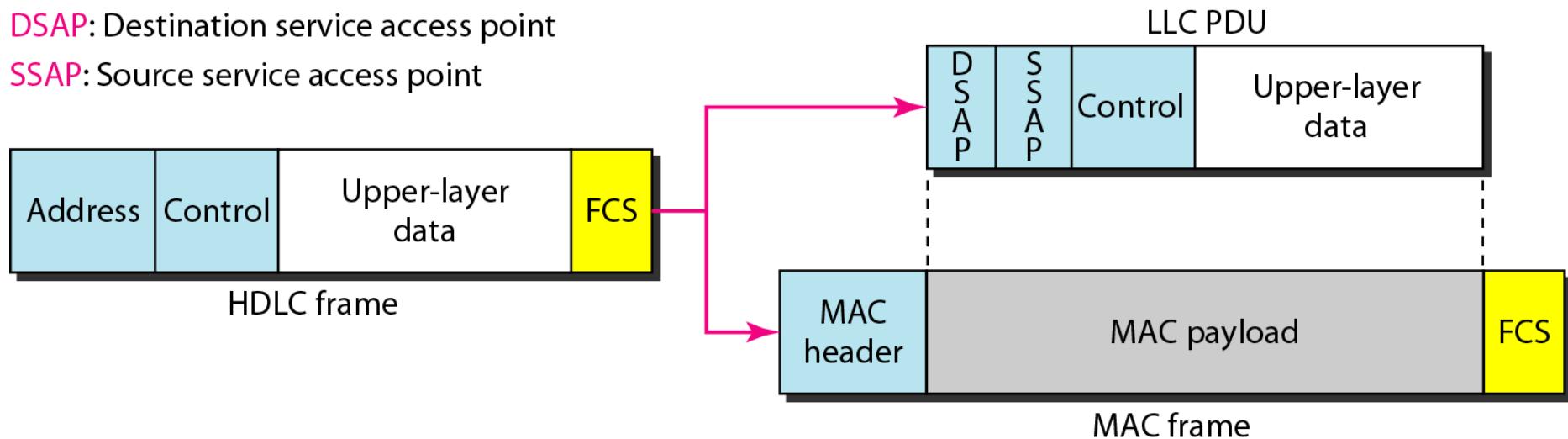


Figure 13.2 HDLC frame compared with LLC and MAC frames

PDU: Protocol Data Unit

DSAP: Destination service access point

SSAP: Source service access point



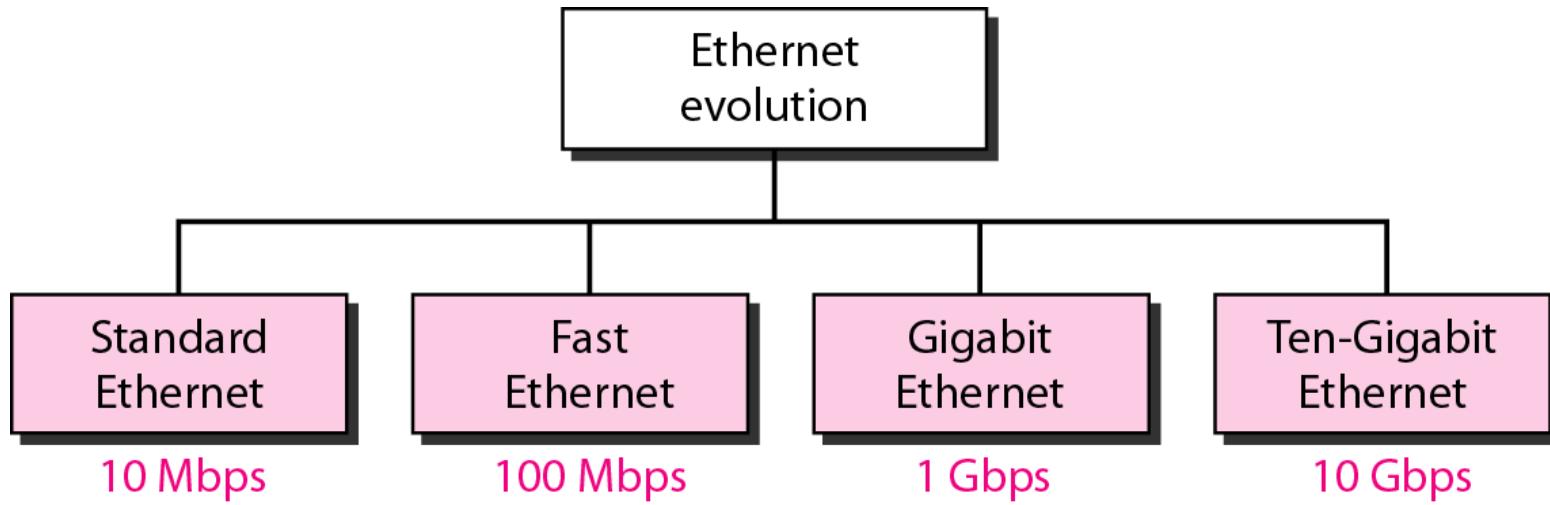
13-2 STANDARD ETHERNET

*The original Ethernet was created in 1976 at Xerox's Palo Alto Research Center (PARC). Since then, it has gone through four generations: Standard Ethernet(10 Mbps), Fast Ethernet(100 Mbps), Gigabit Ethernet(1Gbps) and Ten-Gigabit Ethernet(10 Gbps). We briefly discuss the **Standard (or traditional) Ethernet** in this section.*

Topics discussed in this section:

MAC Sublayer
Physical Layer

Figure 13.3 *Ethernet evolution through four generations*

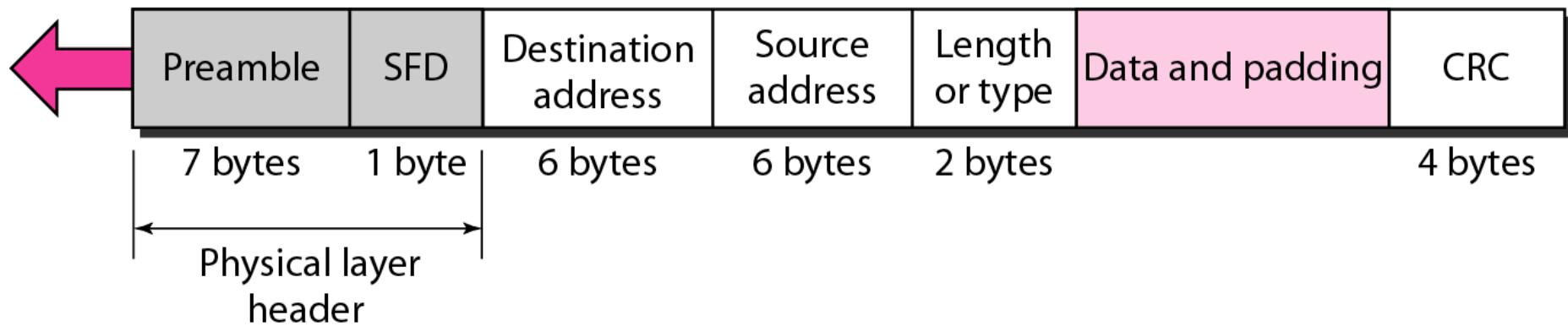


MAC Sublayer – governs the operation of the access method. And also frames data received from the upper layer and passes them to the physical layer.

Figure 13.4 802.3 MAC frame

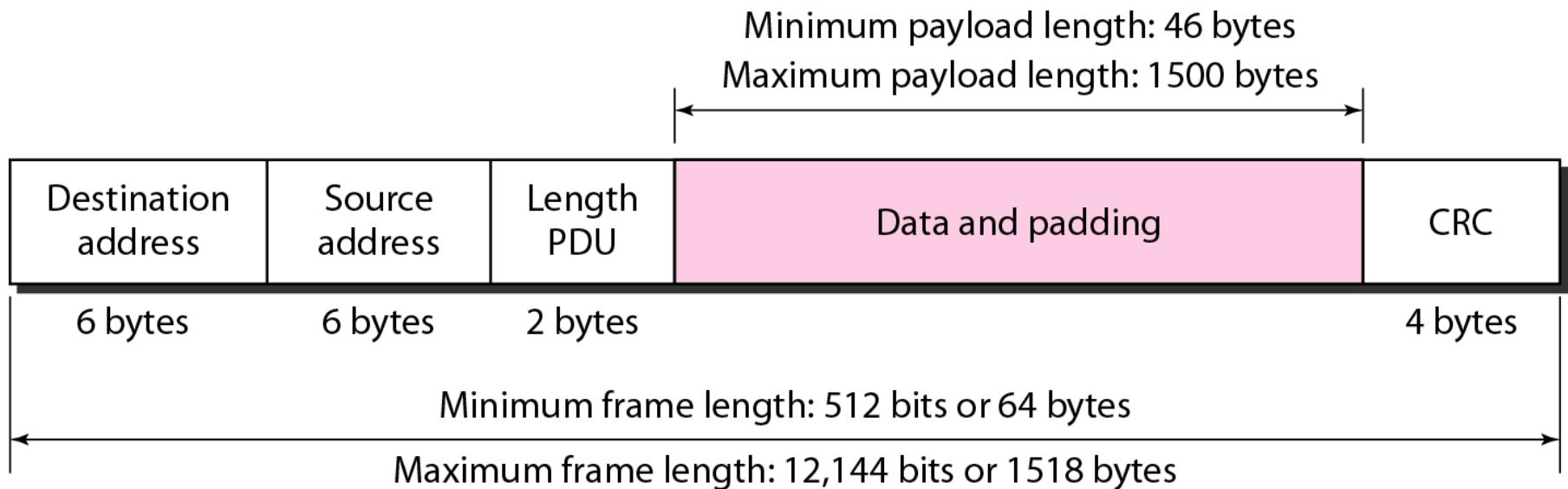
Preamble: 56 bits of alternating 1s and 0s.

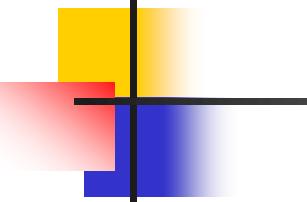
SFD: Start frame delimiter, flag (10101011)



Data field : 46-1500 bytes

Figure 13.5 *Minimum and maximum lengths of a Frame*





Note

Frame length:

Minimum: 64 bytes (512 bits)

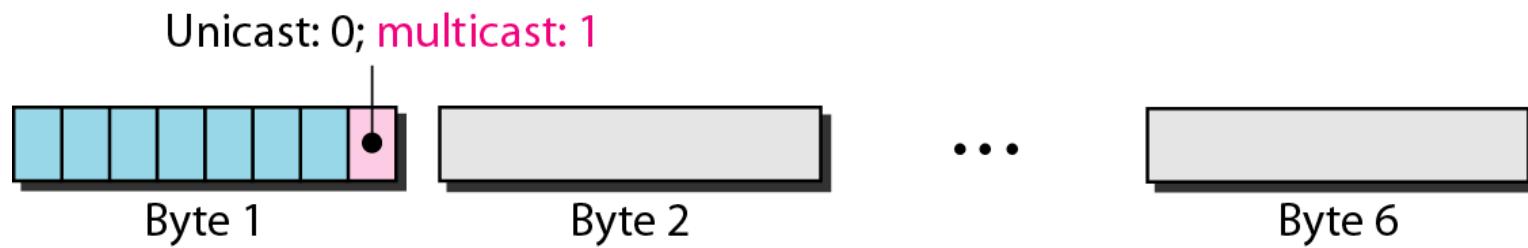
Maximum: 1518 bytes (12,144 bits)

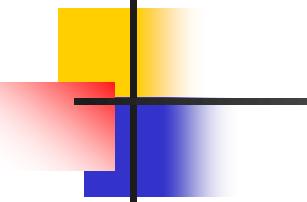
Figure 13.6 *Example of an Ethernet address in hexadecimal notation*

06 : 01 : 02 : 01 : 2C : 4B

6 bytes = 12 hex digits = 48 bits

Figure 13.7 *Unicast and multicast addresses*

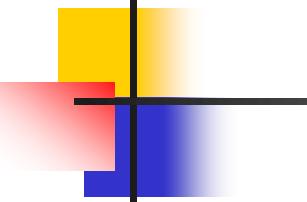




Note

The least significant bit of the first byte defines the type of address.

**If the bit is 0, the address is unicast;
otherwise, it is multicast.**



Note

The broadcast destination address is a special case of the multicast address in which all bits are 1s.

Example 13.1

Define the type of the following destination addresses:

- a.* **4A:30:10:21:10:1A**
- b.* **47:20:1B:2E:08:EE**
- c.* **FF:FF:FF:FF:FF:FF**

Solution

To find the type of the address, we need to look at the second hexadecimal digit from the left. If it is even, the address is unicast. If it is odd, the address is multicast. If all digits are F's, the address is broadcast. Therefore, we have the following:

- a.* **This is a unicast address because A in binary is 1010.**
- b.* **This is a multicast address because 7 in binary is 0111.**
- c.* **This is a broadcast address because all digits are F's.**

Example 13.2

Show how the address 47:20:1B:2E:08:EE is sent out on line.

Solution

The address is sent left-to-right, byte by byte; for each byte, it is sent right-to-left, bit by bit, as shown below:

← 11100010 00000100 11011000 01110100 00010000 01110111

Figure 13.8 *Categories of Standard Ethernet*

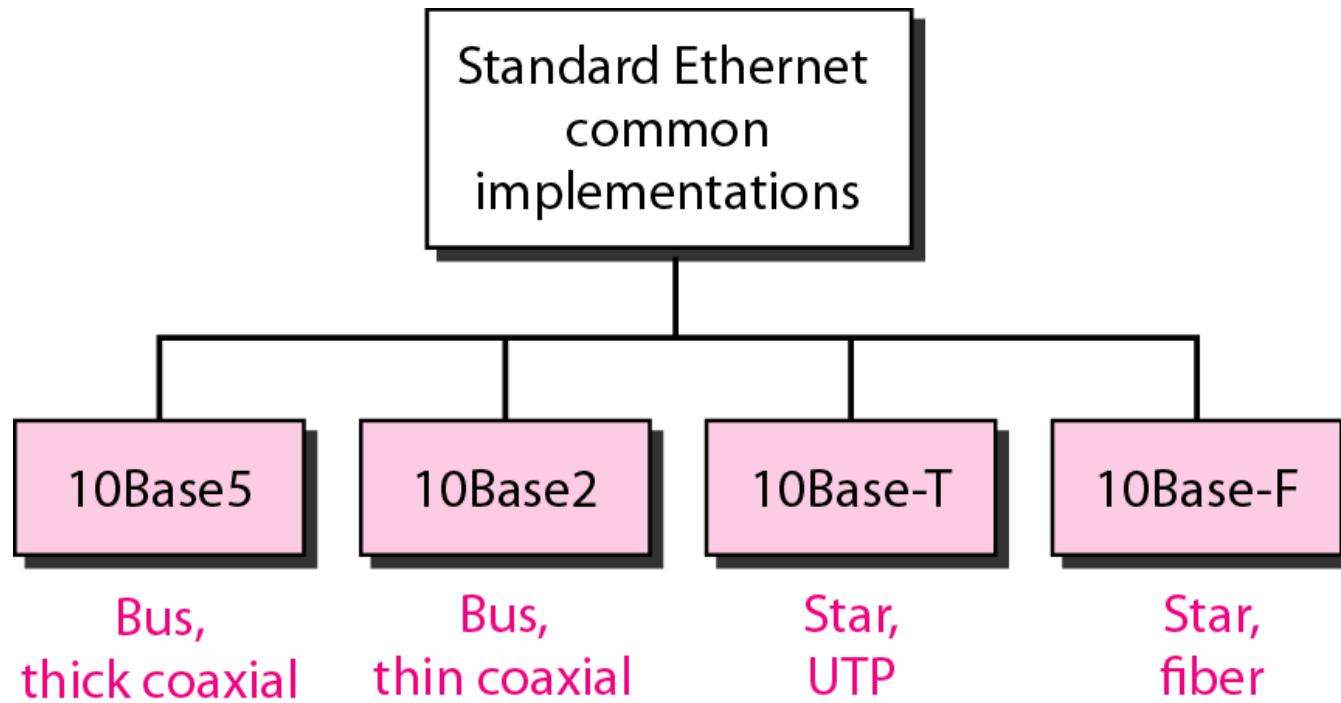


Figure 13.9 *Encoding in a Standard Ethernet implementation*

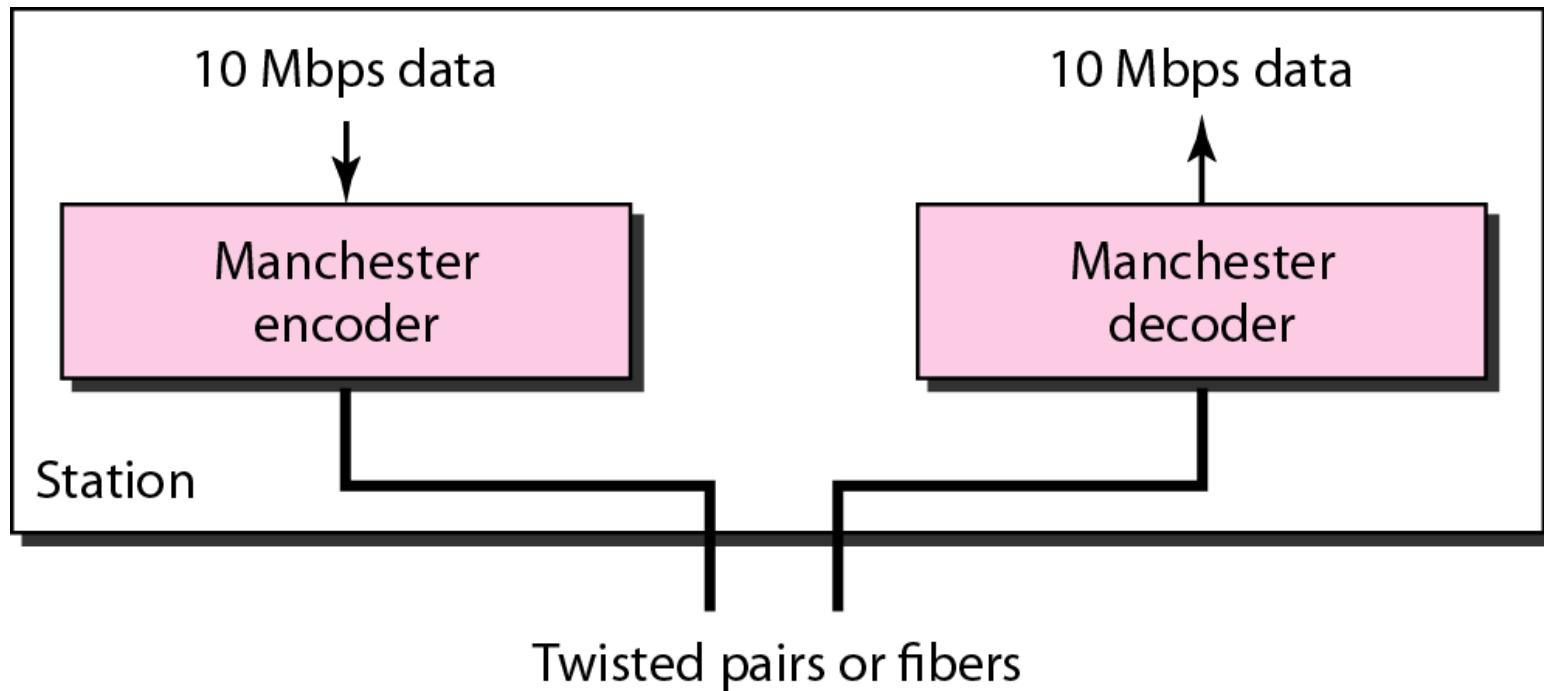


Figure 13.10 10Base5 implementation

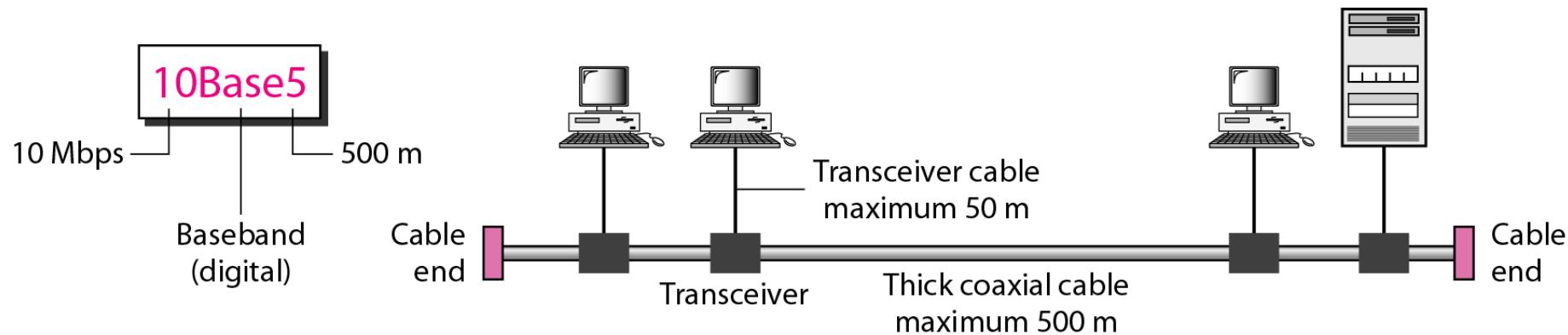


Figure 13.11 *10Base2 implementation*

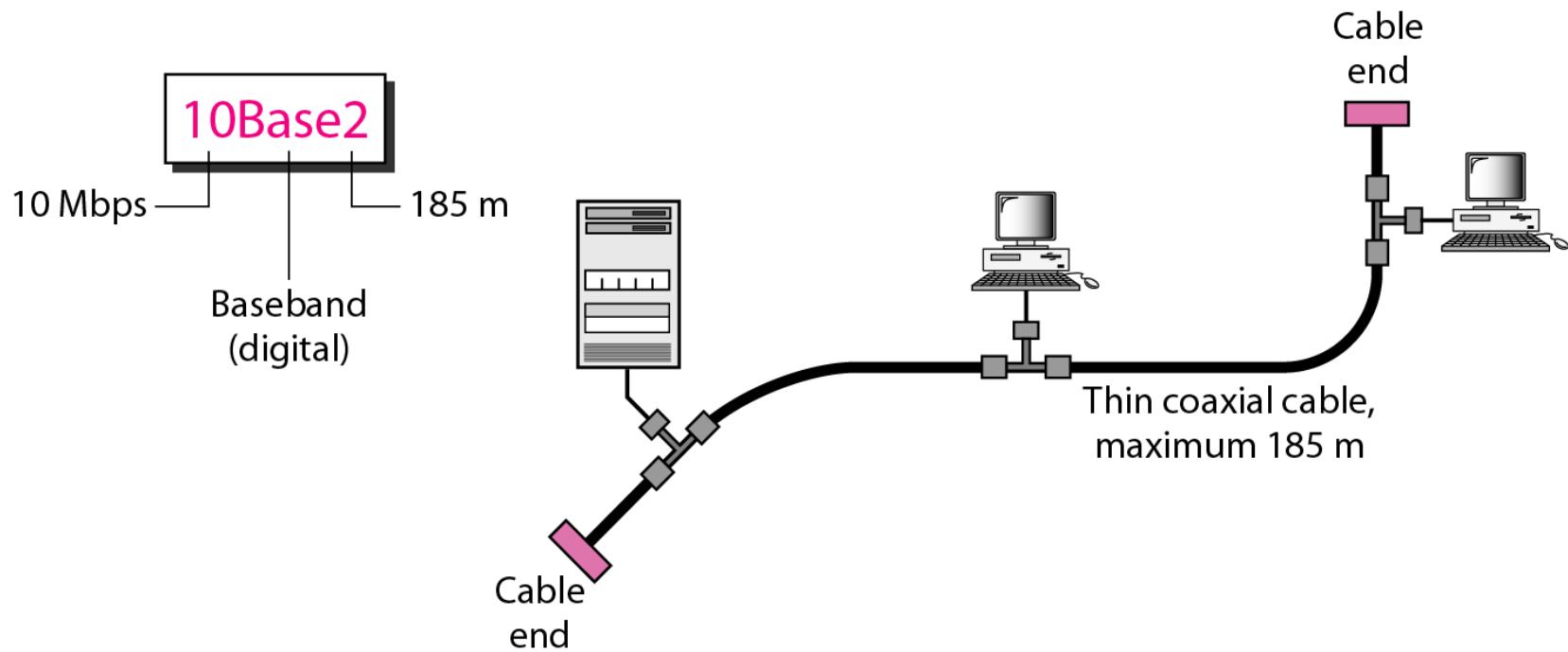


Figure 13.12 *10Base-T implementation*

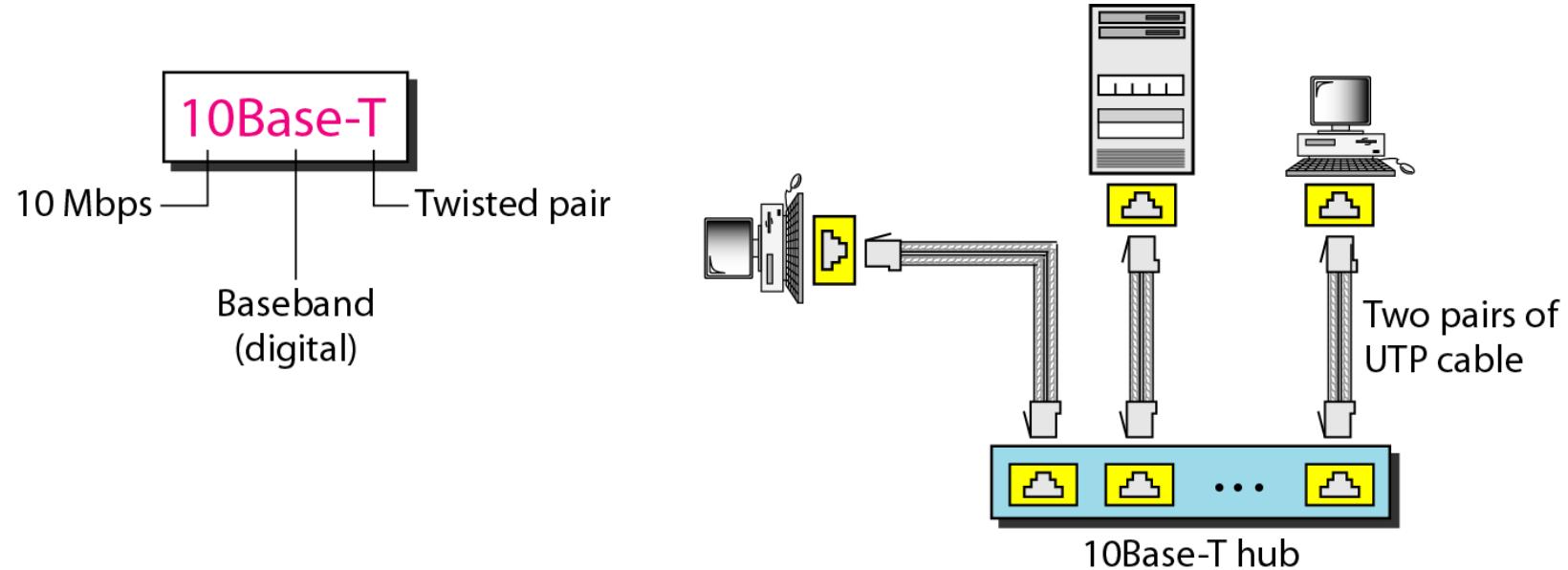


Figure 13.13 *10Base-F implementation*

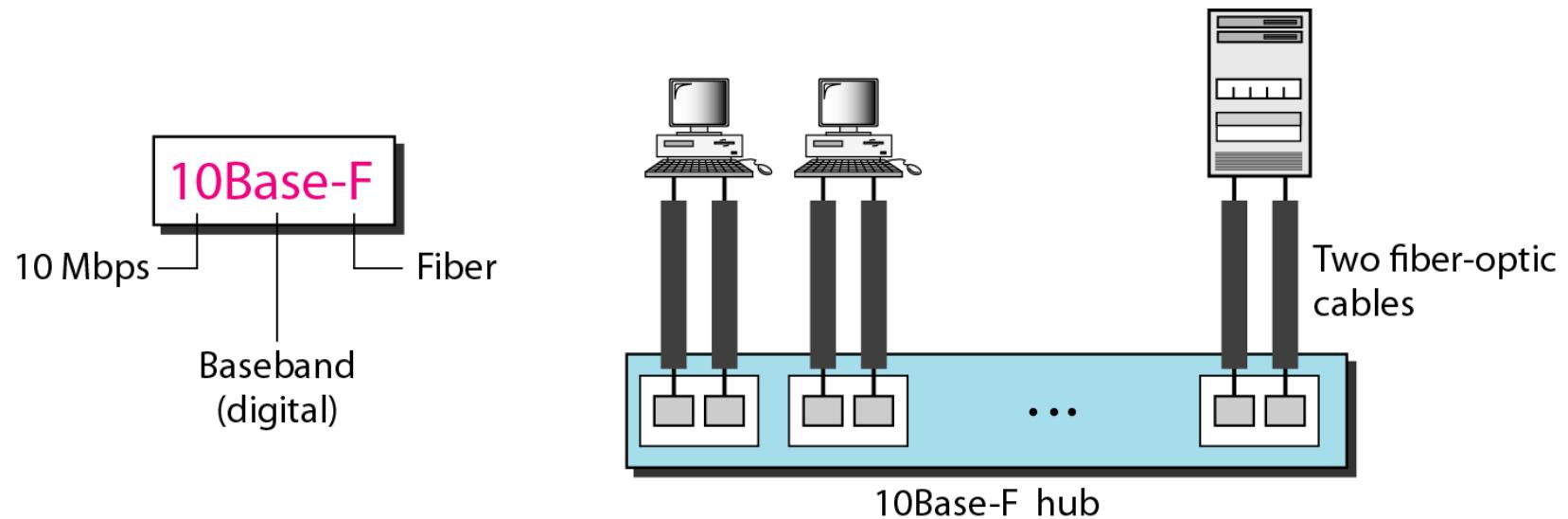


Table 13.1 *Summary of Standard Ethernet implementations*

<i>Characteristics</i>	<i>10Base5</i>	<i>10Base2</i>	<i>10Base-T</i>	<i>10Base-F</i>
Media	Thick coaxial cable	Thin coaxial cable	2 UTP	2 Fiber
Maximum length	500 m	185 m	100 m	2000 m
Line encoding	Manchester	Manchester	Manchester	Manchester

13-3 CHANGES IN THE STANDARD

The 10-Mbps Standard Ethernet has gone through several changes before moving to the higher data rates. These changes actually opened the road to the evolution of the Ethernet to become compatible with other high-data-rate LANs.

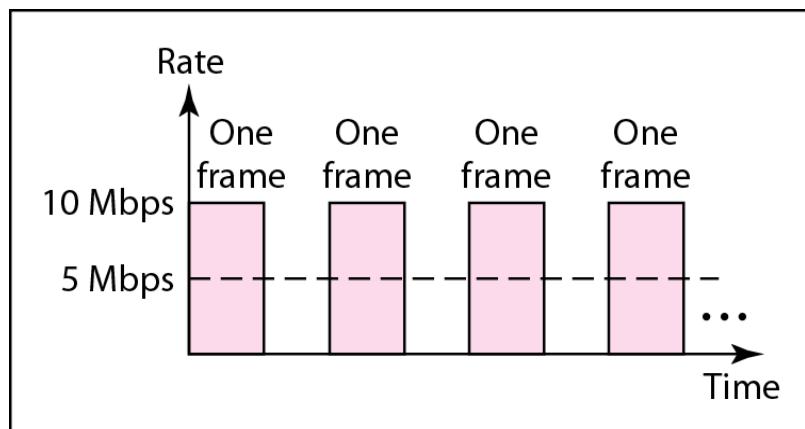
Topics discussed in this section:

Bridged Ethernet

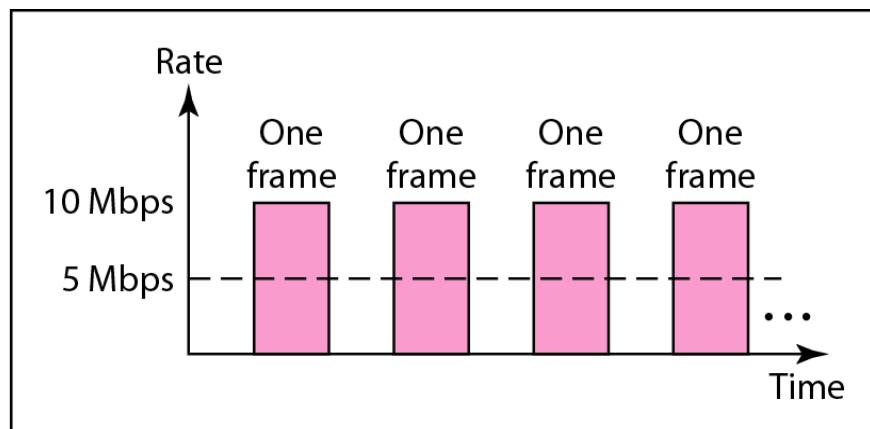
Switched Ethernet

Full-Duplex Ethernet

Figure 13.14 *Sharing bandwidth*



a. First station

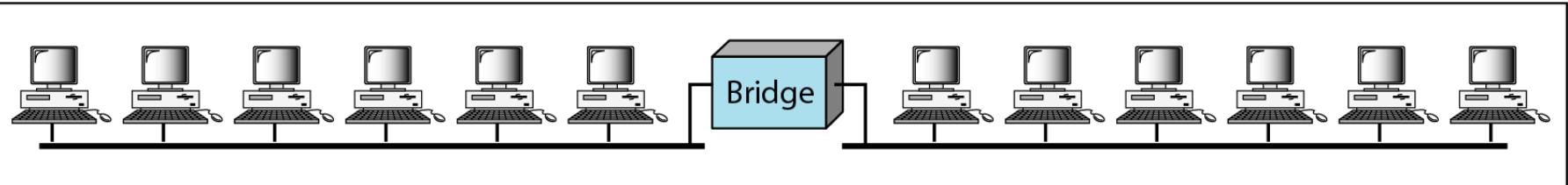


b. Second station

Figure 13.15 *A network with and without a bridge*



a. Without bridging



b. With bridging

Figure 13.16 Collision domains in an unbridged network and a bridged network

Domain



a. Without bridging

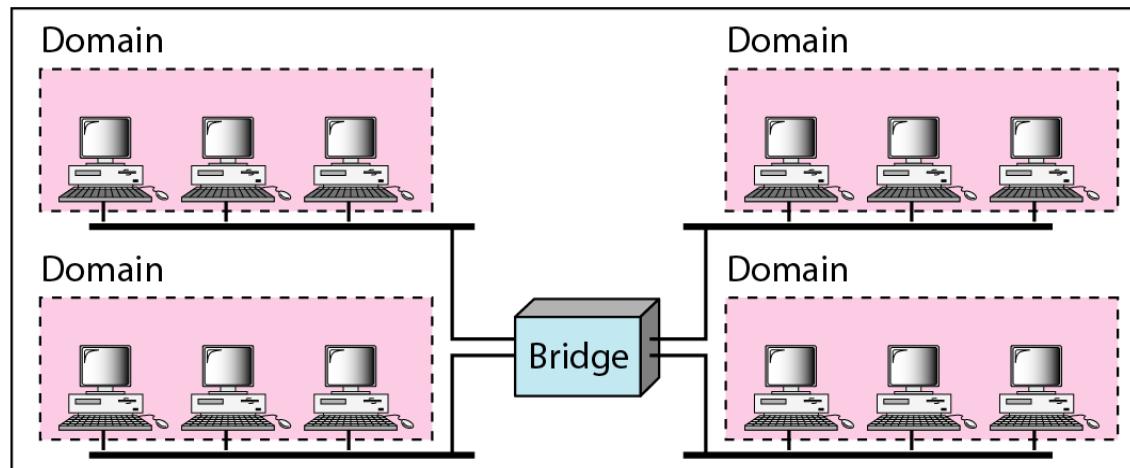
Domain

Domain

Domain

Domain

Bridge



b. With bridging

Figure 13.17 *Switched Ethernet*

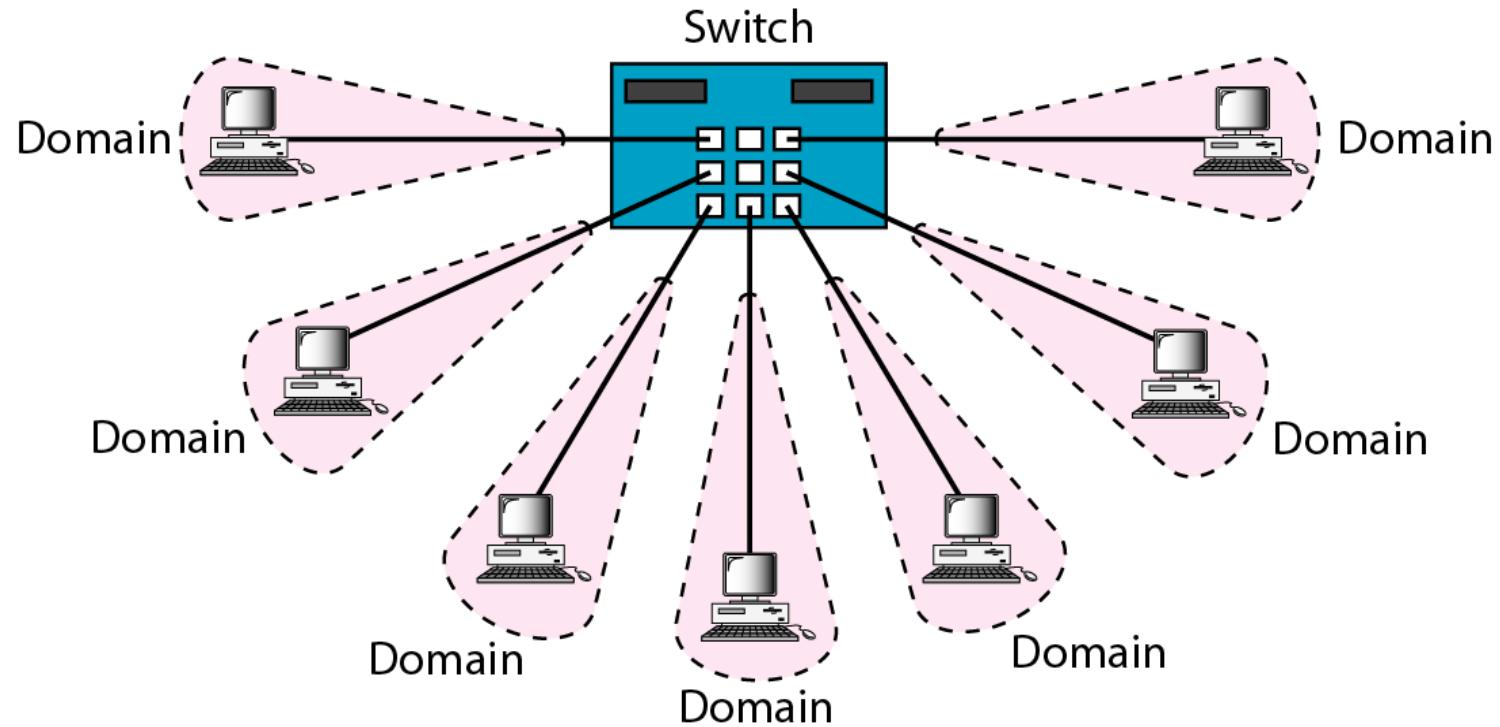
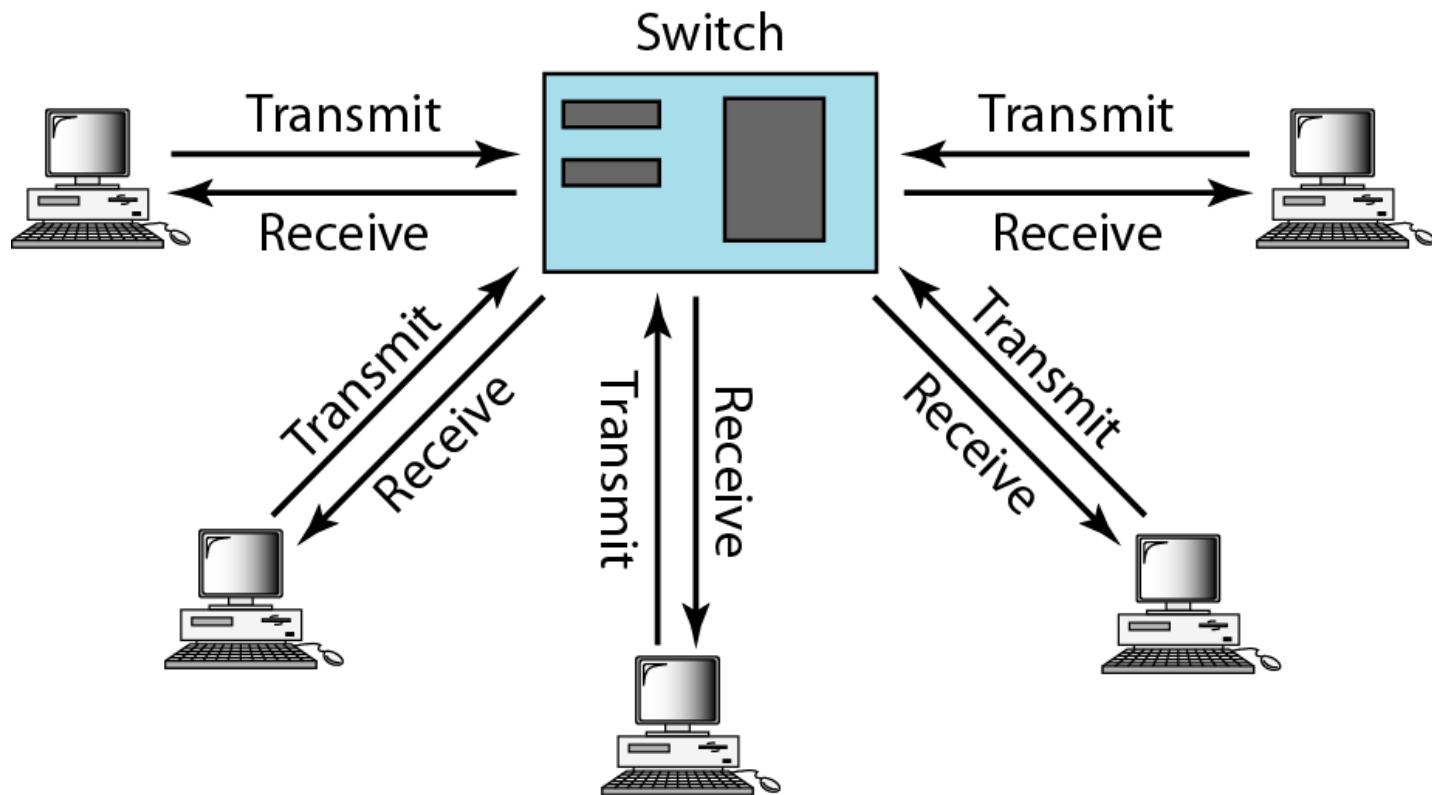


Figure 13.18 Full-duplex switched Ethernet



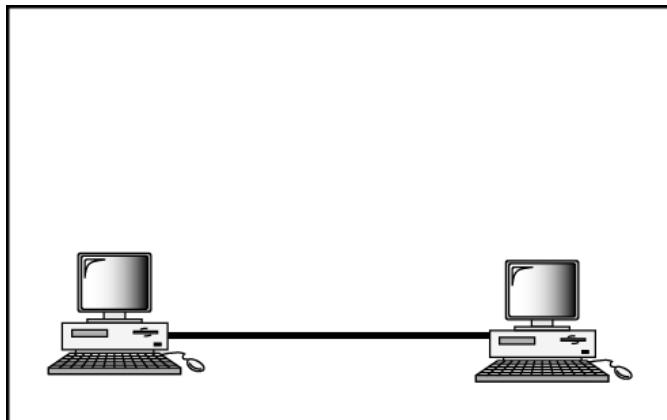
13-4 FAST ETHERNET

Fast Ethernet was designed to compete with LAN protocols such as FDDI(Fiber Distributed Data Interface) or Fiber Channel. IEEE created Fast Ethernet under the name 802.3u. Fast Ethernet is backward-compatible with Standard Ethernet, but it can transmit data 10 times faster at a rate of 100 Mbps.

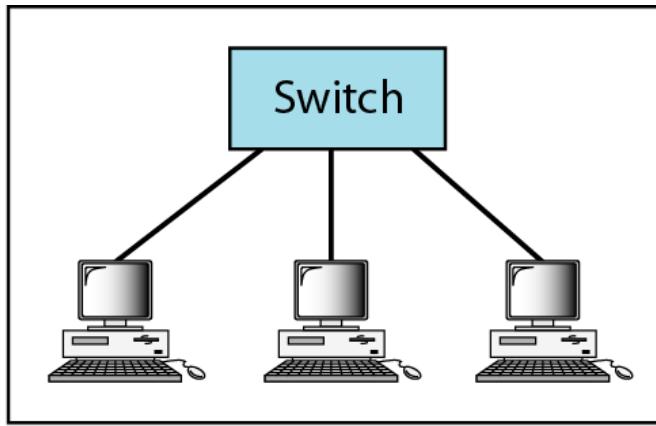
Topics discussed in this section:

MAC Sublayer
Physical Layer

Figure 13.19 *Fast Ethernet topology*



a. Point-to-point



b. Star

Figure 13.20 *Fast Ethernet implementations*

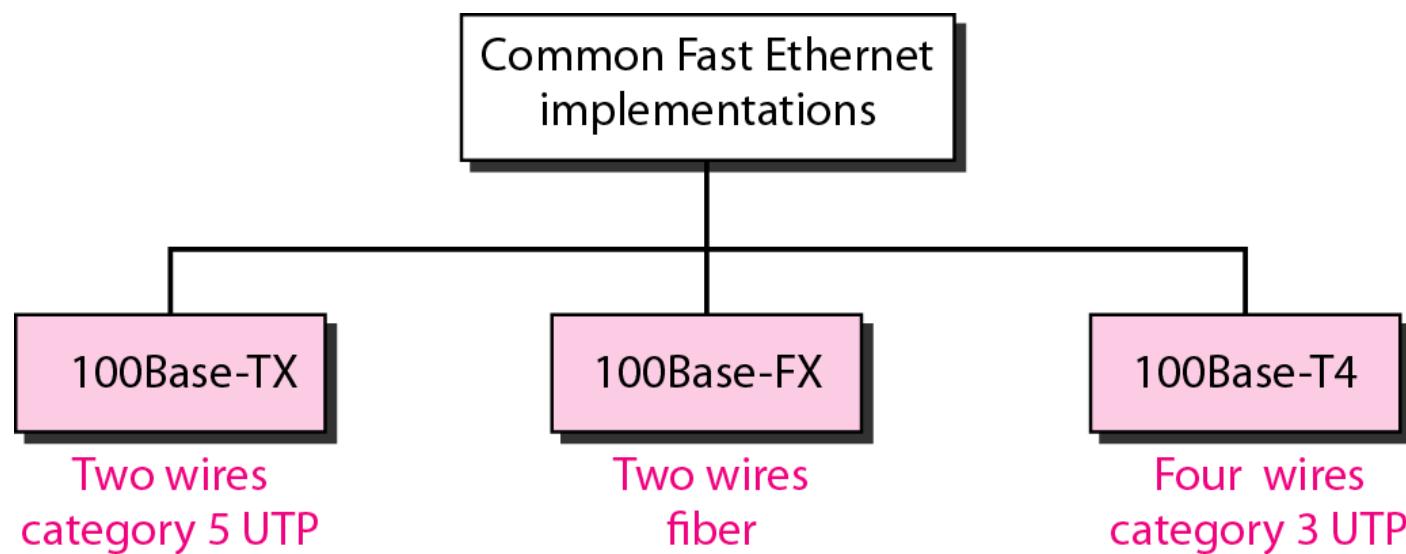


Figure 13.21 Encoding for Fast Ethernet implementation

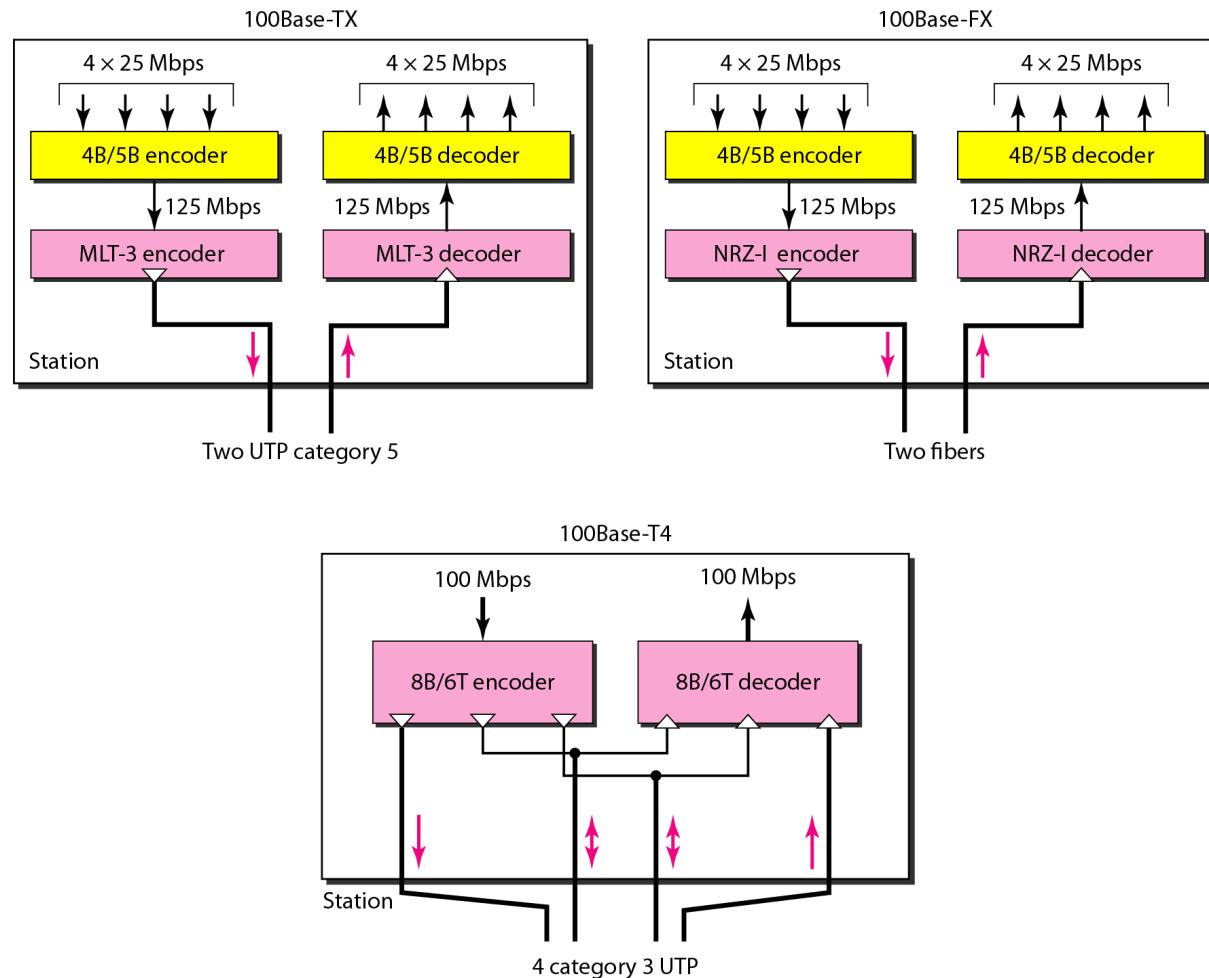


Table 13.2 *Summary of Fast Ethernet implementations*

<i>Characteristics</i>	<i>100Base-TX</i>	<i>100Base-FX</i>	<i>100Base-T4</i>
Media	Cat 5 UTP or STP	Fiber	Cat 4 UTP
Number of wires	2	2	4
Maximum length	100 m	100 m	100 m
Block encoding	4B/5B	4B/5B	
Line encoding	MLT-3	NRZ-I	8B/6T

13-5 GIGABIT ETHERNET

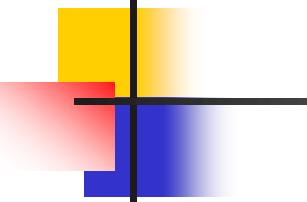
The need for an even higher data rate resulted in the design of the Gigabit Ethernet protocol (1000 Mbps). The IEEE committee calls the standard 802.3z.

Topics discussed in this section:

MAC Sublayer

Physical Layer

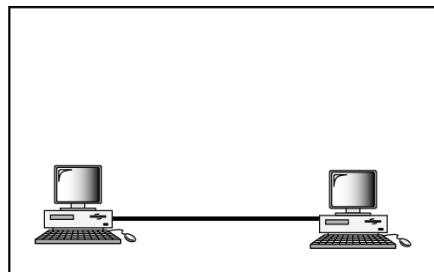
Ten-Gigabit Ethernet



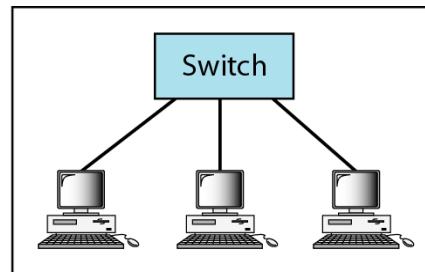
Note

In the full-duplex mode of Gigabit Ethernet, there is no collision; the maximum length of the cable is determined by the signal attenuation in the cable.

