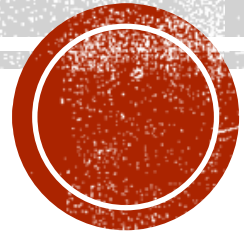
MODULE 2

DATA LINK LAYER: Flow Control and Error control – Stop and Wait – Go Back N ARQ – Selective Repeat ARQ – Sliding Window Techniques – HDLC – LAN – Ethernet IEEE 802.3 (Only Description) IEEE 802.4 and IEEE 802.5 – IEEE 802.11–FDDI – Bridges



FRAMING

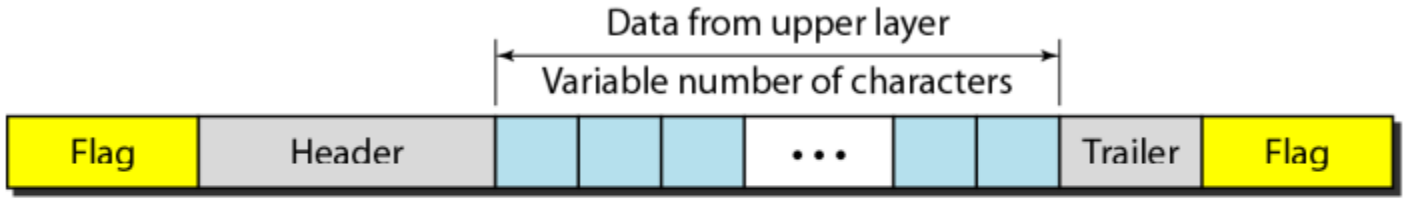
- Data transmission in the physical layer means moving bits in the form of a signal from source to destination.
- The physical layer provides bit synchronization to ensure that the sender and receiver use the same bit duration and timing.
- Data link layer packs data into frames.
- **Framing** in the data link layer separates a message from one source to a destination or some other message to other destinations, by adding a sender address and a destination address.

TYPES OF FRAMING

- Fixed Size Framing
  - ❖ Boundaries need not be defined
  - ❖ Ex: ATM WAN
- Variable Size Framing
  - ❖ End of frame and beginning of frame need to be defined
  - ❖ Ex: LAN