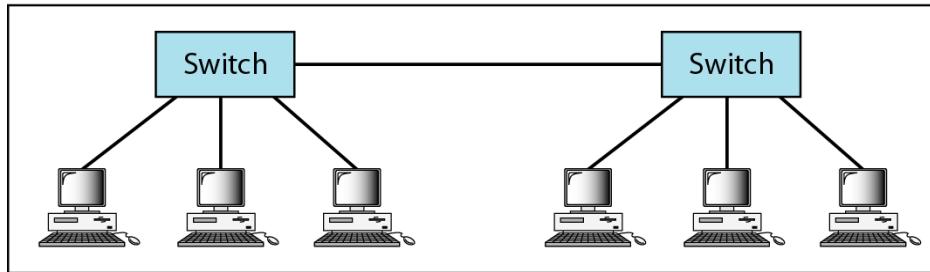
Figure 13.22 *Topologies of Gigabit Ethernet*



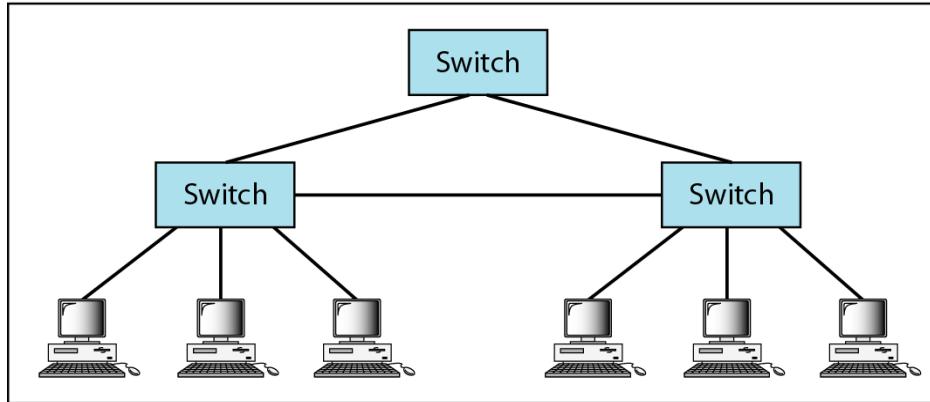
a. Point-to-point



b. Star



c. Two stars



d. Hierarchy of stars

Figure 13.23 *Gigabit Ethernet implementations*

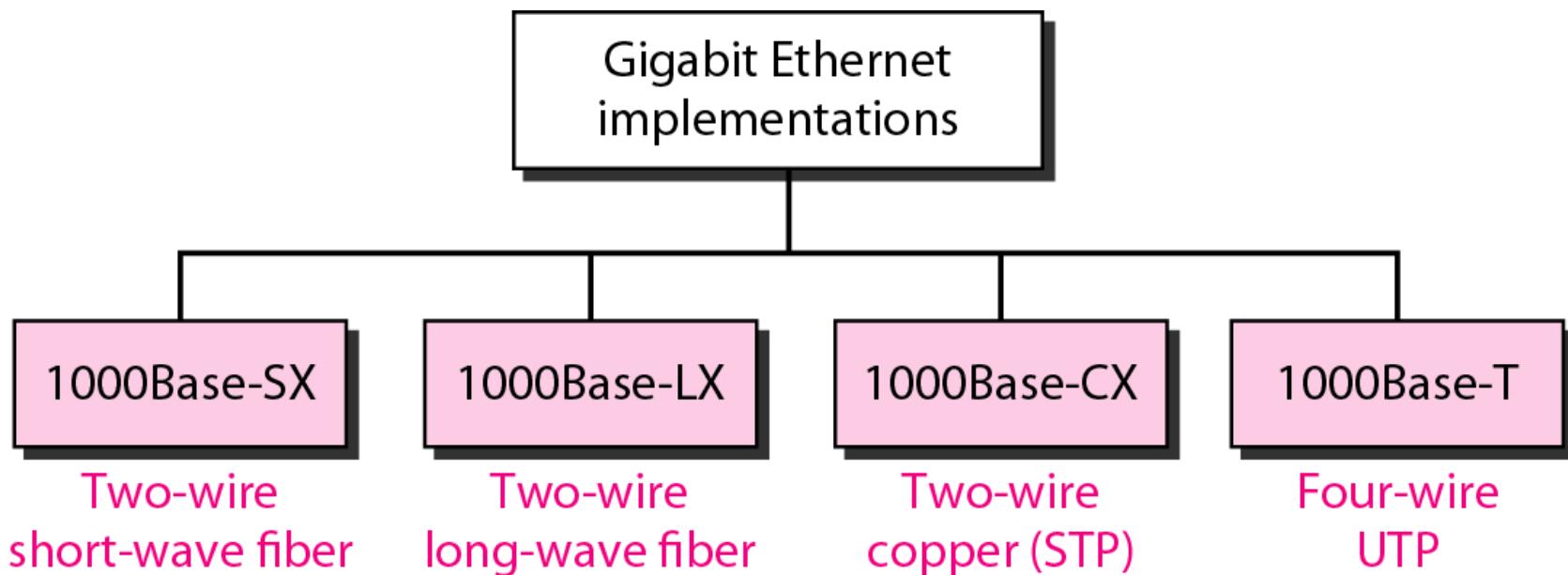


Figure 13.24 Encoding in Gigabit Ethernet implementations

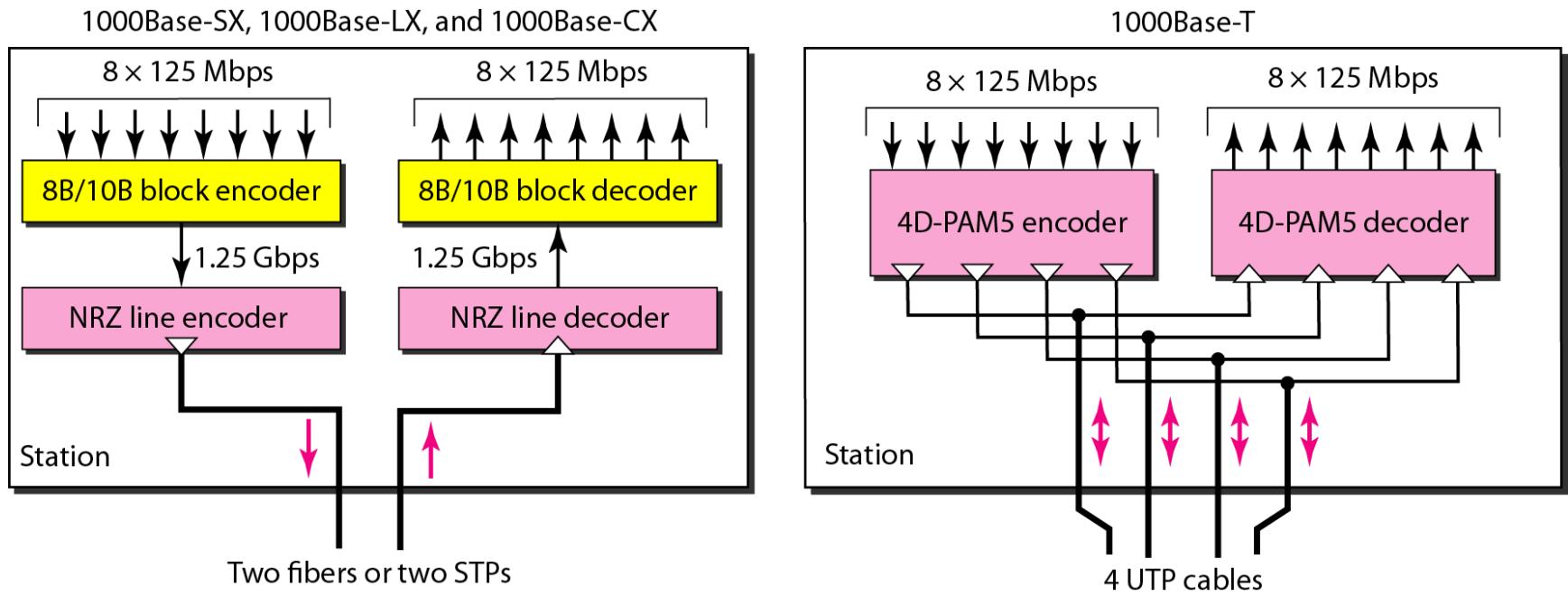


Table 13.3 *Summary of Gigabit Ethernet implementations*

<i>Characteristics</i>	<i>1000Base-SX</i>	<i>1000Base-LX</i>	<i>1000Base-CX</i>	<i>1000Base-T</i>
Media	Fiber short-wave	Fiber long-wave	STP	Cat 5 UTP
Number of wires	2	2	2	4
Maximum length	550 m	5000 m	25 m	100 m
Block encoding	8B/10B	8B/10B	8B/10B	
Line encoding	NRZ	NRZ	NRZ	4D-PAM5

Table 13.4 *Summary of Ten-Gigabit Ethernet implementations*

<i>Characteristics</i>	<i>10GBase-S</i>	<i>10GBase-L</i>	<i>10GBase-E</i>
Media	Short-wave 850-nm multimode	Long-wave 1310-nm single mode	Extended 1550-mm single mode
Maximum length	300 m	10 km	40 km

Chapter 15

Connecting LANs, Backbone Networks, and Virtual LANs

15-1 CONNECTING DEVICES

In this section, we divide connecting devices into five different categories based on the layer in which they operate in a network.

Topics discussed in this section:

Passive Hubs

Active Hubs

Bridges

Two-Layer Switches

Routers

Three-Layer Switches

Gateways

Figure 15.1 *Five categories of connecting devices*

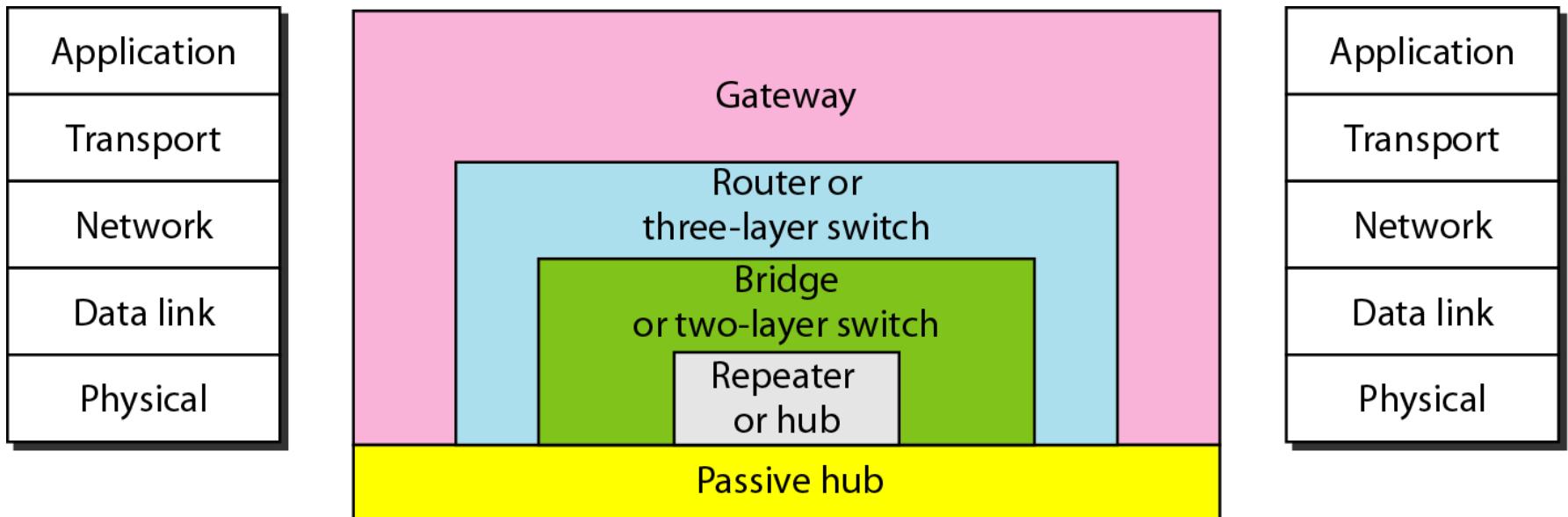
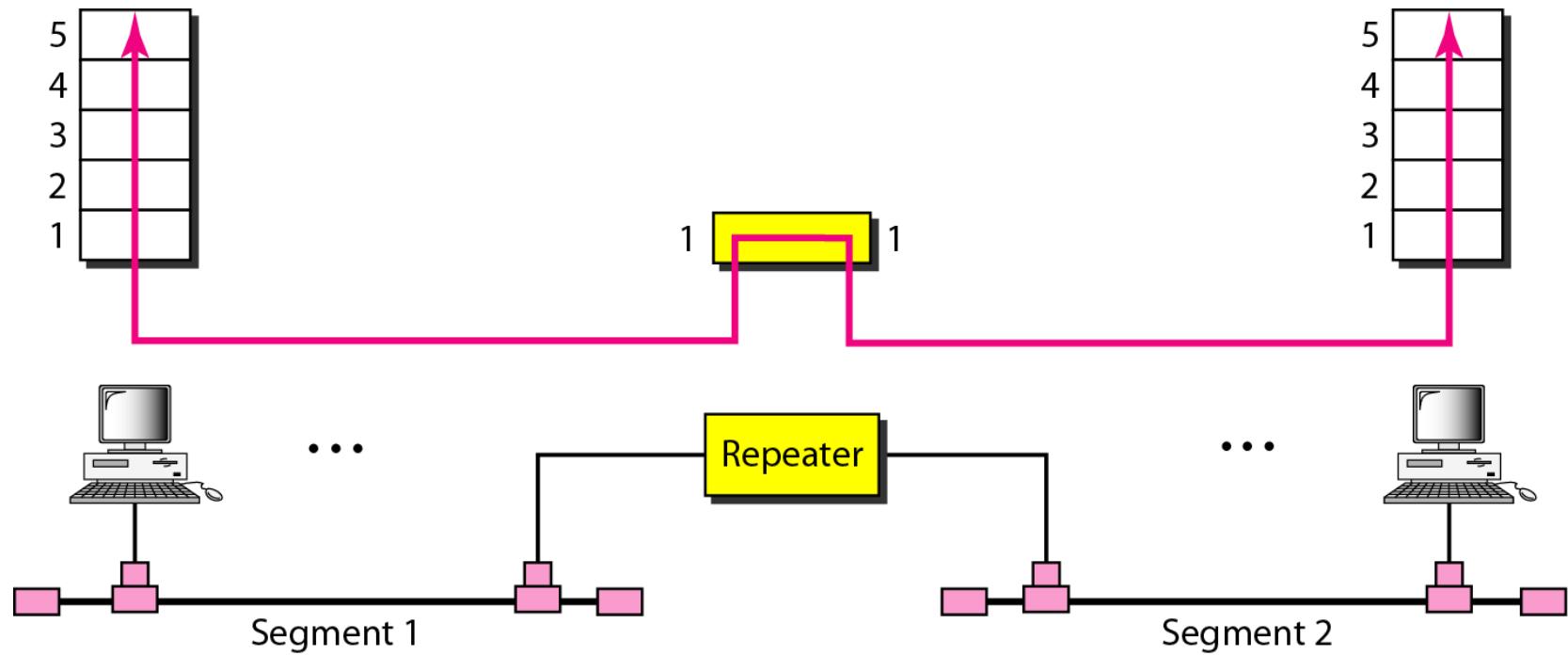
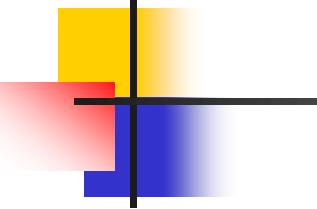


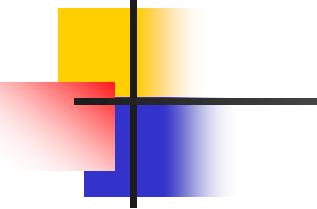
Figure 15.2 A repeater connecting two segments of a LAN





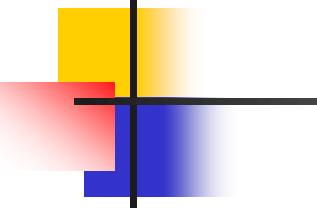
Note

A repeater connects segments of a LAN.



Note

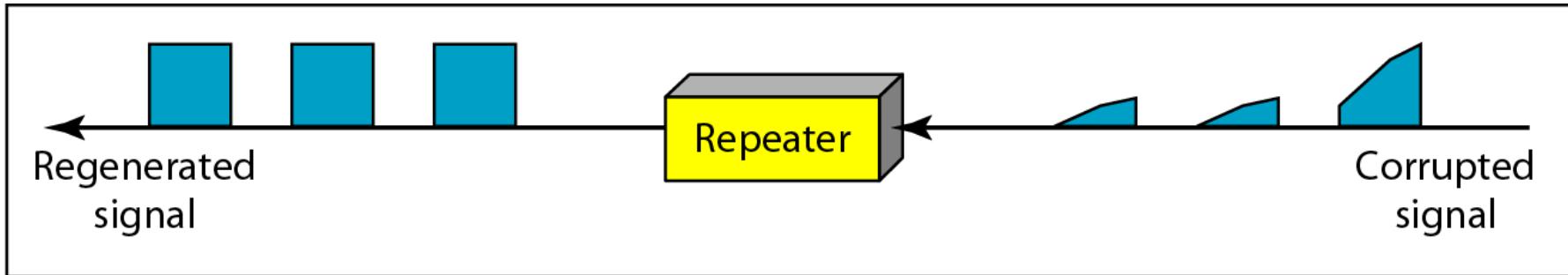
**A repeater forwards every frame;
it has no filtering capability.**



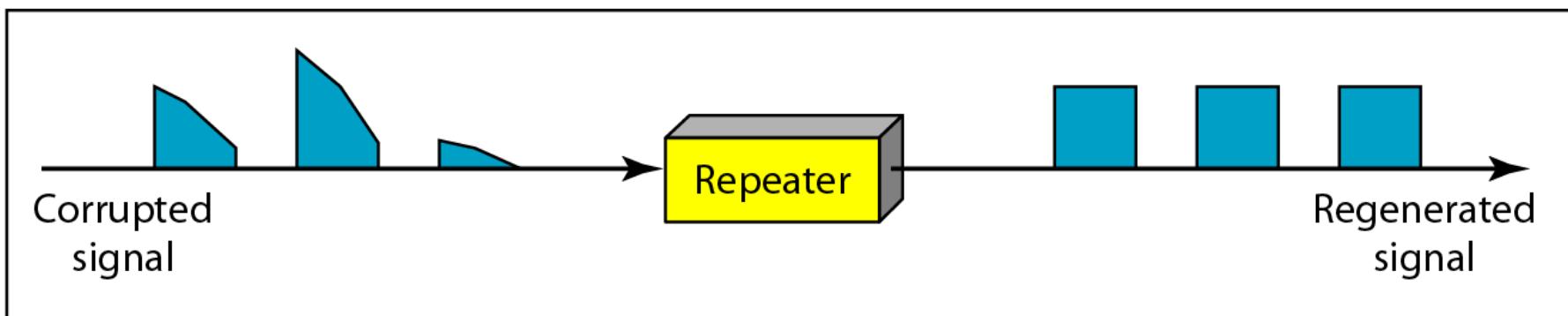
Note

**A repeater is a regenerator,
not an amplifier.**

Figure 15.3 Function of a repeater

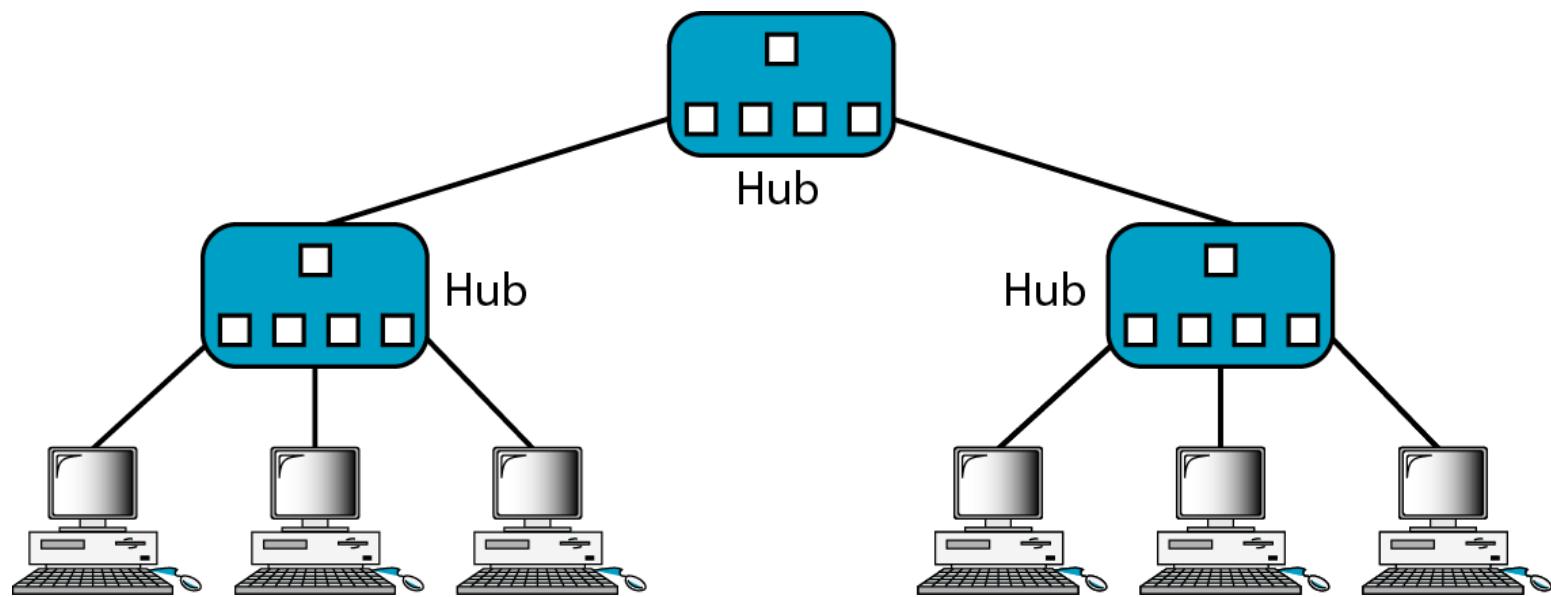


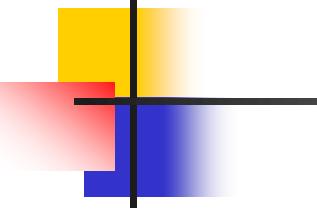
a. Right-to-left transmission.



b. Left-to-right transmission.

Figure 15.4 *A hierarchy of hubs*

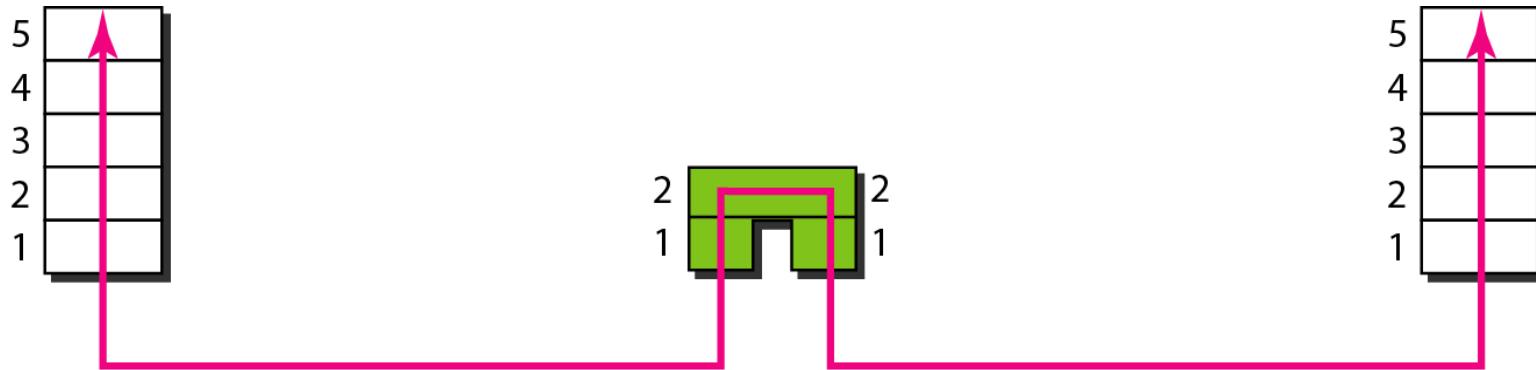




Note

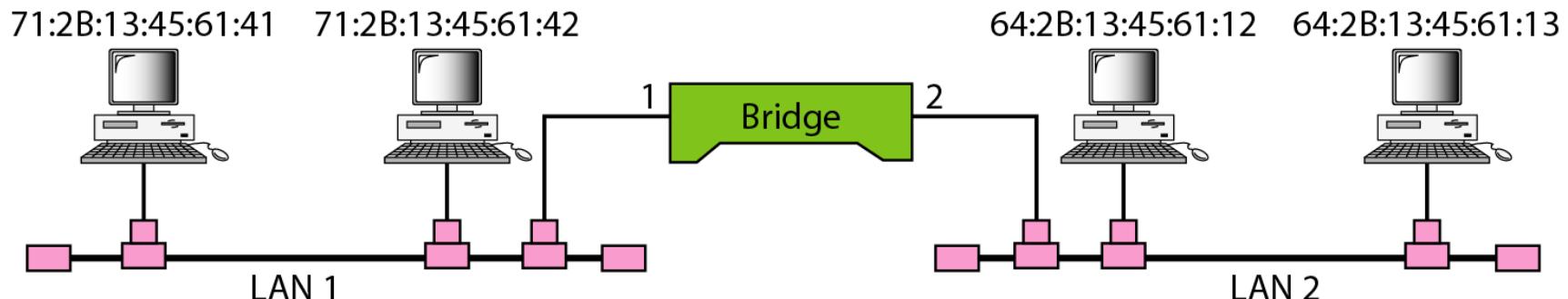
A bridge has a table used in filtering decisions.

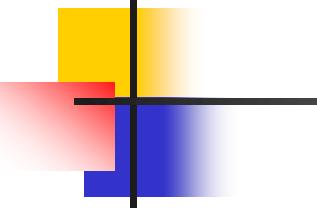
Figure 15.5 A bridge connecting two LANs



Address	Port
71:2B:13:45:61:41	1
71:2B:13:45:61:42	1
64:2B:13:45:61:12	2
64:2B:13:45:61:13	2

Bridge Table

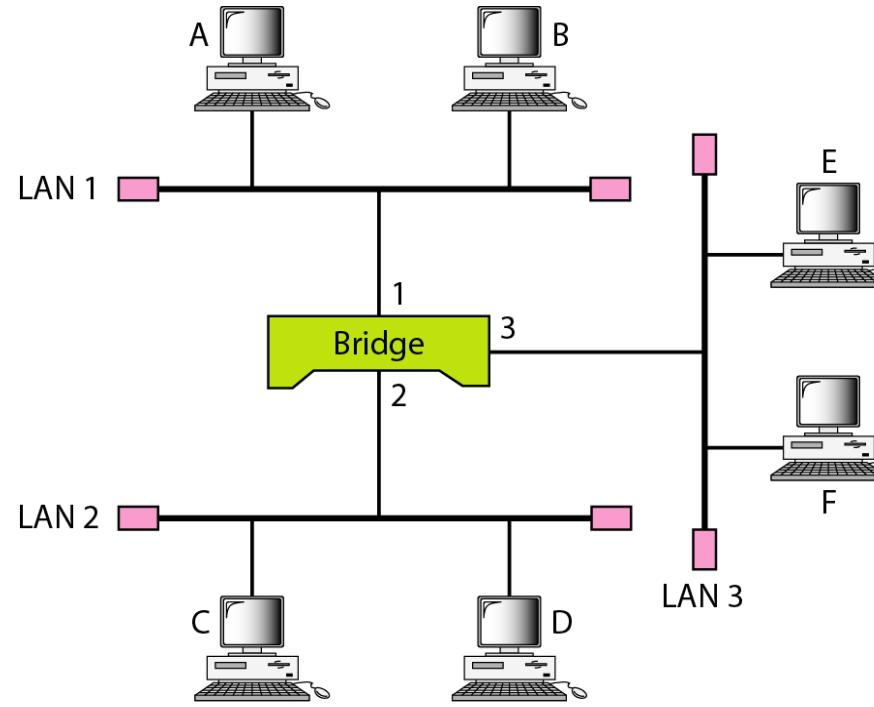




Note

**A bridge does not change the physical
(MAC) addresses in a frame.**

Figure 15.6 A learning bridge and the process of learning



Address	Port

a. Original

Address	Port
A	1

b. After A sends
a frame to D

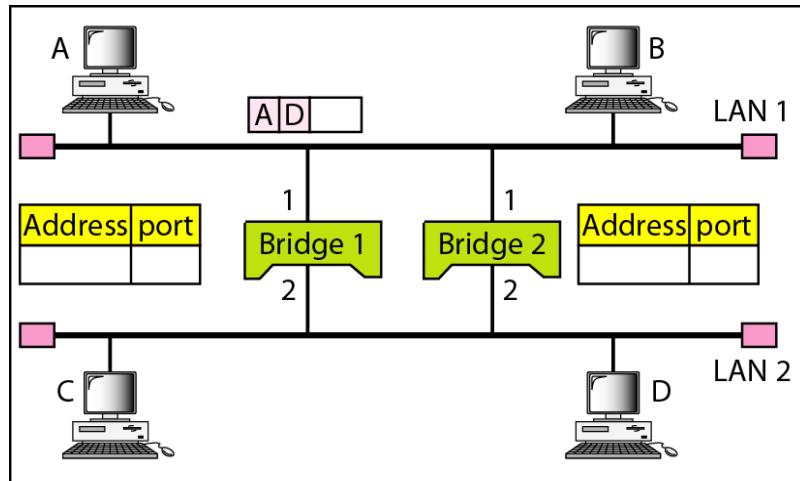
Address	Port
A	1
E	3

c. After E sends
a frame to A

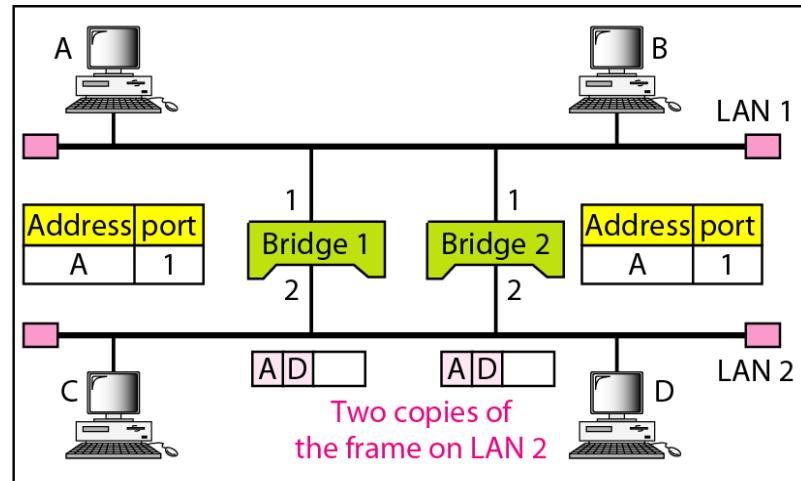
Address	Port
A	1
E	3
B	1

d. After B sends
a frame to C

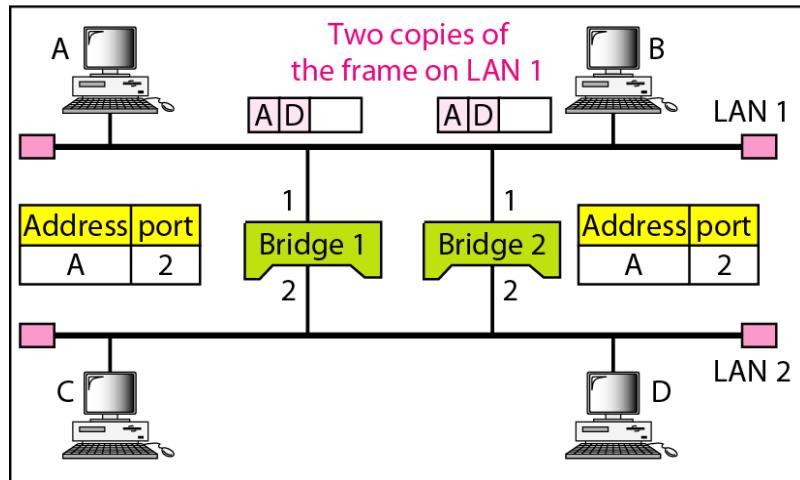
Figure 15.7 Loop problem in a learning bridge



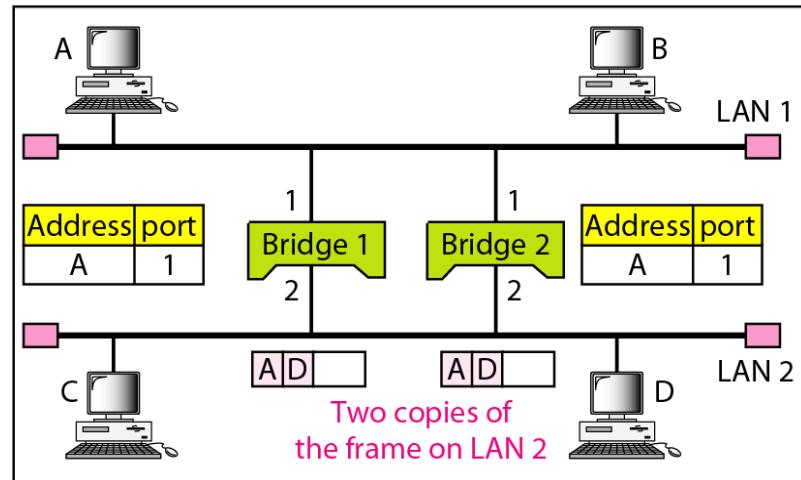
a. Station A sends a frame to station D



b. Both bridges forward the frame

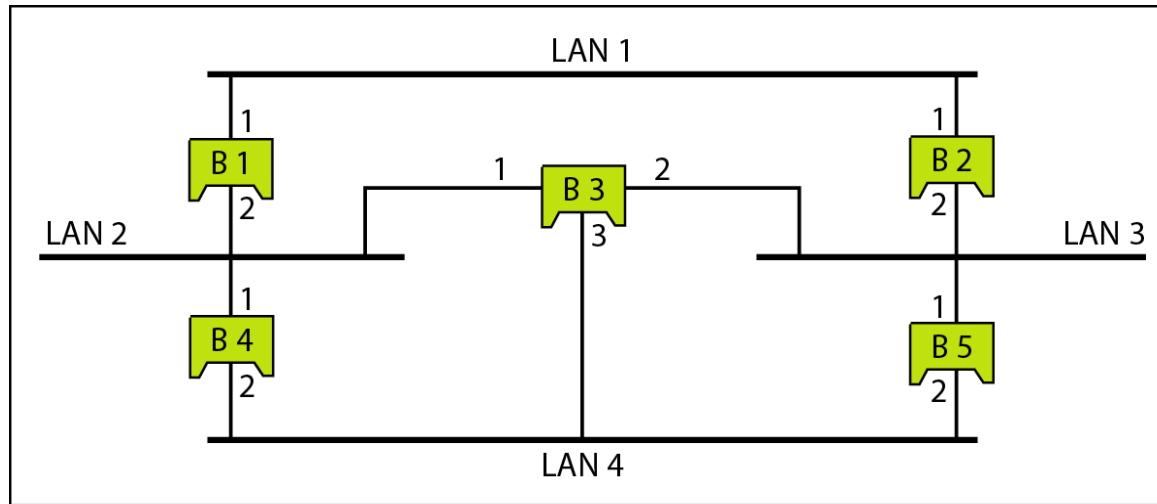


c. Both bridges forward the frame

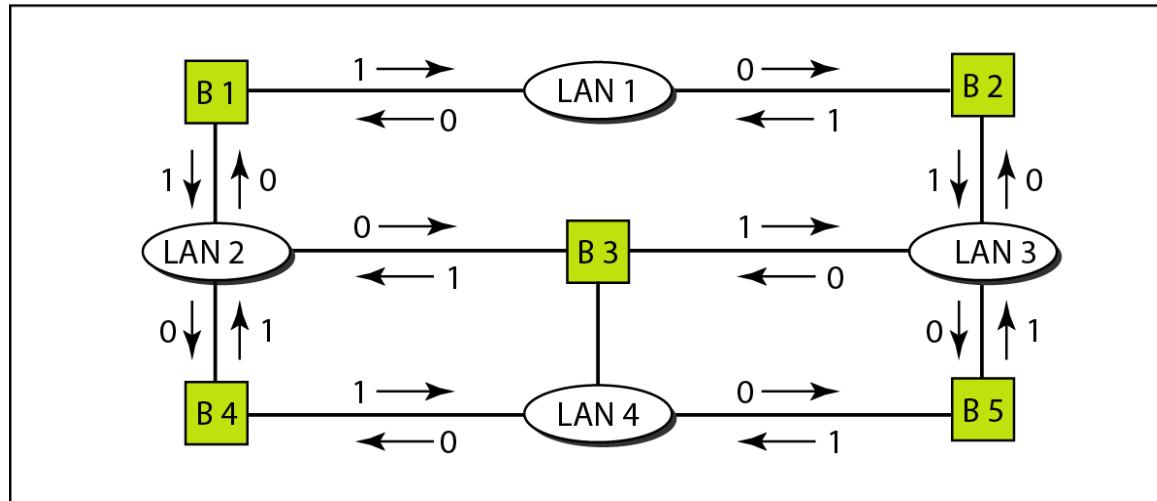


d. Both bridges forward the frame

Figure 15.8 A system of connected LANs and its graph representation

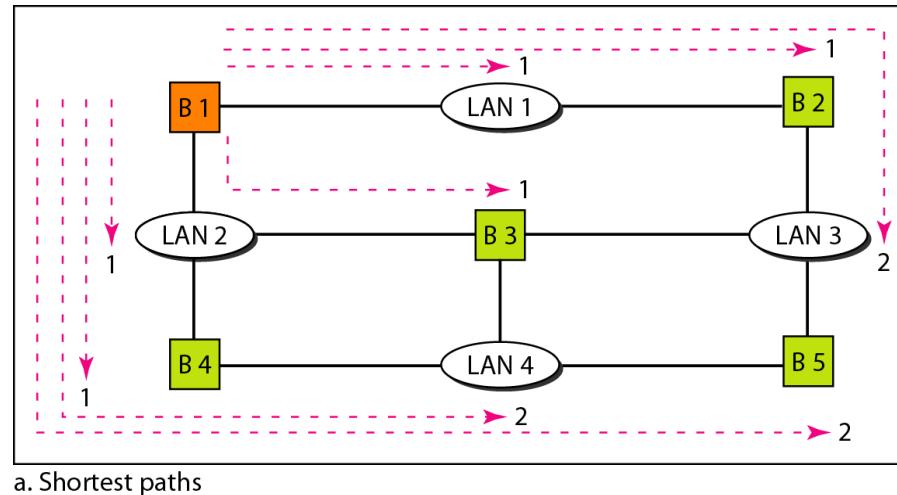


a. Actual system

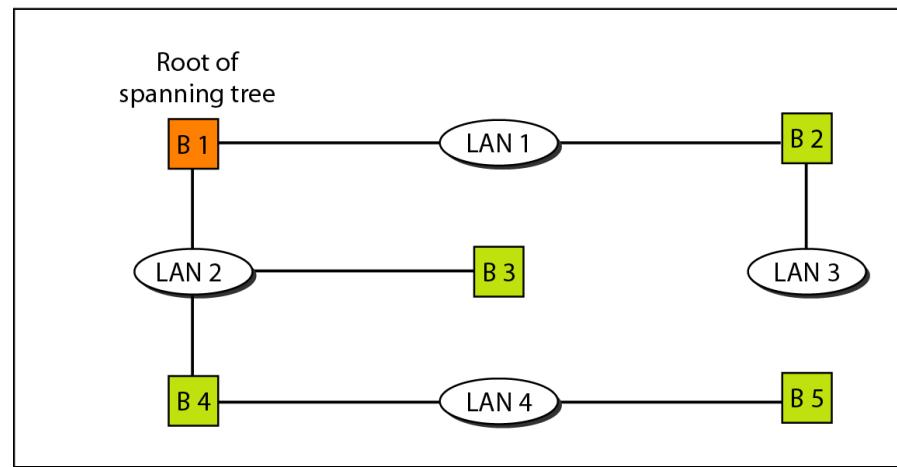


b. Graph representation with cost assigned to each arc

Figure 15.9 Finding the shortest paths and the spanning tree in a system of bridges

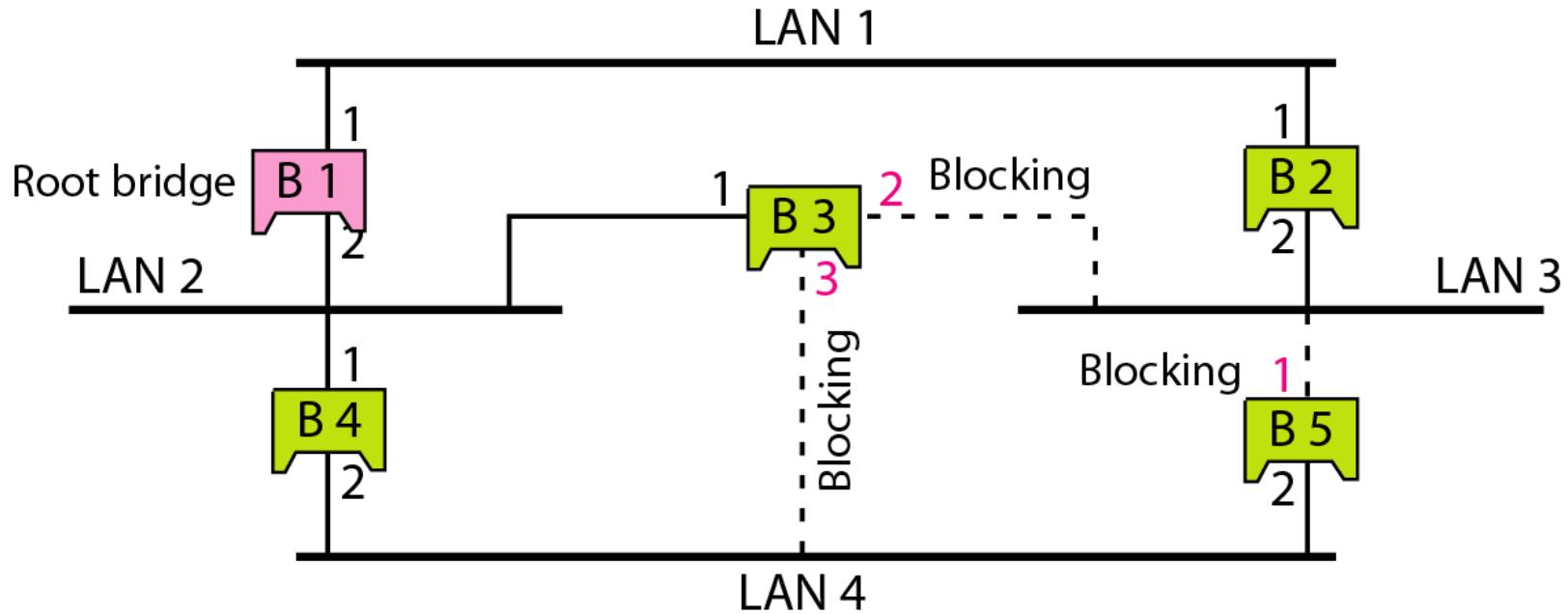


a. Shortest paths



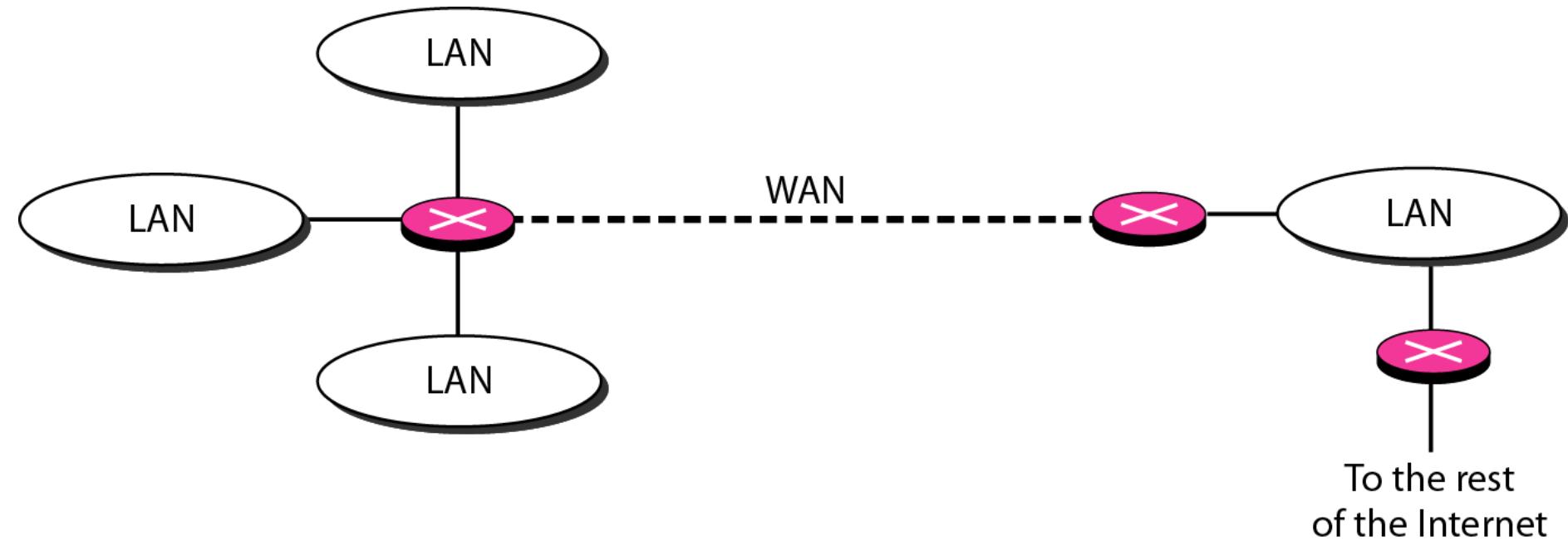
b. Spanning tree

Figure 15.10 Forwarding and blocking ports after using spanning tree algorithm



Ports 2 and 3 of bridge B3 are blocking ports (no frame is sent out of these ports). Port 1 of bridge B5 is also a blocking port (no frame is sent out of this port).

Figure 15.11 *Routers connecting independent LANs and WANs*



15-2 BACKBONE NETWORKS

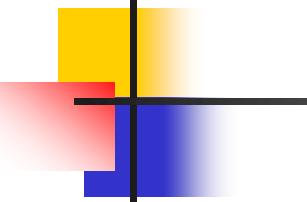
A backbone network allows several LANs to be connected. In a backbone network, no station is directly connected to the backbone; the stations are part of a LAN, and the backbone connects the LANs.

Topics discussed in this section:

Bus Backbone

Star Backbone

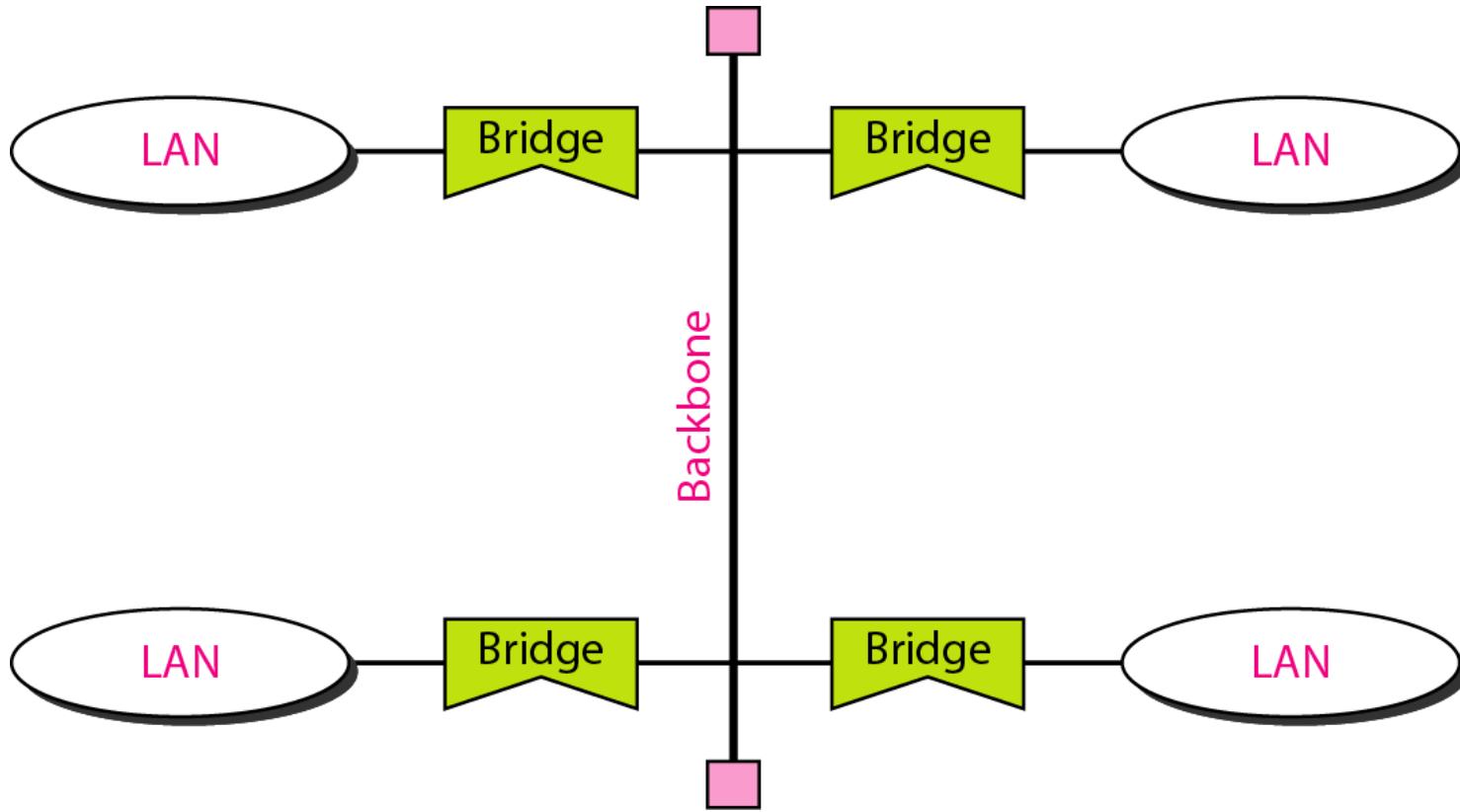
Connecting Remote LANs

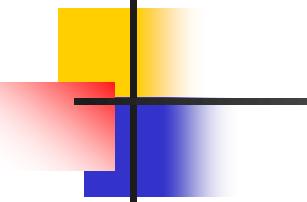


Note

**In a bus backbone, the topology
of the backbone is a bus.**

Figure 15.12 Bus backbone





Note

**In a star backbone, the topology of the backbone is a star;
the backbone is just one switch.**

Figure 15.13 *Star backbone*

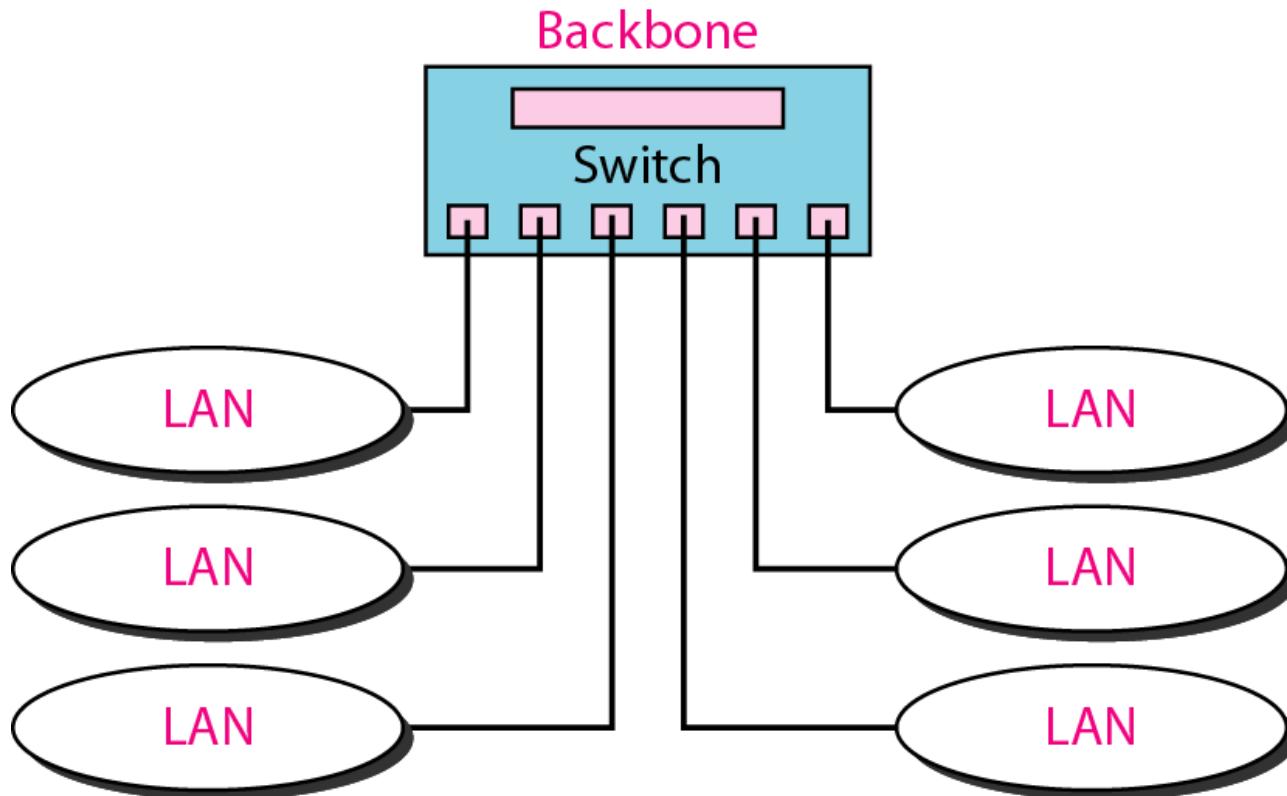
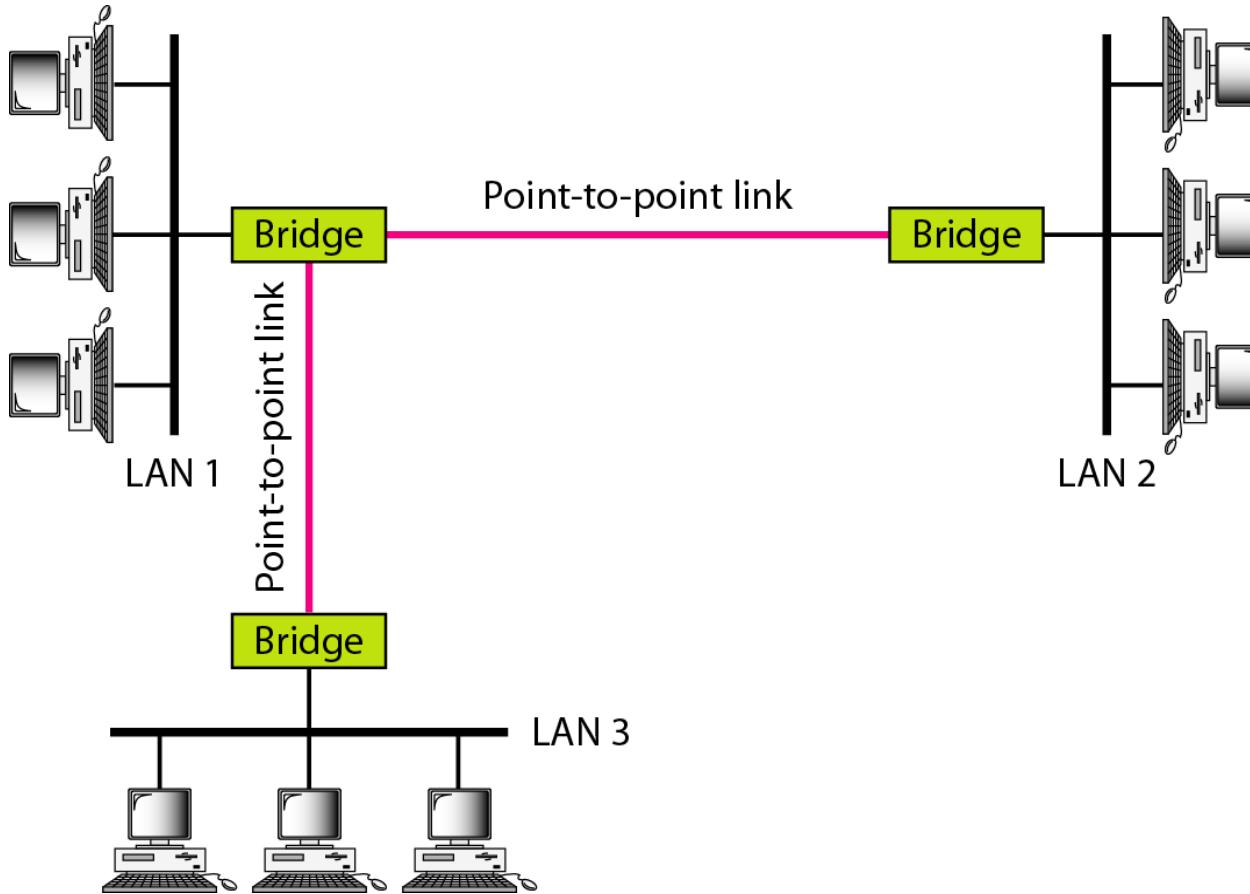
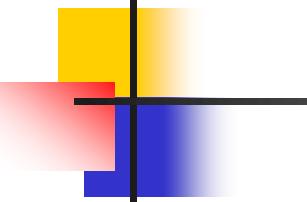


Figure 15.14 Connecting remote LANs with bridges





Note

A point-to-point link acts as a LAN in a remote backbone connected by remote bridges.