  - ✓ Character Oriented Protocols
  - ✓ Bit Oriented Protocols

Figure 11.1 *A frame in a character-oriented protocol*

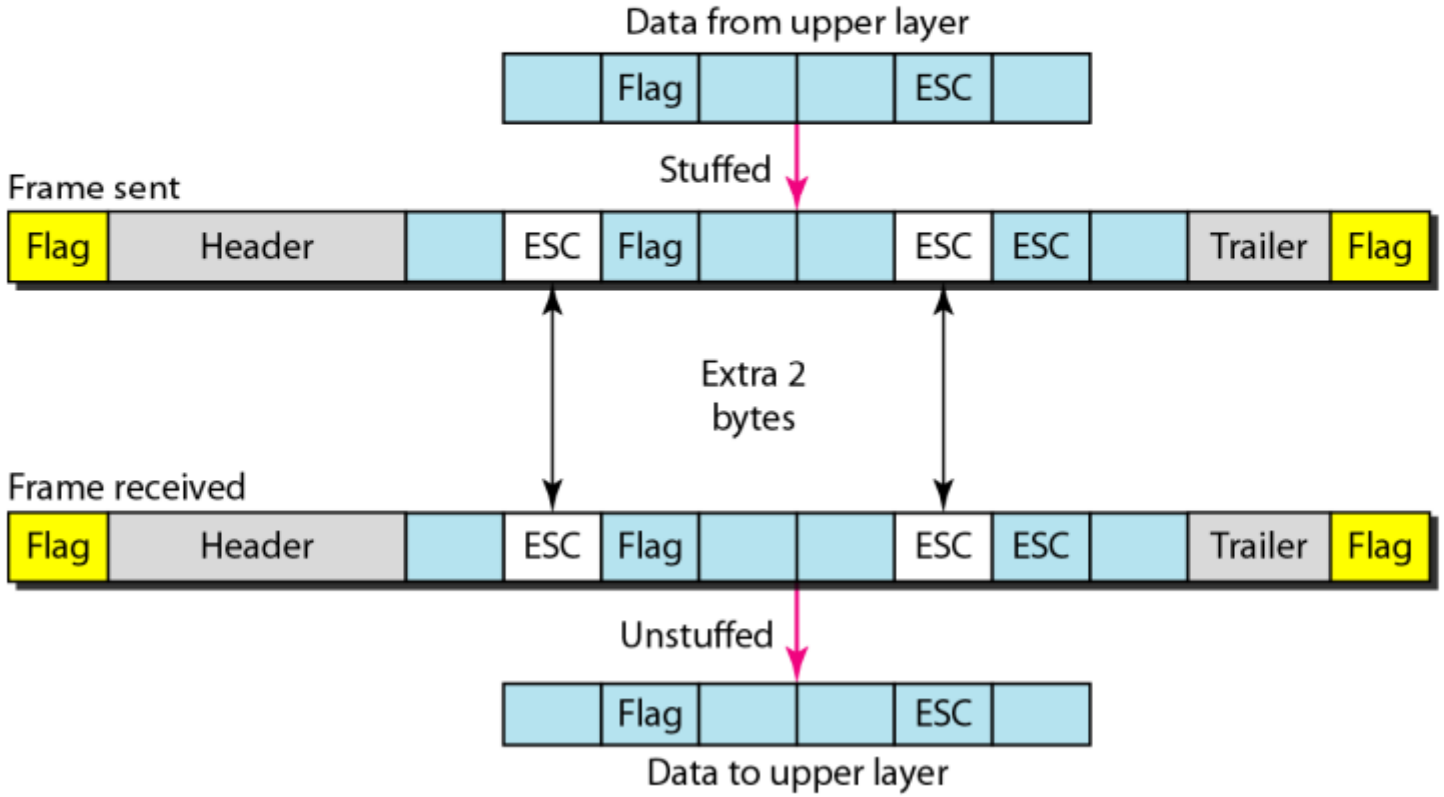


BYTE STUFFING OR CHARACTER STUFFING

- A special byte is added to the data section of the frame when there is a character with the same pattern as the flag.

- This byte is usually called the ESCAPE CHARACTER (ESC), which has a predefined bit pattern.

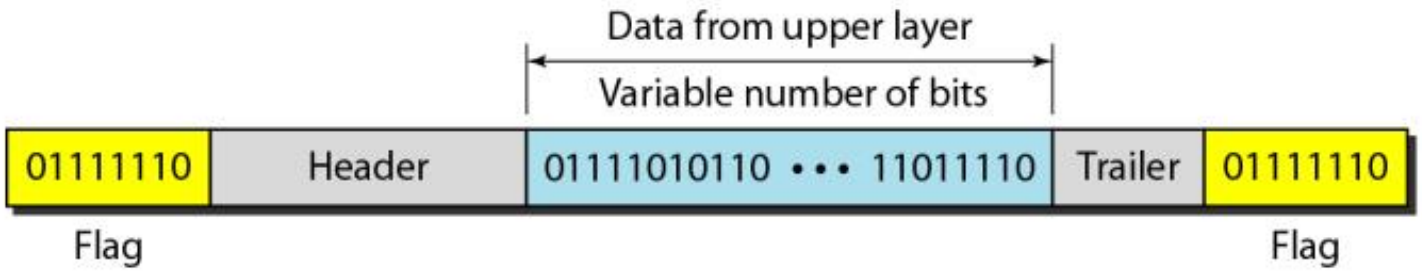
Figure 11.2 *Byte stuffing and unstuffing*