15-3 VIRTUAL LANs

*We can roughly define a **virtual local area network** (VLAN) as a local area network configured by software, not by physical wiring.*

Topics discussed in this section:

Membership

Configuration

Communication between Switches

IEEE Standard

Advantages

Figure 15.15 A switch connecting three LANs

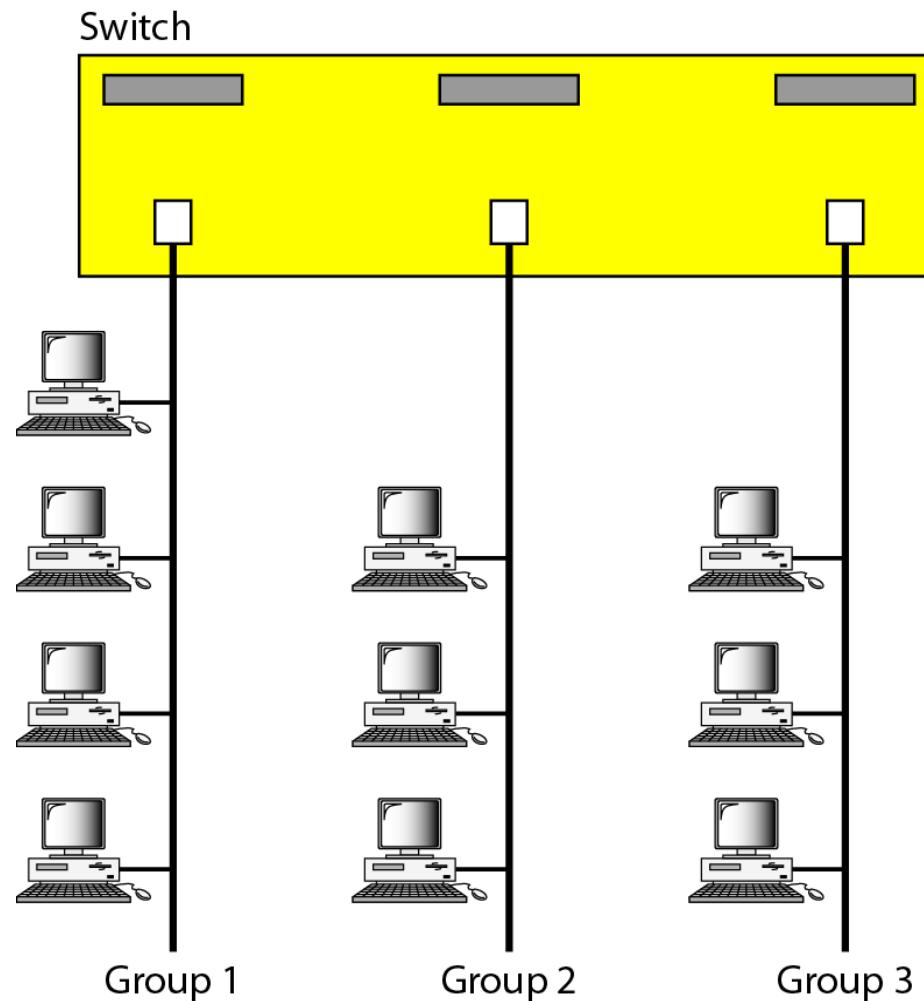


Figure 15.16 A switch using VLAN software

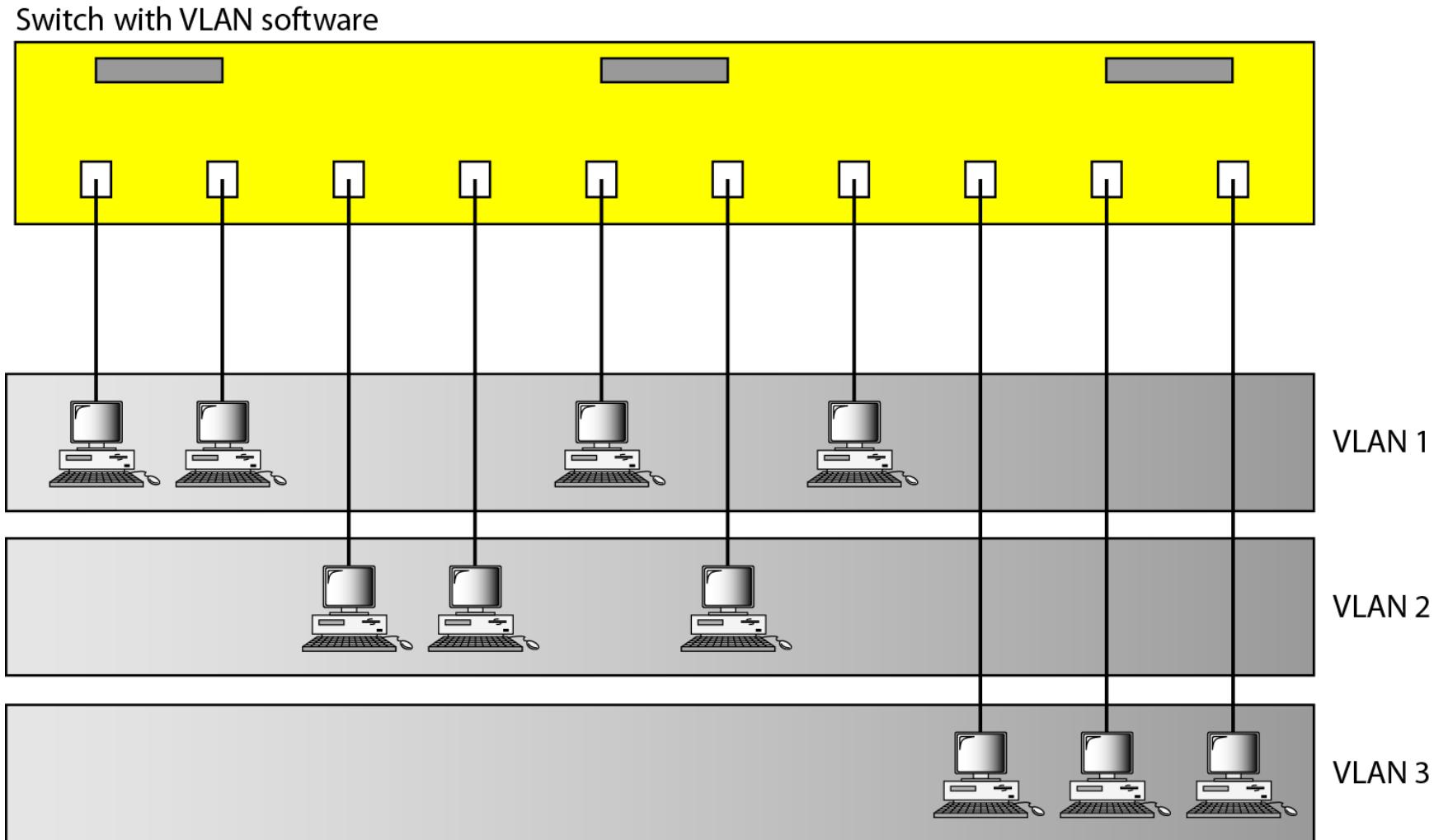
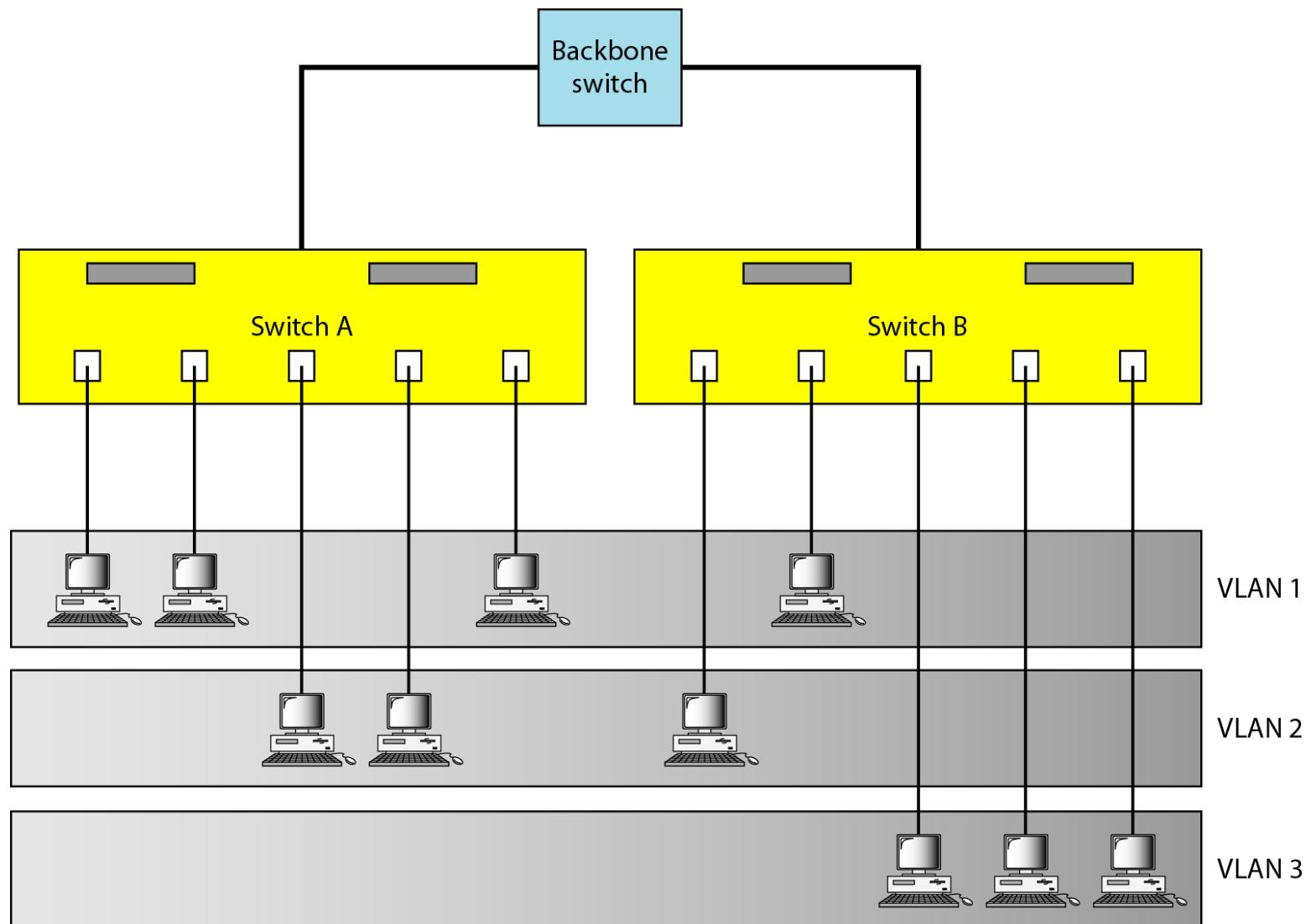
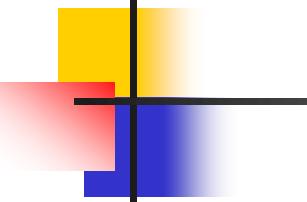


Figure 15.17 Two switches in a backbone using VLAN software





Note

VLANs create broadcast domains.

VLAN Characteristics

- ***Membership:*** Port no., MAC Addresses, IP Addresses, Multicast IP Addresses, Combination
- ***Configuration:*** Manual, Semiautomatic, Automatic
- ***Communication between Switches :*** Table Maintenance, Frame Tagging, TDM
- ***IEEE Standard***
- ***Advantages :*** Cost & Time reduction, creating virtual work groups, Security



Data Communications
and Networking

Fourth Edition

Forouzan

Chapter 19

Network Layer: Logical Addressing

19-1 IPv4 ADDRESSES

An **IPv4 address** is a **32-bit** address that uniquely and universally defines the connection of a device (for example, a computer or a router) to the Internet.

Topics discussed in this section:

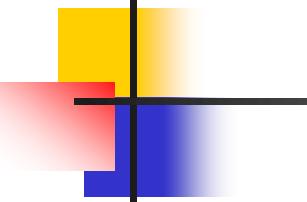
Address Space

Notations

Classful Addressing

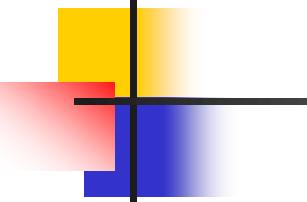
Classless Addressing

Network Address Translation (NAT)



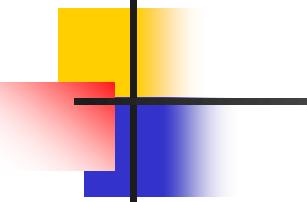
Note

An IPv4 address is 32 bits long.



Note

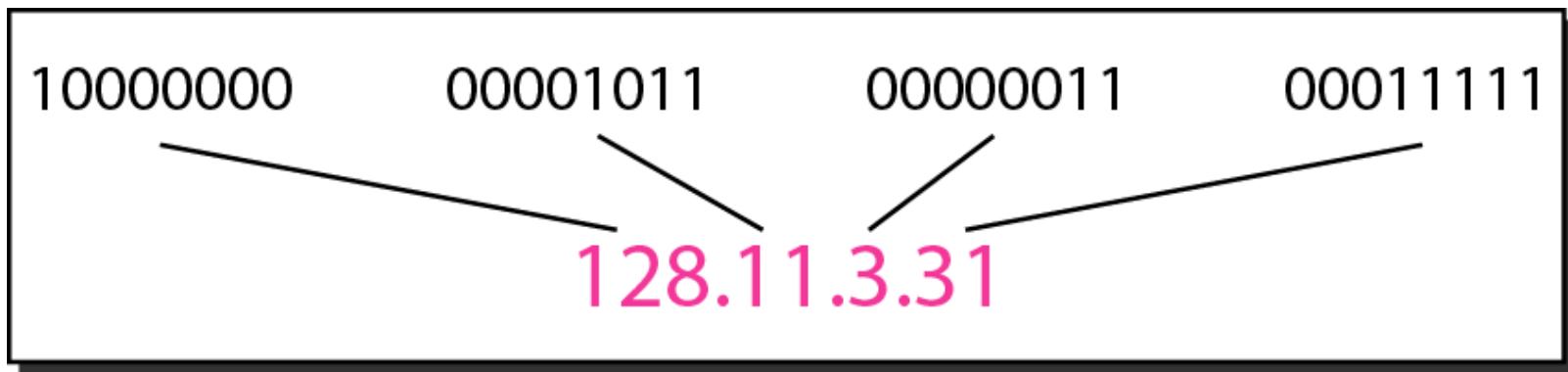
**The IPv4 addresses are unique
and universal.**

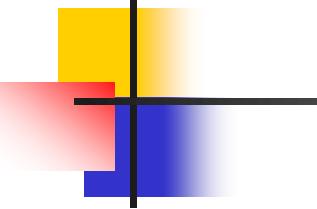


Note

**The address space of IPv4 is
 2^{32} or 4,294,967,296.**

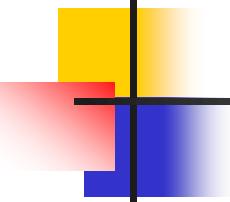
Figure 19.1 Dotted-decimal notation and binary notation for an IPv4 address





Note

**Numbering systems are reviewed in
Appendix B.**



Example 19.1

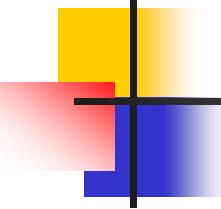
Change the following IPv4 addresses from binary notation to dotted-decimal notation.

- a. 10000001 00001011 00001011 11101111
- b. 11000001 10000011 00011011 11111111

Solution

We replace each group of 8 bits with its equivalent decimal number (see Appendix B) and add dots for separation.

- a. 129.11.11.239
- b. 193.131.27.255



Example 19.2

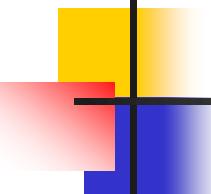
Change the following IPv4 addresses from dotted-decimal notation to binary notation.

- a. 111.56.45.78
- b. 221.34.7.82

Solution

We replace each decimal number with its binary equivalent (see Appendix B).

- a. 01101111 00111000 00101101 01001110
- b. 11011101 00100010 00000111 01010010



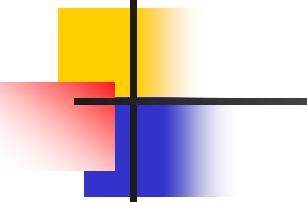
Example 19.3

Find the error, if any, in the following IPv4 addresses.

- a. 111.56.045.78
- b. 221.34.7.8.20
- c. 75.45.301.14
- d. 11100010.23.14.67

Solution

- a. *There must be no leading zero (045).*
- b. *There can be no more than four numbers.*
- c. *Each number needs to be less than or equal to 255.*
- d. *A mixture of binary notation and dotted-decimal notation is not allowed.*



Note

**In classful addressing, the address space is divided into five classes:
A, B, C, D, and E.**

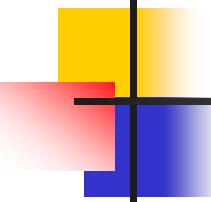
Figure 19.2 Finding the classes in binary and dotted-decimal notation

	First byte	Second byte	Third byte	Fourth byte
Class A	0			
Class B	10			
Class C	110			
Class D	1110			
Class E	1111			

a. Binary notation

	First byte	Second byte	Third byte	Fourth byte
Class A	0–127			
Class B	128–191			
Class C	192–223			
Class D	224–239			
Class E	240–255			

b. Dotted-decimal notation



Example 19.4

Find the class of each address.

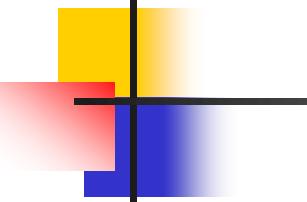
- a. 00000001 00001011 00001011 11101111
- b. 11000001 10000011 00011011 11111111
- c. 14.23.120.8
- d. 252.5.15.111

Solution

- a. *The first bit is 0. This is a class A address.*
- b. *The first 2 bits are 1; the third bit is 0. This is a class C address.*
- c. *The first byte is 14; the class is A.*
- d. *The first byte is 252; the class is E.*

Table 19.1 *Number of blocks and block size in classful IPv4 addressing*

<i>Class</i>	<i>Number of Blocks</i>	<i>Block Size</i>	<i>Application</i>
A	128	16,777,216	Unicast
B	16,384	65,536	Unicast
C	2,097,152	256	Unicast
D	1	268,435,456	Multicast
E	1	268,435,456	Reserved

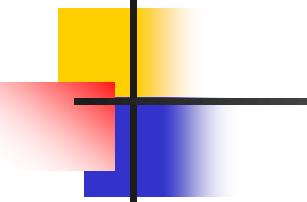


Note

In classful addressing, a large part of the available addresses were wasted.

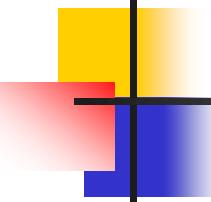
Table 19.2 *Default masks for classful addressing*

Class	Binary	Dotted-Decimal	CIDR
A	11111111 00000000 00000000 00000000	255.0.0.0	/8
B	11111111 11111111 00000000 00000000	255.255.0.0	/16
C	11111111 11111111 11111111 00000000	255.255.255.0	/24



Note

Classful addressing, which is almost obsolete, is replaced with classless addressing.

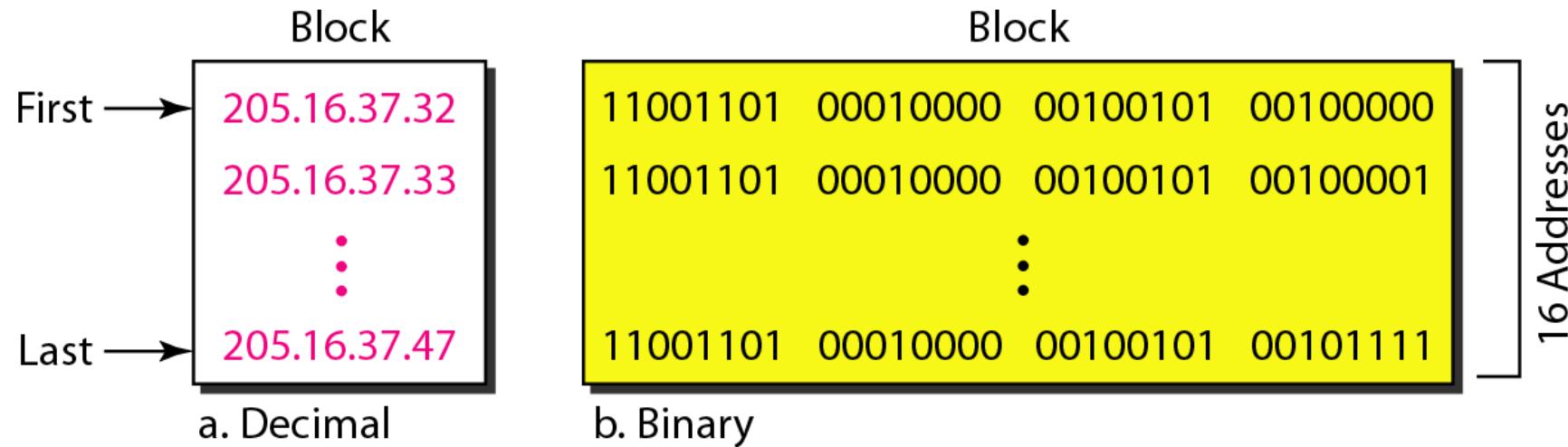


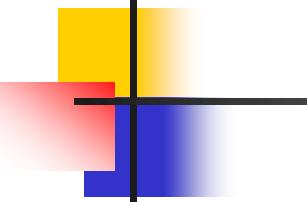
Example 19.5

Figure 19.3 shows a block of addresses, in both binary and dotted-decimal notation, granted to a small business that needs 16 addresses.

We can see that the restrictions are applied to this block. The addresses are contiguous. The number of addresses is a power of 2 ($16 = 2^4$), and the first address is divisible by 16. The first address, when converted to a decimal number, is 3,440,387,360, which when divided by 16 results in 215,024,210.

Figure 19.3 A block of 16 addresses granted to a small organization



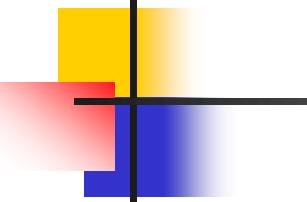


Note

In IPv4 addressing, a block of addresses can be defined as

x.y.z.t /n

in which x.y.z.t defines one of the addresses and the /n defines the mask.



Note

The first address in the block can be found by setting the rightmost $32 - n$ bits to 0s.

Example 19.6

A block of addresses is granted to a small organization. We know that one of the addresses is 205.16.37.39/28. What is the first address in the block?

Solution

The binary representation of the given address is

11001101 00010000 00100101 00100111

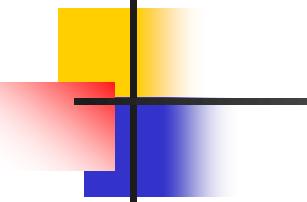
If we set 32–28 rightmost bits to 0, we get

11001101 00010000 00100101 0010000

or

205.16.37.32.

This is actually the block shown in Figure 19.3.



Note

The last address in the block can be found by setting the rightmost $32 - n$ bits to 1s.

Example 19.7

Find the last address for the block in Example 19.6.

Solution

The binary representation of the given address is

11001101 00010000 00100101 00100111

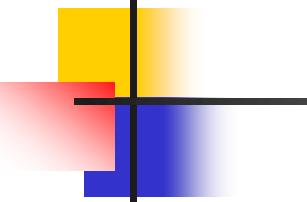
If we set 32 – 28 rightmost bits to 1, we get

11001101 00010000 00100101 00101111

or

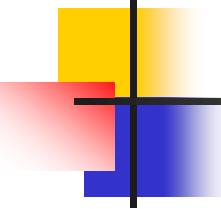
205.16.37.47

This is actually the block shown in Figure 19.3.



Note

**The number of addresses in the block
can be found by using the formula
 2^{32-n} .**

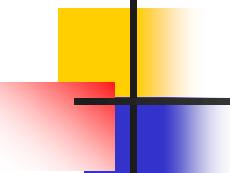


Example 19.8

Find the number of addresses in Example 19.6.

Solution

The value of n is 28, which means that number of addresses is 2^{32-28} or 16.



Example 19.9

Another way to find the first address, the last address, and the number of addresses is to represent the mask as a 32-bit binary (or 8-digit hexadecimal) number. This is particularly useful when we are writing a program to find these pieces of information. In Example 19.6 the /28 can be represented as

11111111 11111111 11111111 11110000

(twenty-eight 1s and four 0s).

Find

- a. The first address*
- b. The last address*
- c. The number of addresses.*

Example 19.9 (continued)

Solution

a. The first address can be found by ANDing the given addresses with the mask. ANDing here is done bit by bit. The result of ANDing 2 bits is 1 if both bits are 1s; the result is 0 otherwise.

Address: 11001101 00010000 00100101 00100111

Mask: **11111111 11111111 11111111 11110000**

First address: 11001101 00010000 00100101 00100000

Example 19.9 (continued)

b. The last address can be found by ORing the given addresses with the complement of the mask. ORing here is done bit by bit. The result of ORing 2 bits is 0 if both bits are 0s; the result is 1 otherwise. The complement of a number is found by changing each 1 to 0 and each 0 to 1.

Address: 11001101 00010000 00100101 00100111

Mask complement: **00000000 00000000 00000000 00001111**

Last address: 11001101 00010000 00100101 00101111

Example 19.9 (continued)

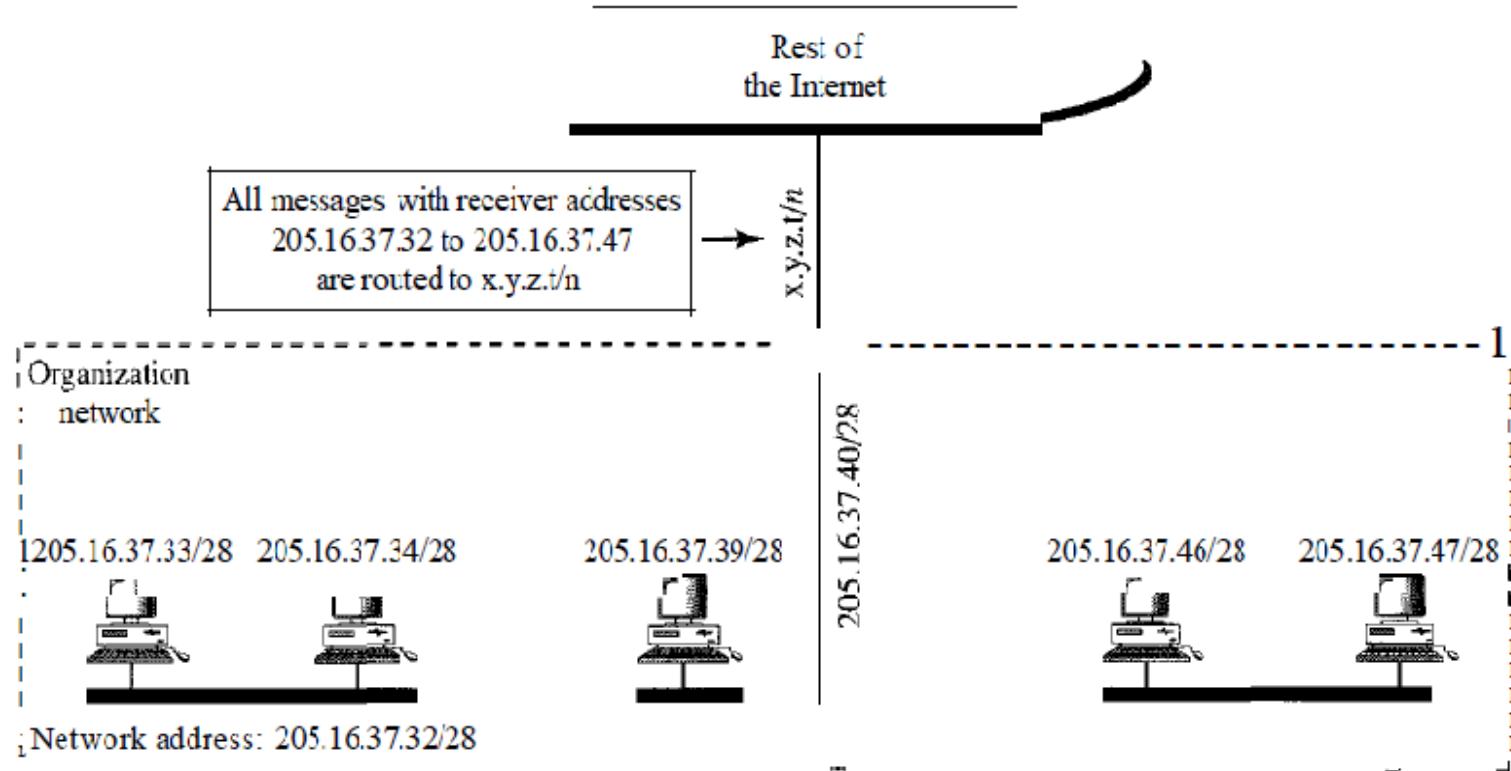
- c. The number of addresses can be found by complementing the mask, interpreting it as a decimal number, and adding 1 to it.

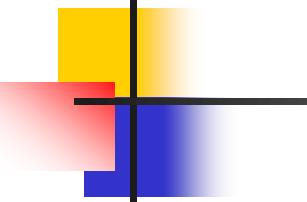
Mask complement: 00000000 00000000 00000000 00001111

Number of addresses: $15 + 1 = 16$

Network Addresses

Figure 19.4 A network configuration for the block 205.16.37.32/28





Note

The first address in a block is normally not assigned to any device; it is used as the network address that represents the organization to the rest of the world.

Figure 19.5 *Hierarchy in a telephone network in North America*

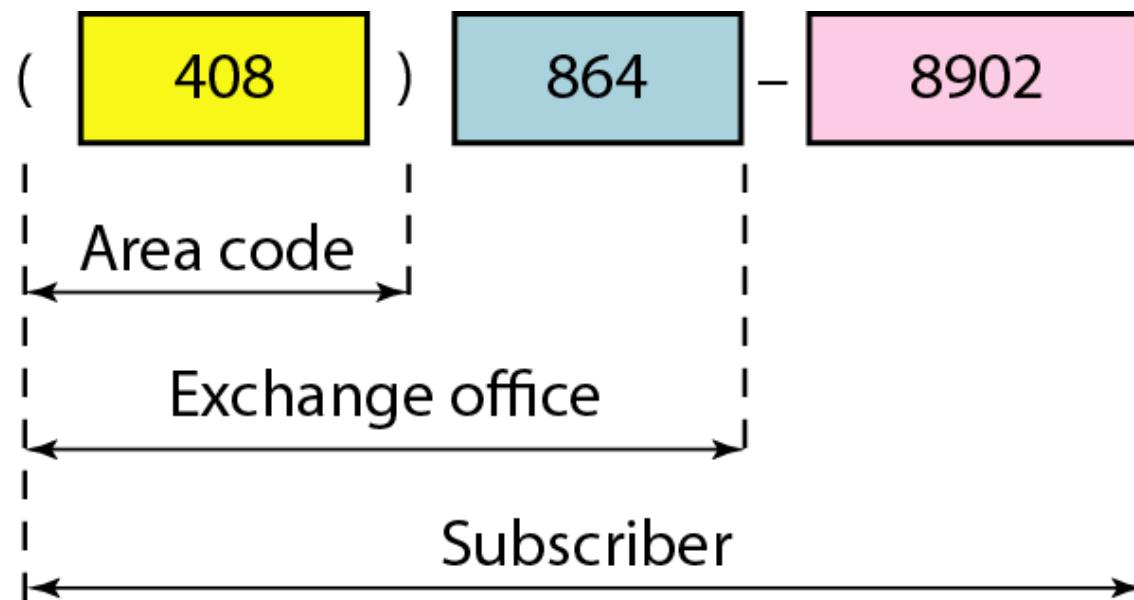
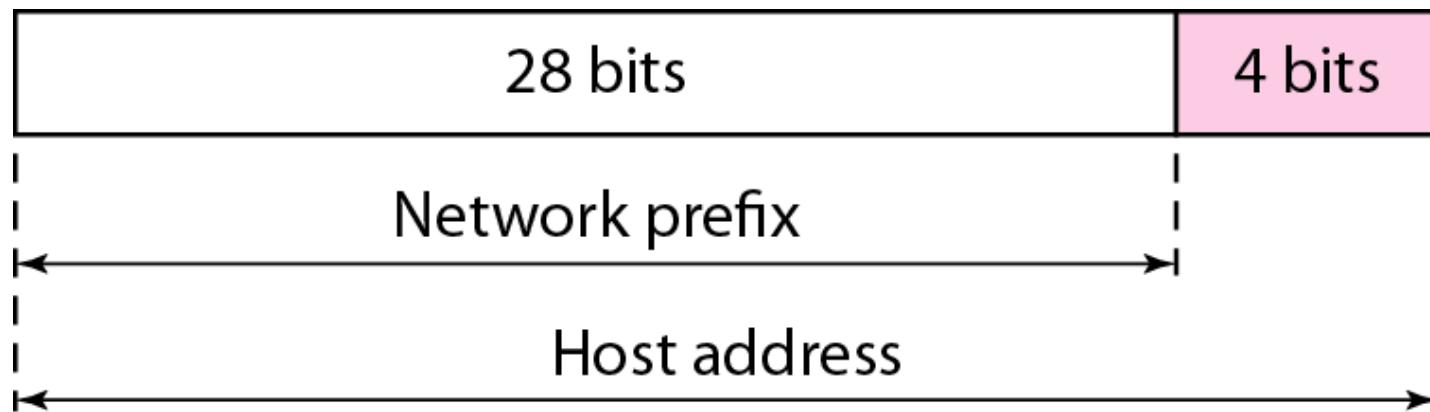
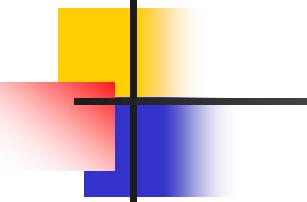


Figure 19.6 Two levels of hierarchy in an IPv4 address

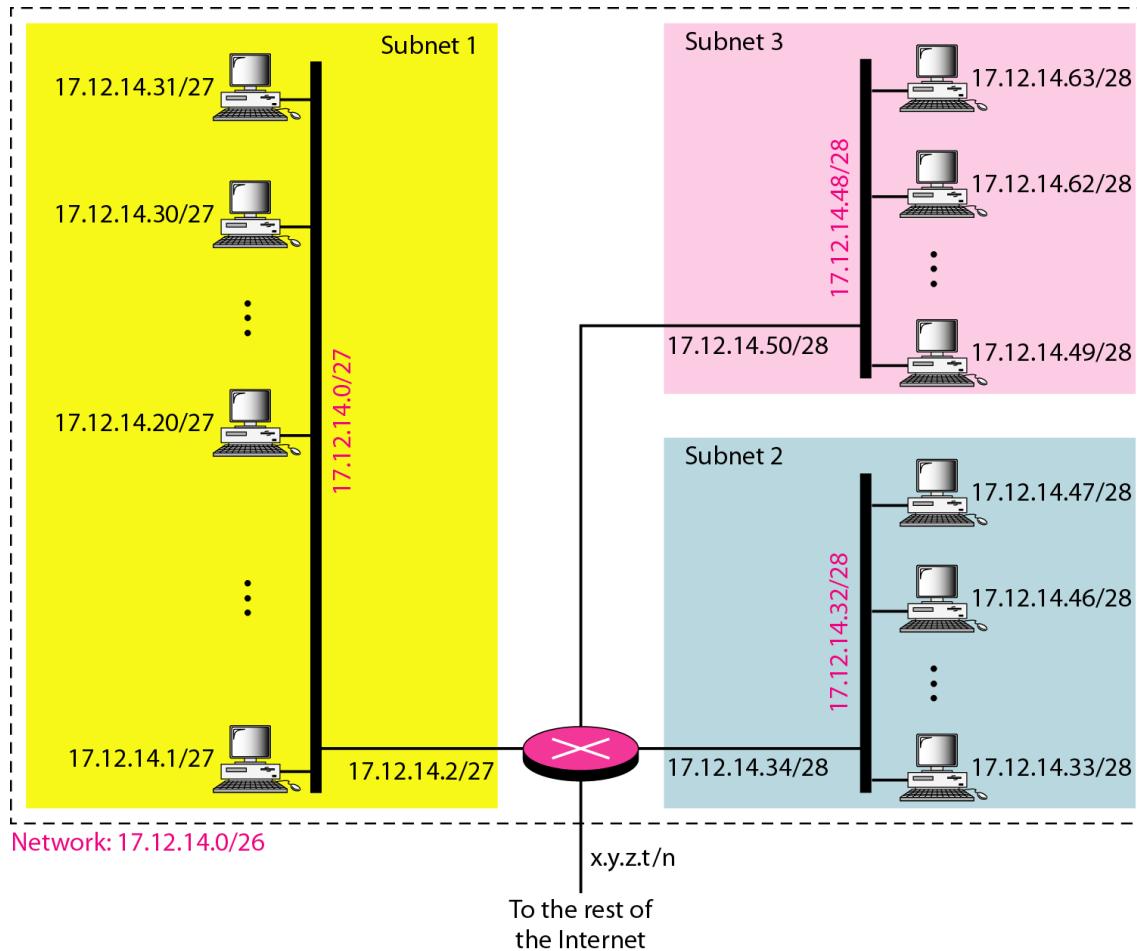


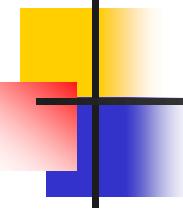


Note

Each address in the block can be considered as a two-level hierarchical structure:
the leftmost n bits (prefix) define the network;
the rightmost $32 - n$ bits define the host.

Figure 19.7 Configuration and addresses in a subnetted network





In subnet 1, the address 17.12.14.29/27 can give us the subnet address if we use the mask /27 because

Host: 00010001 00001100 00001110 00011101

Mask: /27

Subnet: 00010001 00001100 00001110 00000000 (17.12.14.0)

In subnet 2, the address 17.12.14.45/28 can give us the subnet address if we use the mask /28 because

Host: 00010001 00001100 00001110 00101101

Mask: /28

Subnet: 00010001 00001100 00001110 00100000 (17.12.14.32)

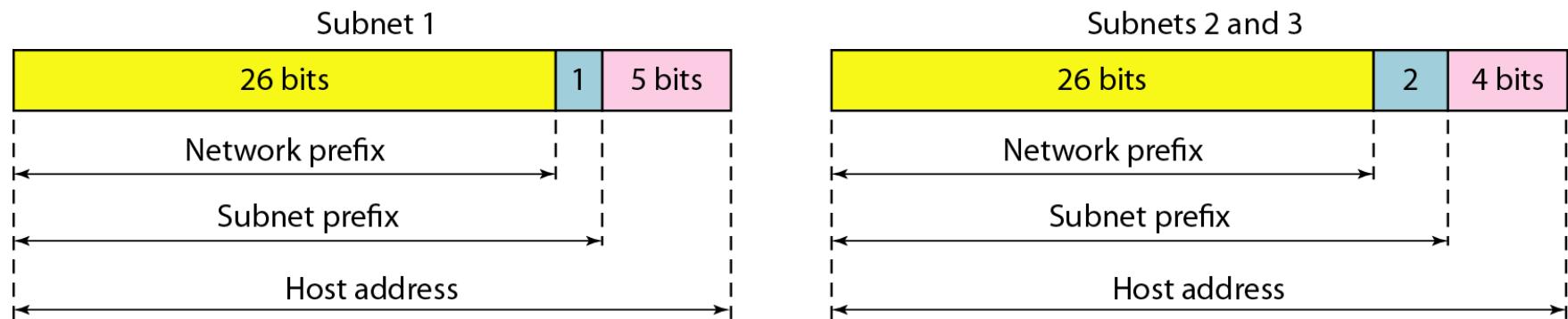
c. In subnet 3, the address 17.12.14.50/28 can give us the subnet address if we use the mask /28 because

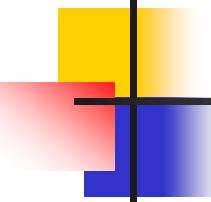
Host: 00010001 00001100 00001110 00110010

Mask: /28

Subnet: 00010001 00001100 00001110 00110000 (17.12.14.48)

Figure 19.8 *Three-level hierarchy in an IPv4 address*





Example 19.10

An ISP is granted a block of addresses starting with 190.100.0.0/16 (65,536 addresses). The ISP needs to distribute these addresses to three groups of customers as follows:

- a. The first group has 64 customers; each needs 256 addresses.*
- b. The second group has 128 customers; each needs 128 addresses.*
- c. The third group has 128 customers; each needs 64 addresses.*

Design the subblocks and find out how many addresses are still available after these allocations.

Example 19.10 (continued)

Solution

Figure 19.9 shows the situation.

Group 1

For this group, each customer needs 256 addresses. This means that 8 ($\log_2 256$) bits are needed to define each host. The prefix length is then $32 - 8 = 24$. The addresses are

1st Customer: 190.100.0.0/24 190.100.0.255/24

2nd Customer: 190.100.1.0/24 190.100.1.255/24

...

64th Customer: 190.100.63.0/24 190.100.63.255/24

Total = $64 \times 256 = 16,384$

Example 19.10 (continued)

Group 2

For this group, each customer needs 128 addresses. This means that 7 ($\log_2 128$) bits are needed to define each host. The prefix length is then $32 - 7 = 25$. The addresses are

1st Customer: 190.100.64.0/25 190.100.64.127/25

2nd Customer: 190.100.64.128/25 190.100.64.255/25

...

128th Customer: 190.100.127.128/25 190.100.127.255/25

Total = $128 \times 128 = 16,384$

Example 19.10 (continued)

Group 3

For this group, each customer needs 64 addresses. This means that 6 ($\log_2 64$) bits are needed to each host. The prefix length is then $32 - 6 = 26$. The addresses are

<i>1st Customer:</i>	$190.100.128.0/26$	$190.100.128.63/26$
<i>2nd Customer:</i>	$190.100.128.64/26$	$190.100.128.127/26$
...		
<i>128th Customer:</i>	$190.100.159.192/26$	$190.100.159.255/26$
<i>Total = $128 \times 64 = 8192$</i>		

Number of granted addresses to the ISP: 65,536

Number of allocated addresses by the ISP: 40,960

Number of available addresses: 24,576

Figure 19.9 An example of address allocation and distribution by an ISP

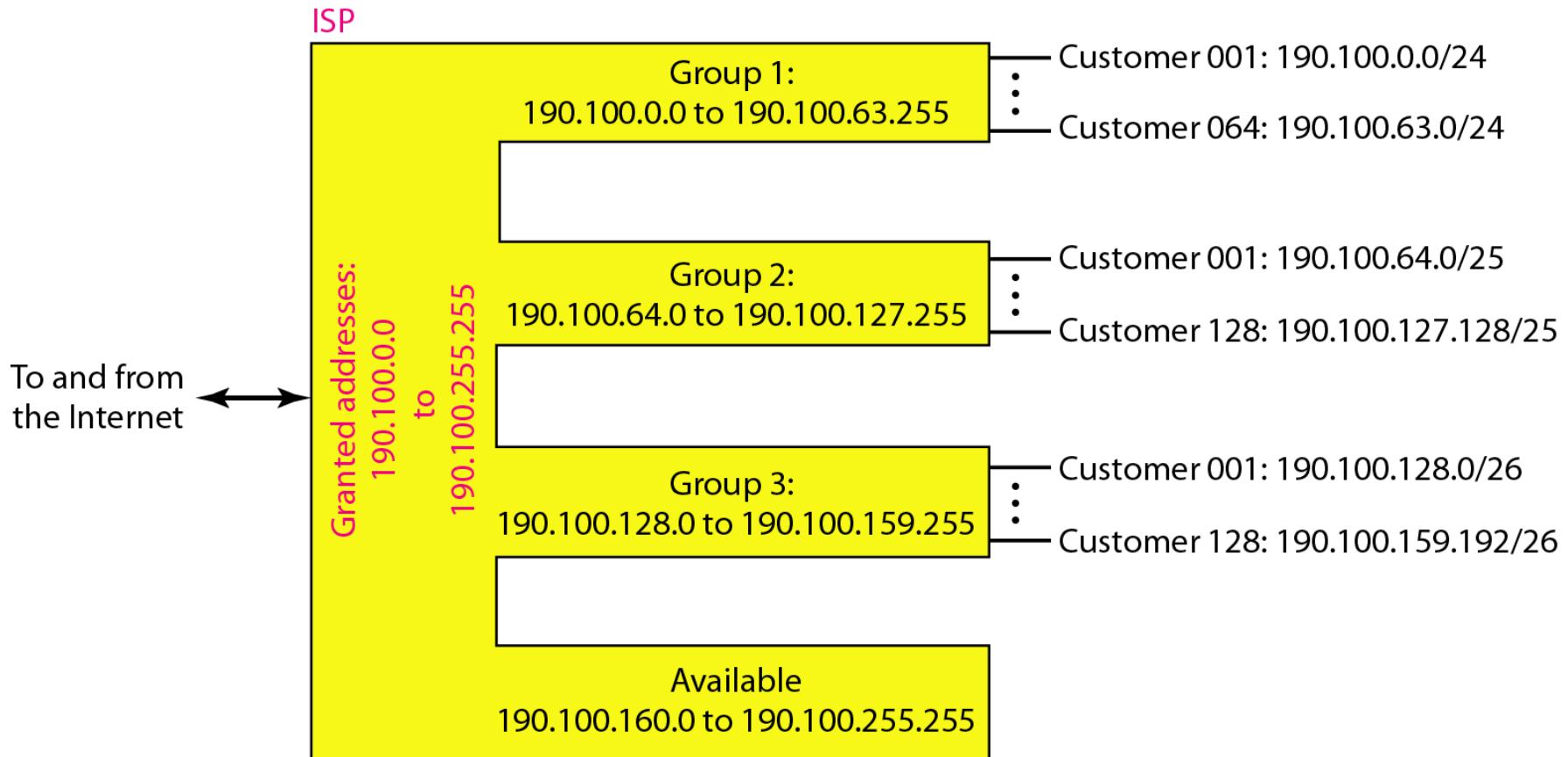


Table 19.3 *Addresses for private networks*

<i>Range</i>	<i>Total</i>
10.0.0.0 to 10.255.255.255	2^{24}
172.16.0.0 to 172.31.255.255	2^{20}
192.168.0.0 to 192.168.255.255	2^{16}

Figure 19.10 A NAT implementation

Site using private addresses

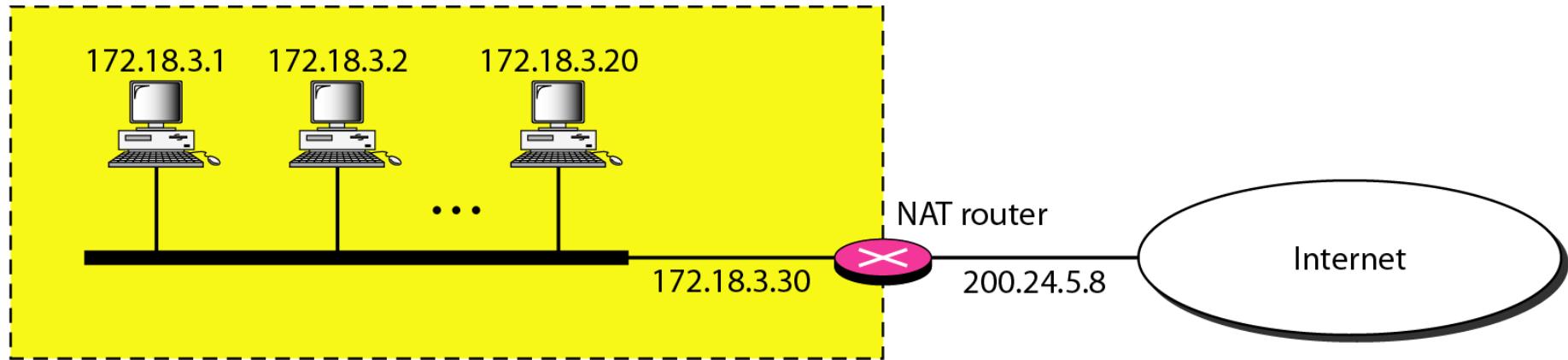


Figure 19.11 Addresses in a NAT

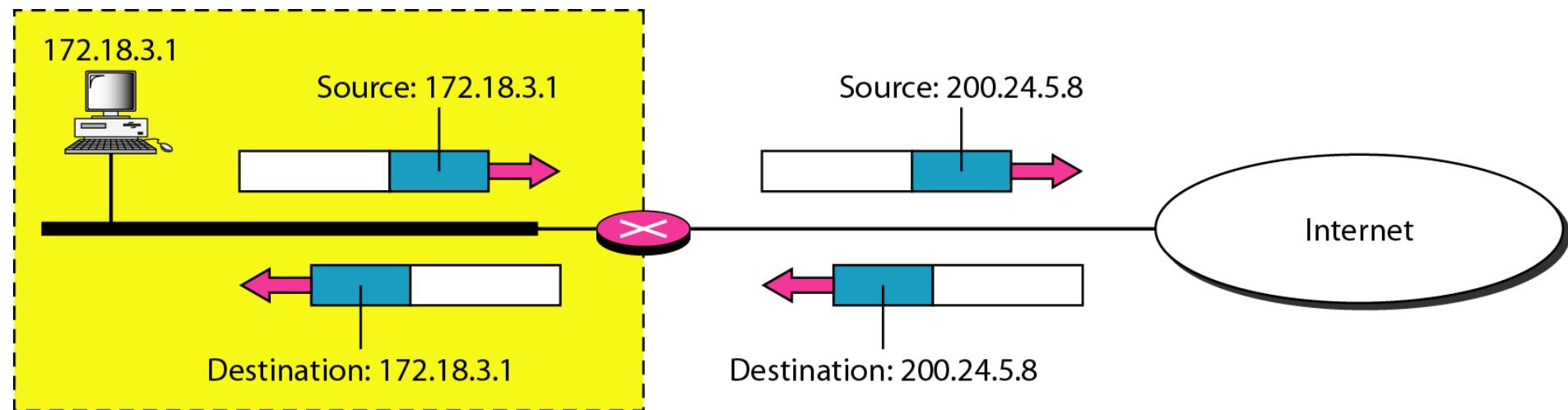


Figure 19.12 NAT address translation

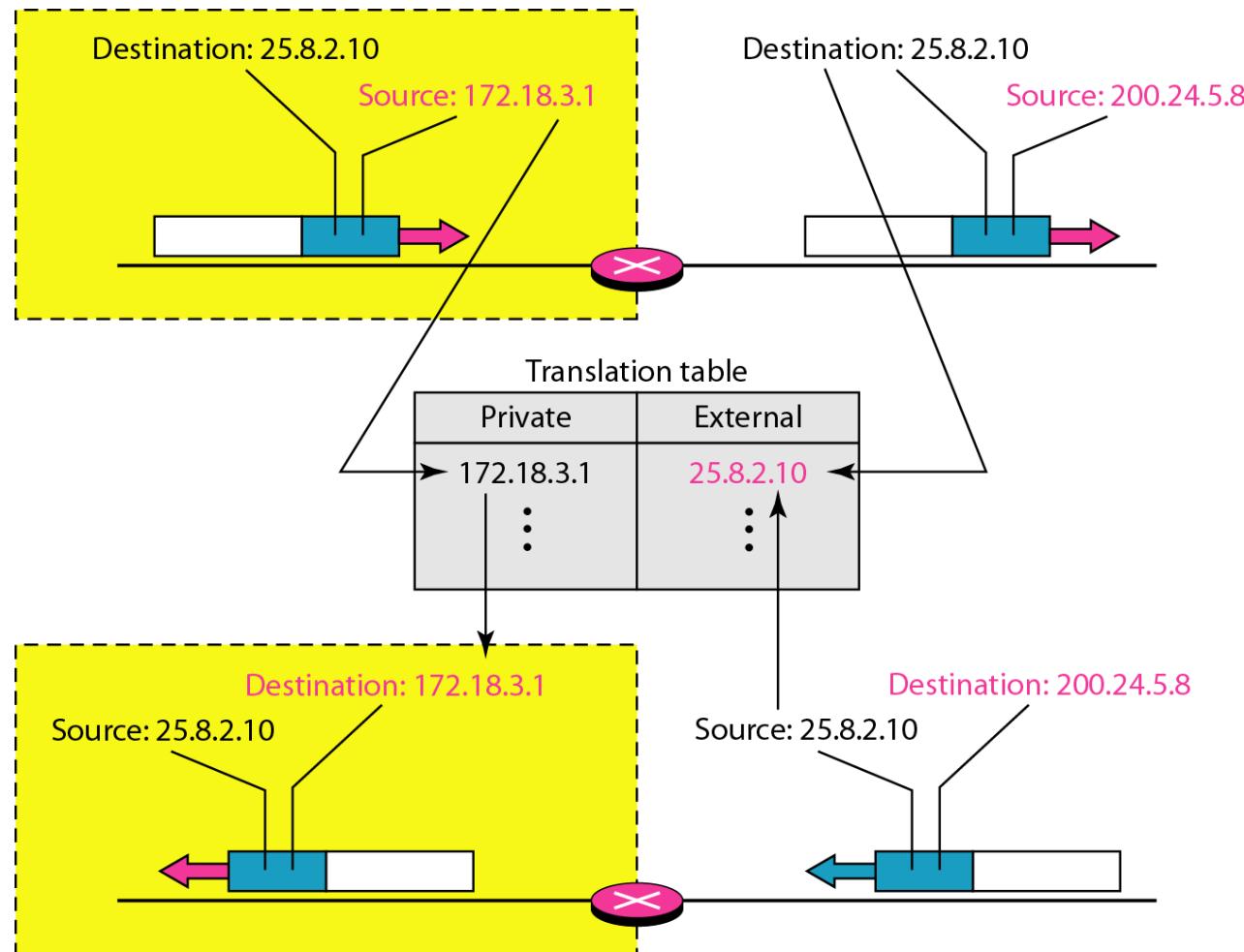
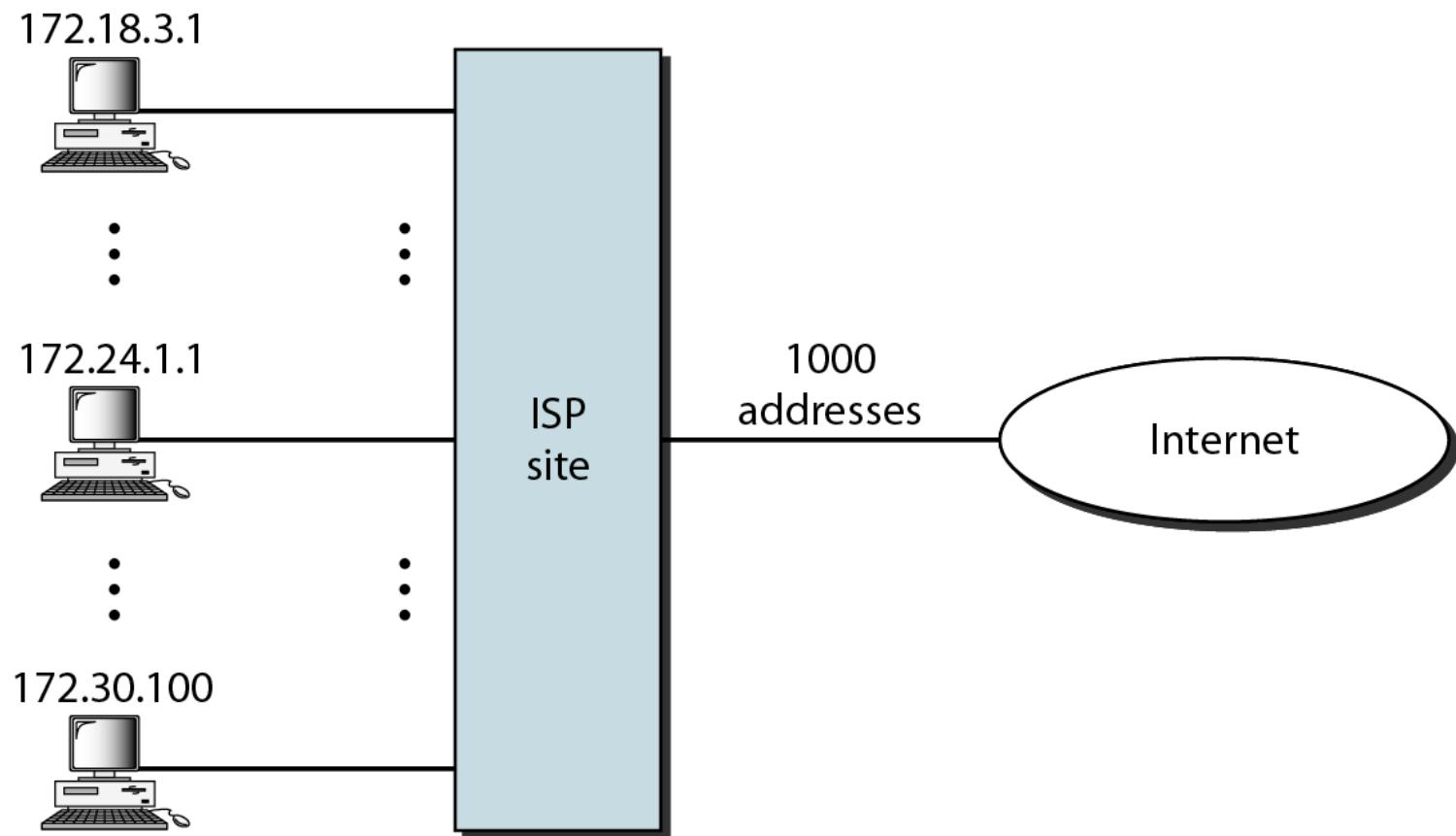


Table 19.4 *Five-column translation table*

<i>Private Address</i>	<i>Private Port</i>	<i>External Address</i>	<i>External Port</i>	<i>Transport Protocol</i>
172.18.3.1	1400	25.8.3.2	80	TCP
172.18.3.2	1401	25.8.3.2	80	TCP
...