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*



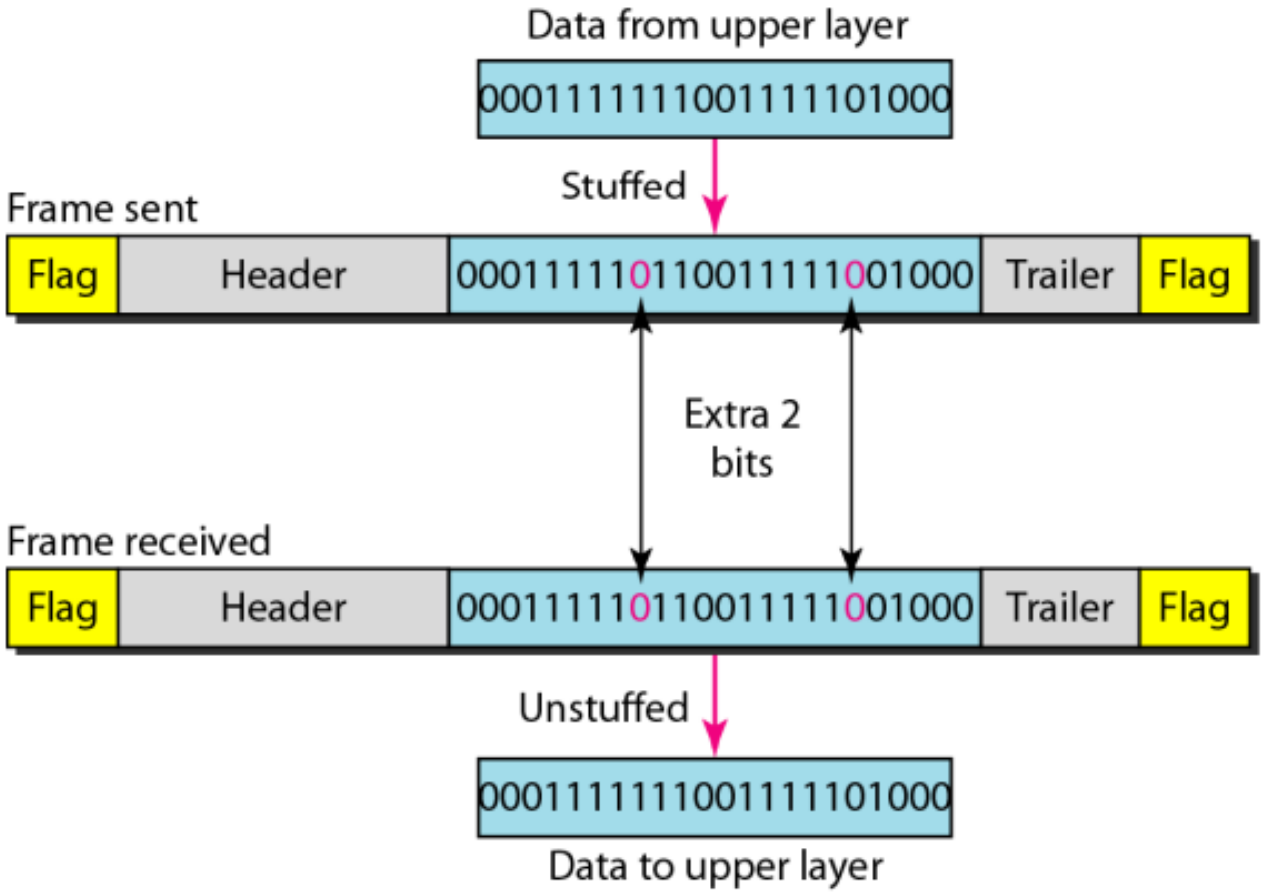
BIT STUFFING

- Here if a 0 and five consecutive 1 bits are encountered, an extra 0 is added.
- This extra stuffed bit is eventually removed from the data by the receiver.

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*



FLOW AND ERROR CONTROL

- Why is it required ?

- Flow control + Error control = Data Link Control

FLOW CONTROL

- Flow control is a primary duty of the data link layer.
- Flow control coordinates the amount of data that can be sent before receiving an acknowledgement
- Flow control protocol answers the following question:

Q How much data the sender can transmit before it must wait for an acknowledgement from receiver

- The flow of data must not overwhelm the receiver
- Incoming data must be checked and processed before they can be used.
- The rate of such processing is slower than the rate of transmission.
- Thus, the receiver has a block of memory, called BUFFER
- If Buffer is full, the receiver must be able to tell the sender to halt transmission until it is once again able to receive.

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.

ERROR CONTROL

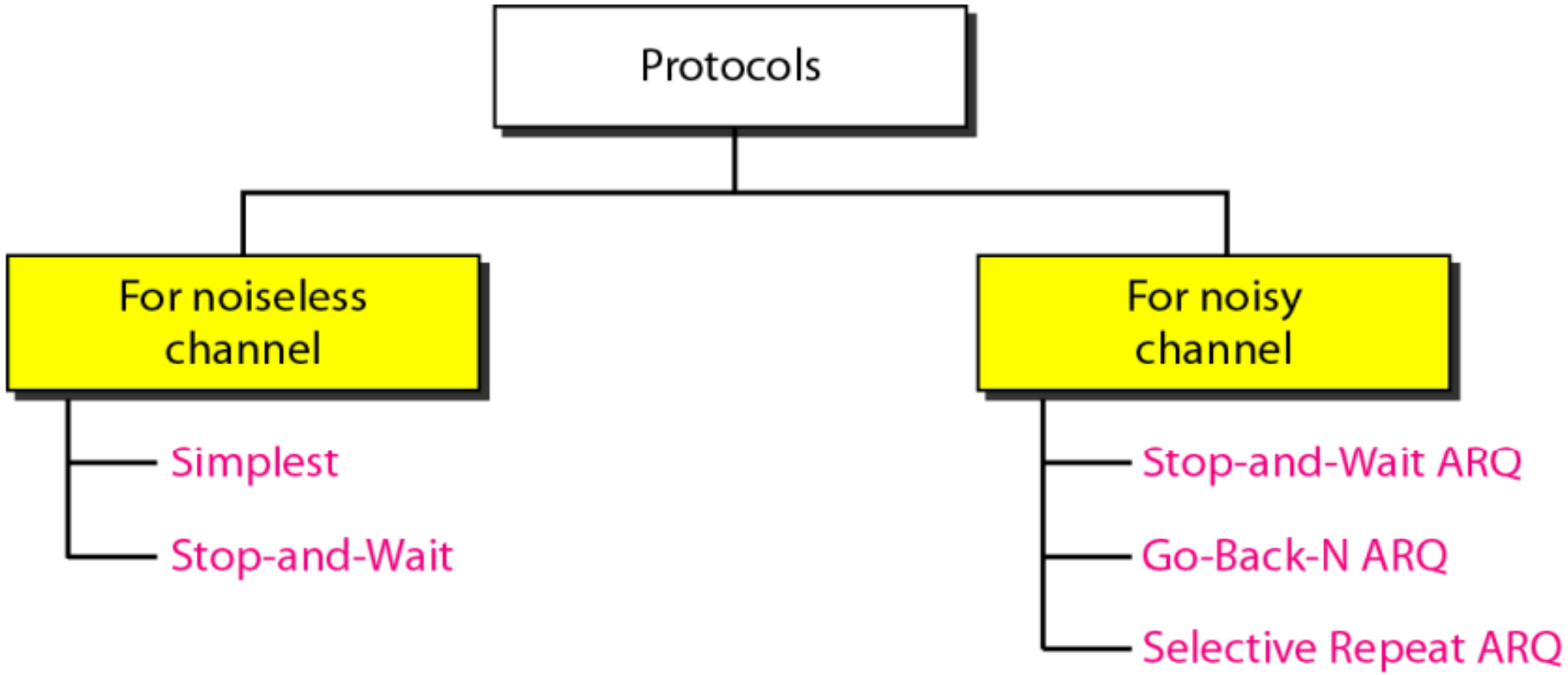
- Error control = Error Detection + Error Correction
- It allows the receiver to inform the sender of any frames lost or damaged in transmission and co-ordinates the retransmission of those frames by the sender
- Anytime an error is detected in an exchange, specified frames are retransmitted. This process is called **AUTOMATIC REPEAT REQUEST (ARQ)**