Figure 19.13 An ISP and NAT



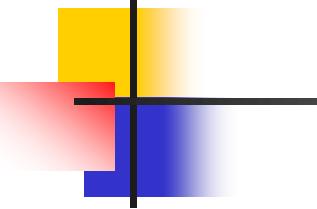
19-2 IPv6 ADDRESSES

Despite all short-term solutions, address depletion is still a long-term problem for the Internet. This and other problems in the IP protocol itself have been the motivation for IPv6.

Topics discussed in this section:

Structure

Address Space



Note

An IPv6 address is 128 bits long.

Figure 19.14 IPv6 address in binary and hexadecimal colon notation

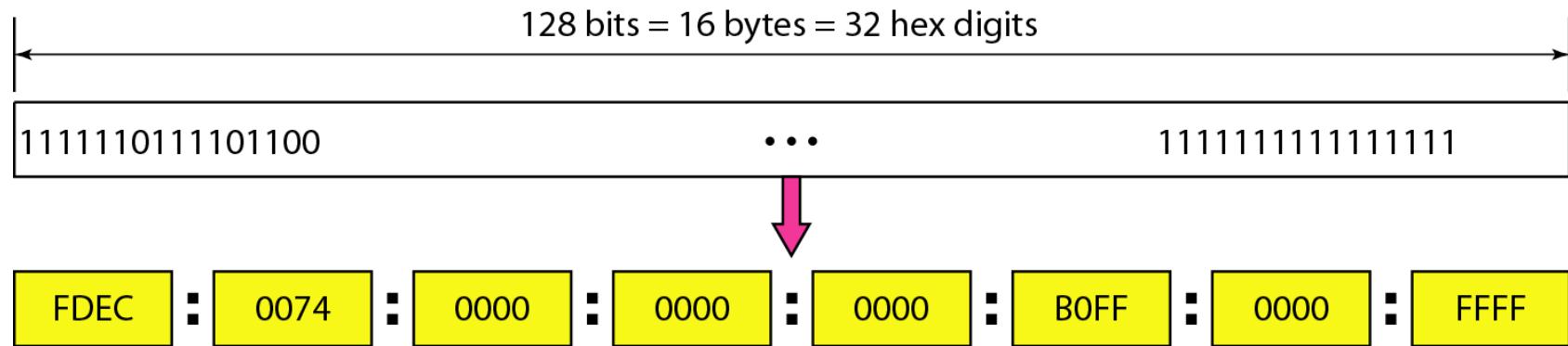
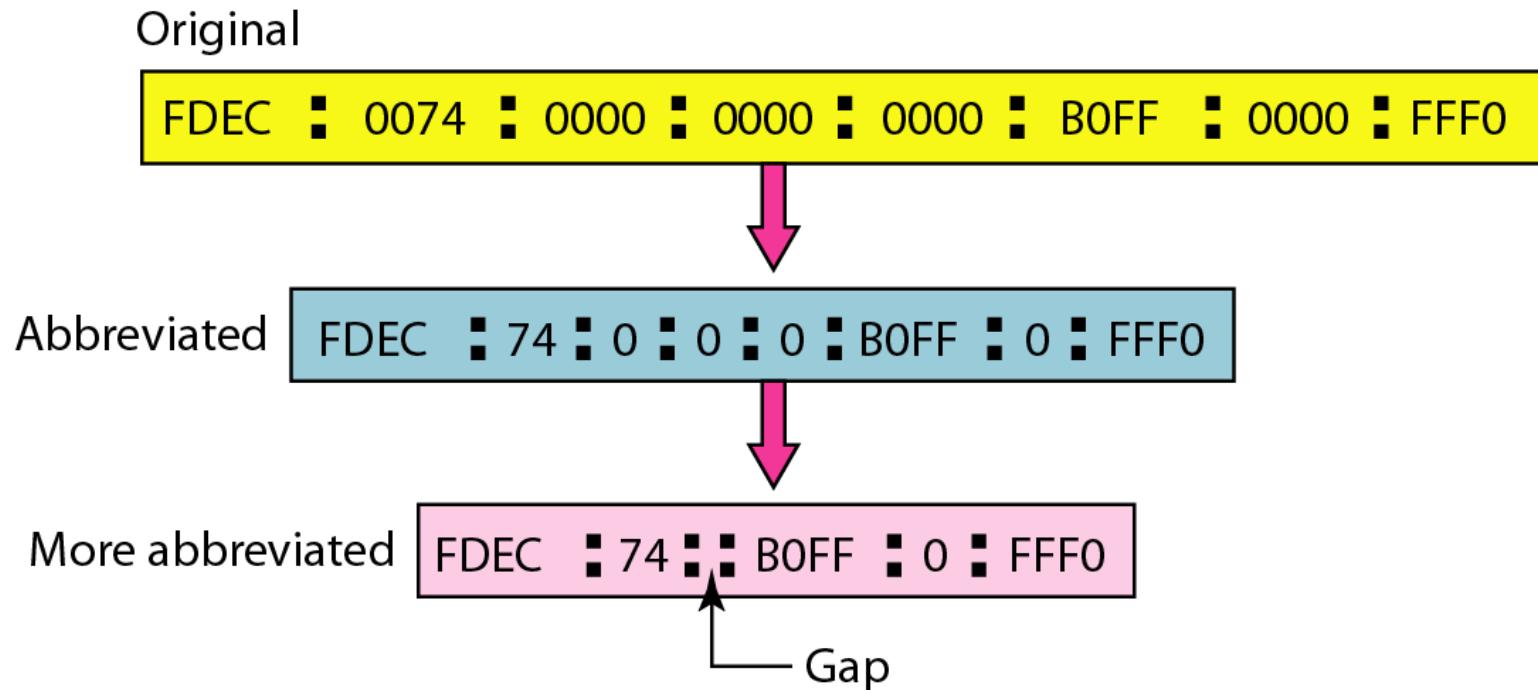


Figure 19.15 Abbreviated IPv6 addresses



Example 19.11

Expand the address 0:15::1:12:1213 to its original.

Solution

We first need to align the left side of the double colon to the left of the original pattern and the right side of the double colon to the right of the original pattern to find how many 0s we need to replace the double colon.

XXXX:XXXX:XXXX:XXXX:XXXX:XXXX:XXXX:XXXX

0: 15: : 1: 12:1213

This means that the original address is.

0000:0015:0000:0000:0000:0001:0012:1213

Table 19.5 *Type prefixes for IPv6 addresses*

Type Prefix	Type	Fraction
0000 0000	Reserved	1/256
0000 0001	Unassigned	1/256
0000 001	ISO network addresses	1/128
0000 010	IPX (Novell) network addresses	1/128
0000 011	Unassigned	1/128
0000 1	Unassigned	1/32
0001	Reserved	1/16
001	Reserved	1/8
010	Provider-based unicast addresses	1/8

Table 19.5 Type prefixes for IPv6 addresses (continued)

Type Prefix	Type	Fraction
011	Unassigned	1/8
100	Geographic-based unicast addresses	1/8
101	Unassigned	1/8
110	Unassigned	1/8
1110	Unassigned	1/16
1111 0	Unassigned	1/32
1111 10	Unassigned	1/64
1111 110	Unassigned	1/128
1111 1110 0	Unassigned	1/512
1111 1110 10	Link local addresses	1/1024
1111 1110 11	Site local addresses	1/1024
1111 1111	Multicast addresses	1/256

Figure 19.16 Prefixes for provider-based unicast address

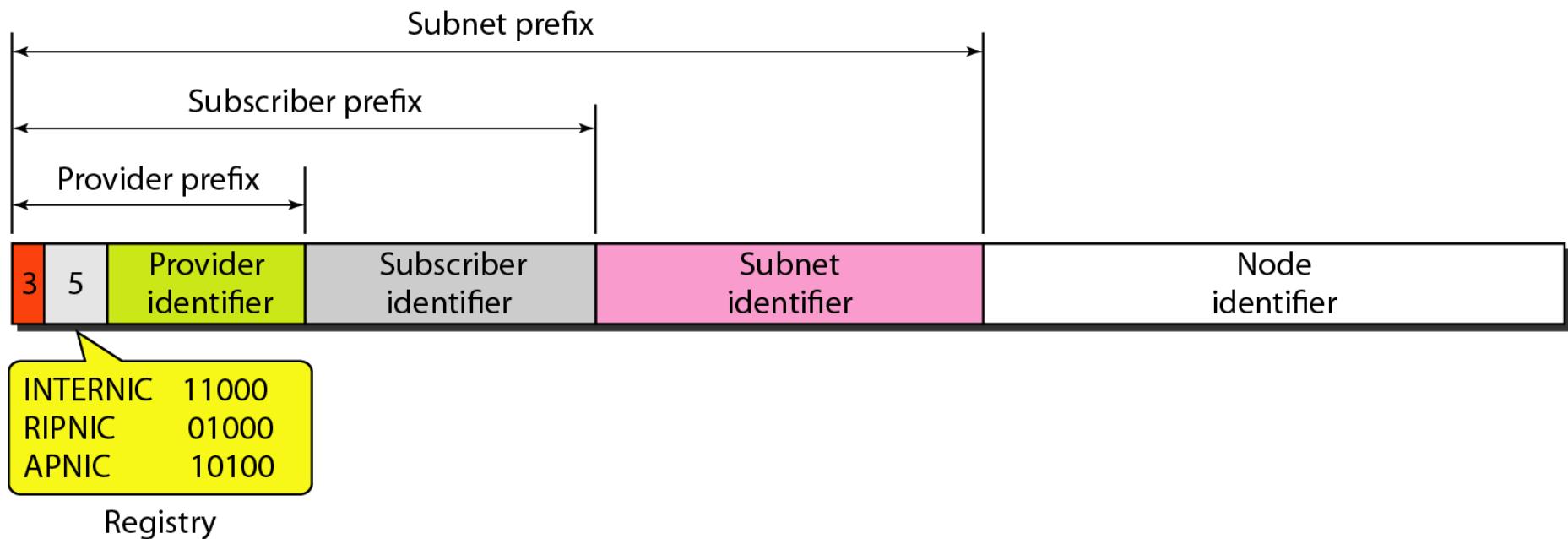


Figure 19.17 Multicast address in IPv6

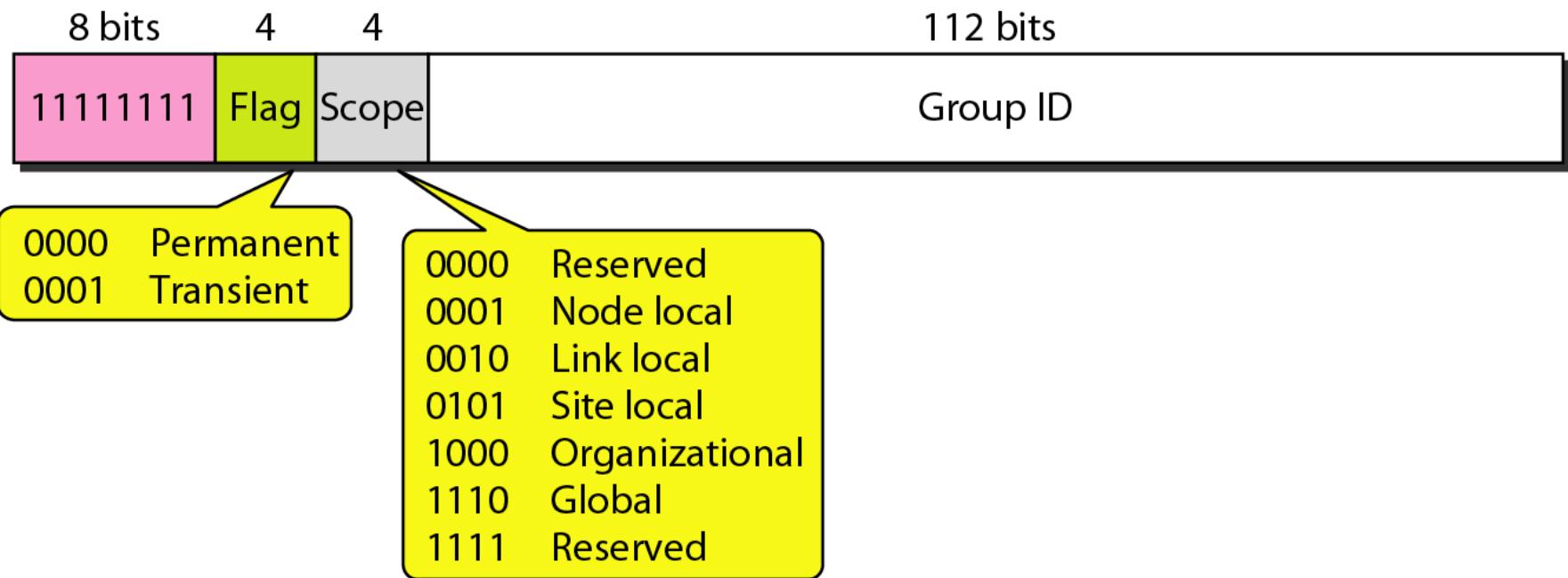


Figure 19.18 *Reserved addresses in IPv6*

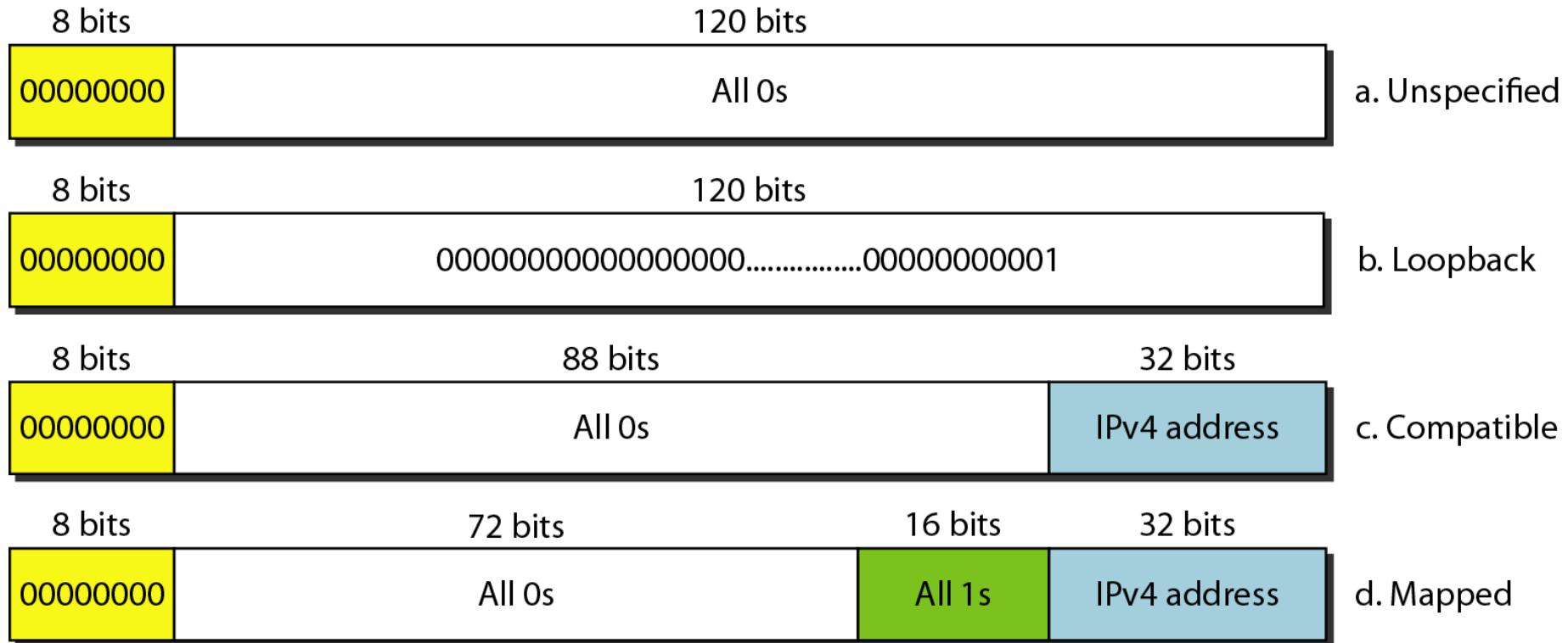


Figure 19.19 Local addresses in IPv6

10 bits

1111111010

70 bits

All 0s

48 bits

Node address

a. Link local

10 bits

1111111011

38 bits

All 0s

32 bits

Subnet
address

48 bits

Node address

b. Site local



**Data Communications
and Networking**

Fourth Edition

Forouzan

Chapter 20

Network Layer: Internet Protocol

20-1 INTERNETWORKING

In this section, we discuss internetworking, connecting networks together to make an internetwork or an internet.

Topics discussed in this section:

Need for Network Layer

Internet as a Datagram Network

Internet as a Connectionless Network

Figure 20.1 *Links between two hosts*

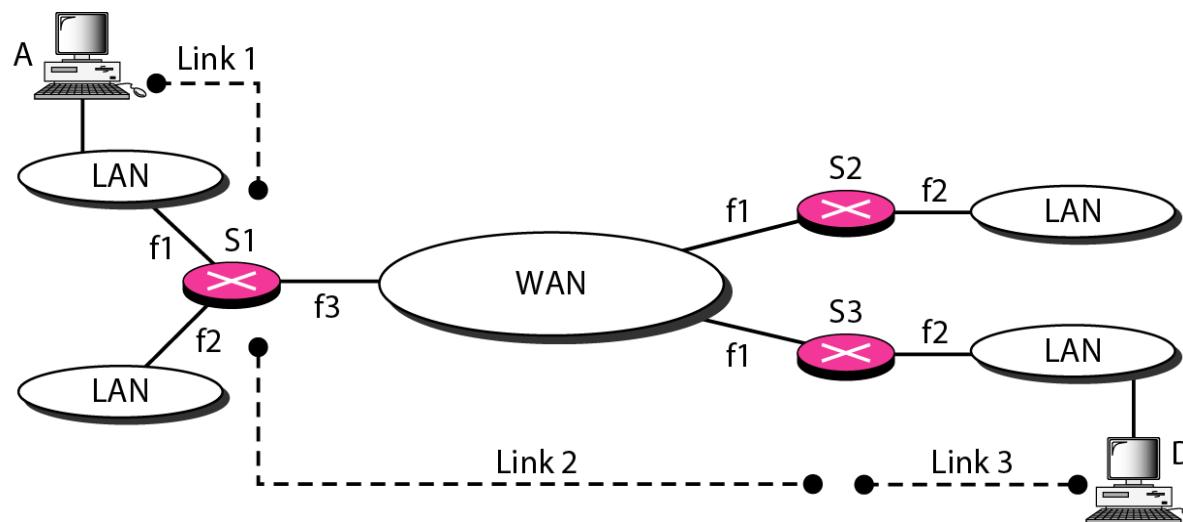
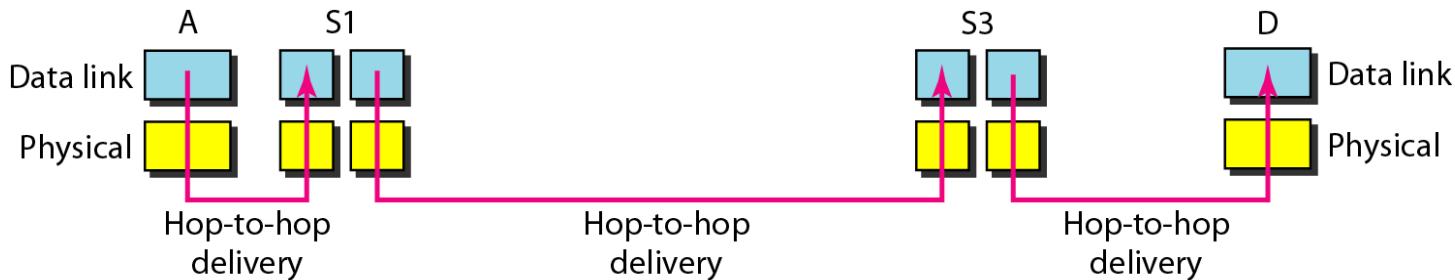


Figure 20.2 Network layer in an internetwork

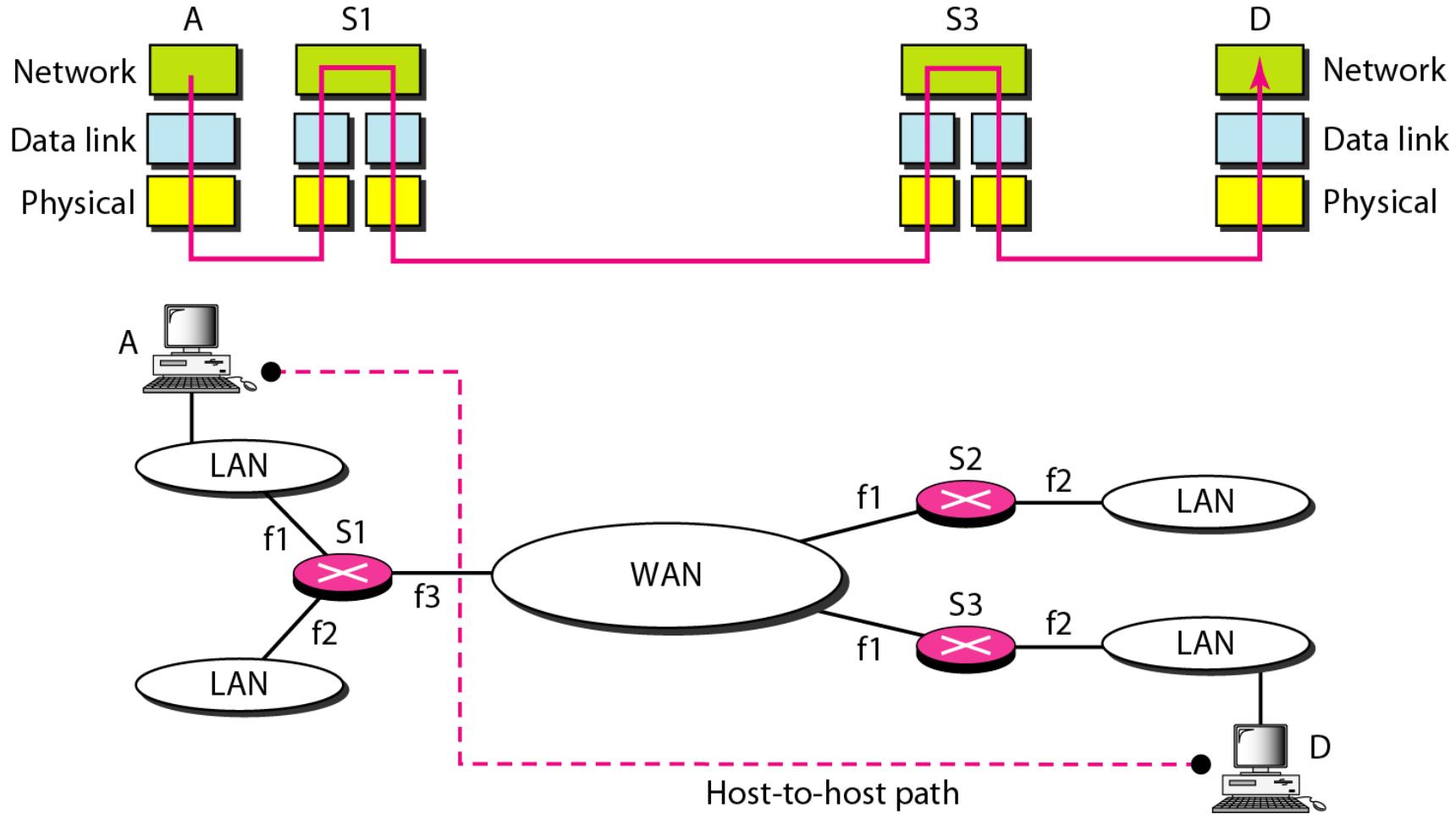
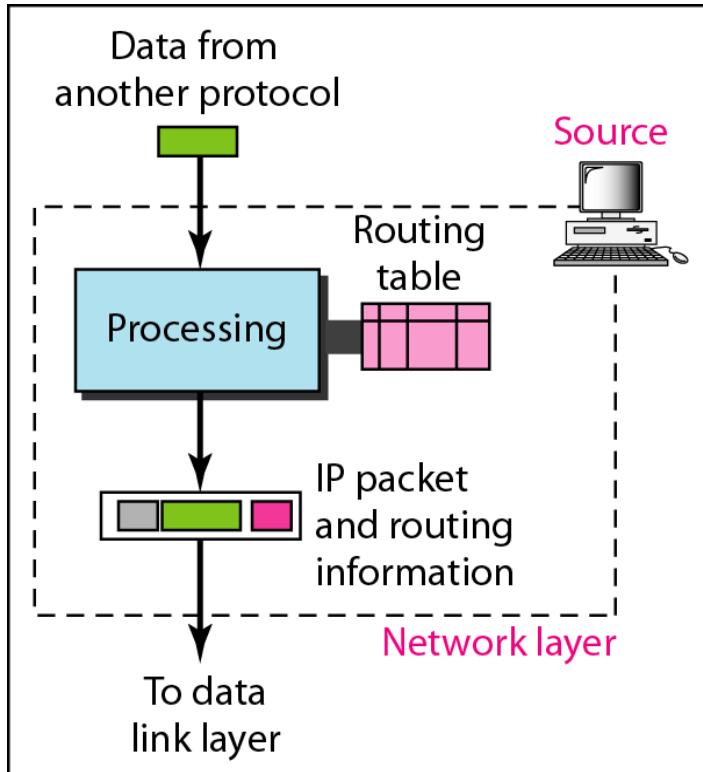
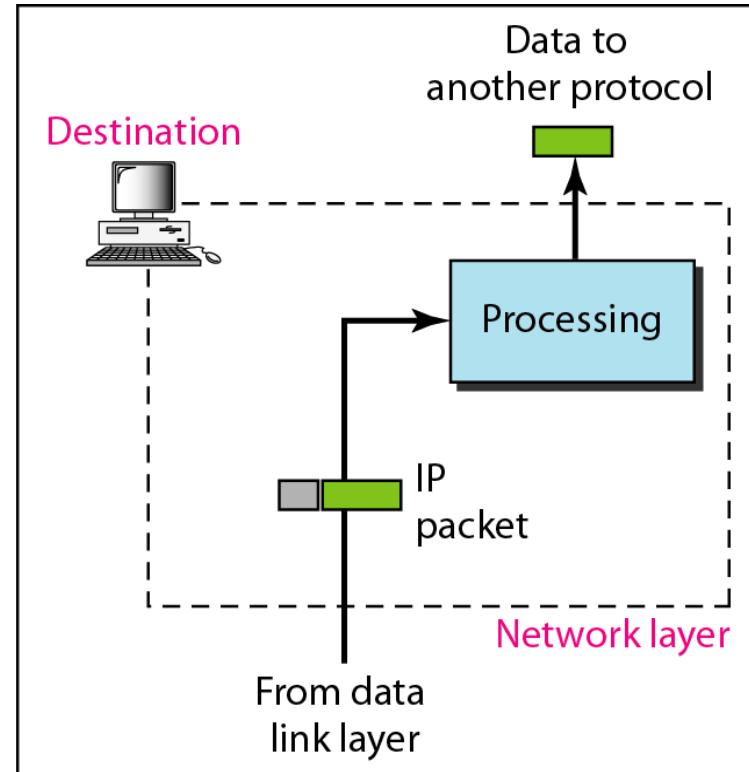


Figure 20.3 Network layer at the source, router, and destination

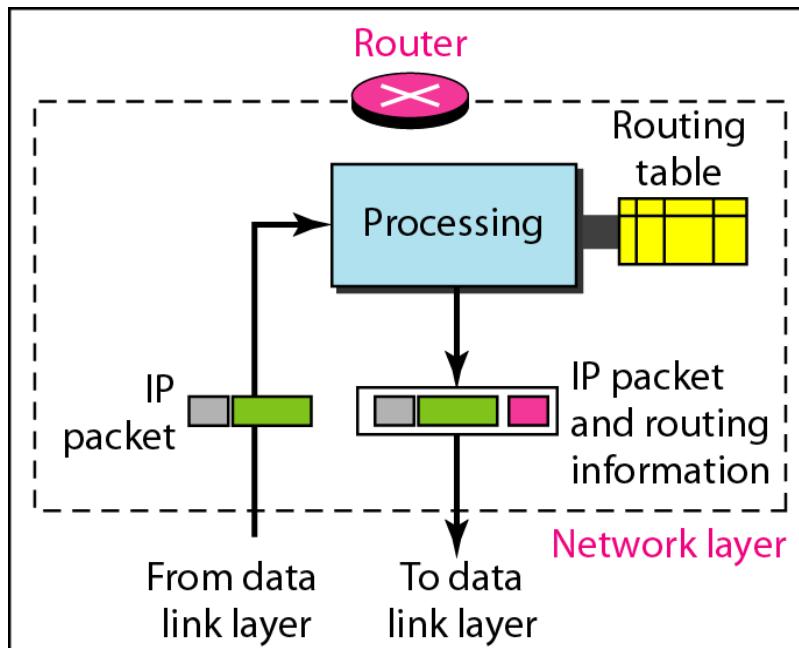


a. Network layer at source

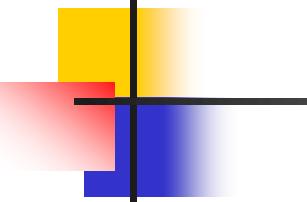


b. Network layer at destination

Figure 20.3 Network layer at the source, router, and destination (continued)



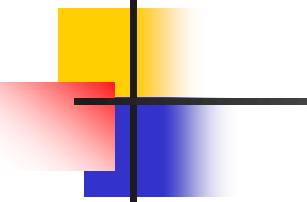
c. Network layer at a router



Note

Switching at the network layer in the Internet uses the datagram approach to packet switching.

It uses the universal addresses defined in the network layer to route the packets from source to the destination



Note

**Communication at the network layer in
the Internet is connectionless.**

20-2 IPv4

*The Internet Protocol version 4 (**IPv4**) is the delivery mechanism used by the TCP/IP protocols.*

Topics discussed in this section:

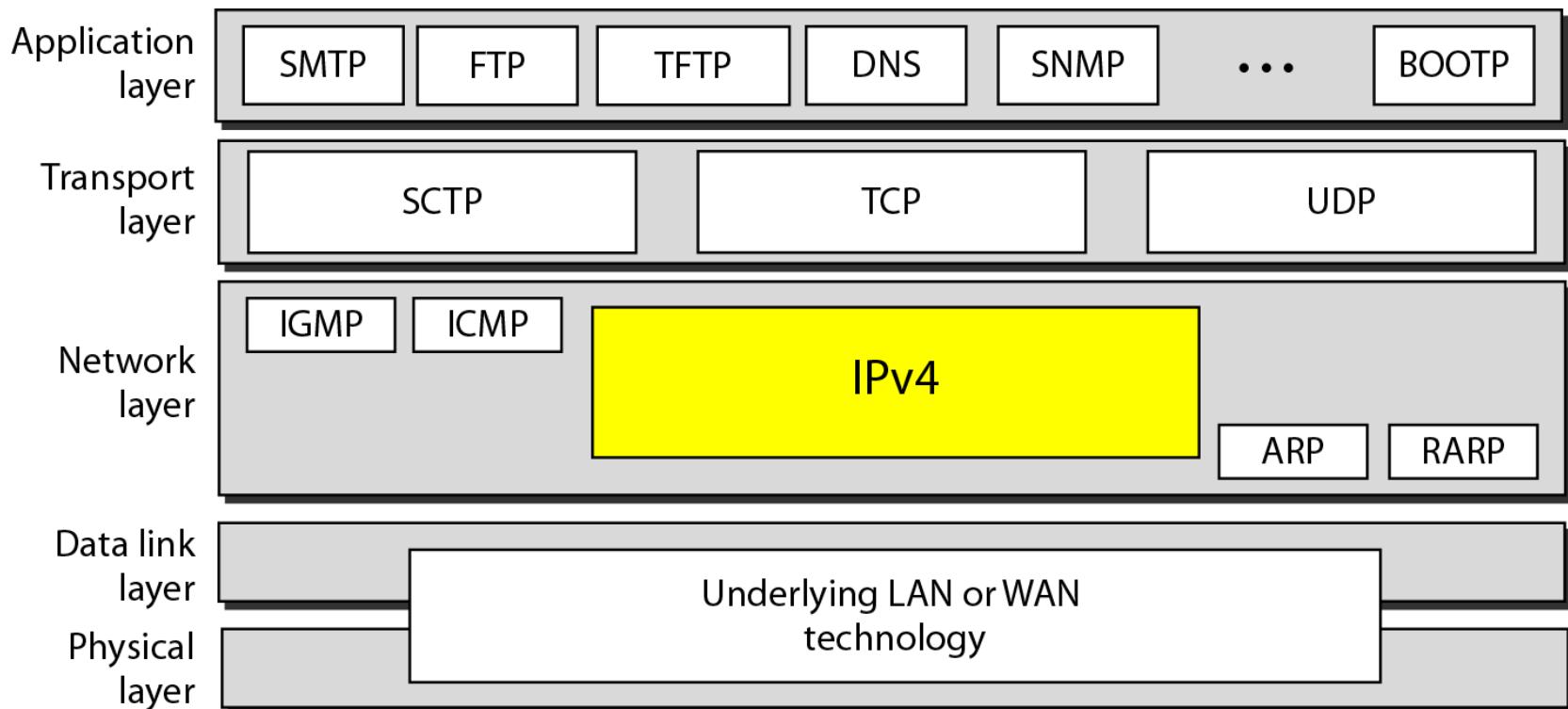
Datagram

Fragmentation

Checksum

Options

Figure 20.4 Position of IPv4 in TCP/IP protocol suite



IPv4 is an unreliable and connectionless protocol- a best effort delivery service

Figure 20.5 IPv4 datagram format

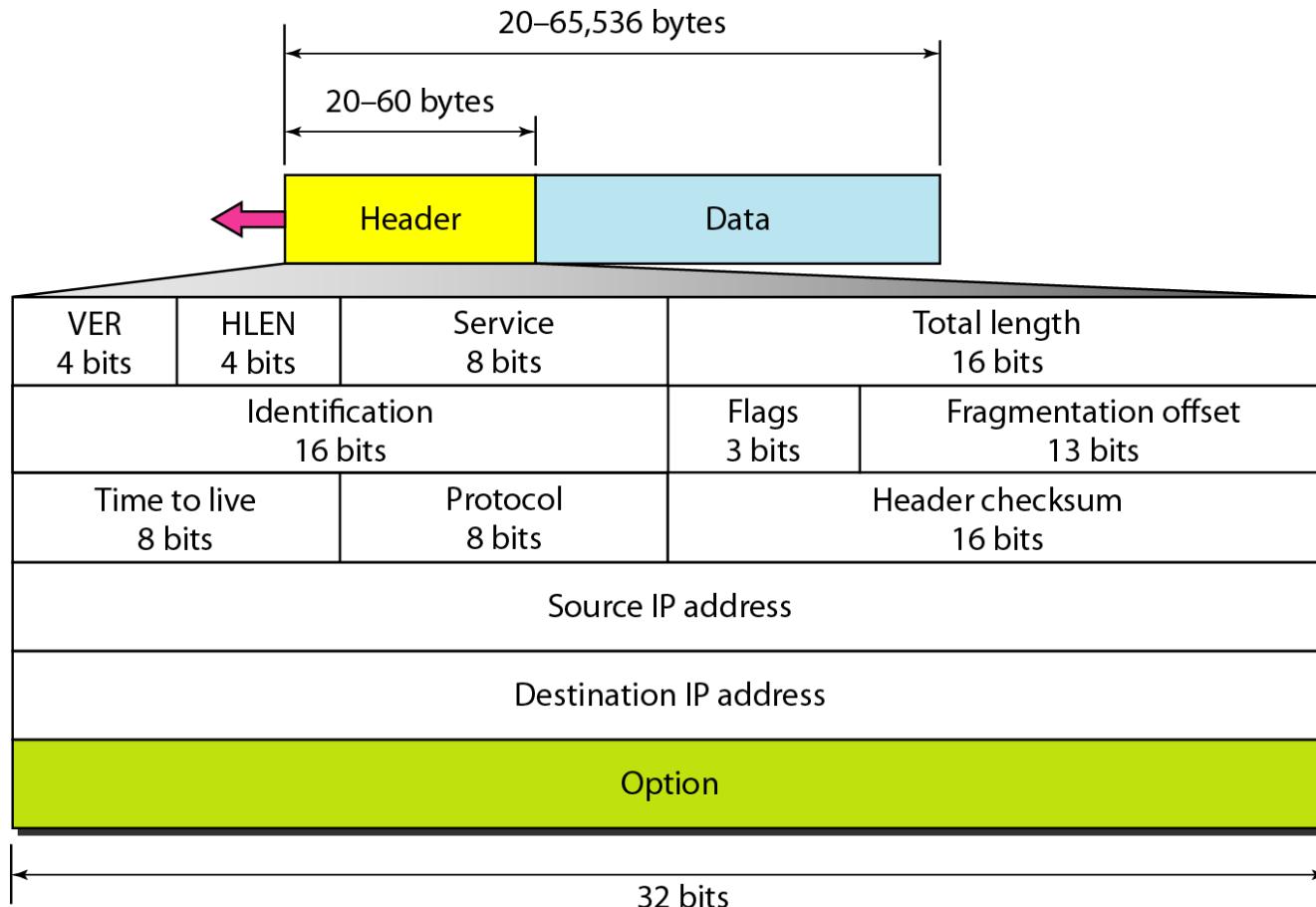
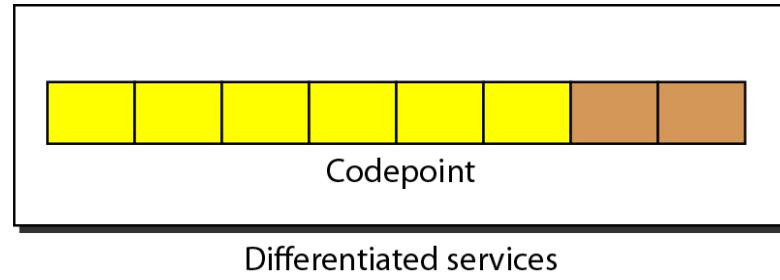
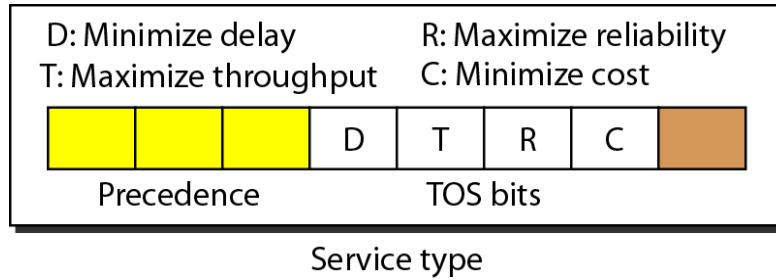
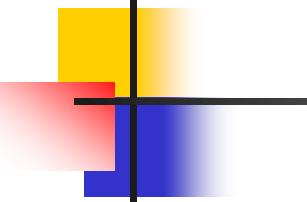


Figure 20.6 *Service type or differentiated services*





Note

The precedence subfield was part of version 4, but never used.

Table 20.1 *Types of service*

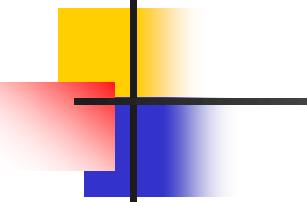
<i>TOS Bits</i>	<i>Description</i>
0000	Normal (default)
0001	Minimize cost
0010	Maximize reliability
0100	Maximize throughput
1000	Minimize delay

Table 20.2 *Default types of service*

<i>Protocol</i>	<i>TOS Bits</i>	<i>Description</i>
ICMP	0000	Normal
BOOTP	0000	Normal
NNTP	0001	Minimize cost
IGP	0010	Maximize reliability
SNMP	0010	Maximize reliability
TELNET	1000	Minimize delay
FTP (data)	0100	Maximize throughput
FTP (control)	1000	Minimize delay
TFTP	1000	Minimize delay
SMTP (command)	1000	Minimize delay
SMTP (data)	0100	Maximize throughput
DNS (UDP query)	1000	Minimize delay
DNS (TCP query)	0000	Normal
DNS (zone)	0100	Maximize throughput

Table 20.3 *Values for codepoints*

<i>Value</i>	<i>Protocol</i>
1	ICMP
2	IGMP
6	TCP
17	UDP
89	OSPF



Note

The total length field defines the total length of the datagram including the header.

Figure 20.7 *Encapsulation of a small datagram in an Ethernet frame*

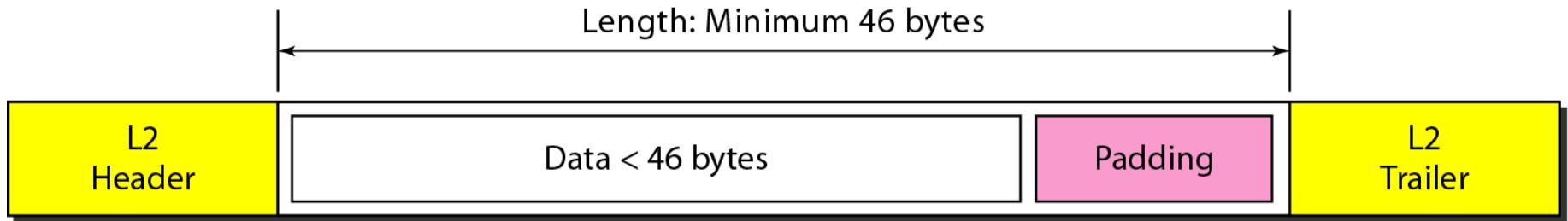


Figure 20.8 *Protocol field and encapsulated data*

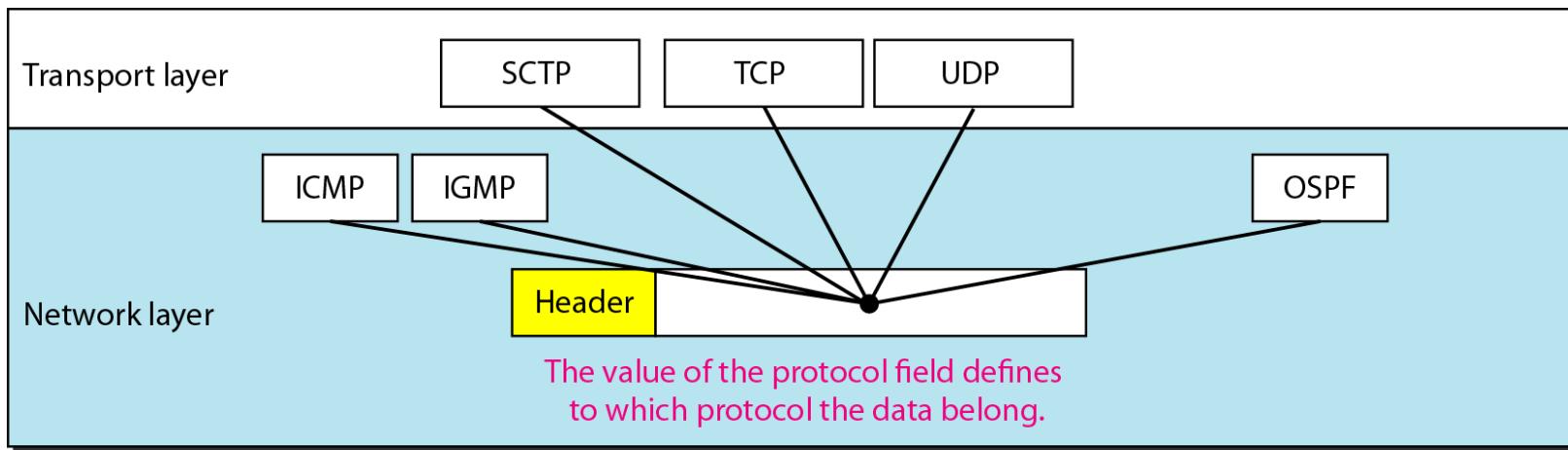


Table 20.4 *Protocol values*

<i>Value</i>	<i>Protocol</i>
1	ICMP
2	IGMP
6	TCP
17	UDP
89	OSPF

Example 20.1

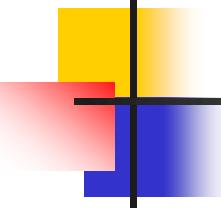
An IPv4 packet has arrived with the first 8 bits as shown:

01000010

The receiver discards the packet. Why?

Solution

There is an error in this packet. The 4 leftmost bits (0100) show the version, which is correct. The next 4 bits (0010) show an invalid header length ($2 \times 4 = 8$). The minimum number of bytes in the header must be 20. The packet has been corrupted in transmission.

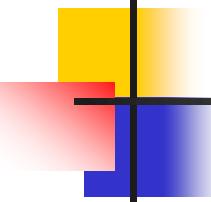


Example 20.2

In an IPv4 packet, the value of HLEN is 1000 in binary. How many bytes of options are being carried by this packet?

Solution

The HLEN value is 8, which means the total number of bytes in the header is 8×4 , or 32 bytes. The first 20 bytes are the base header, the next 12 bytes are the options.

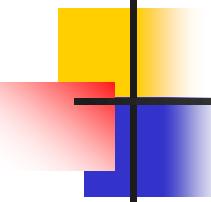


Example 20.3

In an IPv4 packet, the value of HLEN is 5, and the value of the total length field is 0x0028. How many bytes of data are being carried by this packet?

Solution

The HLEN value is 5, which means the total number of bytes in the header is 5×4 , or 20 bytes (no options). The total length is 40 bytes, which means the packet is carrying 20 bytes of data ($40 - 20$).



Example 20.4

An IPv4 packet has arrived with the first few hexadecimal digits as shown.

0x45000028000100000102 ...

How many hops can this packet travel before being dropped? The data belong to what upper-layer protocol?

Solution

To find the time-to-live field, we skip 8 bytes. The time-to-live field is the ninth byte, which is 01. This means the packet can travel only one hop. The protocol field is the next byte (02), which means that the upper-layer protocol is IGMP.

Figure 20.9 *Maximum transfer unit (MTU)*

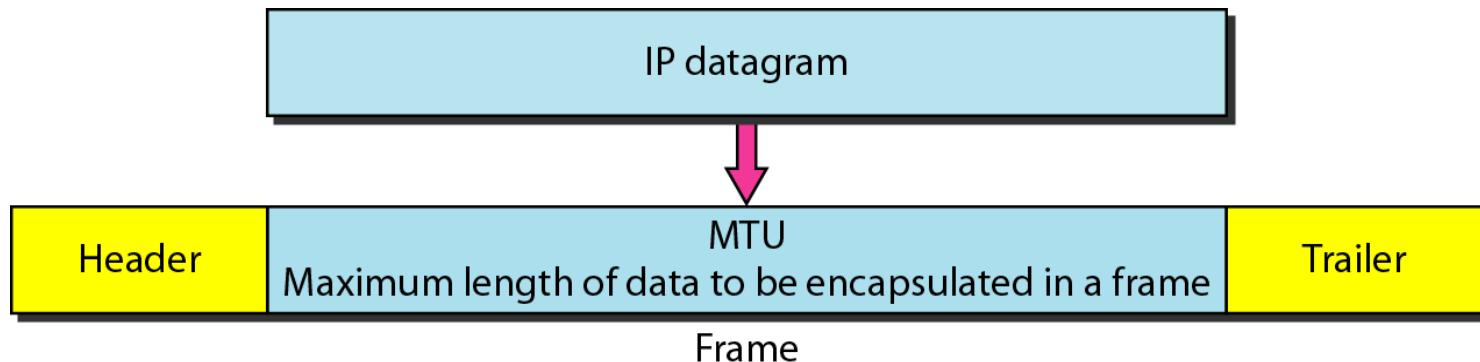


Table 20.5 *MTUs for some networks*

<i>Protocol</i>	<i>MTU</i>
Hyperchannel	65,535
Token Ring (16 Mbps)	17,914
Token Ring (4 Mbps)	4,464
FDDI	4,352
Ethernet	1,500
X.25	576
PPP	296

Figure 20.10 *Flags used in fragmentation*



Figure 20.11 *Fragmentation example*

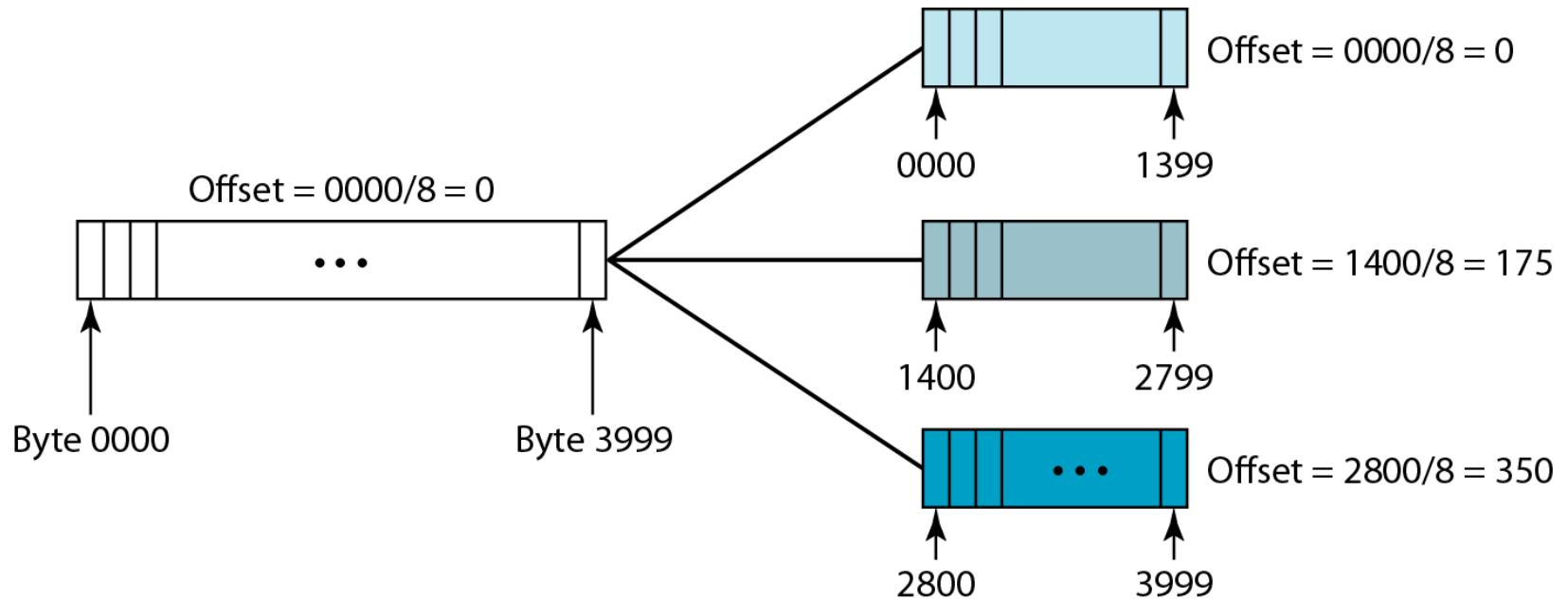
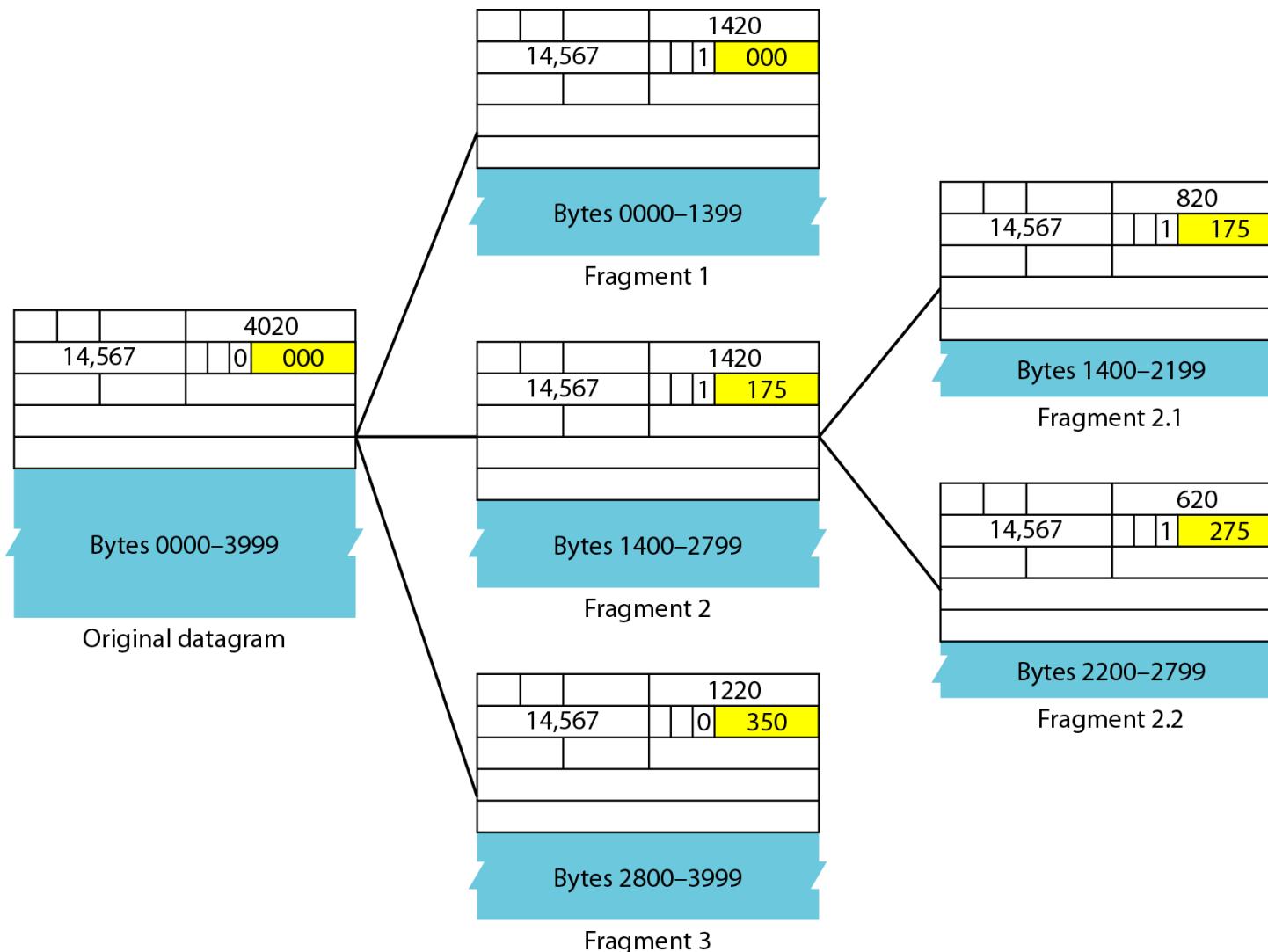
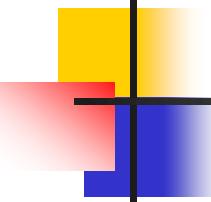


Figure 20.12 Detailed fragmentation example



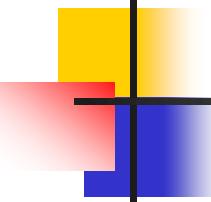


Example 20.5

A packet has arrived with an M bit value of 0. Is this the first fragment, the last fragment, or a middle fragment? Do we know if the packet was fragmented?

Solution

If the M bit is 0, it means that there are no more fragments; the fragment is the last one. However, we cannot say if the original packet was fragmented or not. A non-fragmented packet is considered the last fragment.

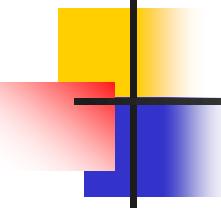


Example 20.6

A packet has arrived with an M bit value of 1. Is this the first fragment, the last fragment, or a middle fragment? Do we know if the packet was fragmented?

Solution

If the M bit is 1, it means that there is at least one more fragment. This fragment can be the first one or a middle one, but not the last one. We don't know if it is the first one or a middle one; we need more information (the value of the fragmentation offset).

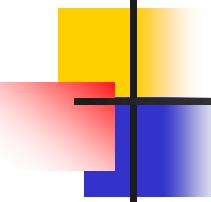


Example 20.7

A packet has arrived with an M bit value of 1 and a fragmentation offset value of 0. Is this the first fragment, the last fragment, or a middle fragment?

Solution

Because the M bit is 1, it is either the first fragment or a middle one. Because the offset value is 0, it is the first fragment.

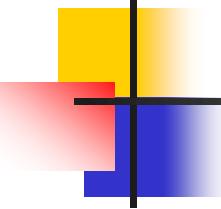


Example 20.8

A packet has arrived in which the offset value is 100. What is the number of the first byte? Do we know the number of the last byte?

Solution

To find the number of the first byte, we multiply the offset value by 8. This means that the first byte number is 800. We cannot determine the number of the last byte unless we know the length.



Example 20.9

A packet has arrived in which the offset value is 100, the value of HLEN is 5, and the value of the total length field is 100. What are the numbers of the first byte and the last byte?

Solution

The first byte number is $100 \times 8 = 800$. The total length is 100 bytes, and the header length is 20 bytes (5×4), which means that there are 80 bytes in this datagram. If the first byte number is 800, the last byte number must be 879.

Example 20.10

Figure 20.13 shows an example of a checksum calculation for an IPv4 header without options. The header is divided into 16-bit sections. All the sections are added and the sum is complemented. The result is inserted in the checksum field.

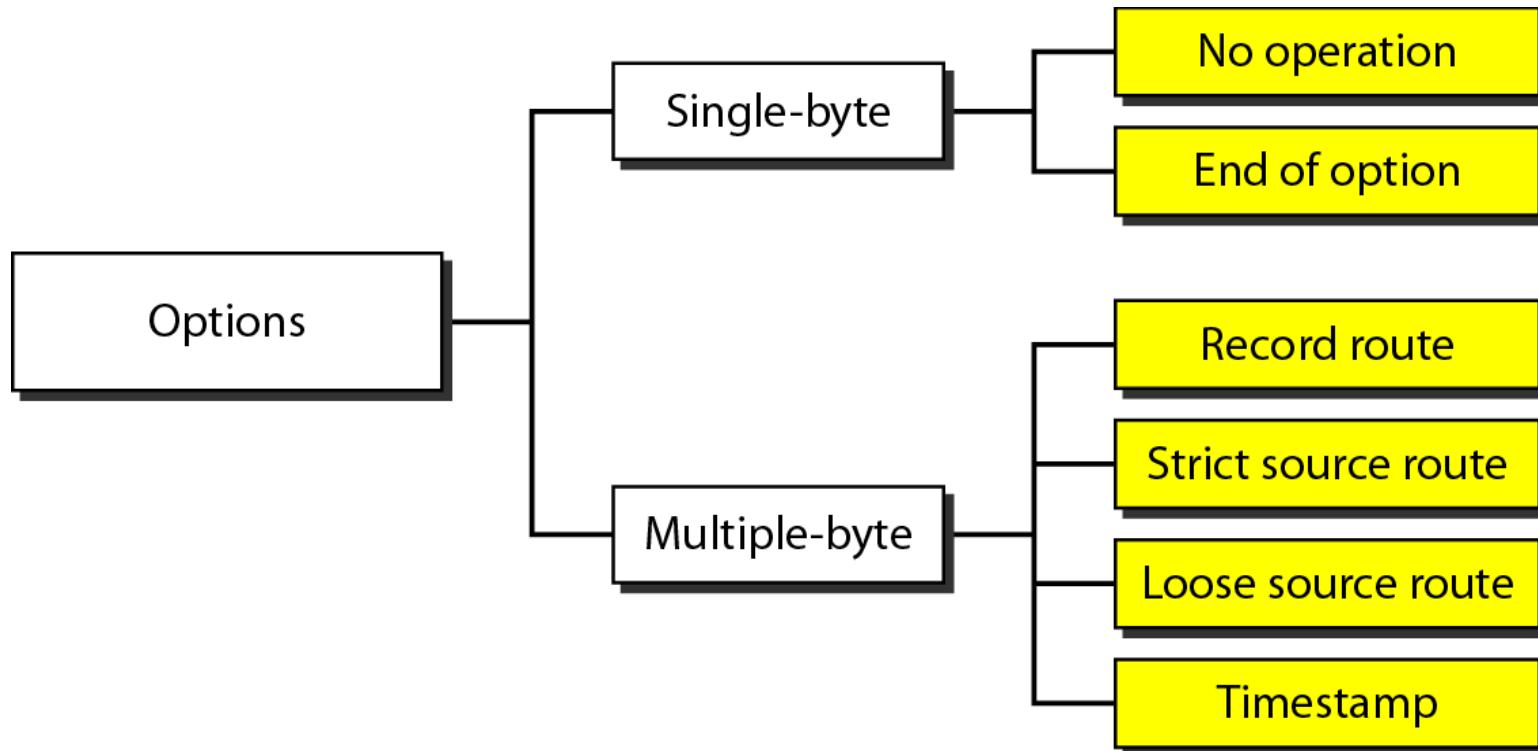
Figure 20.13 Example of checksum calculation in IPv4

4	5	0	28								
1			0	0							
4	17	0				↑					
10.12.14.5											
12.6.7.9											

4, 5, and 0 → 4 5 0 0
28 → 0 0 1 C
1 → 0 0 0 1
0 and 0 → 0 0 0 0
4 and 17 → 0 4 1 1
0 → 0 0 0 0
10.12 → 0 A 0 C
14.5 → 0 E 0 5
12.6 → 0 C 0 6
7.9 → 0 7 0 9

Sum → 7 4 4 E
Checksum → 8 B B 1

Figure 20.14 *Taxonomy of options in IPv4*



20-3 IPv6

The network layer protocol in the TCP/IP protocol suite is currently IPv4. Although IPv4 is well designed, data communication has evolved since the inception of IPv4 in the 1970s. IPv4 has some deficiencies that make it unsuitable for the fast-growing Internet.

Topics discussed in this section:

Advantages

Packet Format

Extension Headers

Figure 20.15 IPv6 datagram header and payload

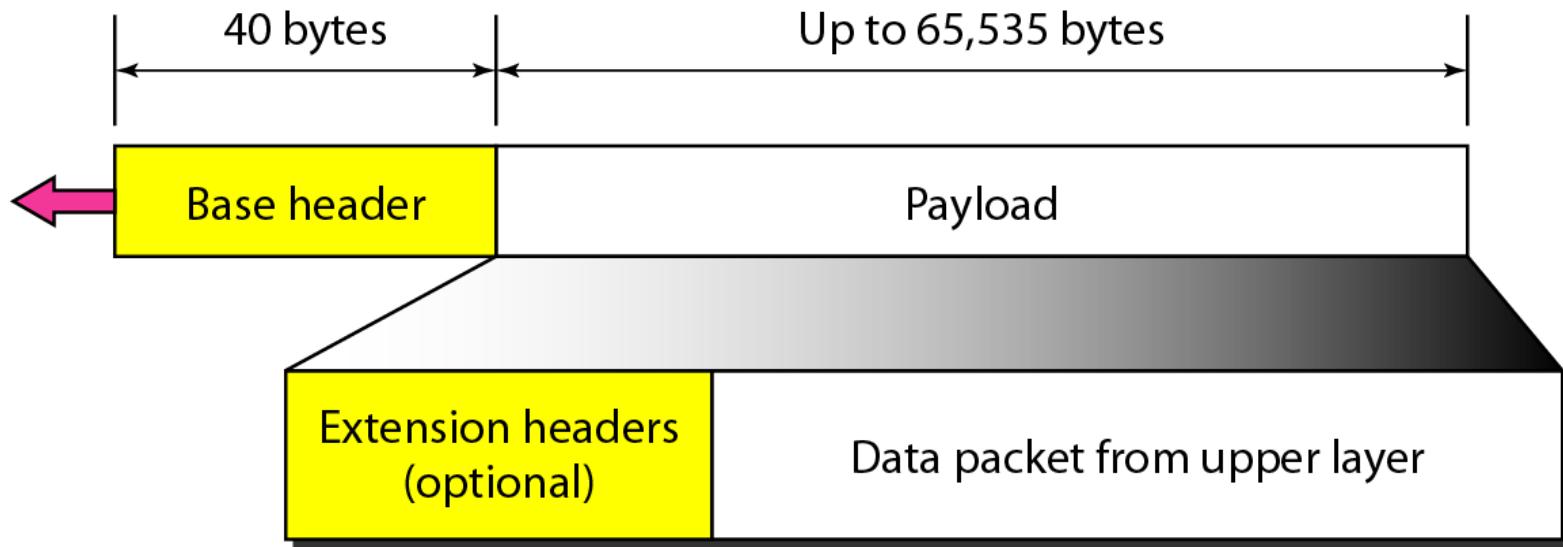


Figure 20.16 Format of an IPv6 datagram

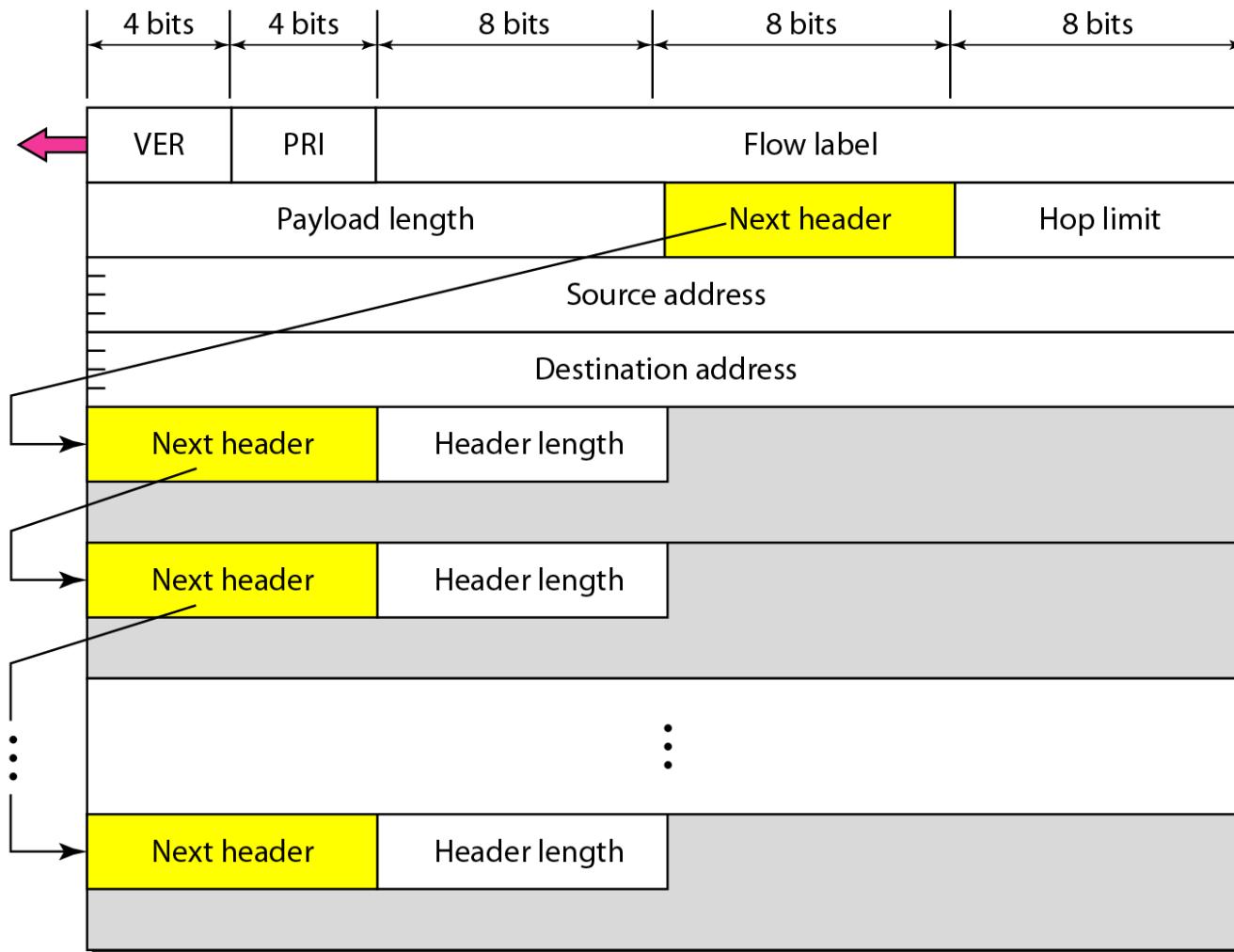


Table 20.6 *Next header codes for IPv6*

<i>Code</i>	<i>Next Header</i>
0	Hop-by-hop option
2	ICMP
6	TCP
17	UDP
43	Source routing
44	Fragmentation
50	Encrypted security payload
51	Authentication
59	Null (no next header)
60	Destination option

Table 20.7 *Priorities for congestion-controlled traffic*

<i>Priority</i>	<i>Meaning</i>
0	No specific traffic
1	Background data
2	Unattended data traffic
3	Reserved
4	Attended bulk data traffic
5	Reserved
6	Interactive traffic
7	Control traffic

Table 20.8 *Priorities for noncongestion-controlled traffic*

<i>Priority</i>	<i>Meaning</i>
8	Data with greatest redundancy
...	...
15	Data with least redundancy

Table 20.9 *Comparison between IPv4 and IPv6 packet headers*

<i>Comparison</i>
1. The header length field is eliminated in IPv6 because the length of the header is fixed in this version.
2. The service type field is eliminated in IPv6. The priority and flow label fields together take over the function of the service type field.
3. The total length field is eliminated in IPv6 and replaced by the payload length field.
4. The identification, flag, and offset fields are eliminated from the base header in IPv6. They are included in the fragmentation extension header.
5. The TTL field is called hop limit in IPv6.
6. The protocol field is replaced by the next header field.
7. The header checksum is eliminated because the checksum is provided by upper-layer protocols; it is therefore not needed at this level.
8. The option fields in IPv4 are implemented as extension headers in IPv6.

Figure 20.17 *Extension header types*

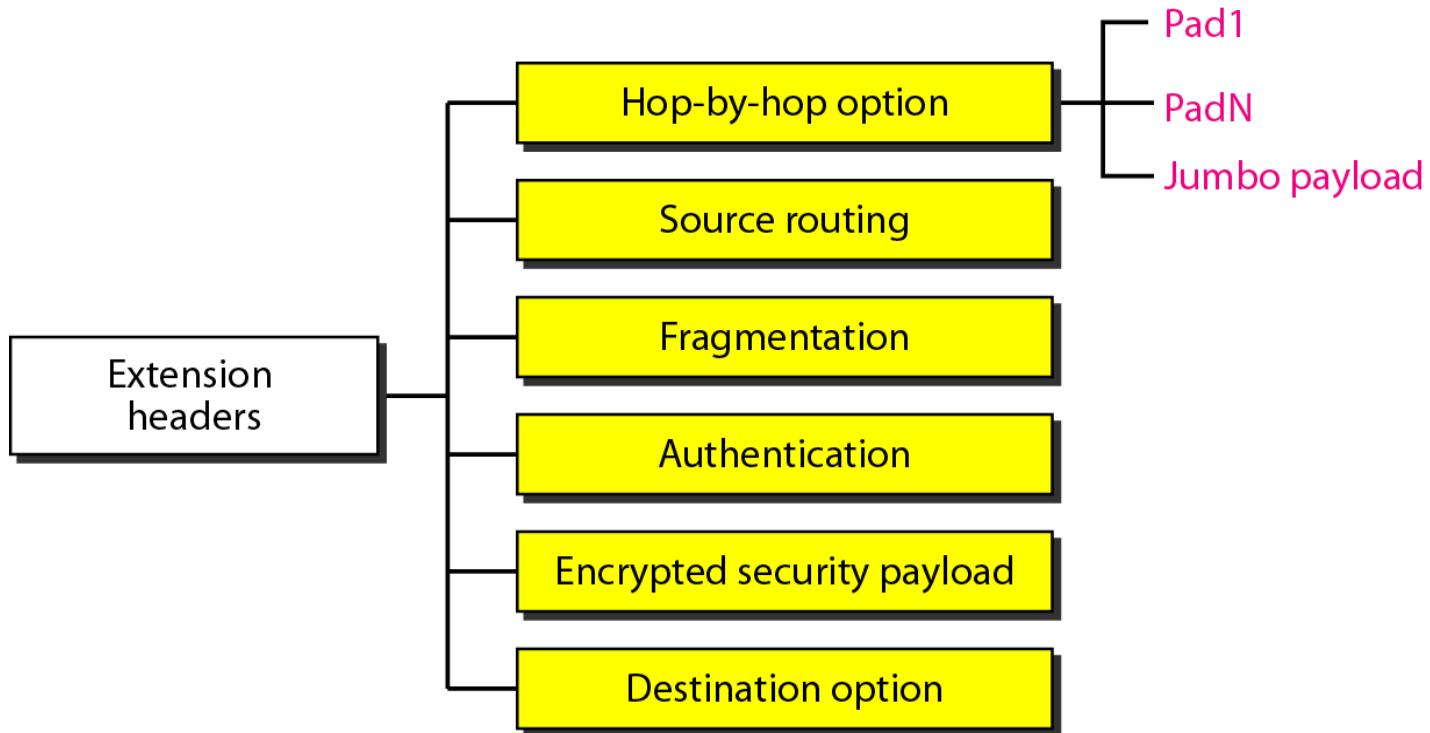


Table 20.10 *Comparison between IPv4 options and IPv6 extension headers*

<i>Comparison</i>
1. The no-operation and end-of-option options in IPv4 are replaced by Pad1 and PadN options in IPv6.
2. The record route option is not implemented in IPv6 because it was not used.
3. The timestamp option is not implemented because it was not used.
4. The source route option is called the source route extension header in IPv6.
5. The fragmentation fields in the base header section of IPv4 have moved to the fragmentation extension header in IPv6.
6. The authentication extension header is new in IPv6.
7. The encrypted security payload extension header is new in IPv6.

20-4 TRANSITION FROM IPv4 TO IPv6

Because of the huge number of systems on the Internet, the transition from IPv4 to IPv6 cannot happen suddenly. It takes a considerable amount of time before every system in the Internet can move from IPv4 to IPv6. The transition must be smooth to prevent any problems between IPv4 and IPv6 systems.

Topics discussed in this section:

Dual Stack

Tunneling

Header Translation

Figure 20.18 *Three transition strategies*

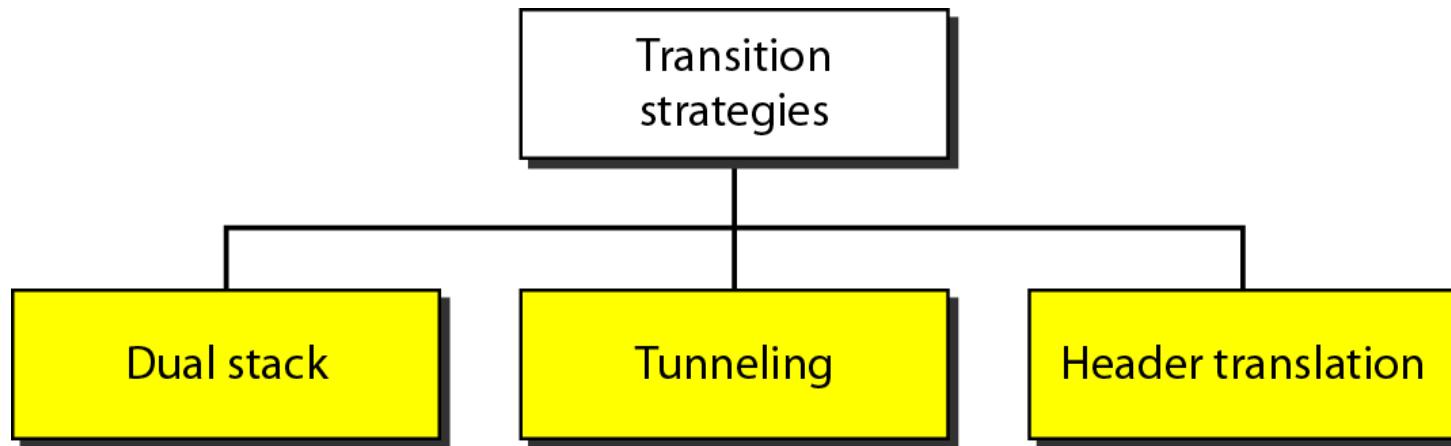


Figure 20.19 Dual stack

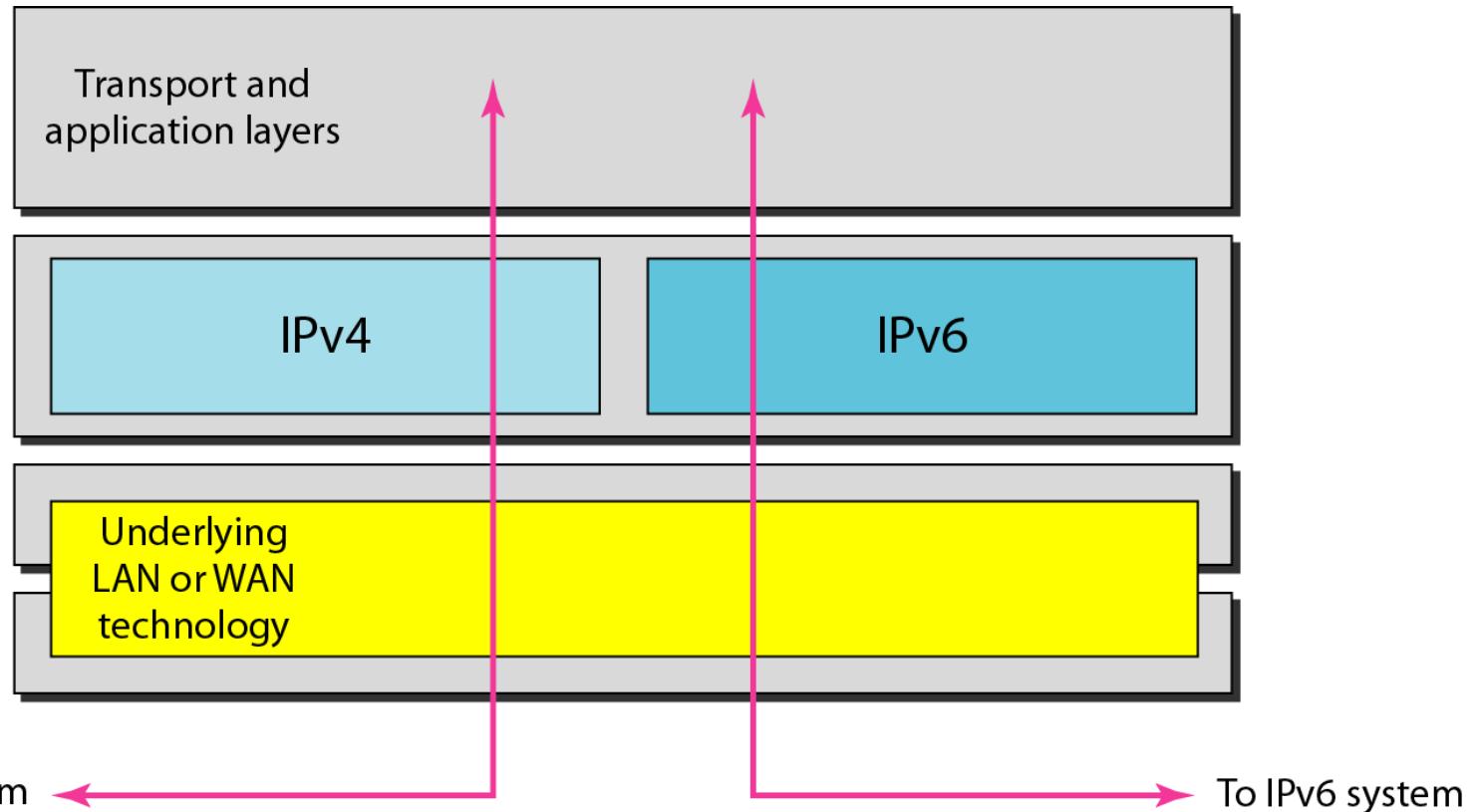


Figure 20.20 *Tunneling strategy*

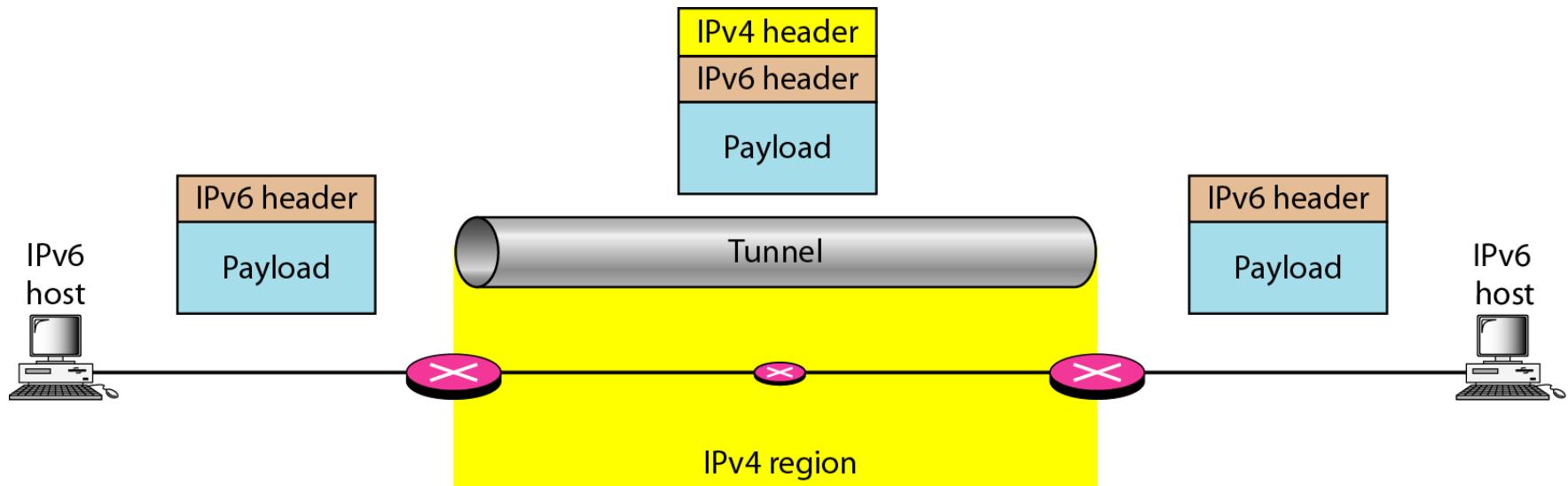


Figure 20.21 *Header translation strategy*

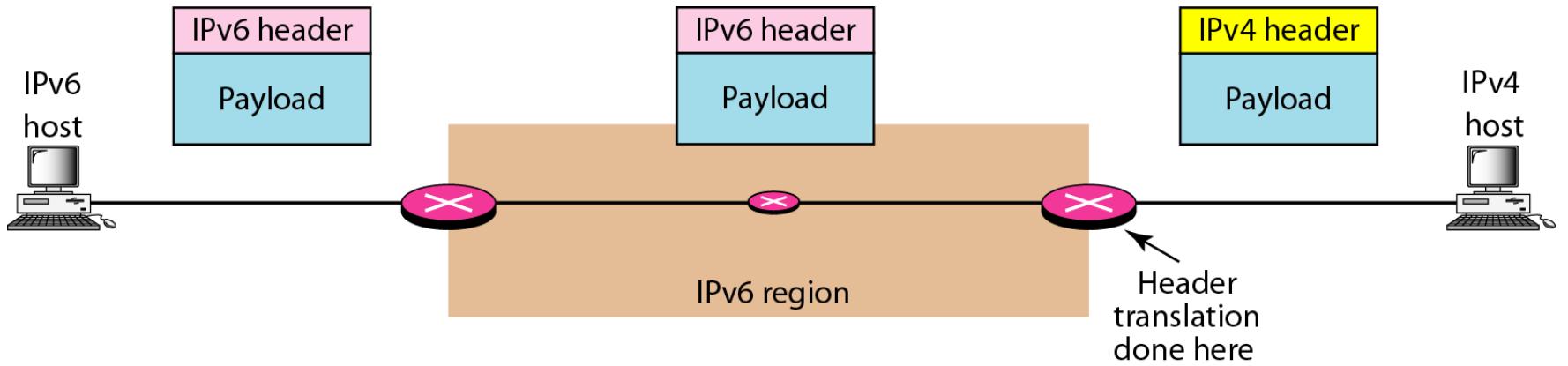


Table 20.11 *Header translation*

<i>Header Translation Procedure</i>
1. The IPv6 mapped address is changed to an IPv4 address by extracting the rightmost 32 bits.
2. The value of the IPv6 priority field is discarded.
3. The type of service field in IPv4 is set to zero.
4. The checksum for IPv4 is calculated and inserted in the corresponding field.
5. The IPv6 flow label is ignored.
6. Compatible extension headers are converted to options and inserted in the IPv4 header. Some may have to be dropped.
7. The length of IPv4 header is calculated and inserted into the corresponding field.
8. The total length of the IPv4 packet is calculated and inserted in the corresponding field.

Chapter 22

Network Layer: Delivery, Forwarding, and Routing

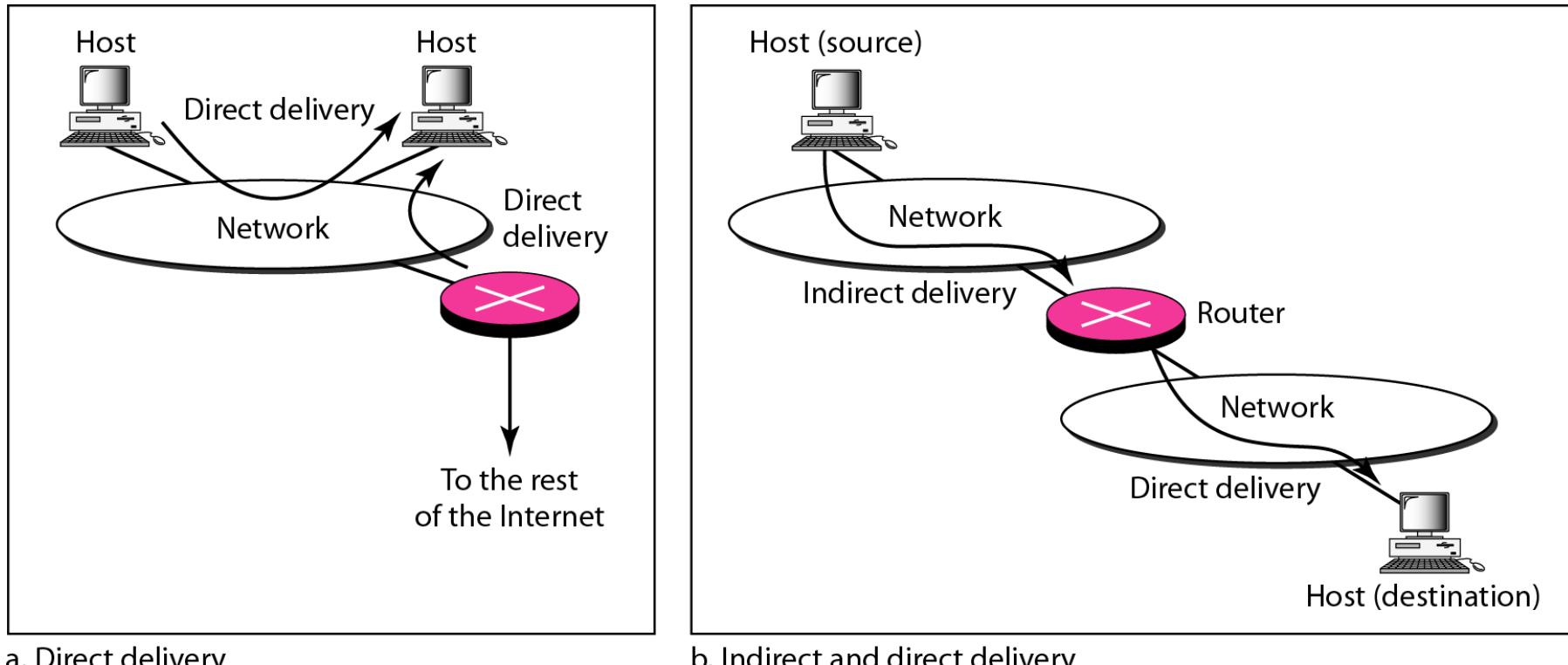
22-1 DELIVERY

The network layer supervises the handling of the packets by the underlying physical networks. We define this handling as the delivery of a packet.

Topics discussed in this section:

Direct Versus Indirect Delivery

Figure 22.1 Direct and indirect delivery



a. Direct delivery

b. Indirect and direct delivery

22-2 FORWARDING

Forwarding means to place the packet in its route to its destination. Forwarding requires a host or a router to have a routing table. When a host has a packet to send or when a router has received a packet to be forwarded, it looks at this table to find the route to the final destination.

Topics discussed in this section:

Forwarding Techniques

Forwarding Process

Routing Table

Figure 22.2 Route method versus next-hop method

a. Routing tables based on route

Destination	Route
Host B	R1, R2, host B

Routing table
for host A

Destination	Route
Host B	R2, host B

Routing table
for R1

Destination	Route
Host B	Host B

Routing table
for R2

b. Routing tables based on next hop

Destination	Next hop
Host B	R1

Destination	Next hop
Host B	R2

Destination	Next hop
Host B	---

Host A



Network

R1

Network

R2

Host B



Network

Figure 22.3 Host-specific versus network-specific method

Routing table for host S based on host-specific method

Destination	Next hop
A	R1
B	R1
C	R1
D	R1

Routing table for host S based on network-specific method

Destination	Next hop
N2	R1

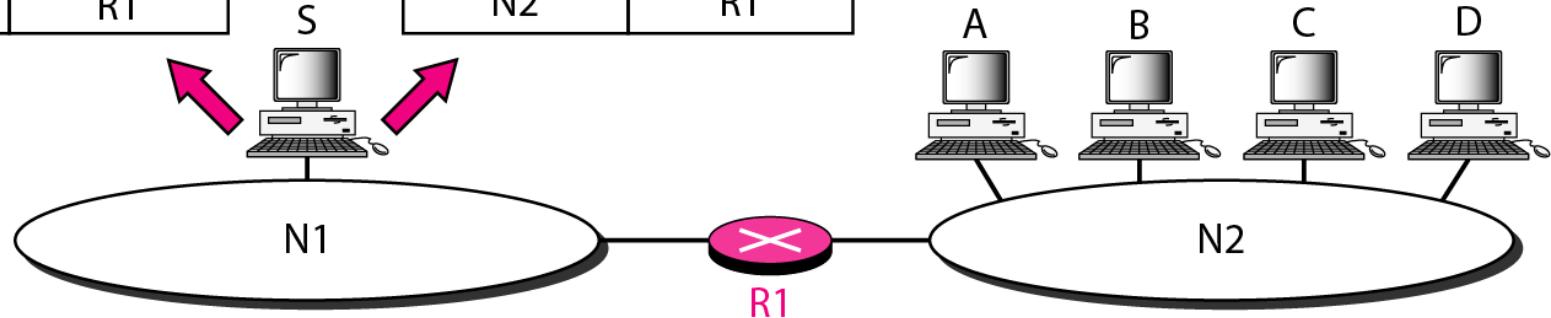


Figure 22.4 Default method

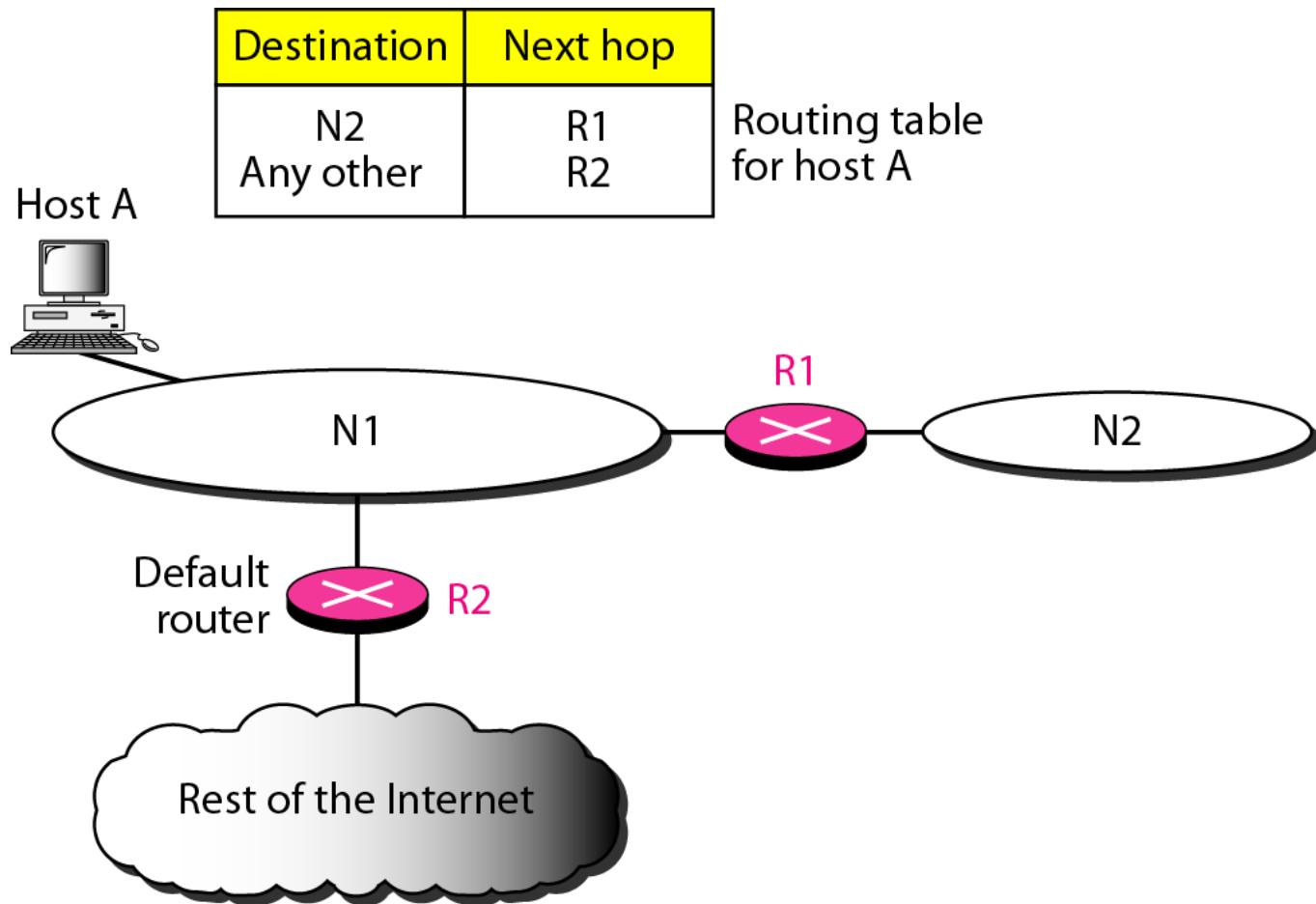
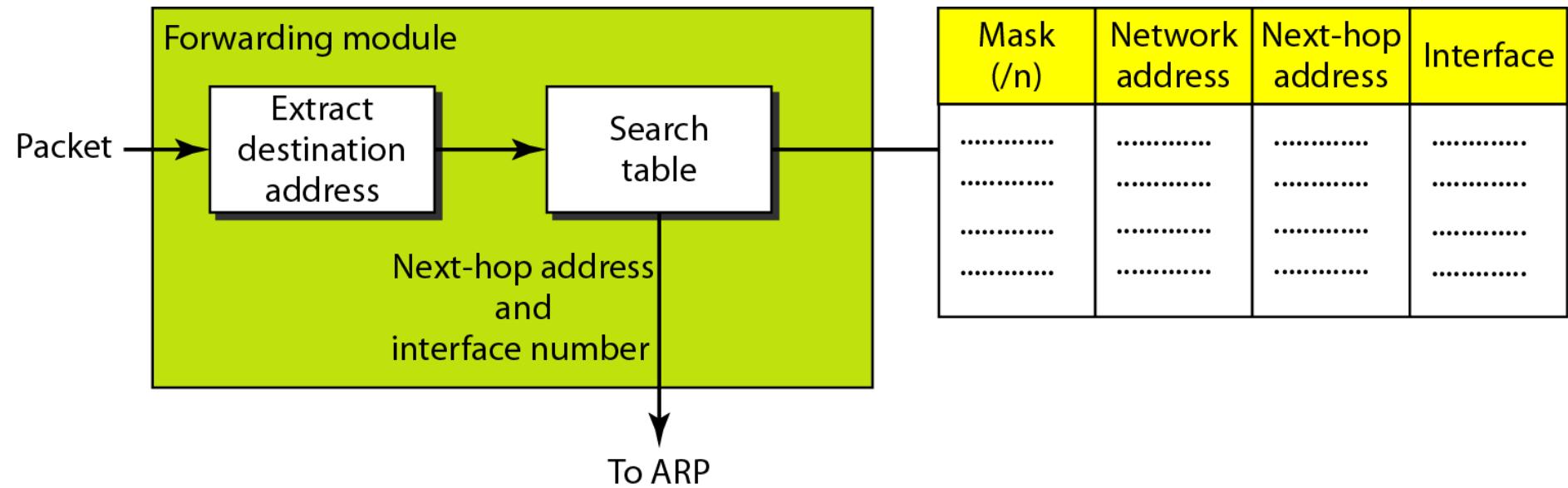
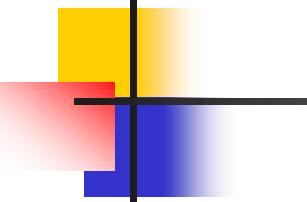


Figure 22.5 Simplified forwarding module in classless address





Note

In classless addressing, we need at least four columns in a routing table.

Example 22.1

Make a routing table for router R1, using the configuration in Figure 22.6.

Solution

Table 22.1 shows the corresponding table.

Figure 22.6 Configuration for Example 22.1

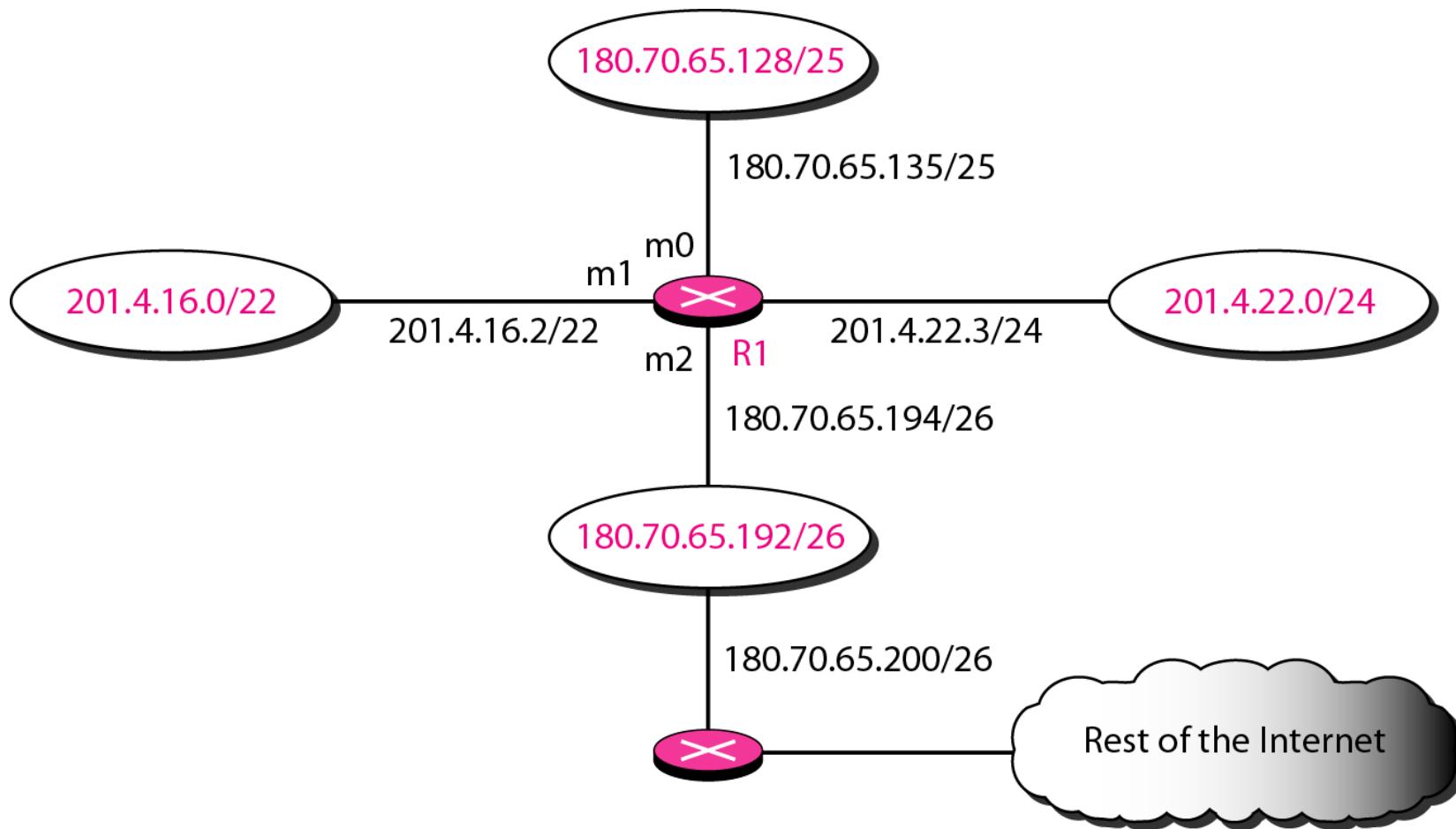


Table 22.1 *Routing table for router R1 in Figure 22.6*

<i>Mask</i>	<i>Network Address</i>	<i>Next Hop</i>	<i>Interface</i>
/26	180.70.65.192	—	m2
/25	180.70.65.128	—	m0
/24	201.4.22.0	—	m3
/22	201.4.16.0	m1
Any	Any	180.70.65.200	m2

Example 22.2

Show the forwarding process if a packet arrives at R1 in Figure 22.6 with the destination address 180.70.65.140.

Solution

The router performs the following steps:

- 1. The first mask (/26) is applied to the destination address.**
The result is 180.70.65.128, which does not match the corresponding network address.
- 2. The second mask (/25) is applied to the destination address.**
The result is 180.70.65.128, which matches the corresponding network address. The next-hop address and the interface number m0 are passed to ARP for further processing.

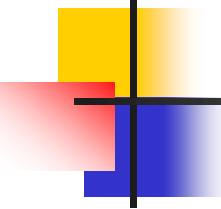
Example 22.3

Show the forwarding process if a packet arrives at R1 in Figure 22.6 with the destination address 201.4.22.35.

Solution

The router performs the following steps:

- 1. The first mask (/26) is applied to the destination address. The result is 201.4.22.0, which does not match the corresponding network address.**
- 2. The second mask (/25) is applied to the destination address. The result is 201.4.22.0, which does not match the corresponding network address (row 2).**



Example 22.3 (continued)

3. The third mask (/24) is applied to the destination address. The result is 201.4.22.0, which matches the corresponding network address. The destination address of the packet and the interface number m3 are passed to ARP.

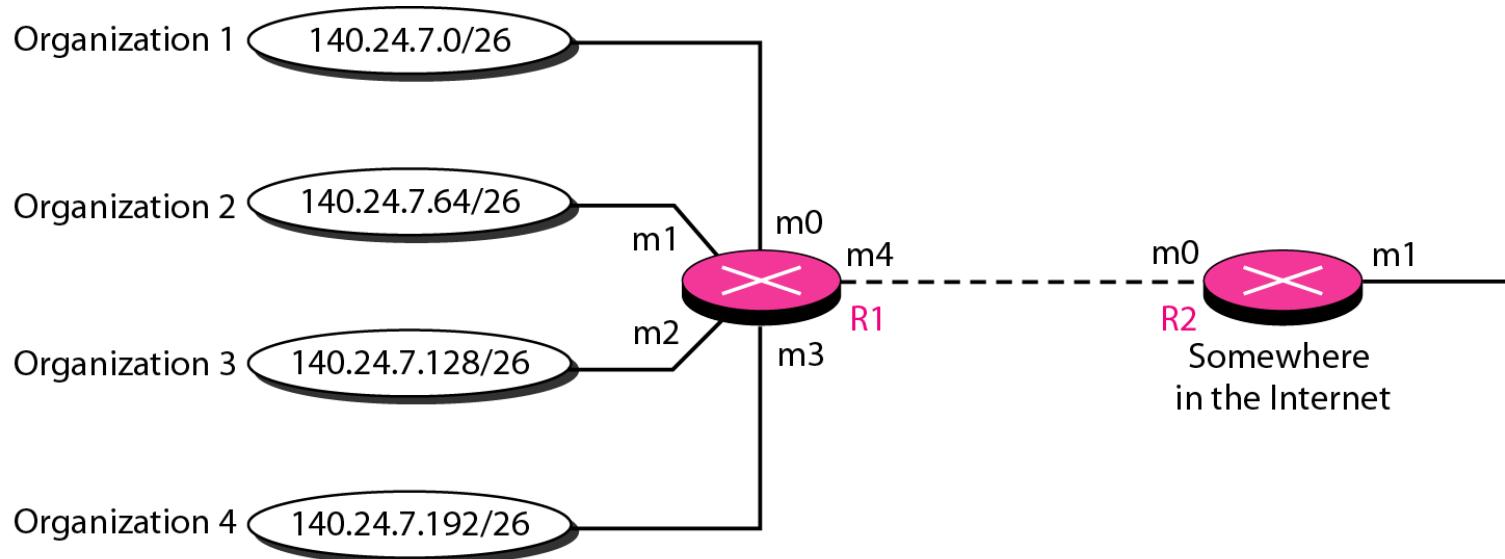
Example 22.4

Show the forwarding process if a packet arrives at R1 in Figure 22.6 with the destination address 18.24.32.78.

Solution

This time all masks are applied, one by one, to the destination address, but no matching network address is found. When it reaches the end of the table, the module gives the next-hop address 180.70.65.200 and interface number m2 to ARP. This is probably an outgoing package that needs to be sent, via the default router, to someplace else in the Internet.