Note

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

PROTOCOLS

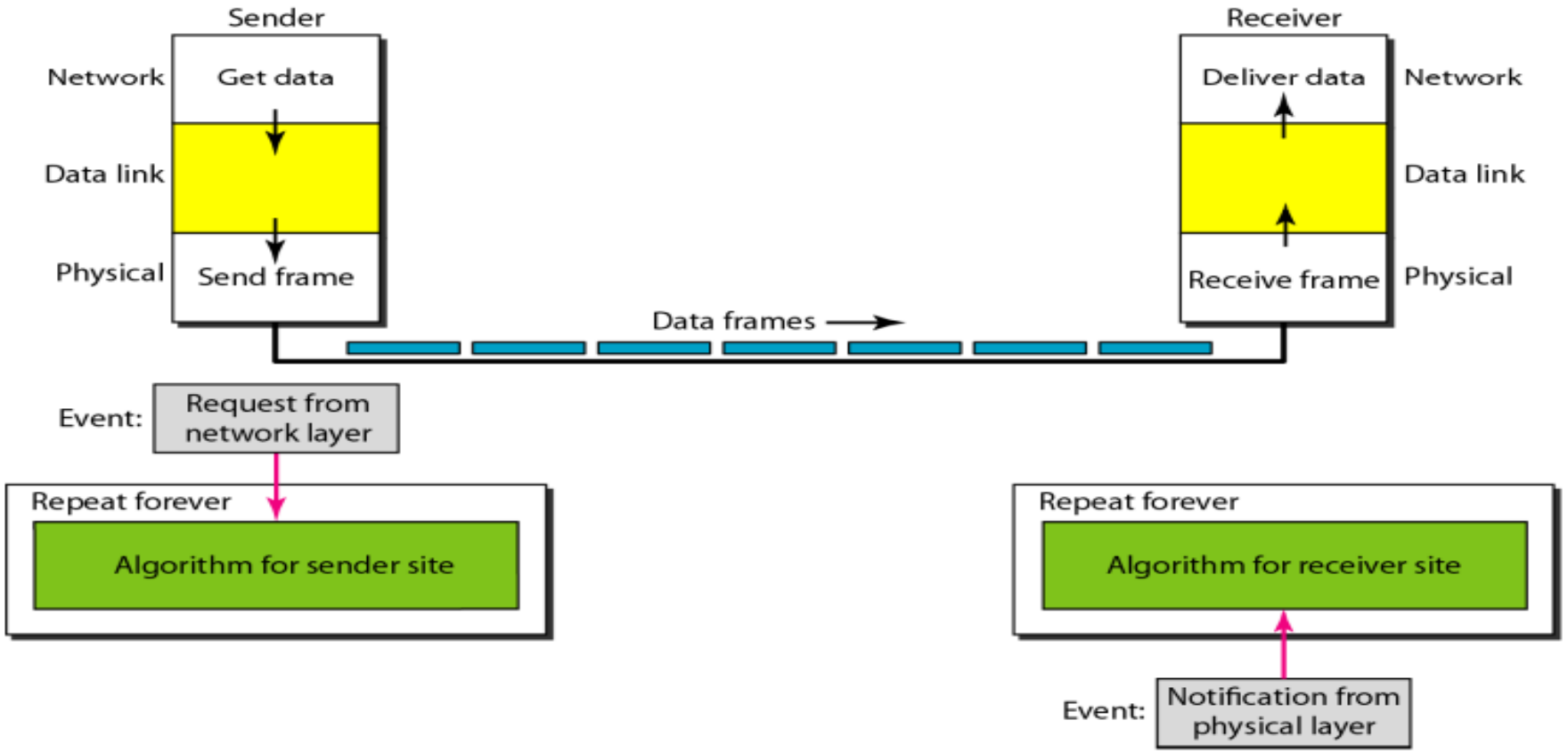
- Data Link Layer = Framing + Flow Control + Error Control
- It delivers data from one node to another node