Figure 22.7 Address aggregation



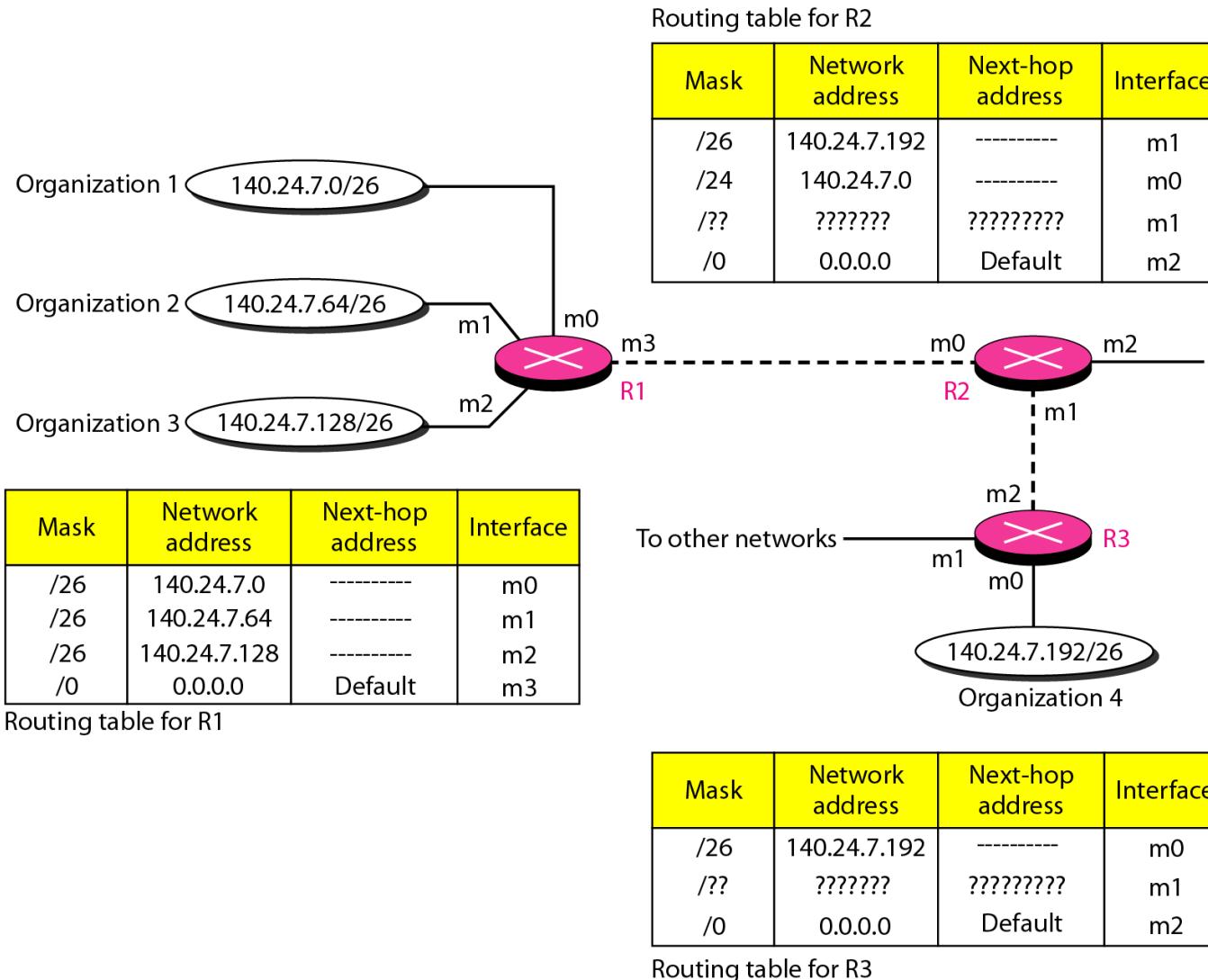
Mask	Network address	Next-hop address	Interface
/26	140.24.7.0	-----	m0
/26	140.24.7.64	-----	m1
/26	140.24.7.128	-----	m2
/26	140.24.7.192	-----	m3
/0	0.0.0.0	Default	m4

Routing table for R1

Mask	Network address	Next-hop address	Interface
/24	140.24.7.0	-----	m0
/0	0.0.0.0	Default	m1

Routing table for R2

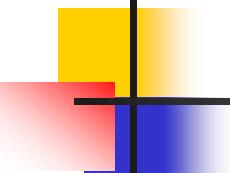
Figure 22.8 Longest mask matching



Example 22.5

As an example of hierarchical routing, let us consider Figure 22.9. A regional ISP is granted 16,384 addresses starting from 120.14.64.0. The regional ISP has decided to divide this block into four subblocks, each with 4096 addresses. Three of these subblocks are assigned to three local ISPs; the second subblock is reserved for future use. Note that the mask for each block is /20 because the original block with mask /18 is divided into 4 blocks.

The first local ISP has divided its assigned subblock into 8 smaller blocks and assigned each to a small ISP. Each small ISP provides services to 128 households, each using four addresses.



Example 22.5 (continued)

The second local ISP has divided its block into 4 blocks and has assigned the addresses to four large organizations.

The third local ISP has divided its block into 16 blocks and assigned each block to a small organization. Each small organization has 256 addresses, and the mask is /24.

There is a sense of hierarchy in this configuration. All routers in the Internet send a packet with destination address 120.14.64.0 to 120.14.127.255 to the regional ISP.

Figure 22.9 *Hierarchical routing with ISPs*

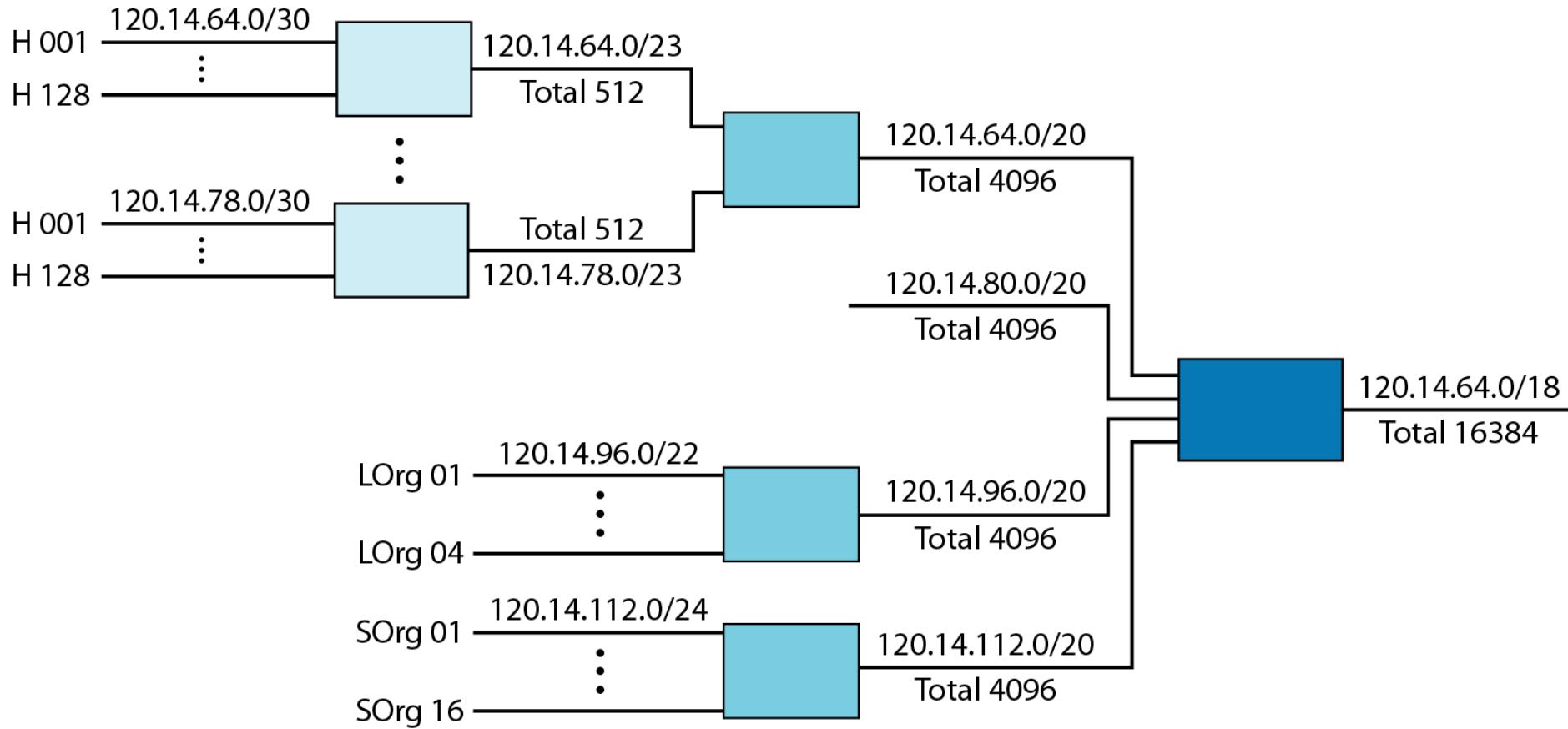
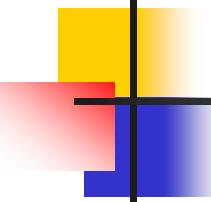


Figure 22.10 *Common fields in a routing table*

Mask	Network address	Next-hop address	Interface	Flags	Reference count	Use
.....



Example 22.6

*One utility that can be used to find the contents of a routing table for a host or router is **netstat** in UNIX or LINUX. The next slide shows the list of the contents of a default server. We have used two options, r and n. The option r indicates that we are interested in the routing table, and the option n indicates that we are looking for numeric addresses. Note that this is a routing table for a host, not a router. Although we discussed the routing table for a router throughout the chapter, a host also needs a routing table.*

Example 22.6 (continued)

```
$ netstat -rn
```

Kernel IP routing table

Destination	Gateway	Mask	Flags	Iface
153.18.16.0	0.0.0.0	255.255.240.0	U	eth0
127.0.0.0	0.0.0.0	255.0.0.0	U	lo
0.0.0.0	153.18.31.254	0.0.0.0	UG	eth0

The destination column here defines the network address. The term gateway used by UNIX is synonymous with router. This column actually defines the address of the next hop. The value 0.0.0.0 shows that the delivery is direct. The last entry has a flag of G, which means that the destination can be reached through a router (default router). The Iface defines the interface.

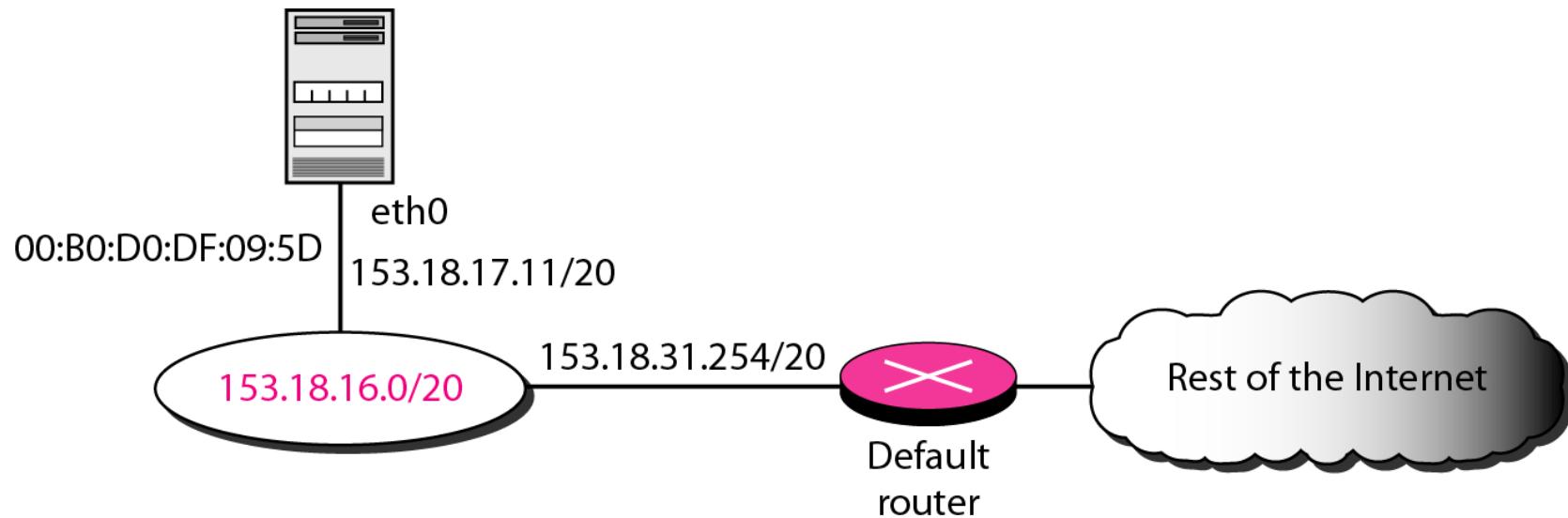
Example 22.6 (continued)

*More information about the IP address and physical address of the server can be found by using the **ifconfig** command on the given interface (eth0).*

```
$ ifconfig eth0
```

```
eth0 Link encap:Ethernet HWaddr 00:B0:D0:DF:09:5D  
inet addr:153.18.17.11 Bcast:153.18.31.255 Mask:255.255.240.0  
...
```

Figure 22.11 Configuration of the server for Example 22.6



22-3 UNICAST ROUTING PROTOCOLS

A routing table can be either static or dynamic. A static table is one with manual entries. A dynamic table is one that is updated automatically when there is a change somewhere in the Internet. A routing protocol is a combination of rules and procedures that lets routers in the Internet inform each other of changes.

Topics discussed in this section:

Optimization

Intra- and Interdomain Routing

Distance Vector Routing and RIP

Link State Routing and OSPF

Path Vector Routing and BGP

Figure 22.12 Autonomous systems

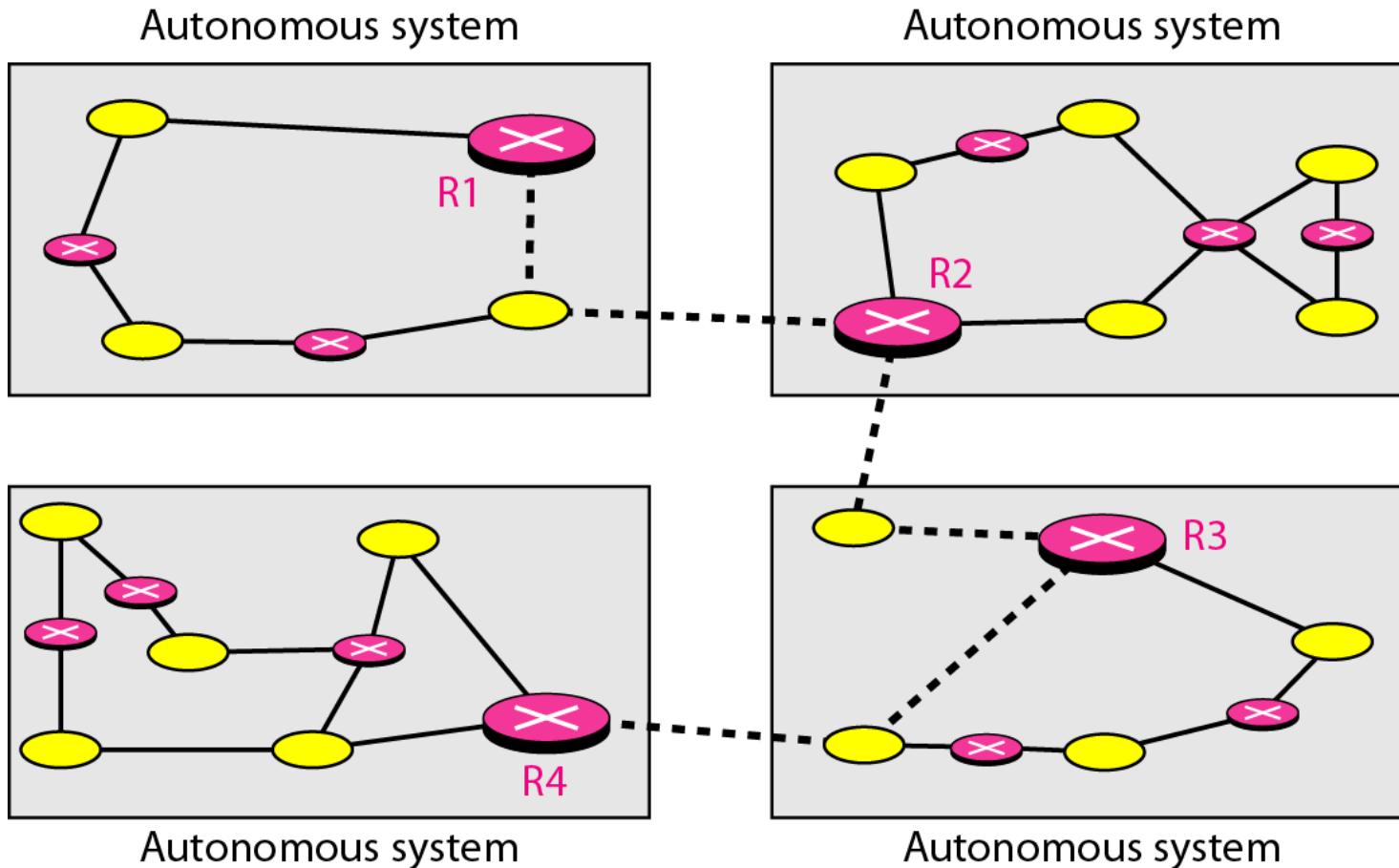


Figure 22.13 *Popular routing protocols*

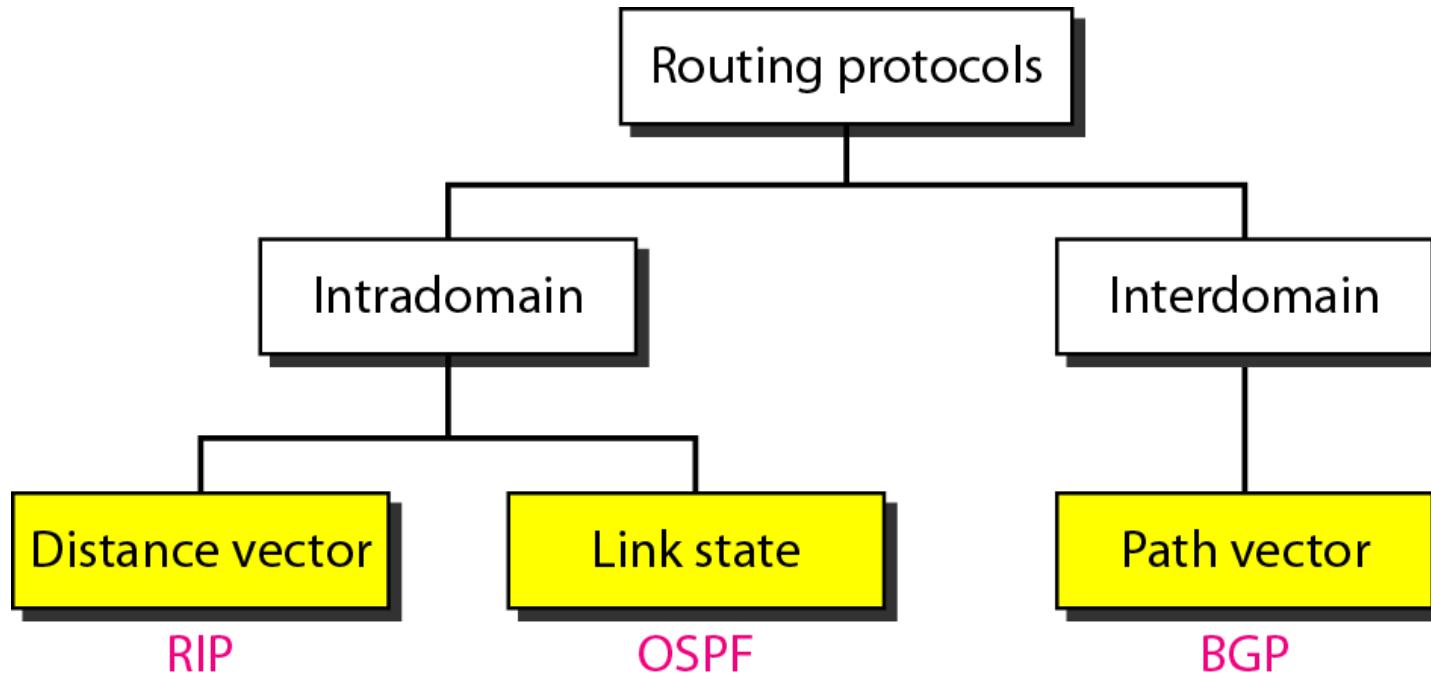


Figure 22.14 Distance vector routing tables

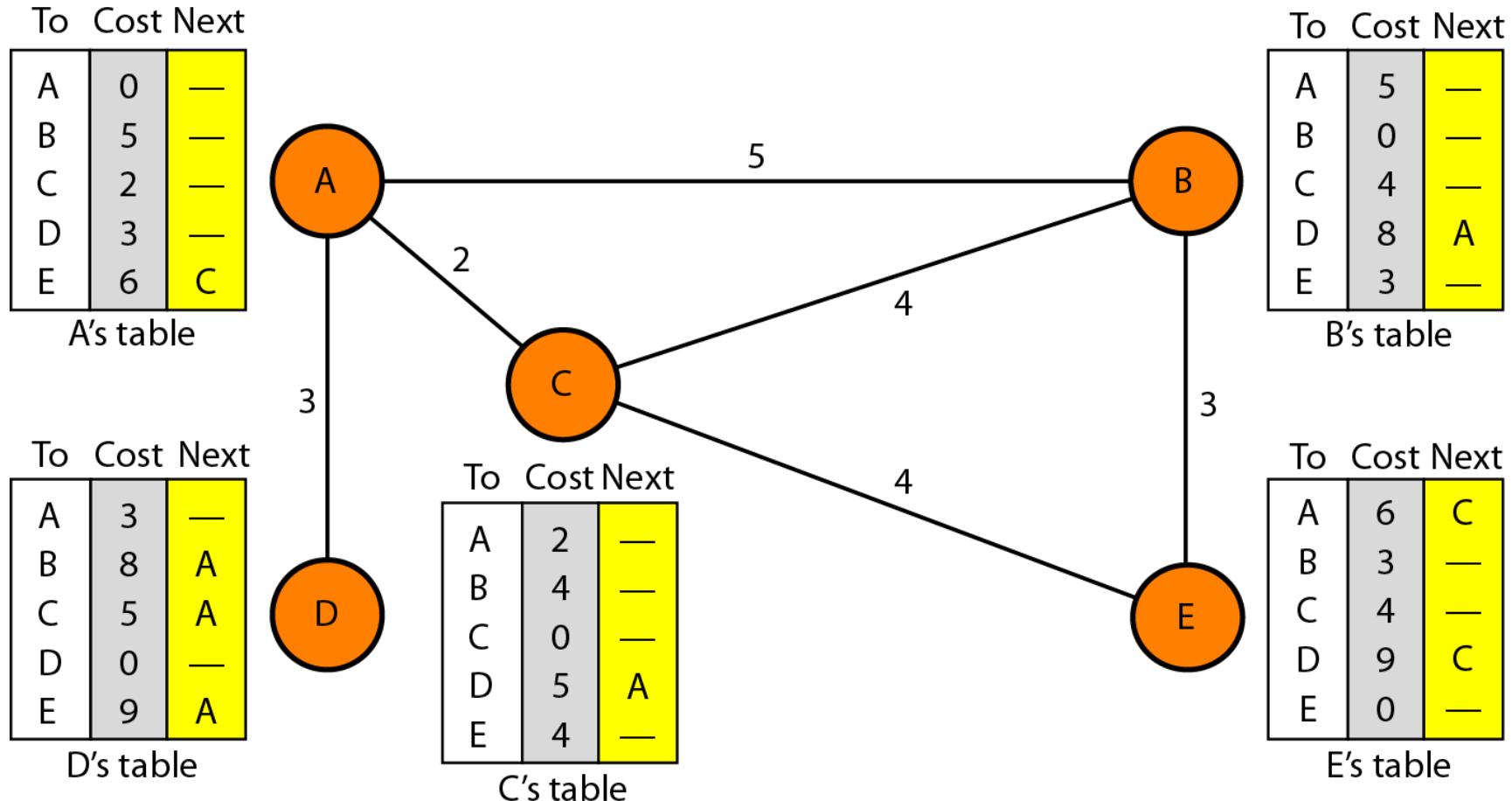
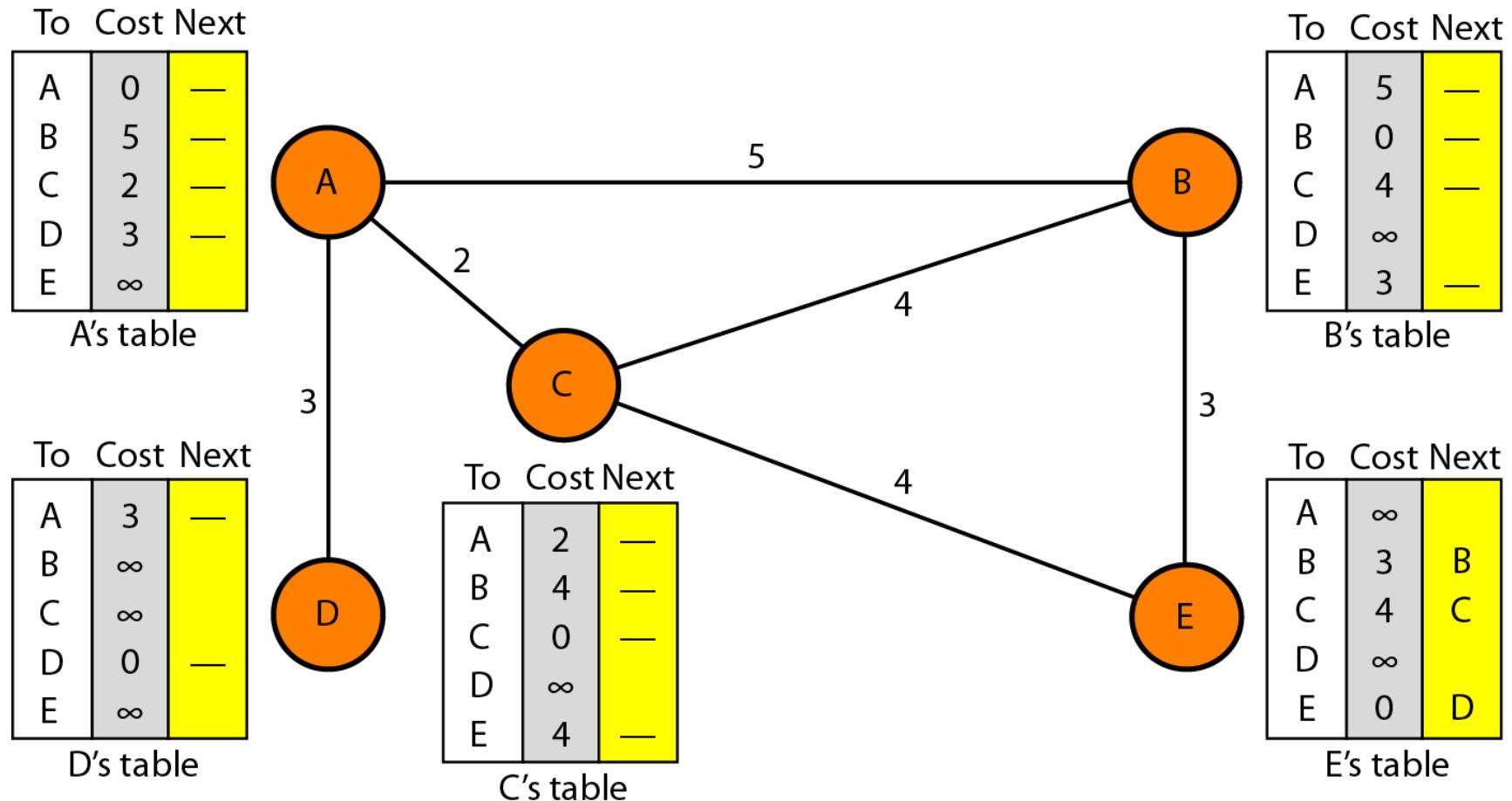
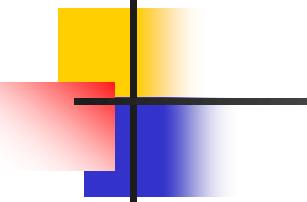


Figure 22.15 Initialization of tables in distance vector routing





Note

In distance vector routing, each node shares its routing table with its immediate neighbors periodically and when there is a change.

Figure 22.16 Updating in distance vector routing

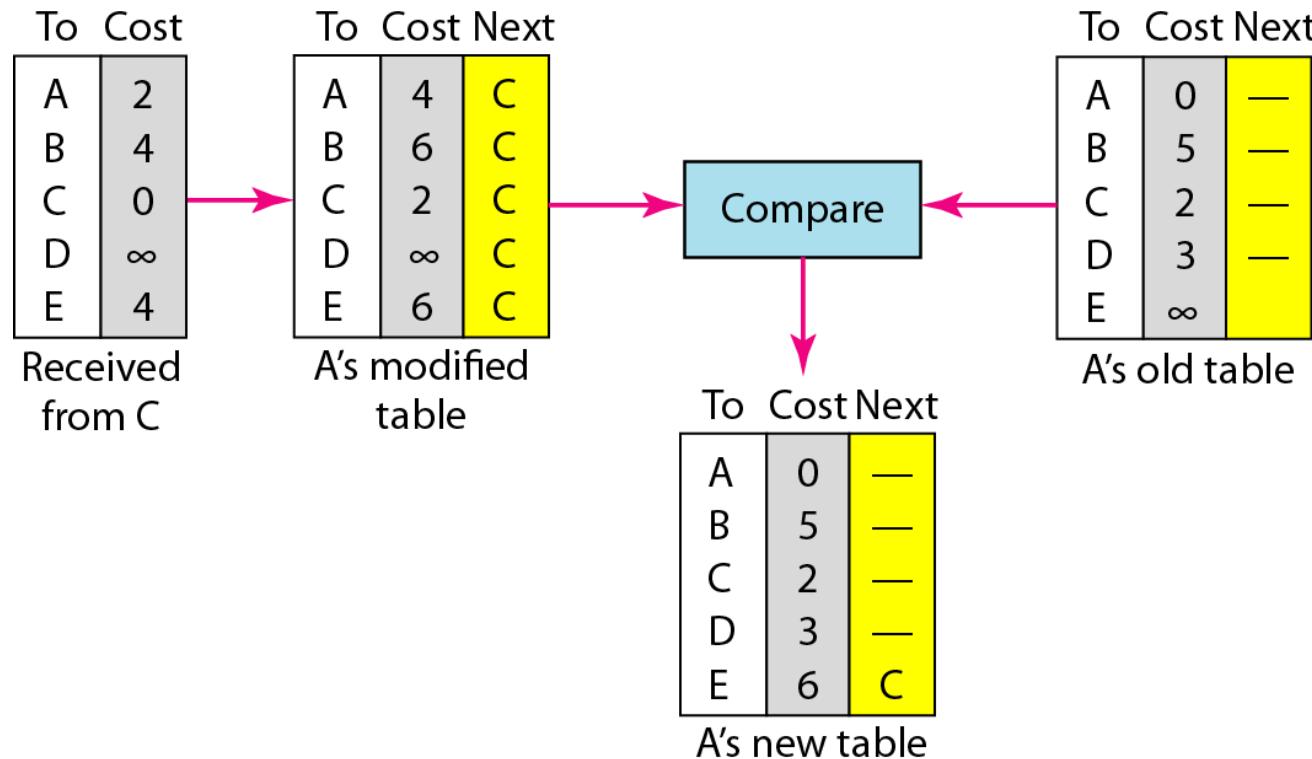


Figure 22.17 Two-node instability

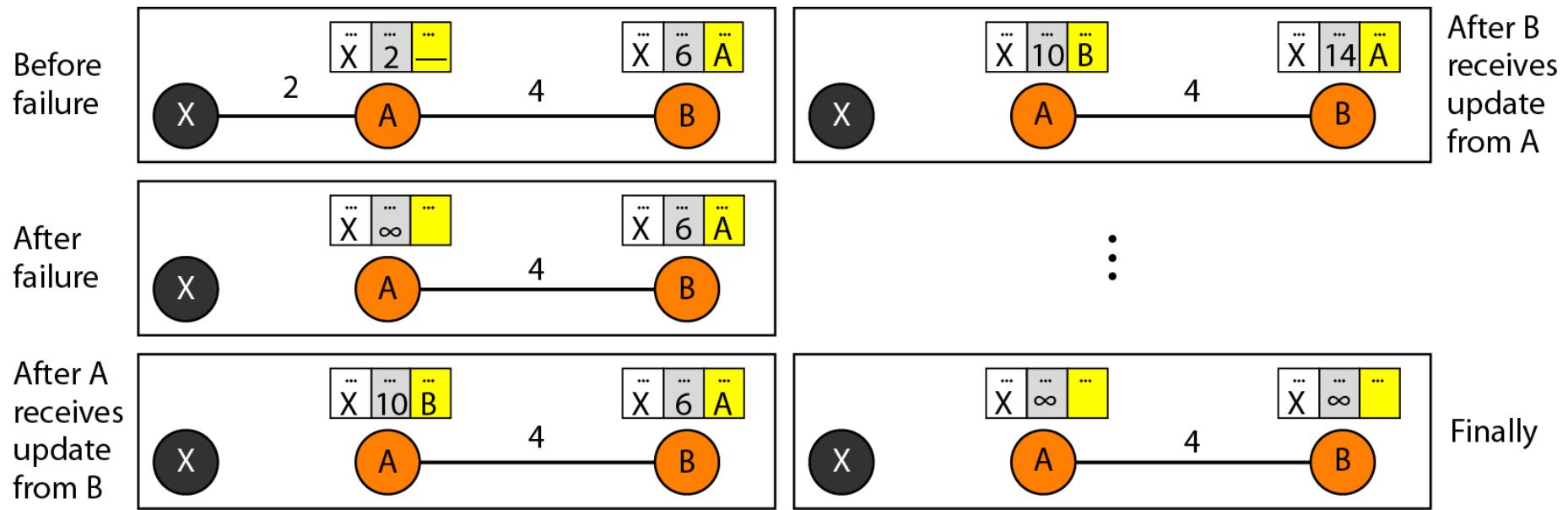


Figure 22.18 Three-node instability

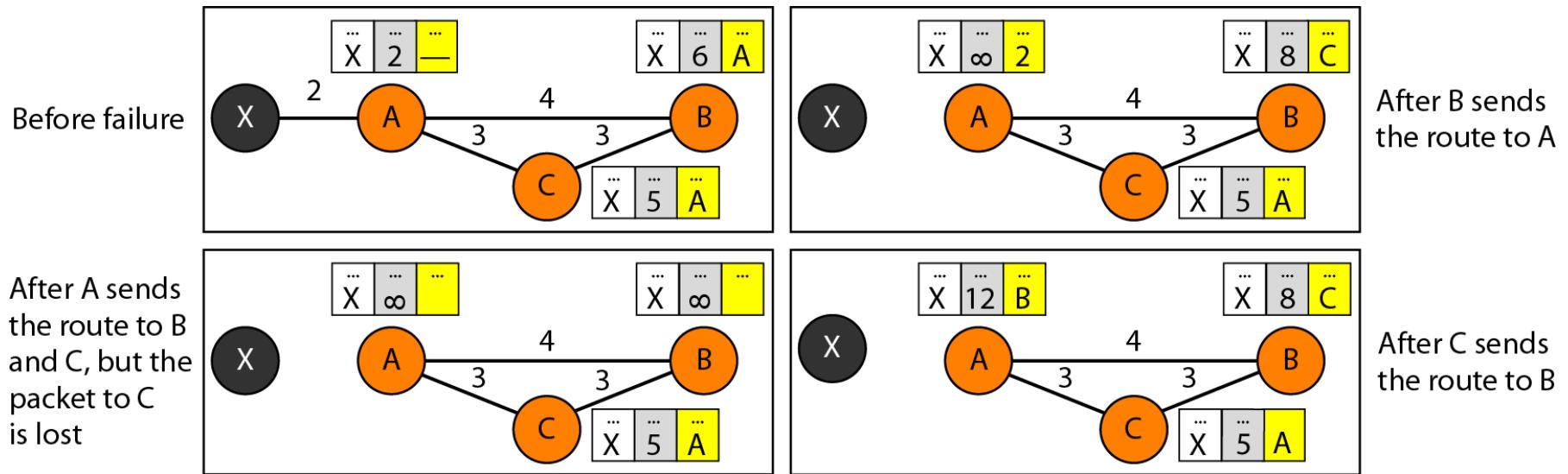


Figure 22.19 Example of a domain using RIP

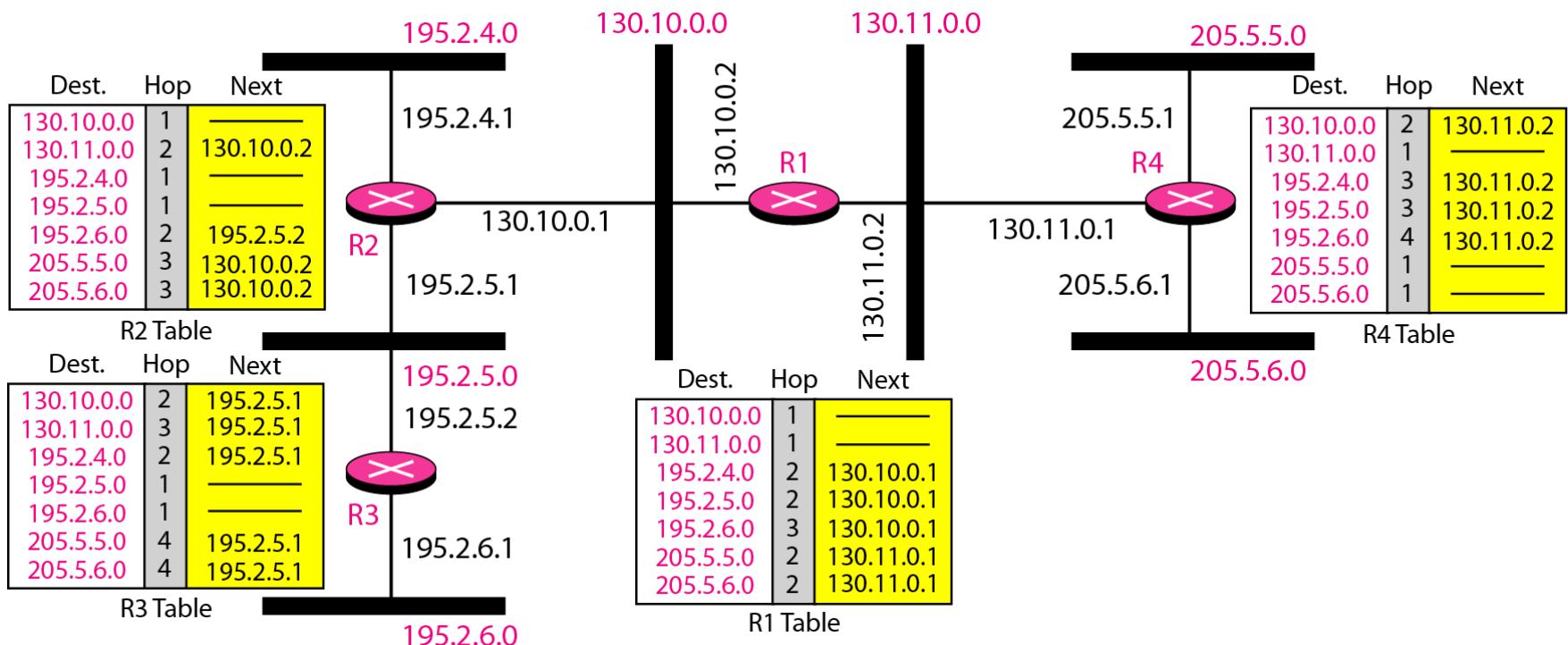


Figure 22.20 Concept of link state routing

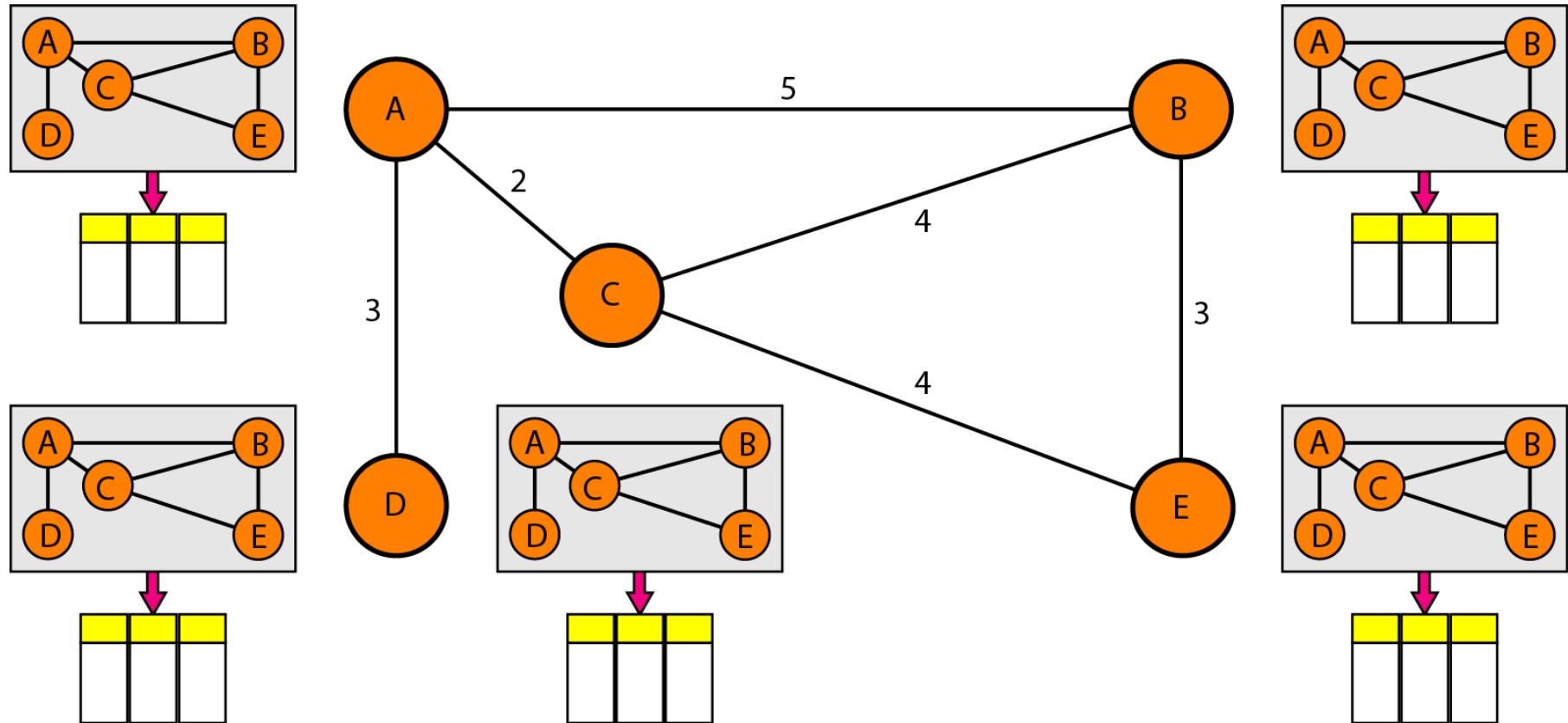


Figure 22.21 *Link state knowledge*

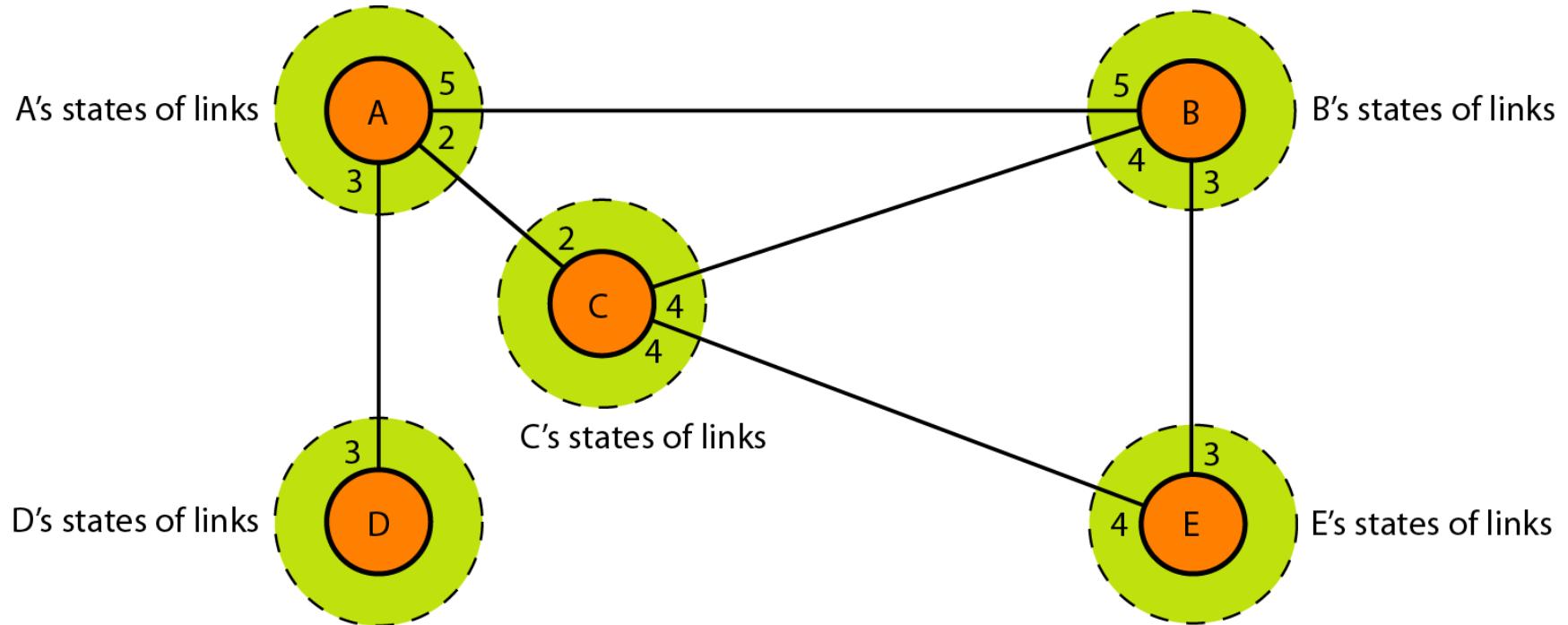


Figure 22.22 Dijkstra algorithm

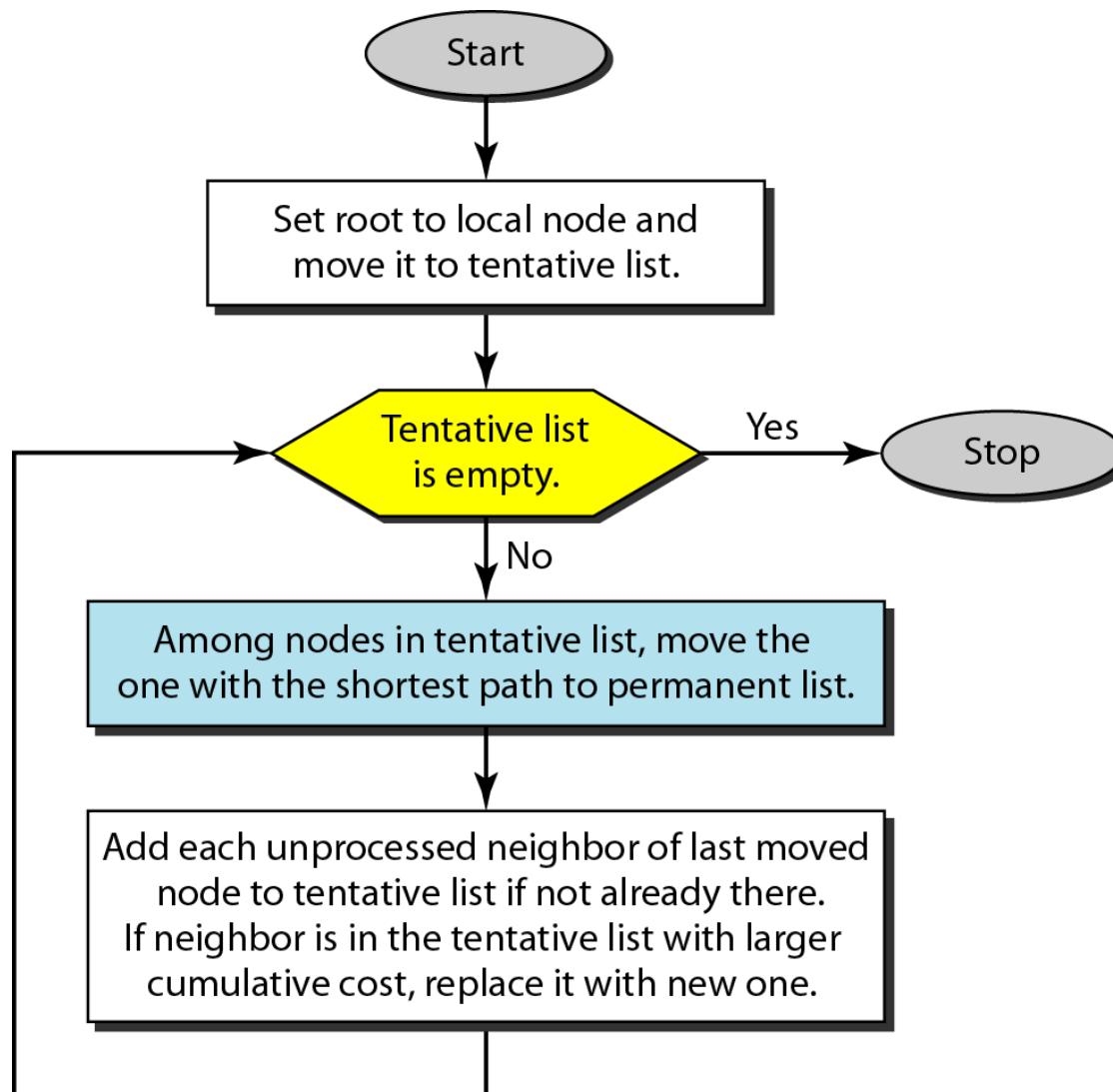


Figure 22.23 Example of formation of shortest path tree

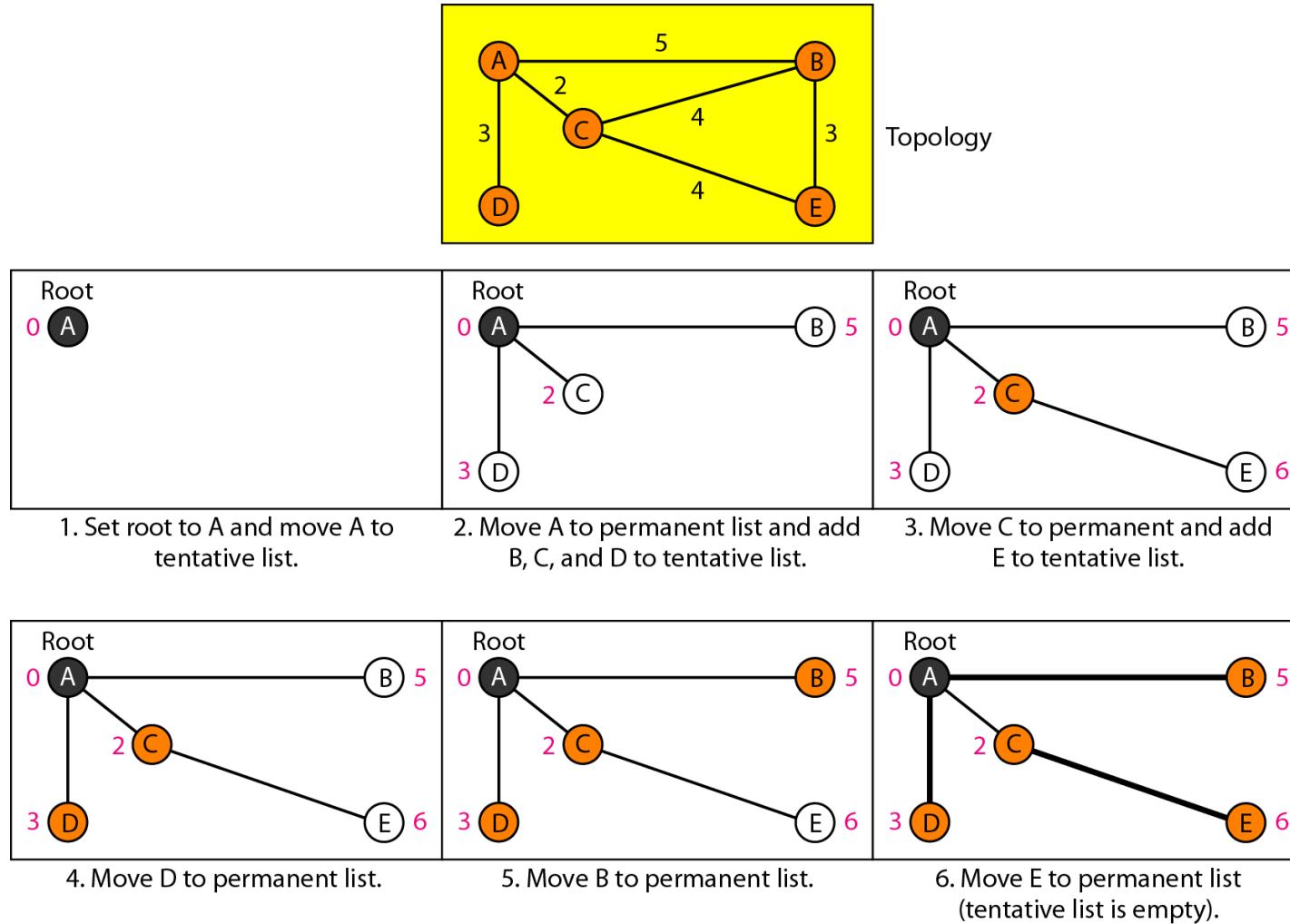


Table 22.2 *Routing table for node A*

<i>Node</i>	<i>Cost</i>	<i>Next Router</i>
A	0	—
B	5	—
C	2	—
D	3	—
E	6	C

22.42

Figure 22.24 Areas in an autonomous system

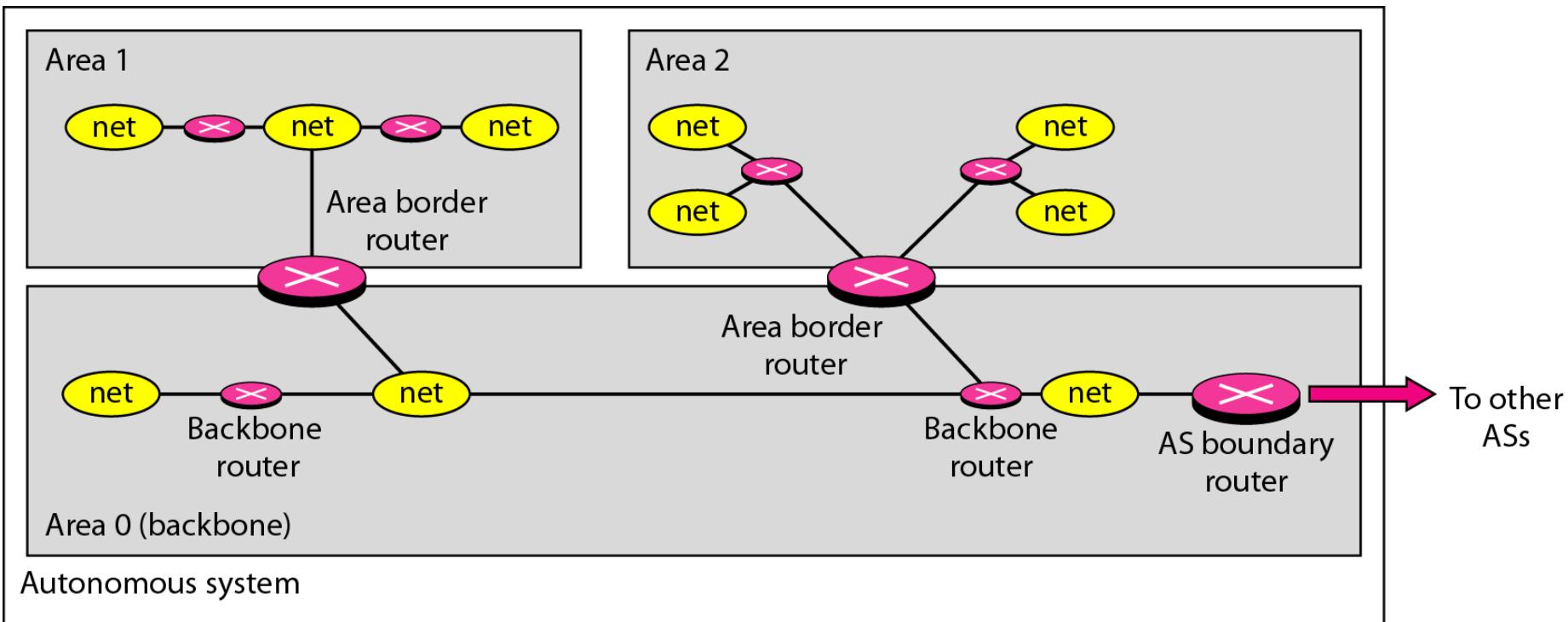


Figure 22.25 *Types of links*

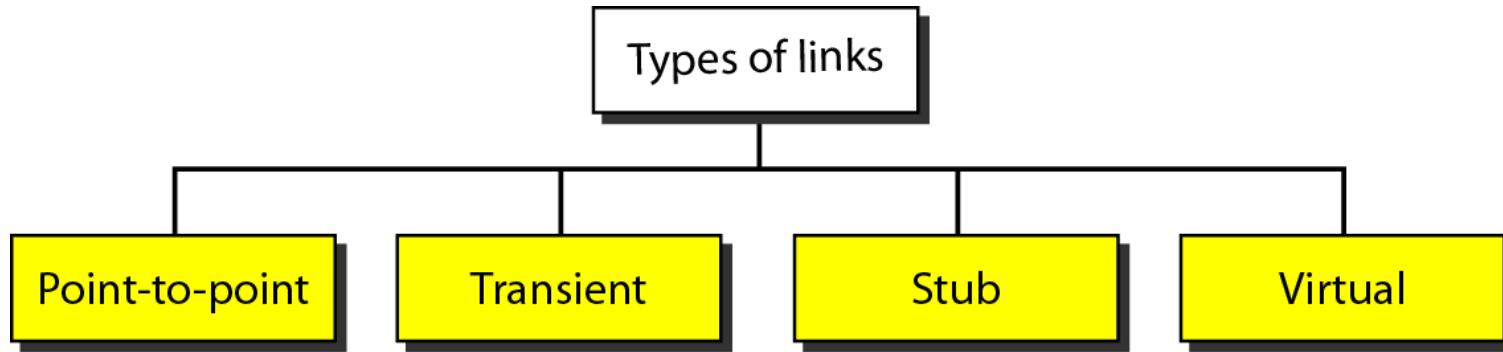


Figure 22.26 *Point-to-point link*

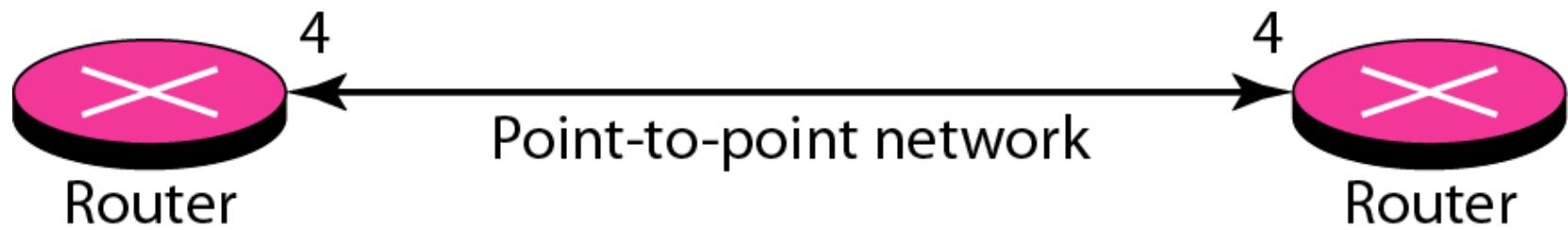
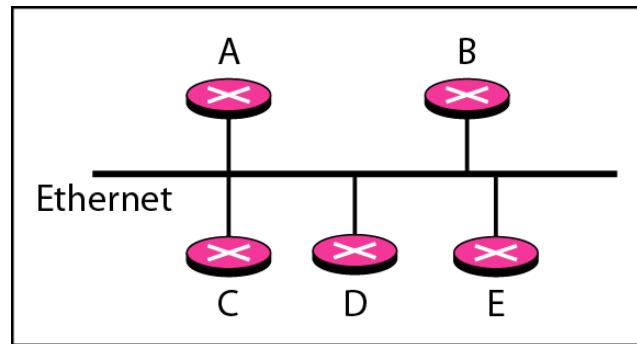
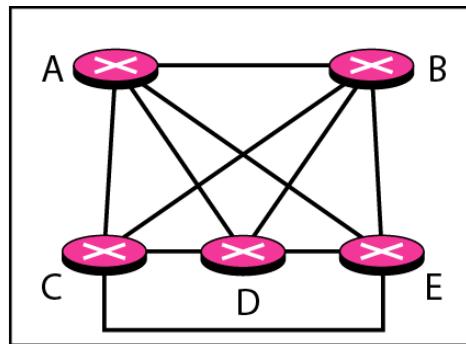


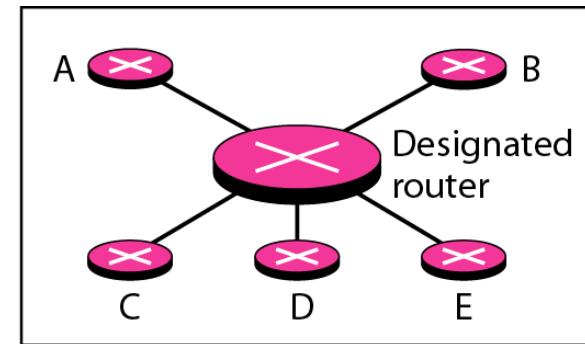
Figure 22.27 Transient link



a. Transient network

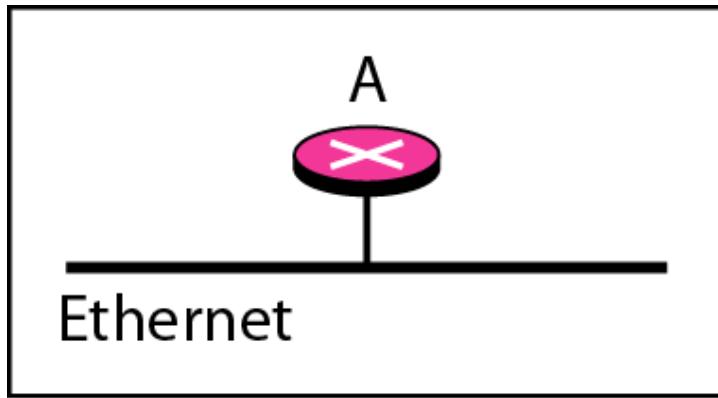


b. Unrealistic representation



c. Realistic representation

Figure 22.28 *Stub link*

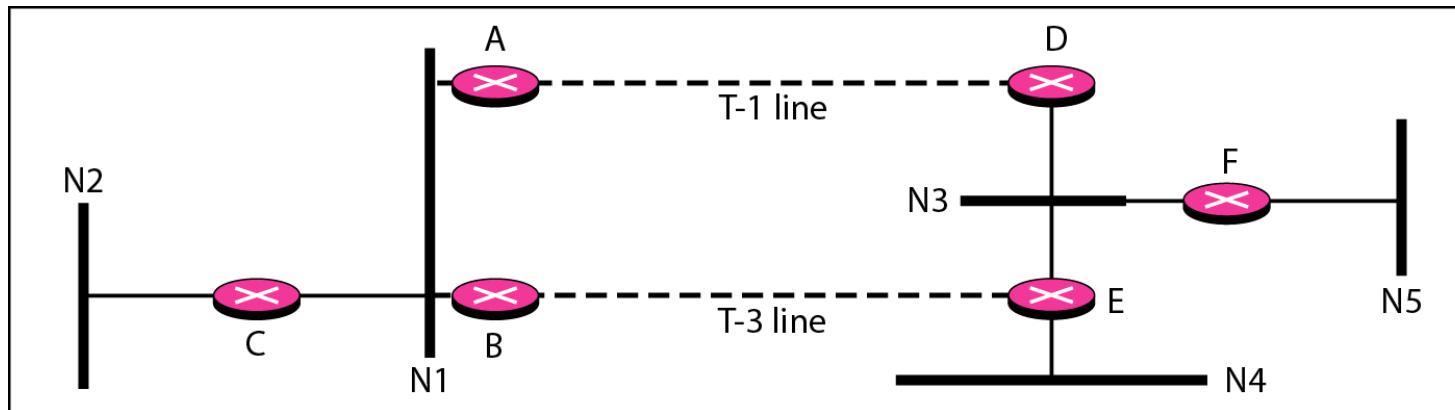


a. Stub network

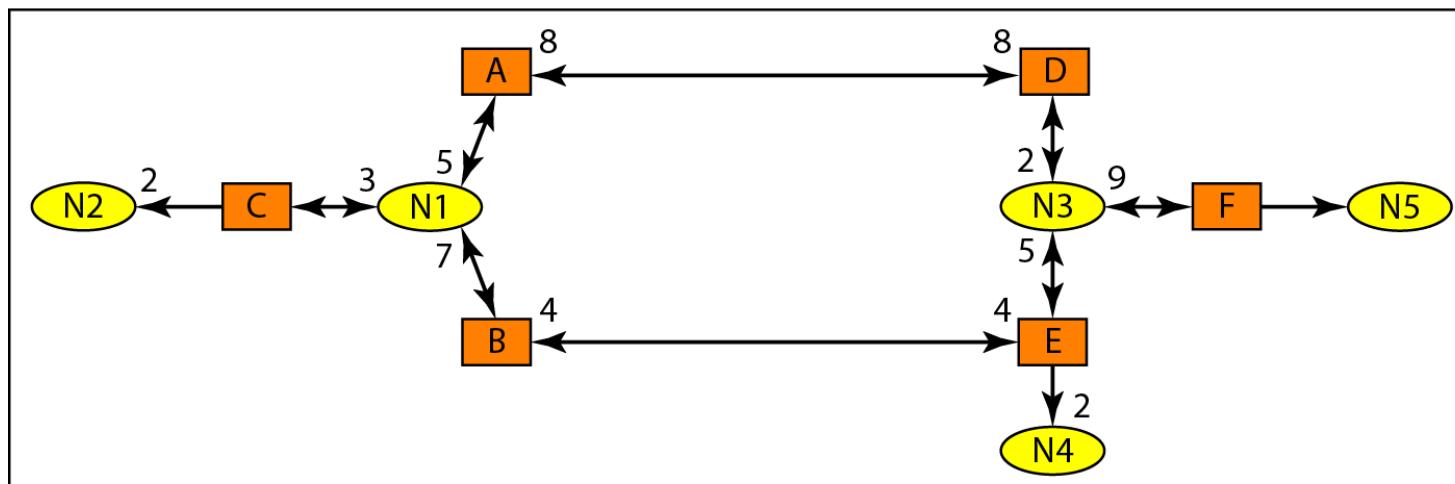


b. Representation

Figure 22.29 Example of an AS and its graphical representation in OSPF



a. Autonomous system



b. Graphical representation

END :- *As per the syllabus*

Figure 22.30 Initial routing tables in path vector routing

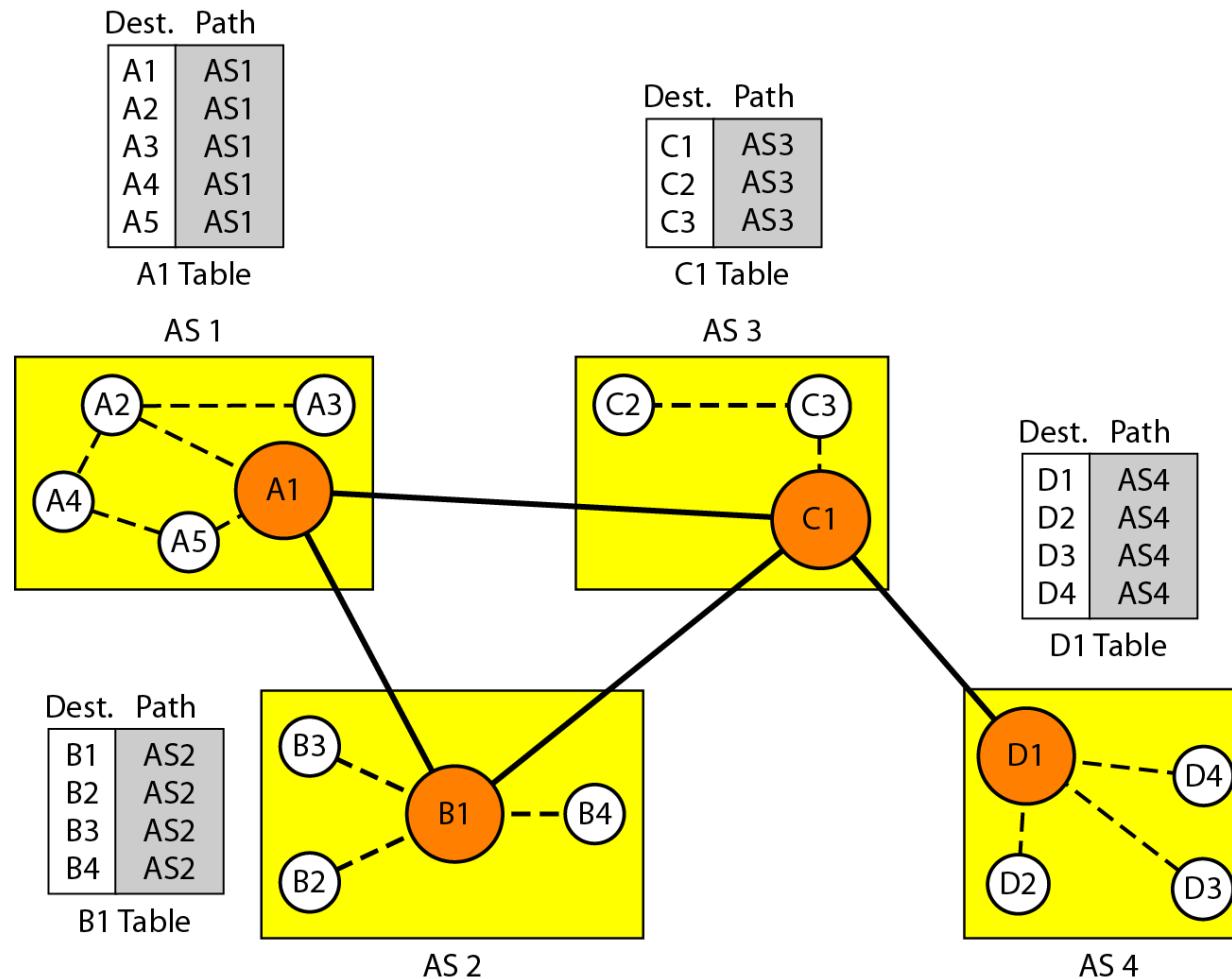


Figure 22.31 *Stabilized tables for three autonomous systems*

Dest.	Path
A1	AS1
...	
A5	AS1
B1	AS1-AS2
...	
B4	AS1-AS2
C1	AS1-AS3
...	
C3	AS1-AS3
D1	AS1-AS2-AS4
...	
D4	AS1-AS2-AS4

A1 Table

Dest.	Path
A1	AS2-AS1
...	
A5	AS2-AS1
B1	AS2
...	
B4	AS2
C1	AS2-AS3
...	
C3	AS2-AS3
D1	AS2-AS3-AS4
...	
D4	AS2-AS3-AS4

B1 Table

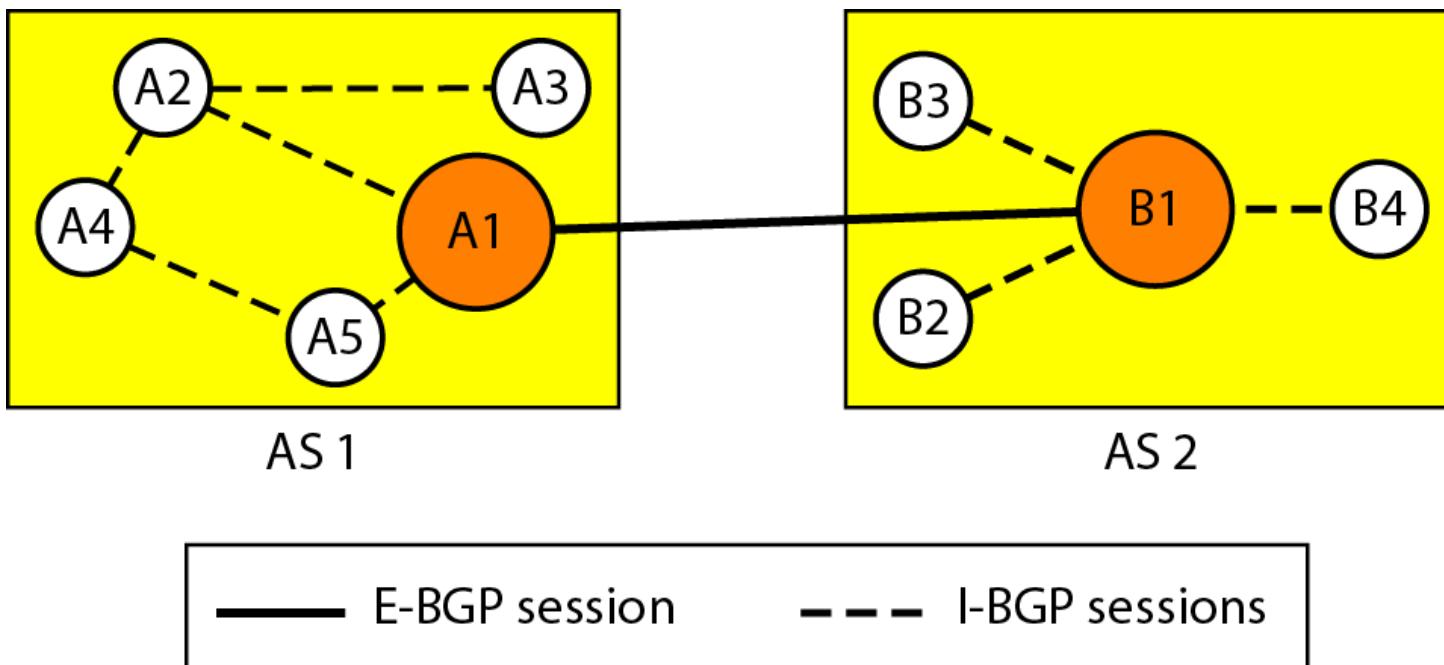
Dest.	Path
A1	AS3-AS1
...	
A5	AS3-AS1
B1	AS3-AS2
...	
B4	AS3-AS2
C1	AS3
...	
C3	AS3
D1	AS3-AS4
...	
D4	AS3-AS4

C1 Table

Dest.	Path
A1	AS4-AS3-AS1
...	
A5	AS4-AS3-AS1
B1	AS4-AS3-AS2
...	
B4	AS4-AS3-AS2
C1	AS4-AS3
...	
C3	AS4-AS3
D1	AS4
...	
D4	AS4

D1 Table

Figure 22.32 Internal and external BGP sessions



22-4 MULTICAST ROUTING PROTOCOLS

In this section, we discuss multicasting and multicast routing protocols.

Topics discussed in this section:

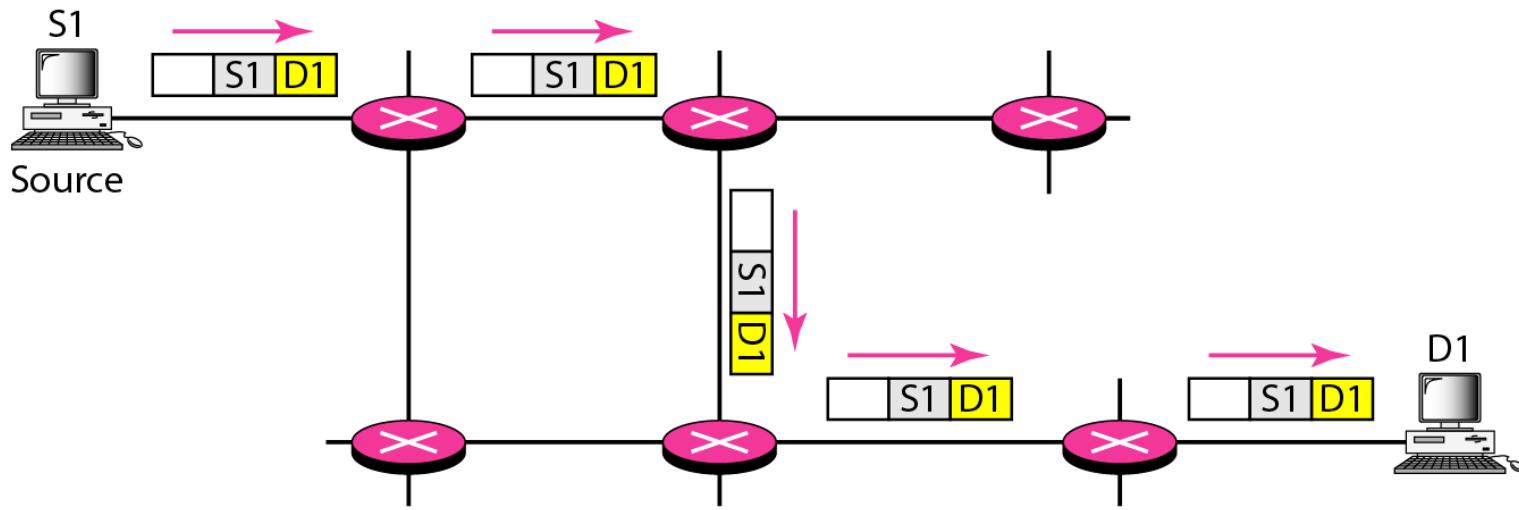
Unicast, Multicast, and Broadcast

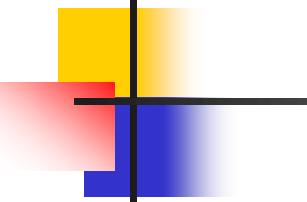
Applications

Multicast Routing

Routing Protocols

Figure 22.33 Unicasting

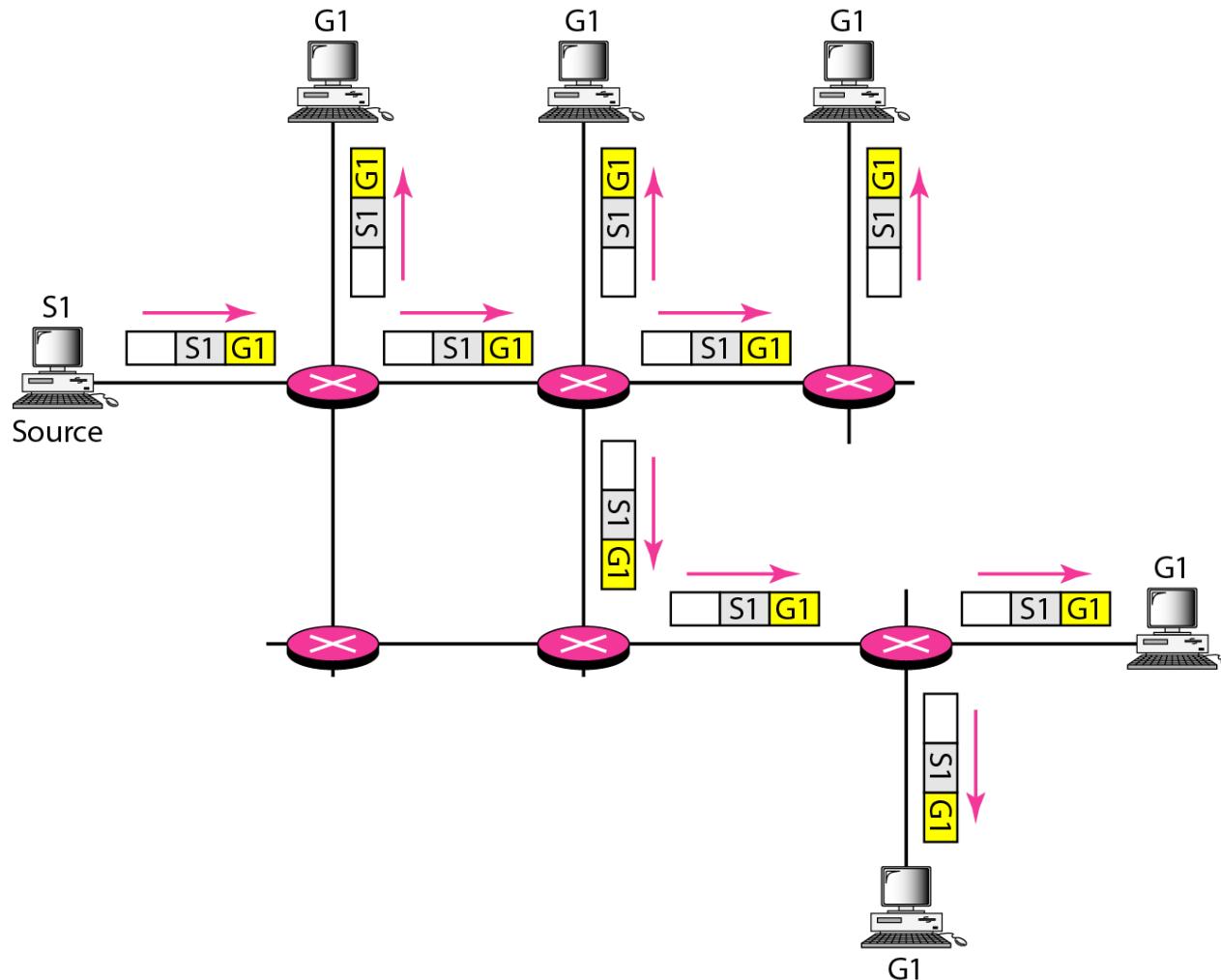


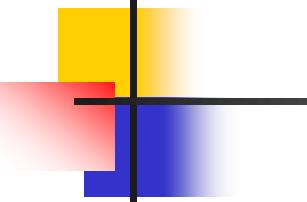


Note

In unicasting, the router forwards the received packet through only one of its interfaces.

Figure 22.34 Multicasting

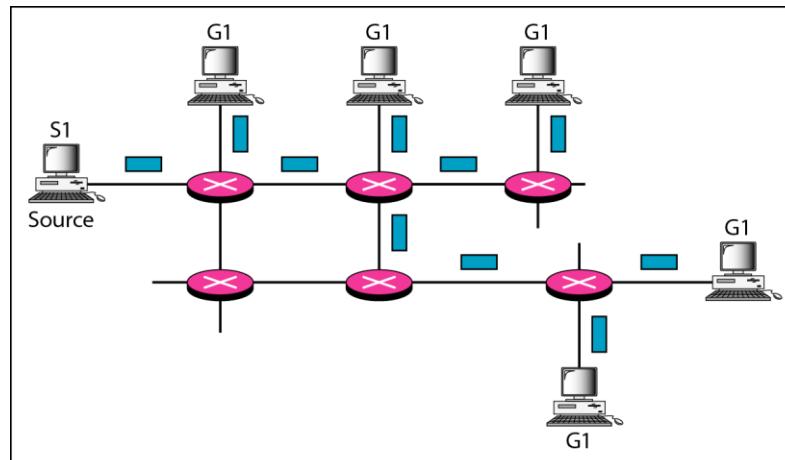




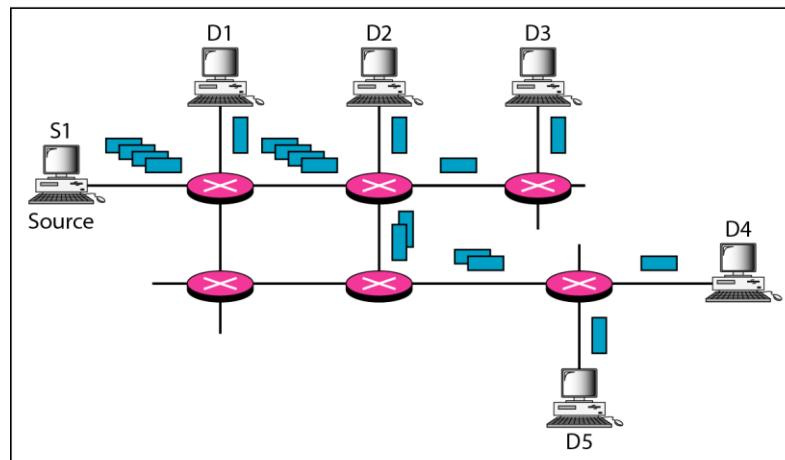
Note

In multicasting, the router may forward the received packet through several of its interfaces.

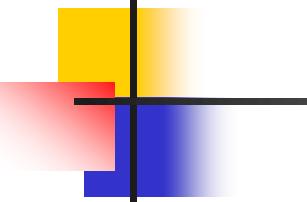
Figure 22.35 Multicasting versus multiple unicasting



a. Multicasting

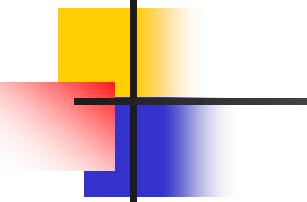


b. Multiple unicasting



Note

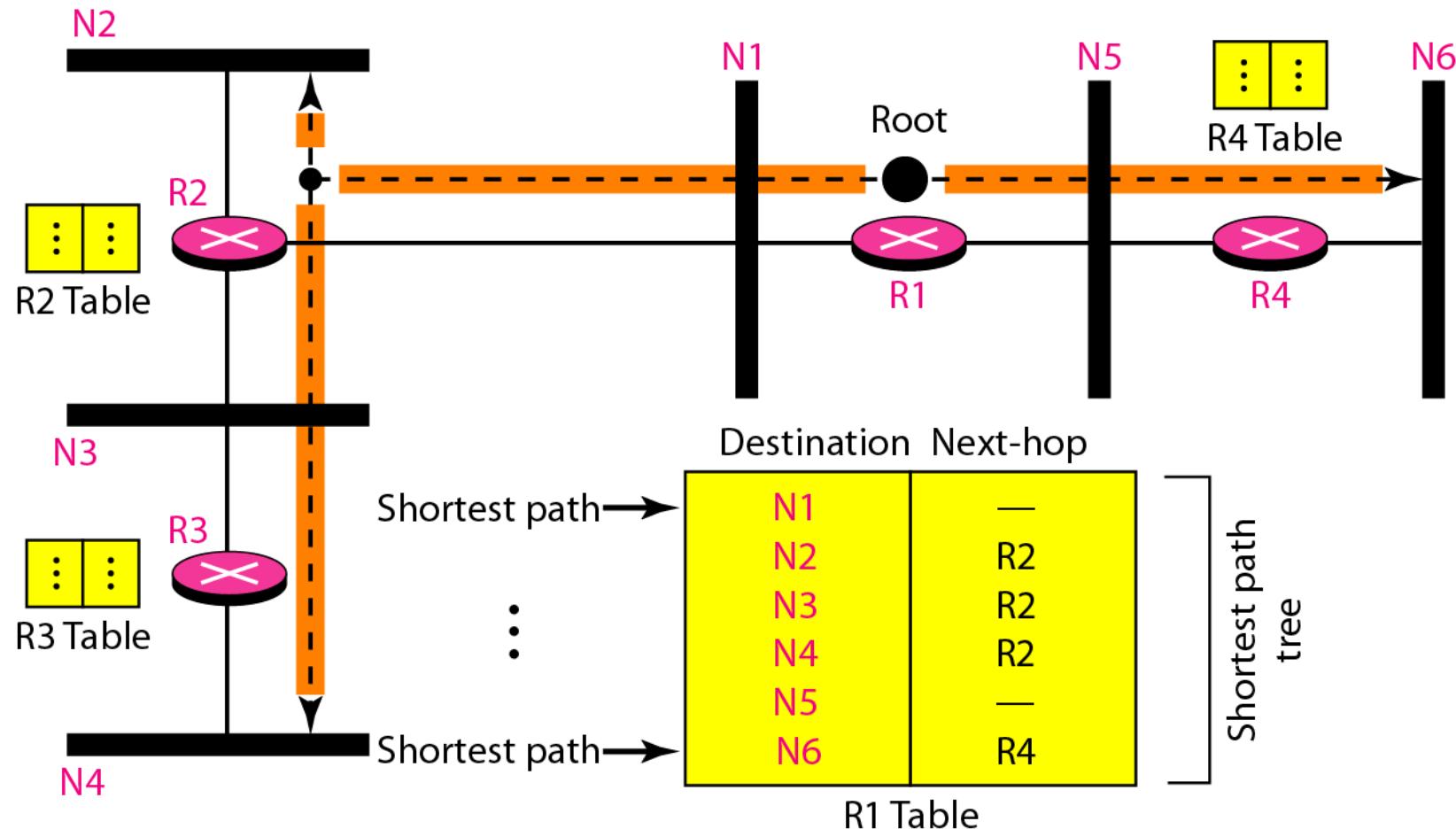
Emulation of multicasting through multiple unicasting is not efficient and may create long delays, particularly with a large group.

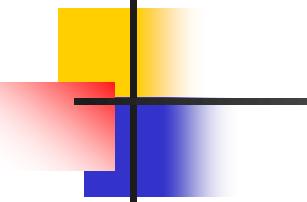


Note

In unicast routing, each router in the domain has a table that defines a shortest path tree to possible destinations.

Figure 22.36 Shortest path tree in unicast routing

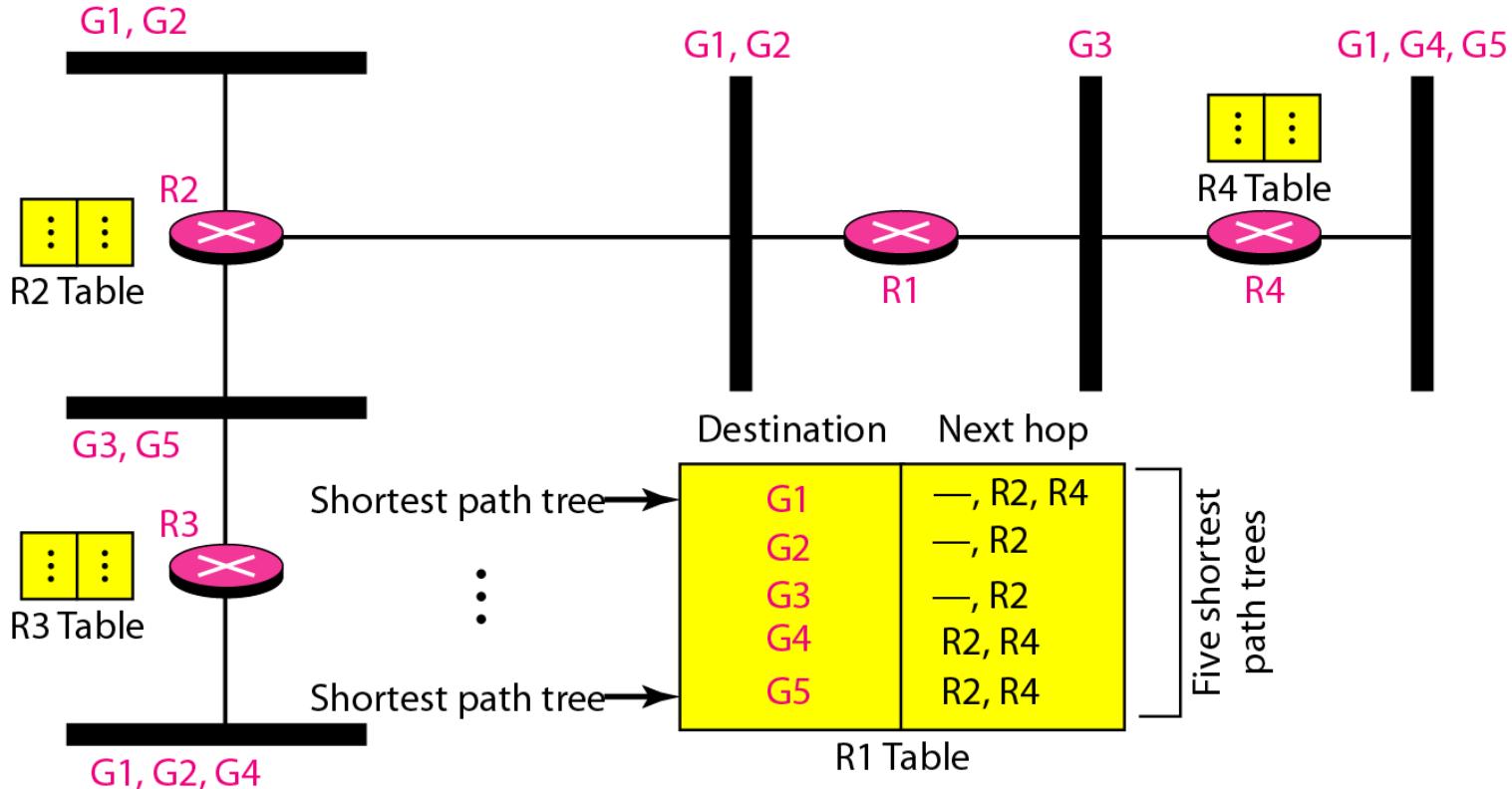


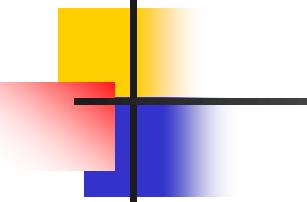


Note

In multicast routing, each involved router needs to construct a shortest path tree for each group.

Figure 22.37 Source-based tree approach

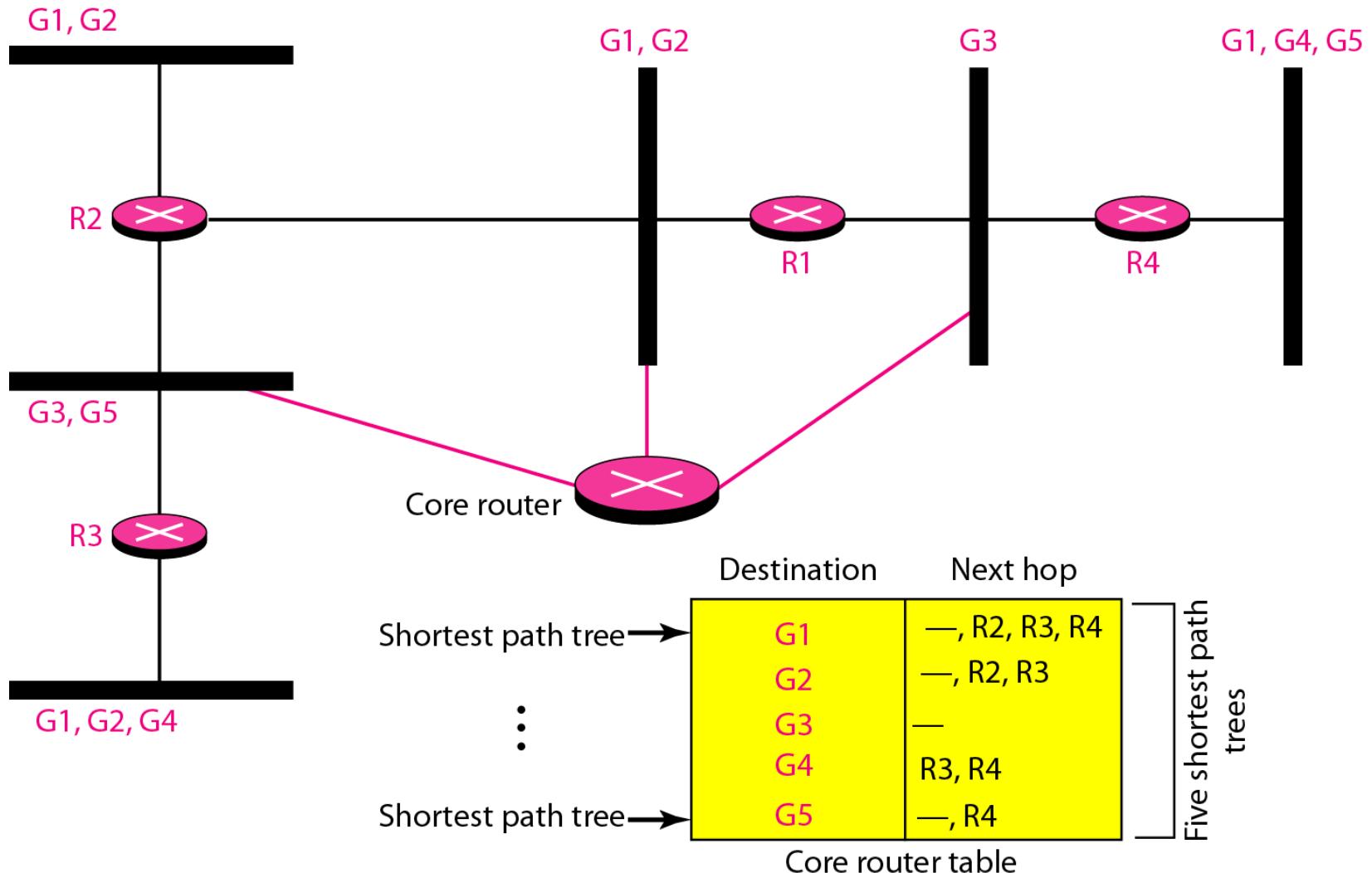


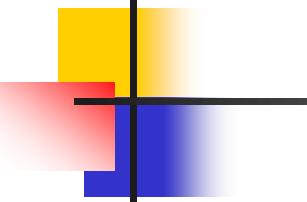


Note

In the source-based tree approach, each router needs to have one shortest path tree for each group.

Figure 22.38 Group-shared tree approach

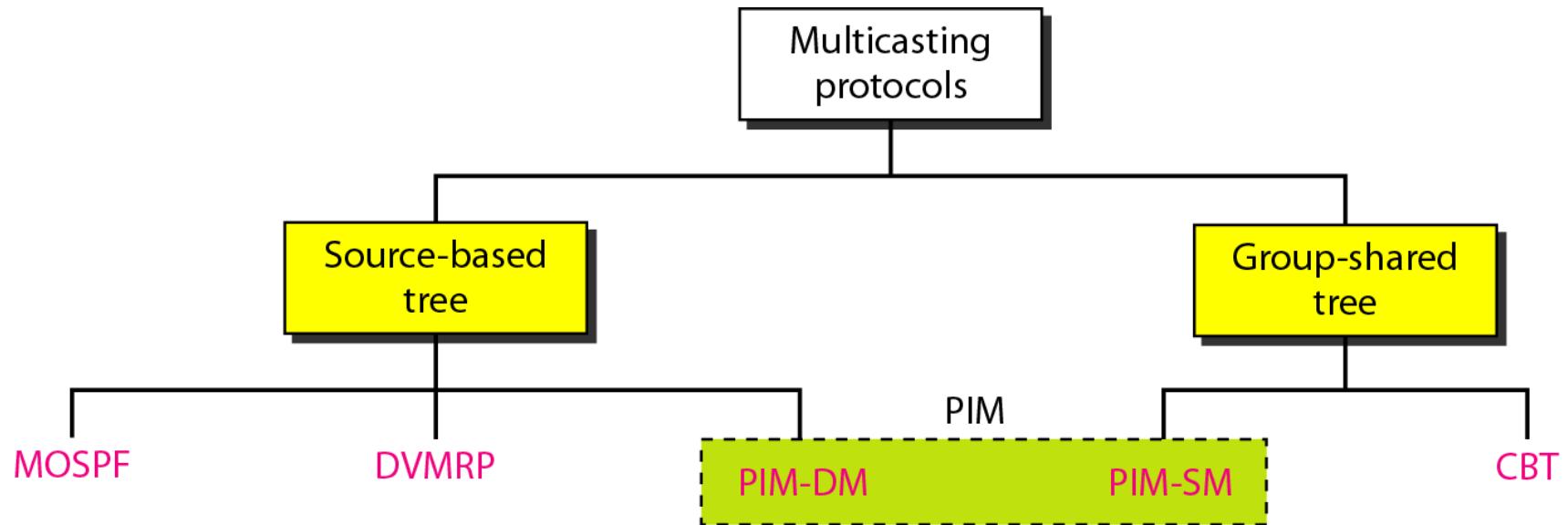


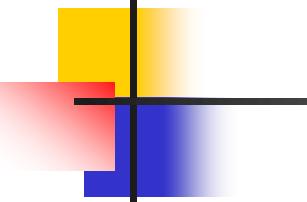


Note

In the group-shared tree approach, only the core router, which has a shortest path tree for each group, is involved in multicasting.

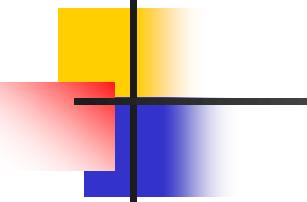
Figure 22.39 *Taxonomy of common multicast protocols*





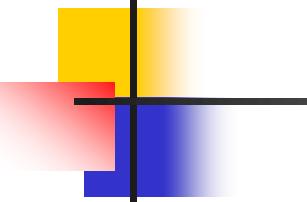
Note

Multicast link state routing uses the source-based tree approach.



Note

Flooding broadcasts packets, but creates loops in the systems.



Note

RPF eliminates the loop in the flooding process.

Figure 22.40 Reverse path forwarding (RPF)

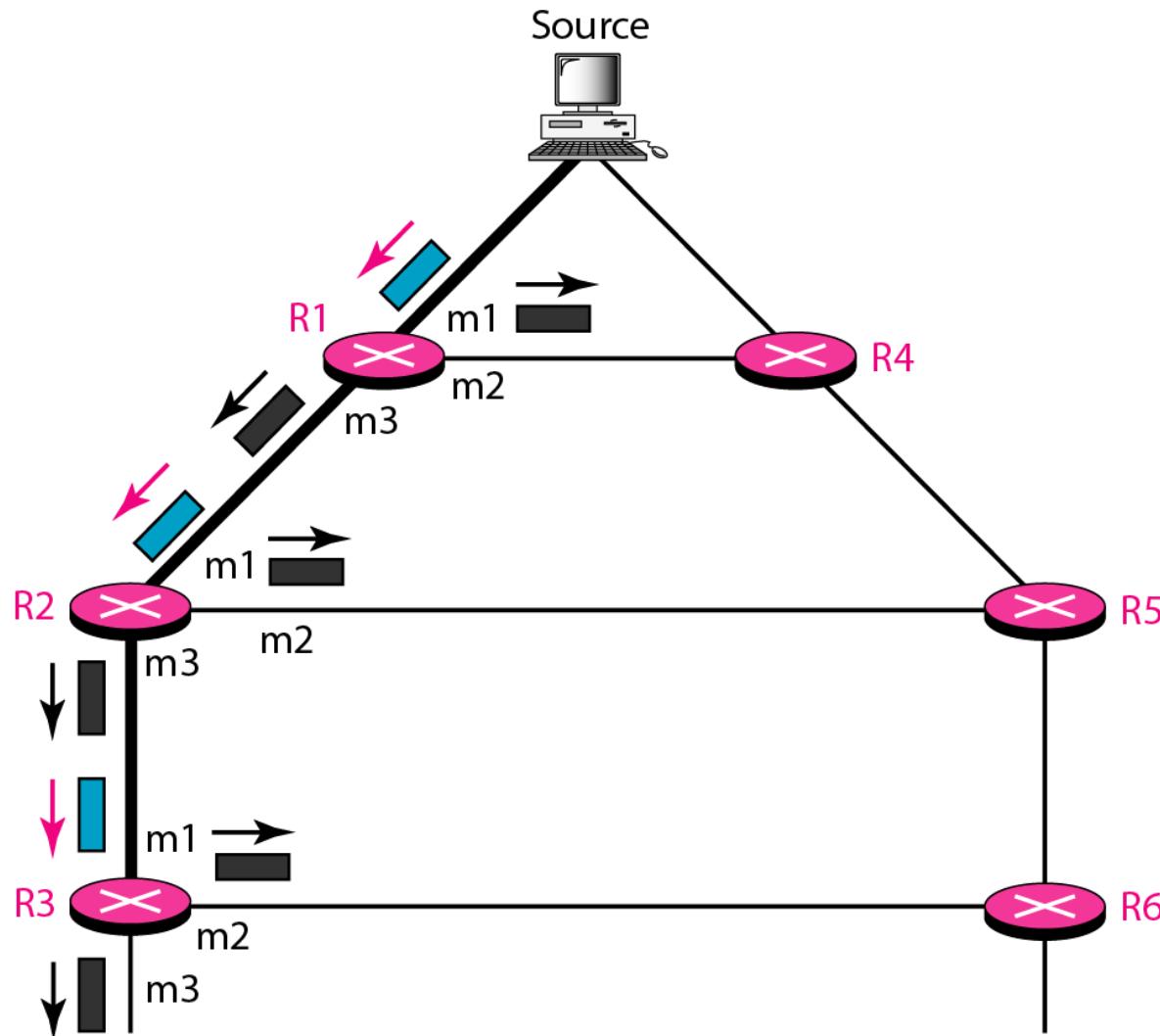


Figure 22.41 Problem with RPF

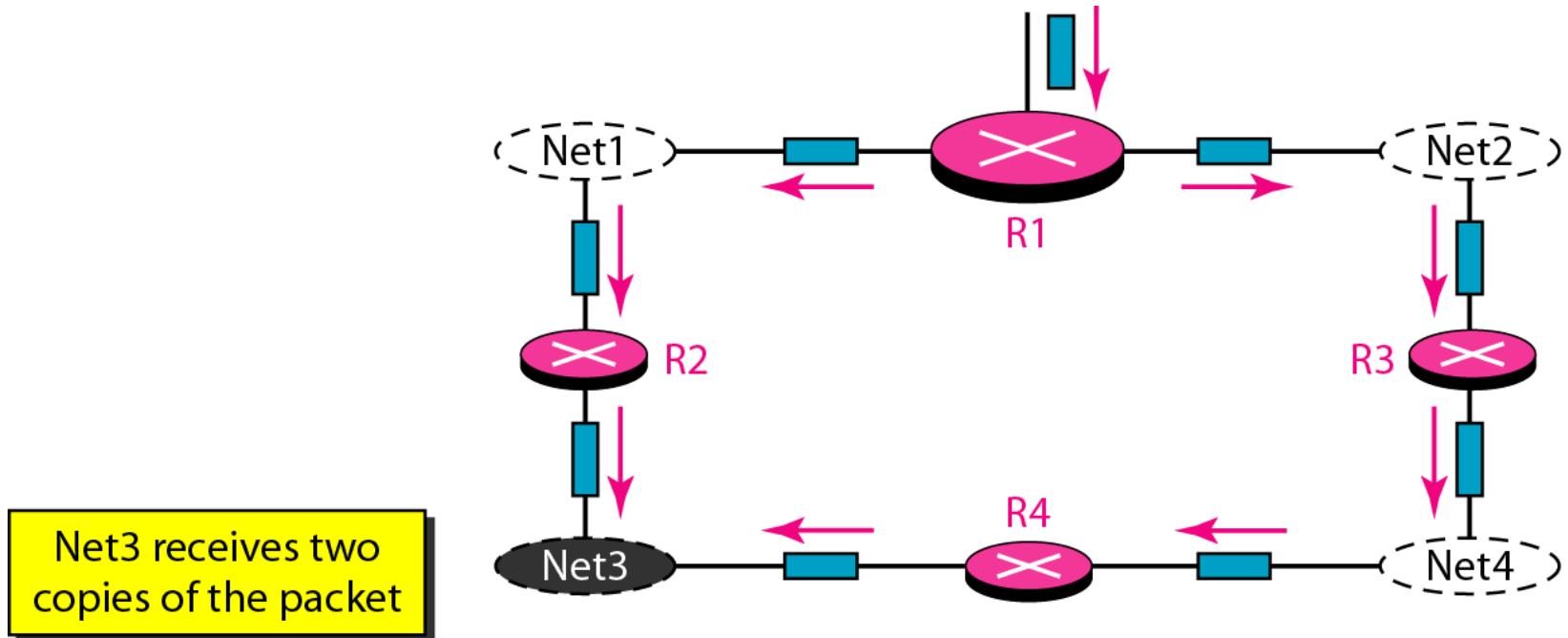
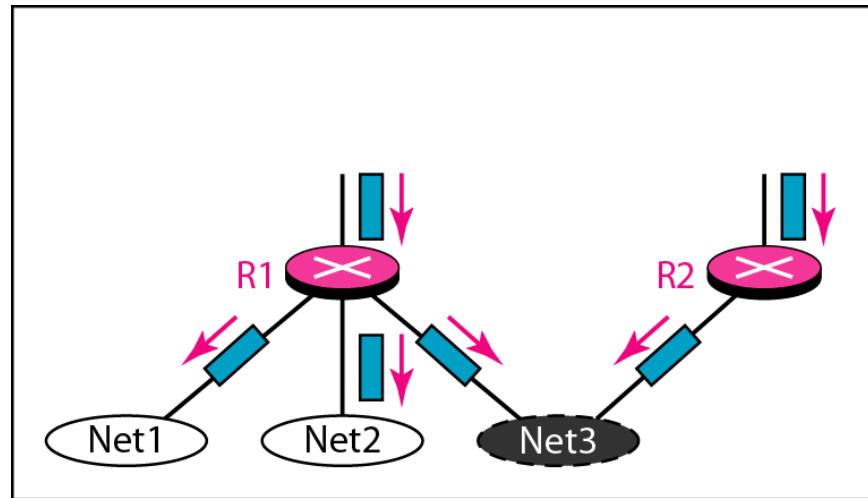
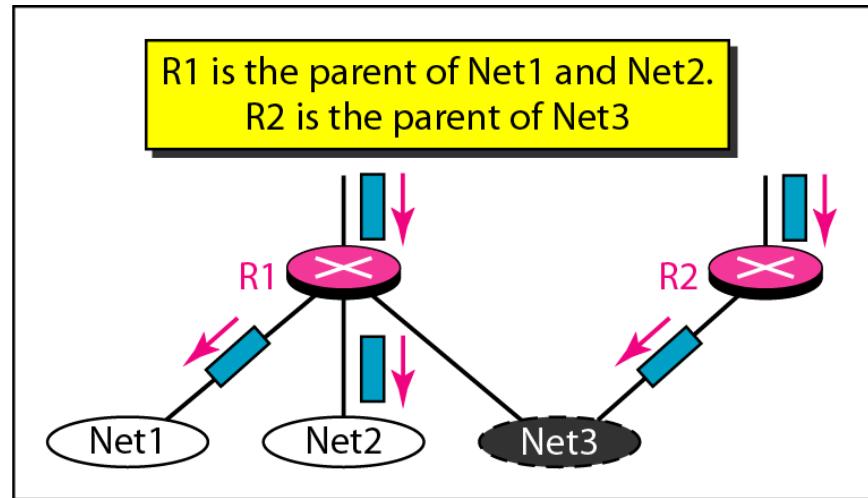


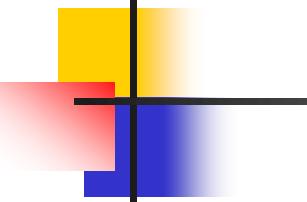
Figure 22.42 RPF Versus RPB



a. RPF



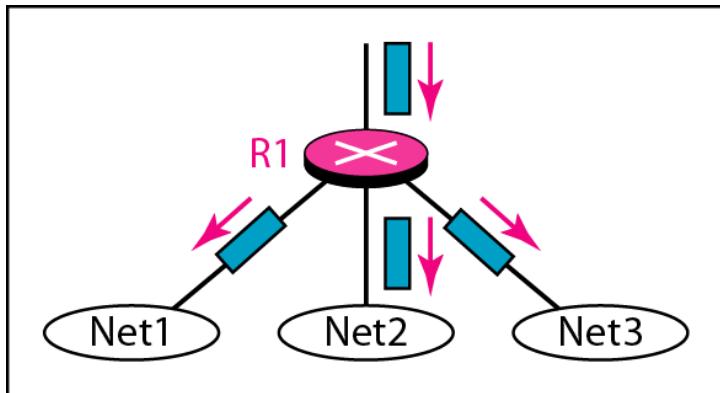
b. RPB



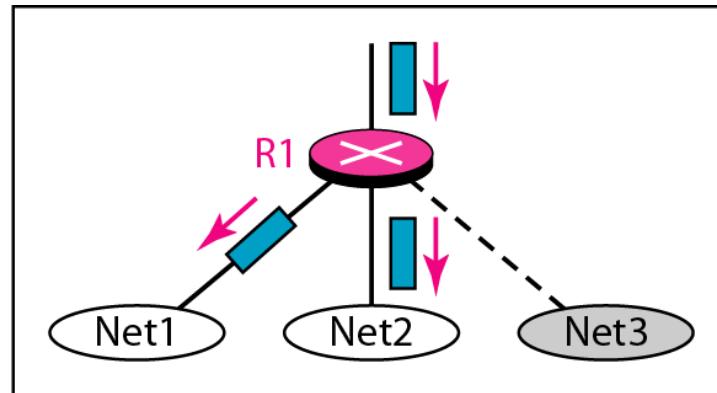
Note

RPB creates a shortest path broadcast tree from the source to each destination. It guarantees that each destination receives one and only one copy of the packet.