Protocols
For noiseless channel
Simplest
Stop-and-Wait
For noisy channel
Stop-and-Wait ARQ
Go-Back-N ARQ
Selective Repeat ARQ

NOISELESS CHANNELS

SIMPLEST PROTOCOL

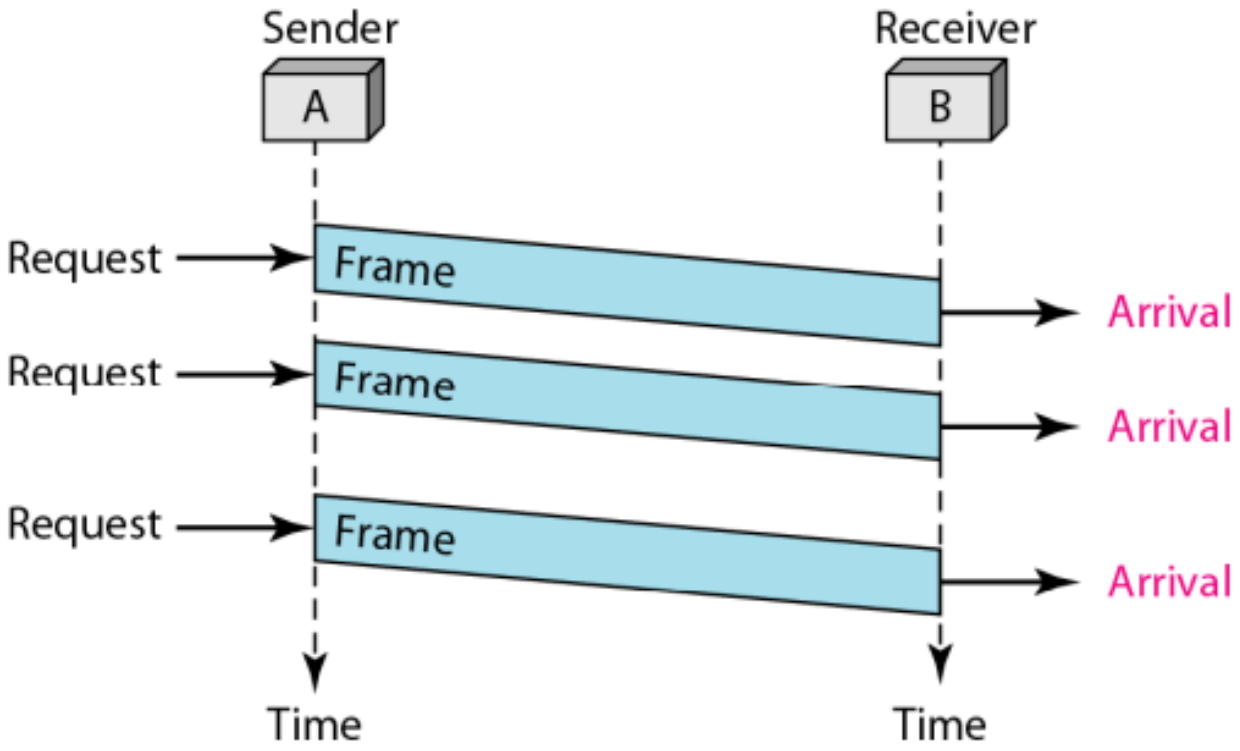
Figure 11.6 *The design of the simplest protocol with no flow or error control*



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*



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*

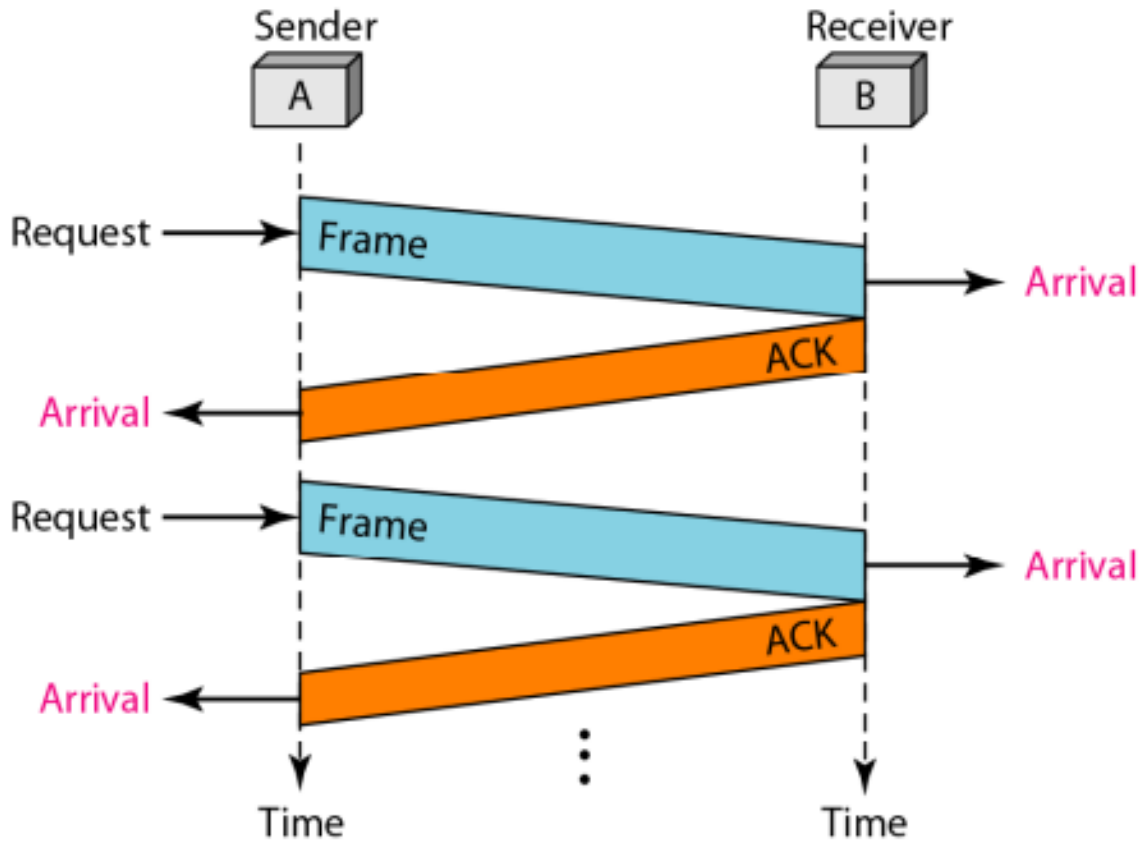
```
1  while(true)                                  // Repeat forever
2  {
3    WaitForEvent();                            // Sleep until an event occurs
4    if(Event(ArrivalNotification))             //Data frame arrived
5    {
6       ReceiveFrame();
7       ExtractData();
8       DeliverData(); 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

STOP AND WAIT PROTOCOL

- If data frames arrive at a rate faster than they can be processed
- Storage
- Feedback

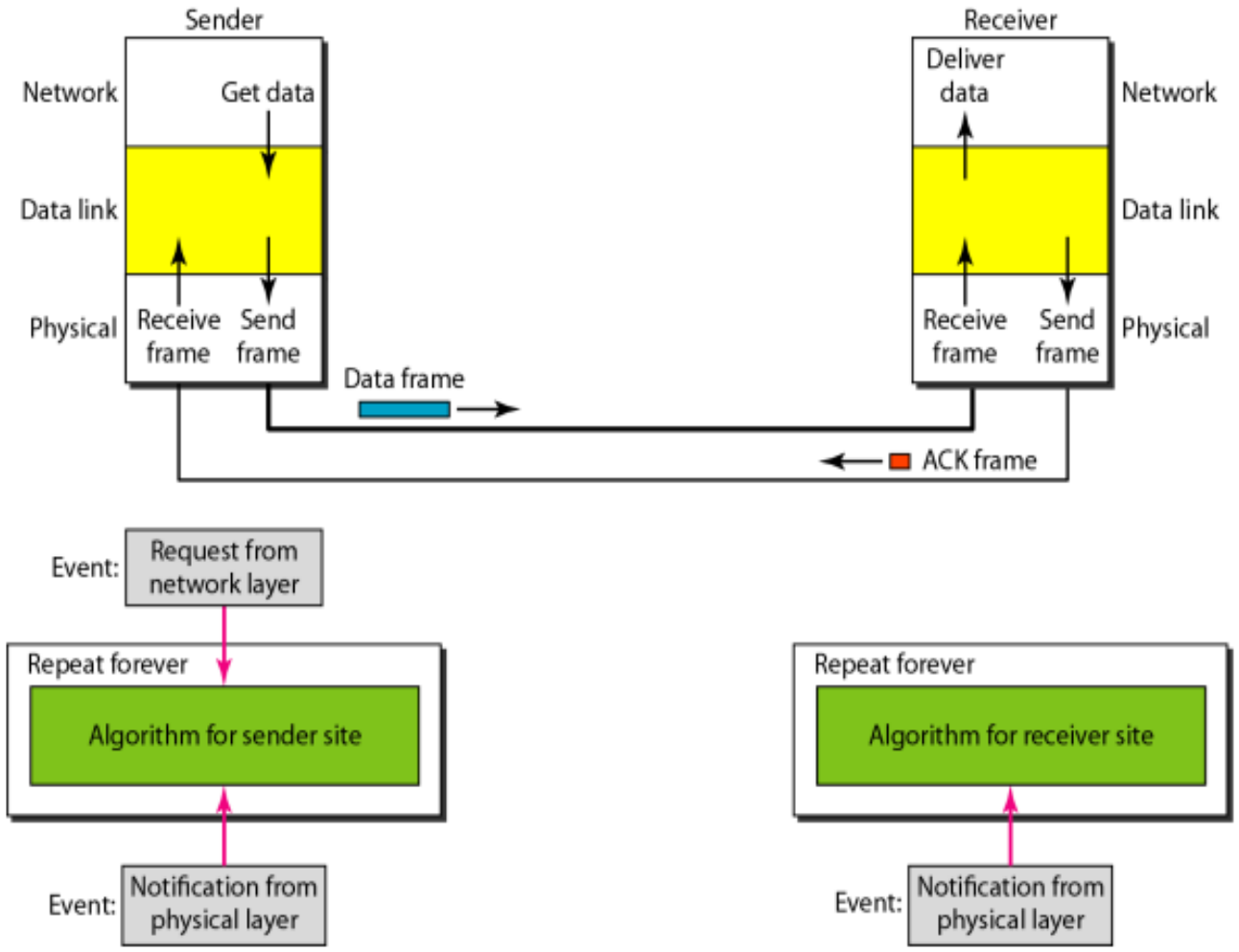
Stop and Wait Protocol : The sender sends one frame , stops until it receives confirmation from the receiver and then sends the next frame.

Figure 11.9 *Flow diagram for Example 11.2*



DESIGN

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 }
```