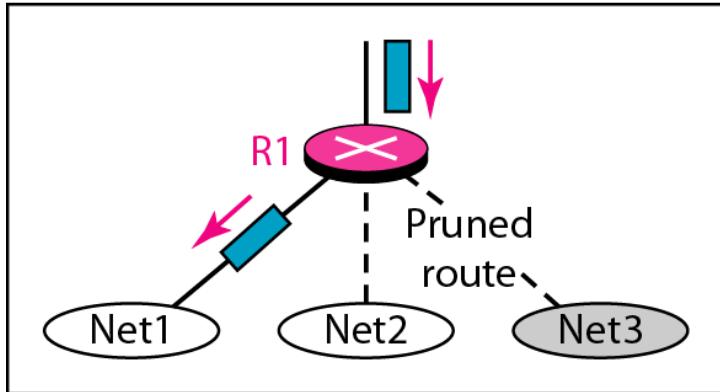
Figure 22.43 RPF, RPB, and RPM



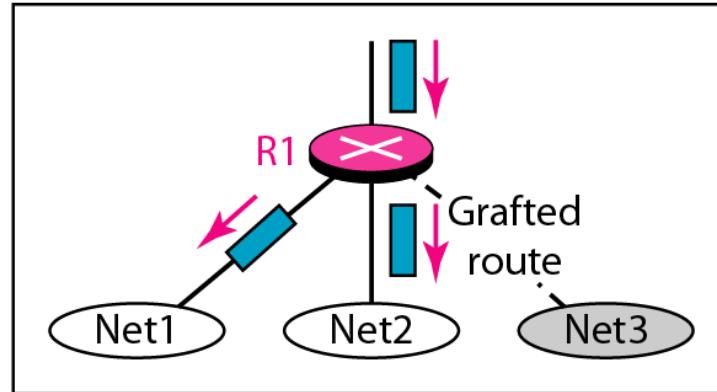
a. RPF



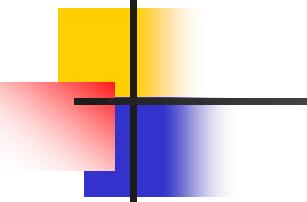
b. RPB



c. RPM (after pruning)



d. RPM (after grafting)



Note

RPM adds pruning and grafting to RPB to create a multicast shortest path tree that supports dynamic membership changes.

Figure 22.44 Group-shared tree with rendezvous router

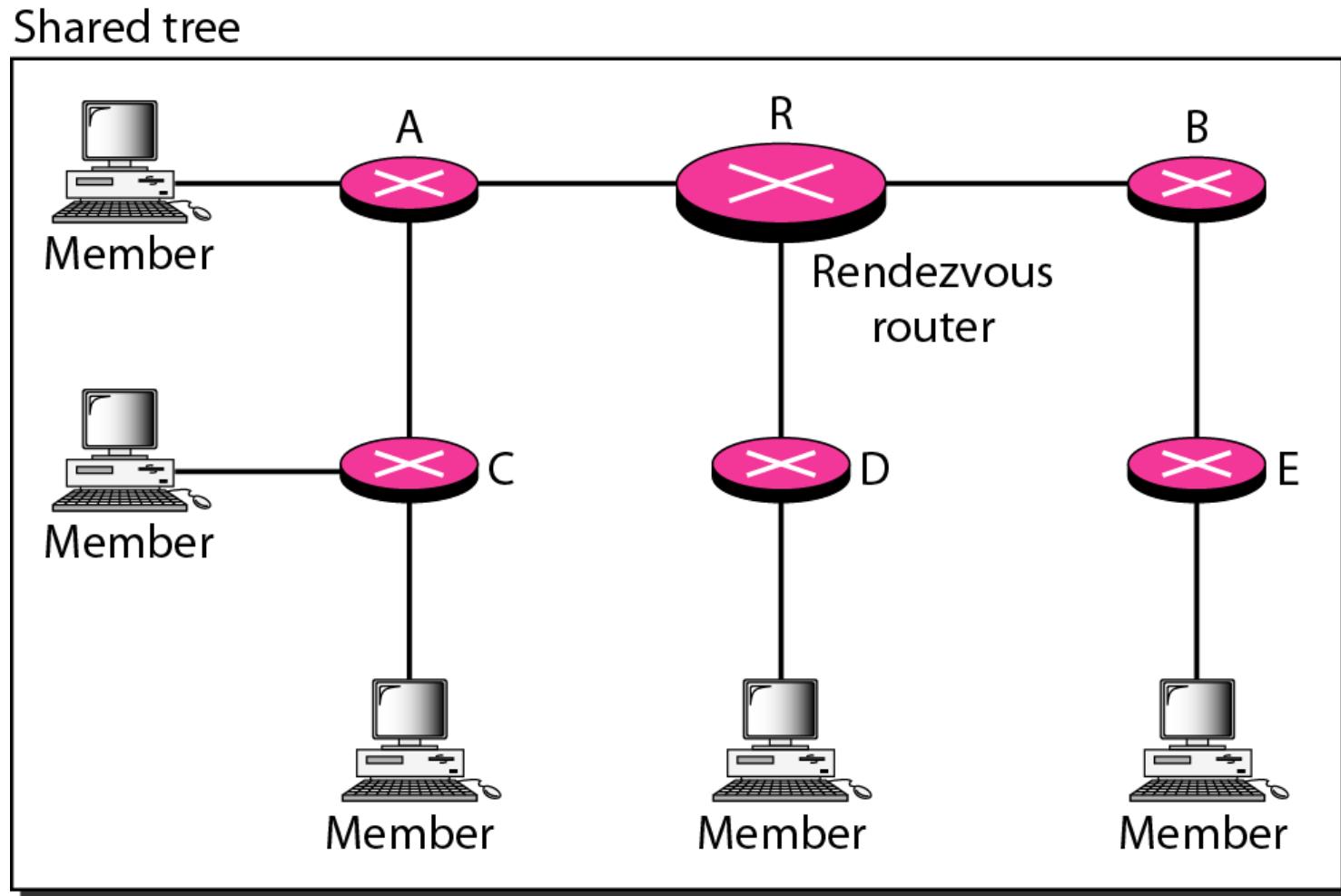
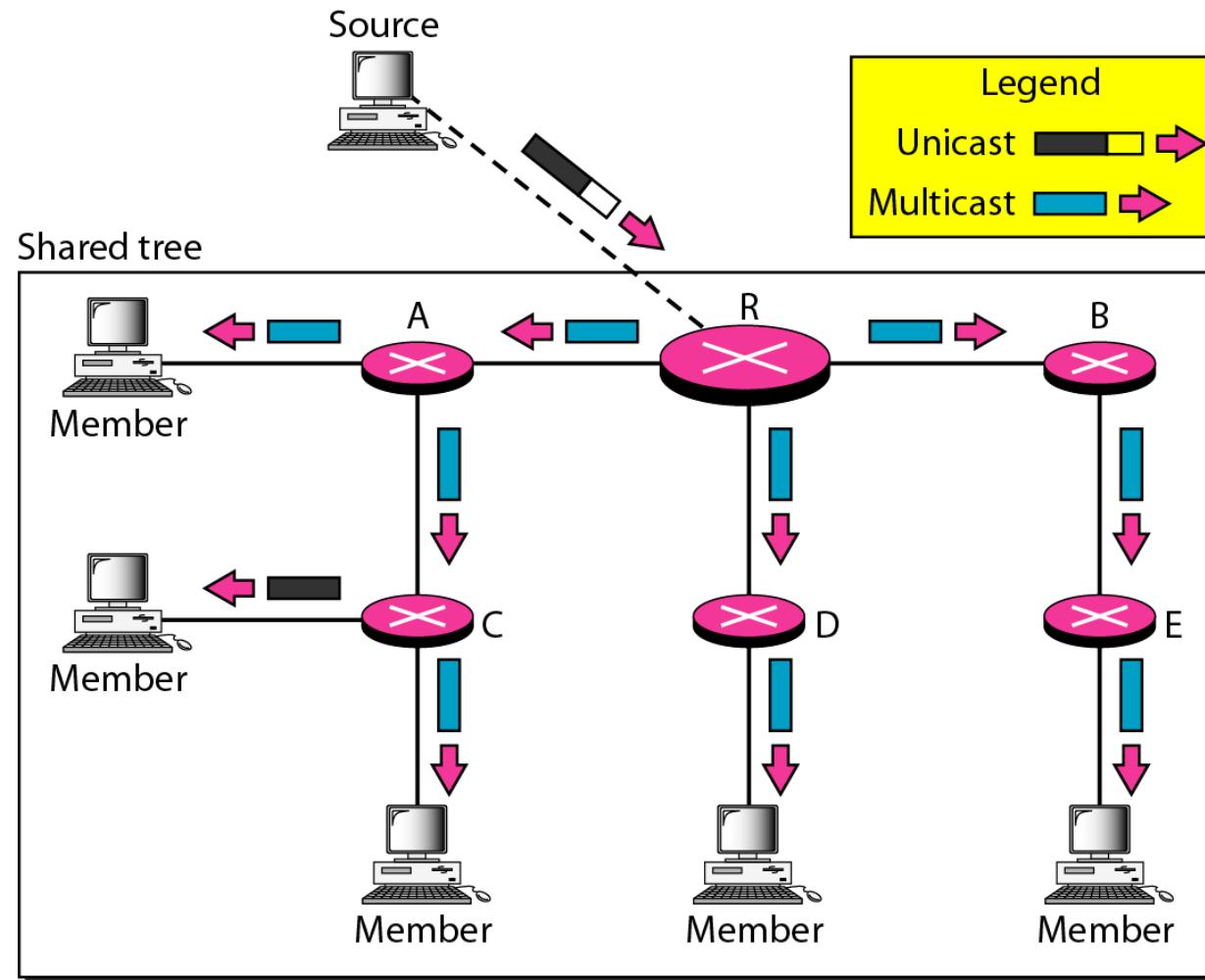
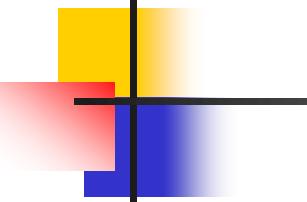


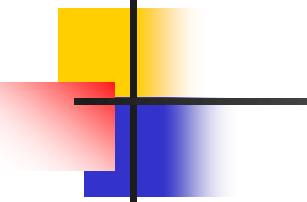
Figure 22.45 *Sending a multicast packet to the rendezvous router*





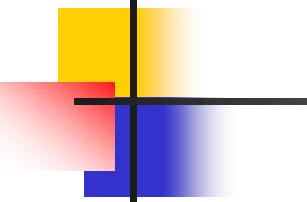
Note

In CBT, the source sends the multicast packet (encapsulated in a unicast packet) to the core router. The core router decapsulates the packet and forwards it to all interested interfaces.



Note

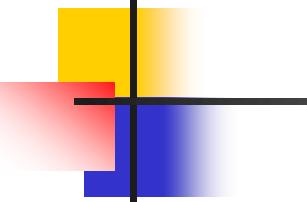
PIM-DM is used in a dense multicast environment, such as a LAN.



Note

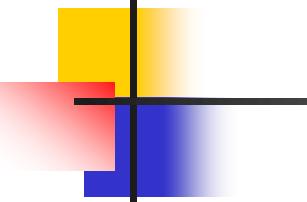
PIM-DM uses RPF and pruning and grafting strategies to handle multicasting.

However, it is independent of the underlying unicast protocol.



Note

PIM-SM is used in a sparse multicast environment such as a WAN.



Note

PIM-SM is similar to CBT but uses a simpler procedure.

Figure 22.46 Logical tunneling

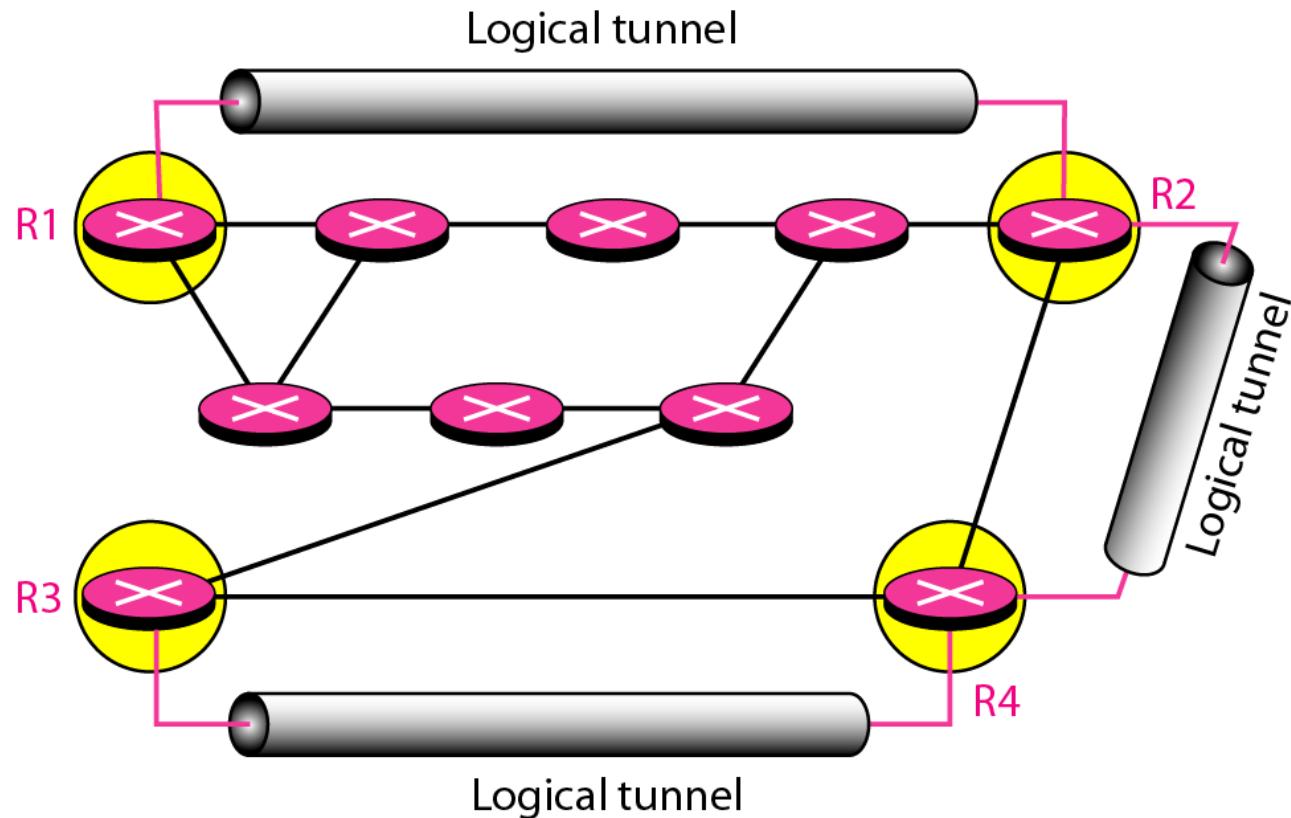
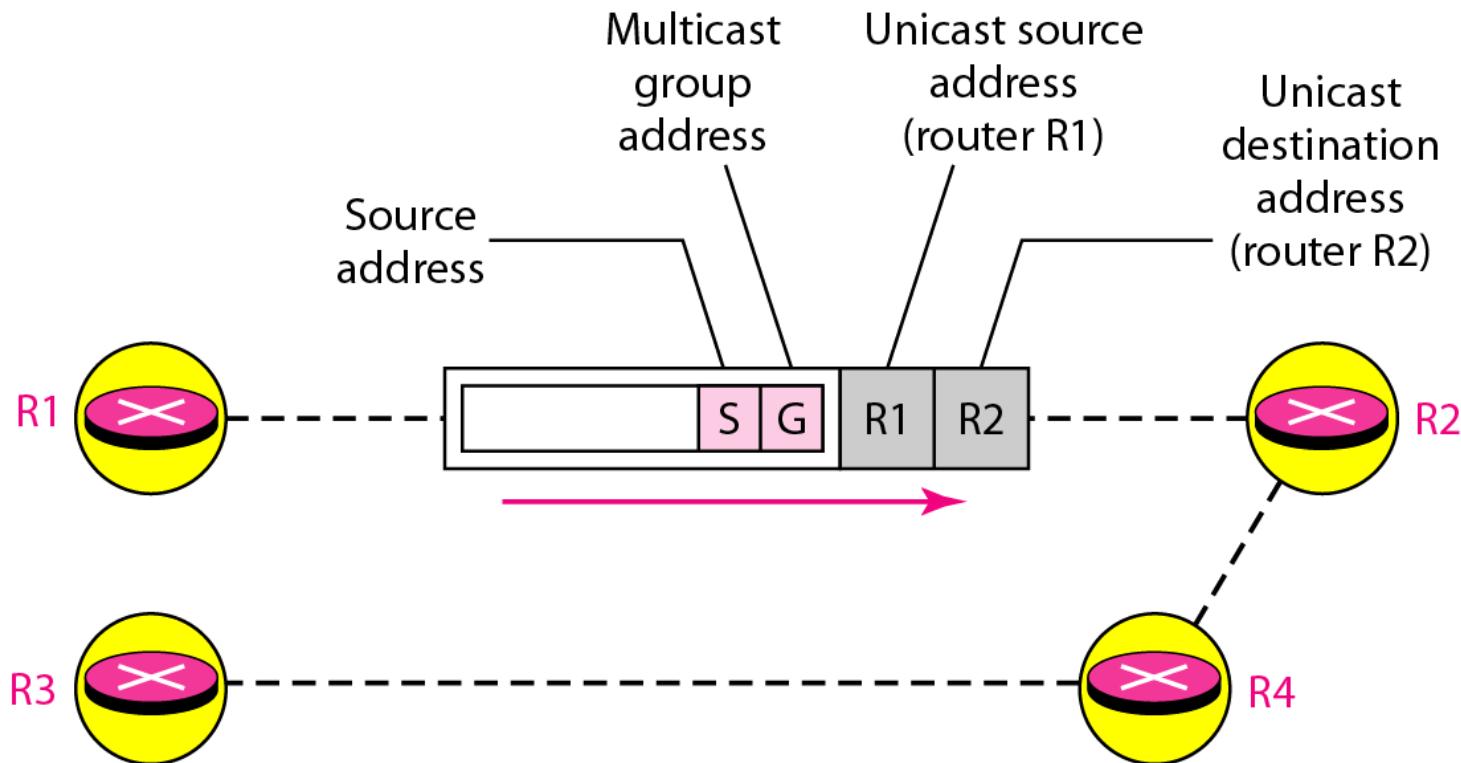


Figure 22.47 MBONE



IEEE STANDARDS

802.3,802.4,802.5

INTRODUCTION

- ❖ IEEE 802 refers to a family of IEEE standards
 - Dealing with local area network and metropolitan area network.
 - Restricted to networks carrying variable-size packets.
 - Specified in IEEE 802 map to the lower two layers
 - ✓ Data link layer
 - ✓ Physical layer
- ❖ The most widely used standards
 - ❖ .802.3 - Ethernet
 - ❖ 802.4 - Token Bus
 - ❖ 802.5 - Token Ring

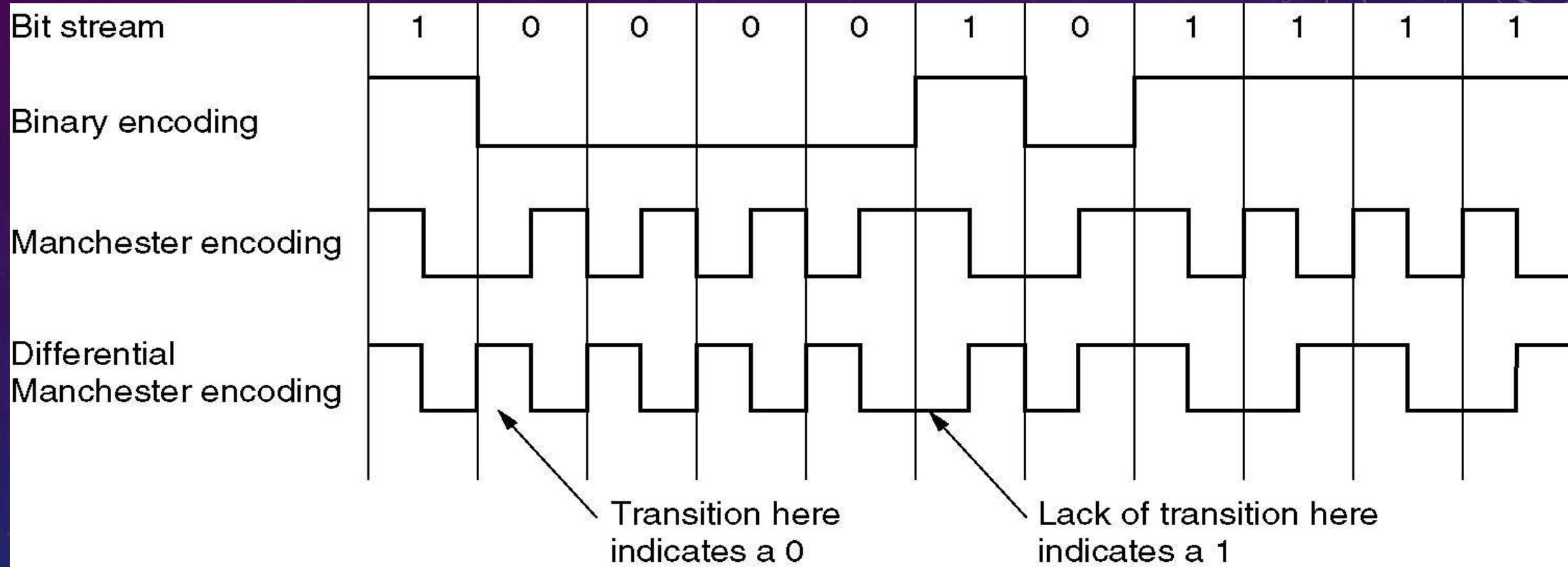
ETHERNET (IEEE 802.3)

- The IEEE 802.3 standard specifies the CSMA/CD (Carrier Sense Multiple Access with Collision Detection) media access control method. CSMA/CD is the most commonly employed access method for LANs using a bus or tree topology. It is the media access control method used by Ethernet[1].
- Most widely type used at present, with a huge installed base and considerable operational experience.
- Protocol is very simple
- Stations can be added without making the network down.
- The delay at low load is practically zero. (no token waiting)

ETHERNET CABLING

Name	Cable	Max. seg.	Nodes/seg.	Advantages
10Base5	Thick coax	500 m	100	Original cable; now obsolete
10Base2	Thin coax	185 m	30	No hub needed
10Base-T	Twisted pair	100 m	1024	Cheapest system
10Base-F	Fiber optics	2000 m	1024	Best between buildings

ETHERNET SIGNALLING



(a) Binary encoding, (b) Manchester encoding,
(c) Differential Manchester encoding.

CSMA/CD TRANSMISSION FRAME

Preamble (7 bytes)	Start of Frame Delimiter (1 byte)	Dest. Address (6 bytes)	Source Address (6 bytes)	Length (2 bytes)	Data bytes (0-1500 bytes)	Pad (0-46 bytes)	Frame Checksum (4 bytes)
-----------------------	---	----------------------------	-----------------------------	---------------------	------------------------------	---------------------	-----------------------------

FIGURE 1: CSMA/CD (IEEE 802.3) FRAME FORMAT

❖ Preamble

The preamble is responsible for providing the synchronization between the sending and receiving device. It is a series of 56 bits (7 bytes) of alternating 1s and 0s found at the beginning of the frame[2].

❖ Start of Frame Delimiter

The start frame delimiter follows the preamble. As its name implies, it indicates the start of the data frame. The start frame delimiter is 1 byte in length—made up of the following 8-bit sequence—10101011[3].

❖ Address Fields

Each of the address fields—the destination address and the source address—can be either 2 bytes or 6 bytes in length. If universal addressing is used, the addresses must be 6 bytes each. But if local addressing is used they may be either 2 or 6 bytes long. Both destination and source addresses must be of the same length for all devices on a given network[2].

❖ Length Count

This is a 2-byte field indicating the length of the data field that follows. It is needed to determine the length of the data field in those cases when a pad field is used[2].

❖ **Information Field**

The information field contains the actual data packet to be transmitted. Its length is variable[2].

❖ **Pad Field**

A pad field is used to ensure that the frame meets a minimum length requirement. A frame must contain a minimum number of bytes in order for stations to detect collisions accurately[2].

❖ **Frame Check Sequence**

The frame check field is used as an error-control mechanism. When the transmitting device assembles a frame, it performs a calculation on the bits in the frame. The algorithm used to perform this calculation always results in a 4-byte value. The sending device stores this value in the frame check sequence field[2].

TOKEN BUS (IEEE 802.4)

- The IEEE 802.4 standard specifies the Token-bus media access control method. It is one of two token passing access methods. IEEE 802.4 is based on a physical bus or tree topology. The Token-bus approach requires a station to have possession of a token in order to transmit. The token is passed from station to station in a logical ring[3].
- Uses highly reliable cable television equipments.
- It is more deterministic than 802.3, although repeated loss of token at critical times can introduce the uncertainness.
- Can easily handle shorter frames. (no limitation on frame size)
- It supports priorities and hence suitable for Real Time traffic.
- It also has excellent throughput and efficiency at high load.

TOKEN-BUS TRANSMISSION FRAME

Preamble (1 bytes)	Start of Frame (1 byte)	Frame Control (1 byte)	Destination Address (2/6 bytes)	Source Address (2/6 bytes)	Data (0-8182bytes)	Frame Checksum (4 bytes)	End of Frame (1 byte)
-----------------------	----------------------------	---------------------------	------------------------------------	-------------------------------	-----------------------	-----------------------------	--------------------------

FIGURE2 : Token Bus (IEEE 802.4) FRAME FORMAT

❖ Preamble

The preamble is responsible for providing the synchronization between the sending and receiving device. The length of this field and its contents depend on the modulation method being used and the speed of the network.

❖ Start of Frame

The start delimiter follows the preamble. As its name implies, it indicates the start of the data frame. The start frame delimiter is 1 byte in length and contains a signaling pattern that is always different from the data—the actual signaling pattern varies with the encoding scheme used[5].

❖ Frame Control

This field identifies the type of frame being sent—Logical Link Control data frames, token control frames, Media Access Control management data frames, or special-purpose data frames.

❖ Address Fields

Each of the address fields—the destination address and the source address—can be either 2 bytes (16-bit addresses) or 6 bytes (48-bit addresses) in length. If universal addressing is used, the addresses must be 6 bytes each. But if local addressing is used they may be either 2 or 6 bytes long. Both destination and source addresses must be of the same length for all devices on a given network. The source address must be for an individual device. The destination address can be an individual address, a group address or a broadcast address[5].

❖ **Information Field**

The information field contains the actual data packet to be transmitted. Its length is variable. It may contain a Logical Link protocol data unit, token control data, management data or special-purpose data—as indicated in the frame control field.

❖ **Frame Checksum**

The frame check field is used as an error control mechanism. When the transmitting device assembles a frame, it performs a calculation on the bits in the frame. The algorithm used to perform this calculation always results in a 4-byte value. The sending device stores this value in the frame check sequence field. When the destination device receives the frame, it performs the same calculation and compares the result to that in the frame check sequence field. If the two values are the same, the transmission is assumed to be correct. If the two values are different, the destination device can request a retransmission of the frame[5].

❖ **End of Frame**

The end delimiter marks the end of the frame and shows the position of the frame check sequence field. Just as with the start delimiter, the signaling value is always different from the data.

TOKEN RING (IEEE 802.5)

- IEEE 802.5 is the second of the token passing access control methods. Token-ring is most commonly used in a network structure following both a logical and physical ring topology. The right to transmit is controlled by a token[3].
- It uses point-to-point connections and hence the engineering is easy.
- Any transmission media can be used.
- The use of wire centers make the token ring the only LAN that can detect and eliminate cable failures automatically.
- Like 802.4, priorities also possible, although the scheme is not as fair.
- Very short and very large frames both are possible.
- At very high load, the throughput and efficiency are excellent.

TOKEN-RING TRANSMISSION FRAME

Start of Frame (1 byte)	Access Control (1 byte)	Frame Control (1 byte)	Source Address (6 bytes)	Destination Address (6 bytes)	Data (unlimited)	Frame Checksum (4 bytes)	Frame Status (1 byte)	End of Frame (1 byte)
----------------------------	----------------------------	---------------------------	-----------------------------	----------------------------------	---------------------	-----------------------------	--------------------------	--------------------------

FIGURE 3: Token-Ring (IEEE 802.5) FRAME FORMAT

❖ Start of frame

The starting delimiter indicates the start of the data frame. It uses a unique signal pattern that does not correspond to either a 0 or 1 bit. These are known as nondata values and ensure that no data sequence will ever be mistaken for a delimiter.

❖ Access Control Field

This field identifies whether the frame is a data frame or a token. It contains a bit used to identify a constantly busy token, a priority bit and reservations bits.

❖ Frame Control Field

This field identifies the frame type and for certain types of control frames, the function it is to perform.

❖ Address Fields

Each of the address fields—the destination address and the source address—can be either 2 bytes (16-bit addresses) or 6 bytes (48-bit addresses) in length. If universal addressing is used, the addresses must be 6 bytes each. But if local addressing is used they may be either 2 or 6 bytes long. Both destination and source addresses must be of the same length for all devices on a given network.

The source address must be for an individual device. The destination address can be an individual address, a group address or a broadcast address.

❖ **Information Field**

The information field contains the actual data packet to be transmitted. This can be either a protocol data unit being passed from the logical link control sublayer or control information supplied by the media access control sublayer. Its length is variable anywhere from 0 to 17800 bytes in length.

❖ **Frame Check Sequence**

The frame check field is used as an error control mechanism. When the transmitting device assembles a frame, it performs a calculation on the bits in the frame. The algorithm used to perform this calculation always results in a 4 byte value. The sending device stores this value in the frame check sequence field. When the destination device receives the frame, it performs the same calculation and compares the result to that in the frame check sequence field. If the two values are the same, the transmission is assumed to be correct. If the two values are different, the destination station can request a retransmission of the frame.

❖ **Ending Delimiter**

This identifies the end of the frame by containing nondata values. It also contains bits used to identify whether or not it is the last frame in a multiframe transmission and if an error has been detected by any station.

❖ **Frame Status Field**

The frame status field contains the address recognized and frame copied control bits.

REFERENCE

- [1]. <http://www.erg.abdn.ac.uk/users/gorry/course/lan-pages/csma-cd.html>
- [2]. <http://www.erg.abdn.ac.uk/users/gorry/course/lan-pages/enet.html>
- [3]. G. Watson, A. Albrecht, J. Curcio, D. Dove, S. Goody, J. Grinham, M.P. Spratt, and P.A. Thaler.
The demand priority MAC protocol. *IEEE Network*, 9(1):28–34, Jan./Feb. 1995.
- [4]. R. Yavatkar, P. Pai, and R. Finkel. A reservation-based CSMA protocol for integrated manufacturing networks. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(8):1247–1258, Aug. 1994.
- [5]. Q. Zheng and K.G. Shin. On the ability of establishing real-time channels in point-to-point packet-switched networks. *IEEE Transactions on Communications*, 42(2/3/4):1096–1105, Feb./Mar./Apr. 1994.

THANK YOU

IEEE STANDARDS

- ✓ Like an architect Who designs the entire layout of a house that includes plumbing, electrical, wiring and so on, a network architect too considers an entire bunch of factors before designing a network. Apart from network topologies, different IEEE standards like Ethernet, Token Ring, and FDDI and so on.
- ✓ The IEEE standards form a basis for promoting standards related to maintaining networks with safe and standardized connections, encryption, and error checking algorithms and network media.
- ✓ All these standards fall under IEEE's Project 802 that was implemented to standardize the logical and physical elements of networks.
- ✓ When it comes to troubleshooting issues related to networks, the most crucial aspect is identifying the problem area and solving it. Problems may occur in connections, in topologies in devices and in many such components or services associated within networks.
- ✓ The IEEE networking specifications pertaining to connectivity, error checking algorithms, encryption, networking media, emerging technologies, etc come under IEEE's Projects 802 which was implemented in order to standardize the physical and logical elements of a network. IEEE developed the 802 standards before ISO came up with the OSI model. But the IEEE 802 standards can be applied to the OSI model's layers.

IEEE 802 Standards

Standard	Name	Topic
802.1	Internetworking	Routing,Bridging, and network-to-network Communications
802.2	Logical Link Control	Error and flow control over data frames
802.3	Ethernet LAN	All forms of Ethernet media and interfaces
802.4	Token BUS LAN	All forms of Token Bus media and interfaces
802.5	Token Ring LAN	All forms of Token Ring media and interfaces
802.6	Metropolitan Area Network	MAN technologies,Addressing, and Services
802.7	Broadband technical Advisory Group	Broadband network media,interfaces, adn other Equipments
802.8	Fiber Optic Technical Advisory Group	Fiber Optic media used in token-passing Networks like FDDI
802.9	Integrated Voice/ Data Network	Integration of voice and data traffic Over a single network medium
802.10	Netwok Security	Network access controls,encryption,Certification, and other Security topics
802.11	Wireless Networks	Standards for wireless networking for many different broadcast frquencies and usage techniques
802.12	High-Speed Networking	A variety of 100 Mbps-plus technologies,including 100 BASE-VG
802.14	Cable Broadband LANs and MANs	Standards for designing network over coaxial cable-based broadband connections.
802.15	Wireless Personal Area Networks	The coexistence of wireless personal area networks with Others wireless devices in unlicensed frequency bands.
802.16	Broadband Wireless Access	The atmospheric interface and related functions associated with Wireless Local Loop(WLL)

- IEEE 802 refers to a family of IEEE standards
 - Dealing with local area network and metropolitan area network.
 - Restricted to networks carrying variable-size packets.
 - Specified in IEEE 802 map to the lower two layers
 - ✓ Data link layer
 - ✓ Physical layer

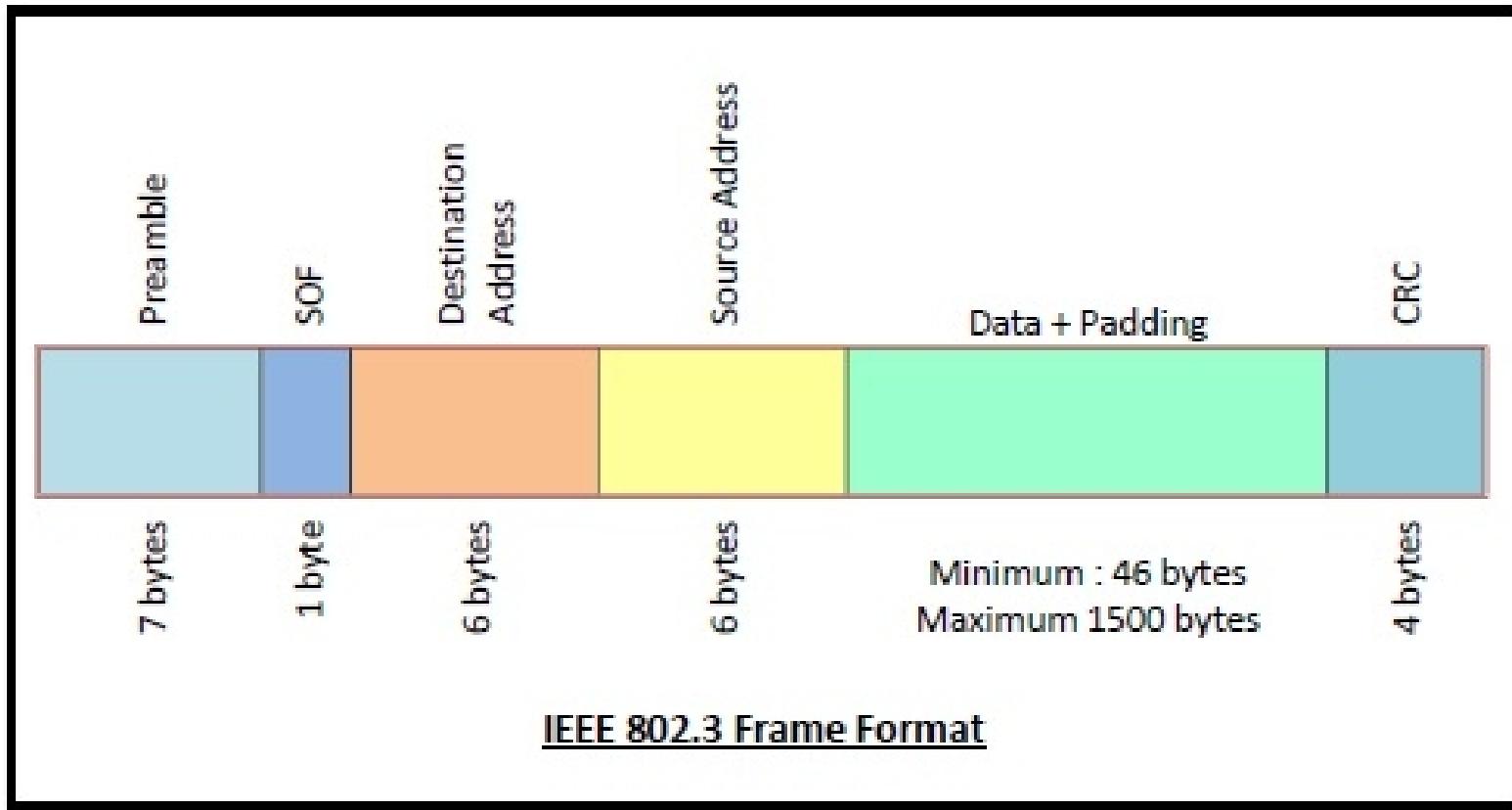
ETHERNET (IEEE 802.3)

- ❑ Ethernet is a set of technologies and protocols that are used primarily in LANs. It was first standardized in 1980s by IEEE 802.3 standard.
- ❑ IEEE 802.3 defines the physical layer and the medium access control (MAC) sub-layer of the data link layer for wired Ethernet networks.
- ❑ Ethernet is classified into two categories: classic Ethernet and switched Ethernet.
- ❑ Classic Ethernet is the original form of Ethernet that provides data rates between 3 to 10 Mbps. The varieties are commonly referred as 10BASE-X. Here, 10 is the maximum throughput, i.e. 10 Mbps, BASE denoted use of baseband transmission, and X is the type of medium used.
- ❑ A switched Ethernet uses switches to connect to the stations in the LAN. It replaces the repeaters used in classic Ethernet and allows full bandwidth utilization.
- ❑ The IEEE 802.3 standard specifies the CSMA/CD (Carrier Sense Multiple Access with Collision Detection) media access control method. CSMA/CD is the most commonly employed access method for LANs using a bus or tree topology. It is the media access control method used by Ethernet.
 - ✓ Most widely used type at present, with a huge installed base and considerable operational experience.
 - ✓ Protocol is very simple
 - ✓ Stations can be added without making the network down.
 - ✓ The delay at low load is practically zero. (no token waiting)

ETHERNET CABLING

Name	Cable	Max. seg.	Nodes/seg.	Advantages
10Base5	Thick coax	500 m	100	Original cable; now obsolete
10Base2	Thin coax	185 m	30	No hub needed
10Base-T	Twisted pair	100 m	1024	Cheapest system
10Base-F	Fiber optics	2000 m	1024	Best between buildings

ETHERNET (IEEE 802.3) Frame Format



ETHERNET (IEEE 802.3) Frame Format

- ✓ **Preamble:** The preamble is responsible for providing the synchronization between the sending and receiving device. It is a series of 56 bits (7 bytes) of alternating 1s and 0s found at the beginning of the frame.
- ✓ **Start of Frame Delimiter:** The start frame delimiter follows the preamble. As its name implies, it indicates the start of the data frame. It is a 1 byte field in a IEEE 802.3 frame that contains an alternating pattern of ones and zeros ending with two ones. The start frame delimiter is 1 byte in length—made up of the following 8-bit sequence—10101011
- ✓ **Address Fields:**
 - ✓ **Destination Address:** It is a 6 byte field containing physical address of destination stations.
 - ✓ **Source Address:** It is a 6 byte field containing the physical address of the sending station.
- ✓ **Length:** This is a 2-byte field indicating the length of the data field that follows. It is needed to determine the length of the data field in those cases when a pad field is used

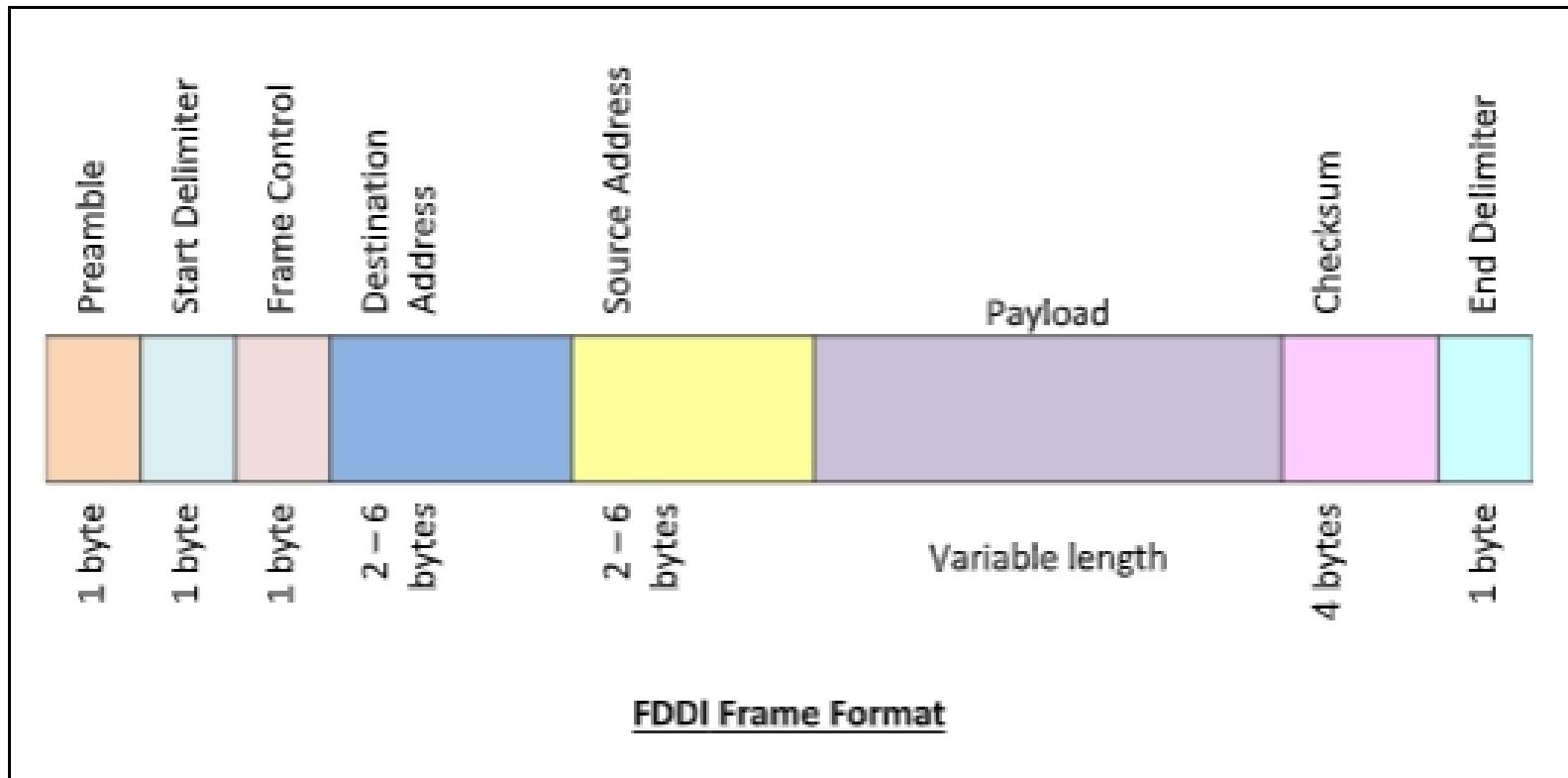
ETHERNET (IEEE 802.3) Frame Format

- ✓ **Data:** This is a variable sized field carries the data from the upper layers. The maximum size of data field is 1500 bytes.
- ✓ **Padding:** This is added to the data to bring its length to the minimum requirement of 46 bytes.
- ✓ **Frame Check Sequence/CRC:** The frame check field is used as an error-control mechanism. When the transmitting device assembles a frame, it performs a calculation on the bits in the frame. The algorithm used to perform this calculation always results in a 4- byte value. The sending device stores this value in the frame check sequence field

TOKEN BUS (IEEE 802.4)

- ✓ The IEEE 802.4 standard specifies the Token-bus media access control method. It is one of two token passing access methods. IEEE 802.4 is based on a physical bus or tree topology. The Token-bus approach requires a station to have possession of a token in order to transmit. The token is passed from station to station in a logical ring.
- ✓ Uses highly reliable cable television equipment's.
- ✓ It is more deterministic than 802.3, although repeated loss of token at critical times can introduce the uncertainness.
- ✓ Can easily handle shorter frames. (no limitation on frame size)
- ✓ It supports priorities and hence suitable for Real Time traffic.
- ✓ It also has excellent throughput and efficiency at high load.

TOKEN BUS (IEEE 802.4) Frame Format



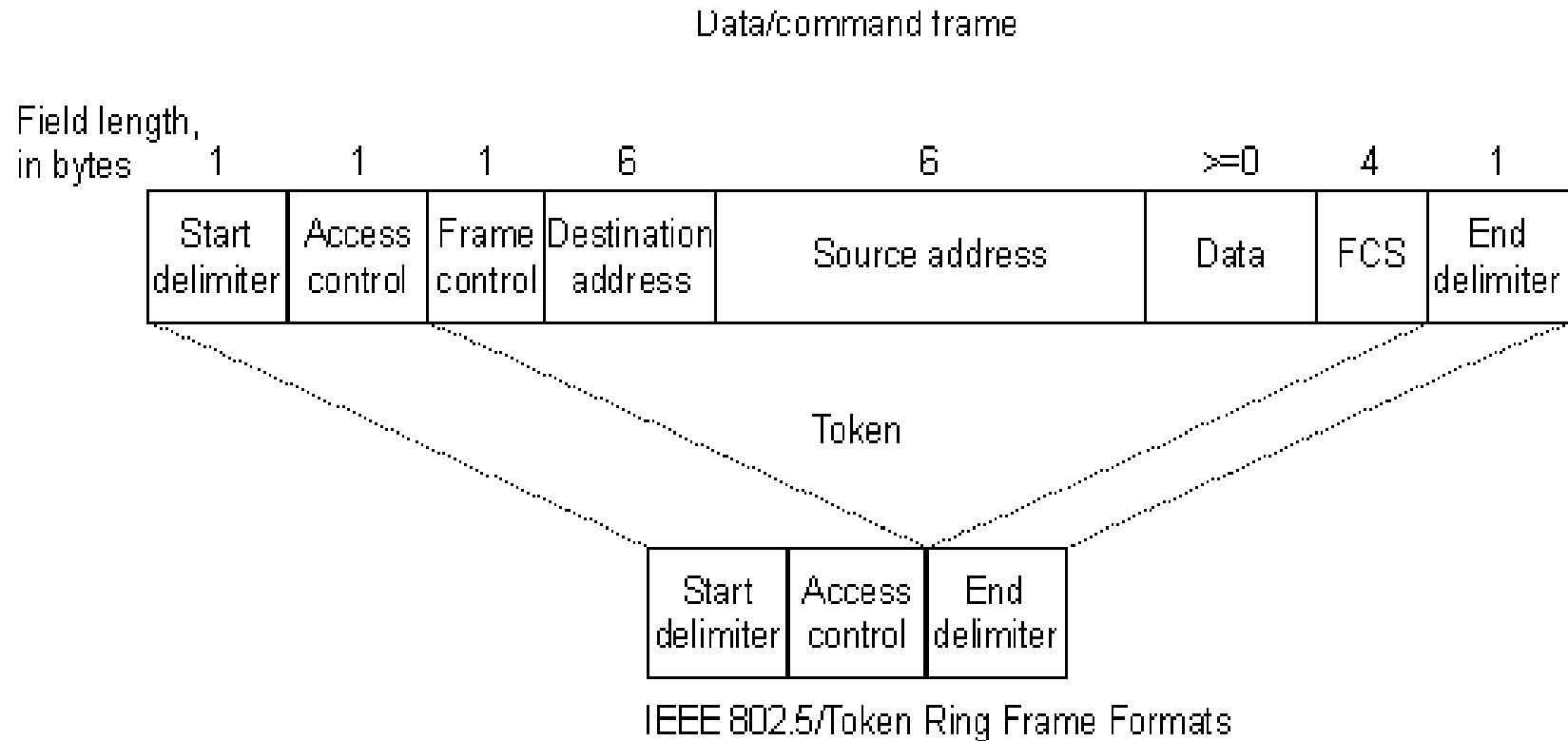
TOKEN BUS (IEEE 802.4) Frame Format

- ✓ **Preamble:** 1 byte for synchronization.
- ✓ **Start Delimiter:** 1 byte that marks the beginning of the frame.
- ✓ **Frame Control:** 1 byte that specifies whether this is a data frame or control frame.
- ✓ **Destination Address:** 2-6 bytes that specifies address of destination station.
- ✓ **Source Address:** 2-6 bytes that specifies address of source station.
- ✓ **Payload:** A variable length field that carries the data from the network layer.
- ✓ **Checksum:** 4 bytes frame check sequence for error detection.
- ✓ **End Delimiter:** 1 byte that marks the end of the frame.

TOKEN RING (IEEE 802.5)

- ✓ IEEE 802.5 is the second of the token passing access control methods. Token-ring is most commonly used in a network structure following both a logical and physical ring topology. The right to transmit is controlled by a token.
- ✓ It uses point-to-point connections and hence the engineering is easy.
- ✓ Any transmission media can be used.
- ✓ The use of wire centers make the token ring the only LAN that can detect and eliminate cable failures automatically.
- ✓ Like 802.4, priorities also possible, although the scheme is not as fair.
- ✓ Very short and very large frames both are possible.
- ✓ At very high load, the throughput and efficiency are excellent.

TOKEN RING (IEEE 802.5)Frame Format



TOKEN RING (IEEE 802.5)Frame Format

- ✓ **Start of frame:** The starting delimiter indicates the start of the data frame. It uses a unique signal pattern that does not correspond to either a 0 or 1 bit. These are known as non data values and ensure that no data sequence will ever be mistaken for a delimiter.
- ✓ **Access Control Field:** This field identifies whether the frame is a data frame or a token. It contains a bit used to identify a constantly busy token, a priority bit and reservations bits.
- ✓ **Frame Control Field:** This field identifies the frame type and for certain types of control frames, the function it is to perform.
- ✓ **Address Fields:** Each of the address fields—the destination address and the source address—can be either 2 bytes (16-bit addresses) or 6 bytes (48-bit addresses) in length. If universal addressing is used, the addresses must be 6 bytes each. But if local addressing is used they may be either 2 or 6 bytes long. Both destination and source addresses must be of the same length for all devices on a given network. The source address must be for an individual device. The destination address can be an individual address, a group address or a broadcast address.

TOKEN RING (IEEE 802.5)Frame Format

- ✓ **Information Field:** The information field contains the actual data packet to be transmitted. This can be either a protocol data unit being passed from the logical link control sub layer or control information supplied by the media access control sub layer. Its length is variable anywhere from 0 to 17800 bytes in length.
- ✓ **Frame Check Sequence:** The frame check field is used as an error control mechanism. When the transmitting device assembles a frame, it performs a calculation on the bits in the frame. The algorithm used to perform this calculation always results in a 4 byte value. The sending device stores this value in the frame check sequence field. When the destination device receives the frame, it performs the same calculation and compares the result to that in the frame check sequence field. If the two values are the same, the transmission is assumed to be correct. If the two values are different, the destination station can request a retransmission of the frame.
- ✓ **Ending Delimiter:** This identifies the end of the frame by containing non data values. It also contains bits used to identify whether or not it is the last frame in a multi frame transmission and if an error has been detected by any station.
- ✓ **Frame Status Field:** The frame status field contains the address recognized and frame copied control bits.

TOKEN RING (IEEE 802.5)Frame Format

- ✓ **Information Field:** The information field contains the actual data packet to be transmitted. This can be either a protocol data unit being passed from the logical link control sub layer or control information supplied by the media access control sub layer. Its length is variable anywhere from 0 to 17800 bytes in length.
- ✓ **Frame Check Sequence:** The frame check field is used as an error control mechanism. When the transmitting device assembles a frame, it performs a calculation on the bits in the frame. The algorithm used to perform this calculation always results in a 4 byte value. The sending device stores this value in the frame check sequence field. When the destination device receives the frame, it performs the same calculation and compares the result to that in the frame check sequence field. If the two values are the same, the transmission is assumed to be correct. If the two values are different, the destination station can request a retransmission of the frame.
- ✓ **Ending Delimiter:** This identifies the end of the frame by containing non data values. It also contains bits used to identify whether or not it is the last frame in a multi frame transmission and if an error has been detected by any station.
- ✓ **Frame Status Field:** The frame status field contains the address recognized and frame copied control bits.