ANALYSIS

- ✓2 events can occur: request from the network layer or an arrival of notification from the physical layer
- ✓The response to these events must alternate.

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 }
```

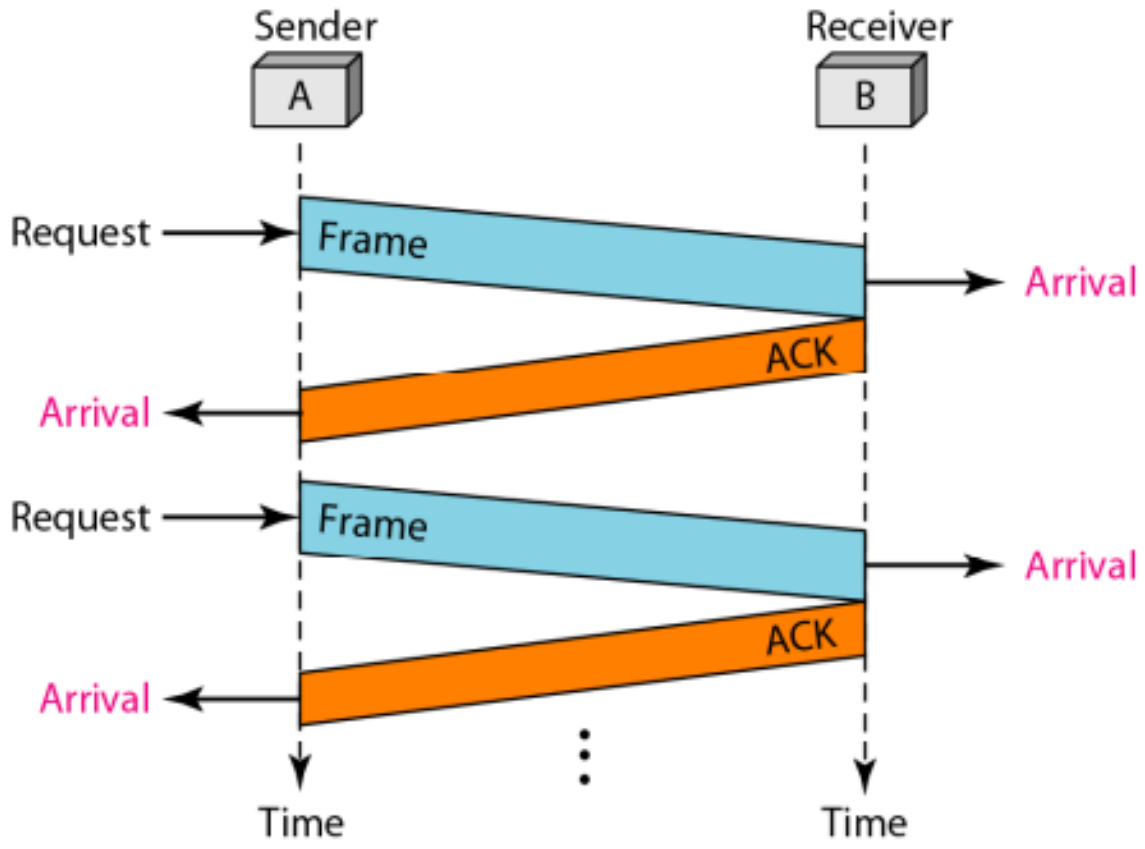
ANALYSIS

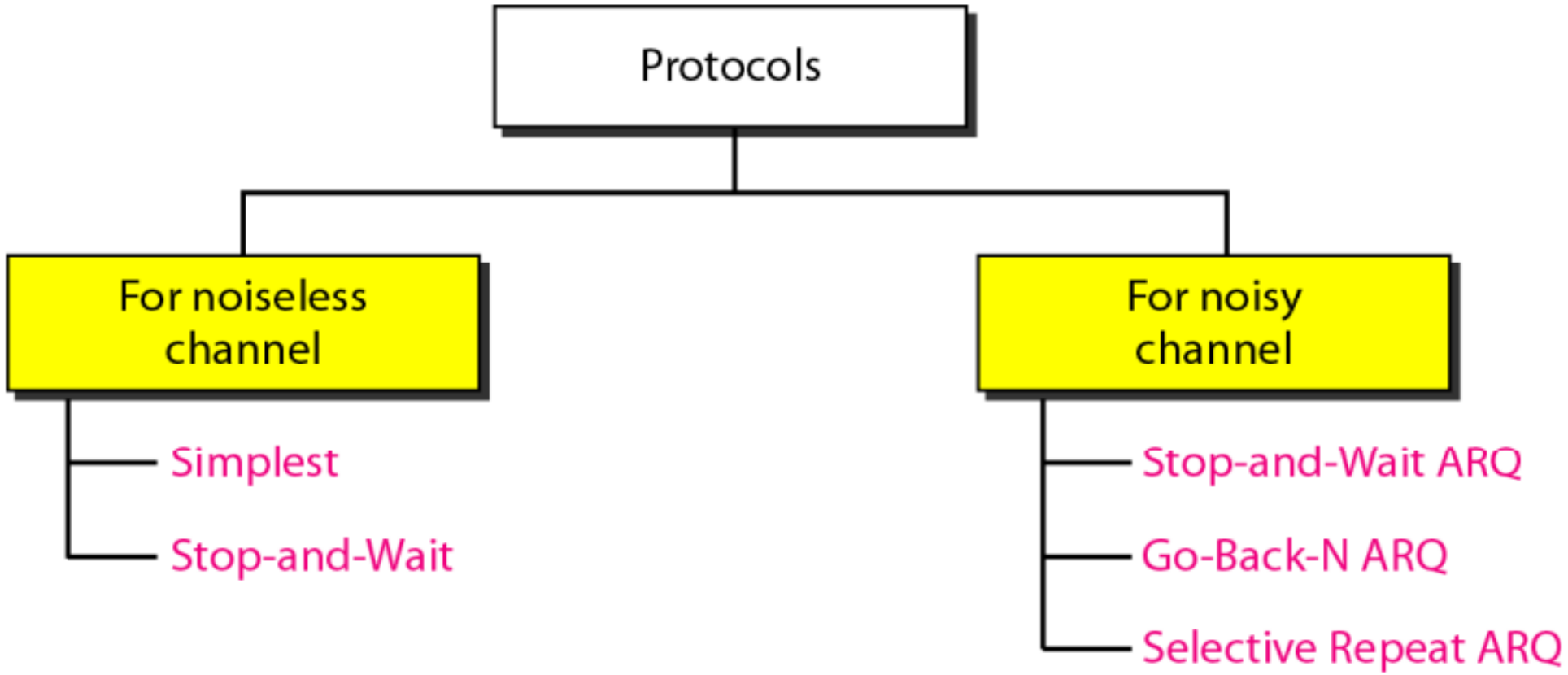
- After the data frame arrives, the receiver sends an ACK frame to acknowledge the receipt and allow the sender to send the next frame.

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*





For noiseless: is error control and flow control carried out ??

NOISY CHANNELS : STOP AND WAIT ARQ

- How is error correction done in Stop and wait ARQ ?
- Received frame can be correct one, duplicate or a out of order frame
- Solution : ??
- The corrupted or lost frames need to be resent
- But when to resend ?

- Two things to be done here
- 1. Sequencing the fames with numbers
- 2. Make a copy and start the timer, send the copy if timer expires

STOP AND WAIT ARQ

- The sender keeps a copy of the sent frame
- At the same time, it starts the timer.
- If the timer expires and there is no ACK for the sent frame, the frame is resent
- The copy of same is held again and timer is restarted.

- Several copies of the same frame will be in network, but ACK is required for only one specific frame

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.

WHAT TO DO IF ACK FRAME IS CORRUPTED OR LOST ??

- It adds some redundant bits to ACK frame too
- ACK frame has a sequence number field.
- However, in this protocol the sender simply discards a corrupted ACK frame or ignores an out of order frame.

SEQUENCE NUMBERS

- Frames are numbered.
- A field is added to the data frame to hold the sequence number of that frame
- If field is m bits : sequence nos. start from 0 to 2^(m-1) and then repeat again
- Range of sequence numbers

RANGE OF SEQUENCE NUMBERS

- Assume we have used x as a sequence no.
- We only need to use x+1 after that.
- There is no need of x+2!!!
- Let's reason it out –

3 things can happen:

1. The frame arrives safe and Rx sends ACK. So, the sender will send next frame ie. x+1.
2. The frame arrives safe but the ACK from receiver is corrupted or lost. Now, the sender resends the copy which the receiver recognizes as it was expecting x+1 frame.
3. The frame is corrupted or lost. The sender resends it

Say x = 0, x+1 = 1 …. We can repeat the sequence 0 1 0 1 0……is this pattern familiar to you ?

Note

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

ACKNOWLEDGEMENT NUMBERS

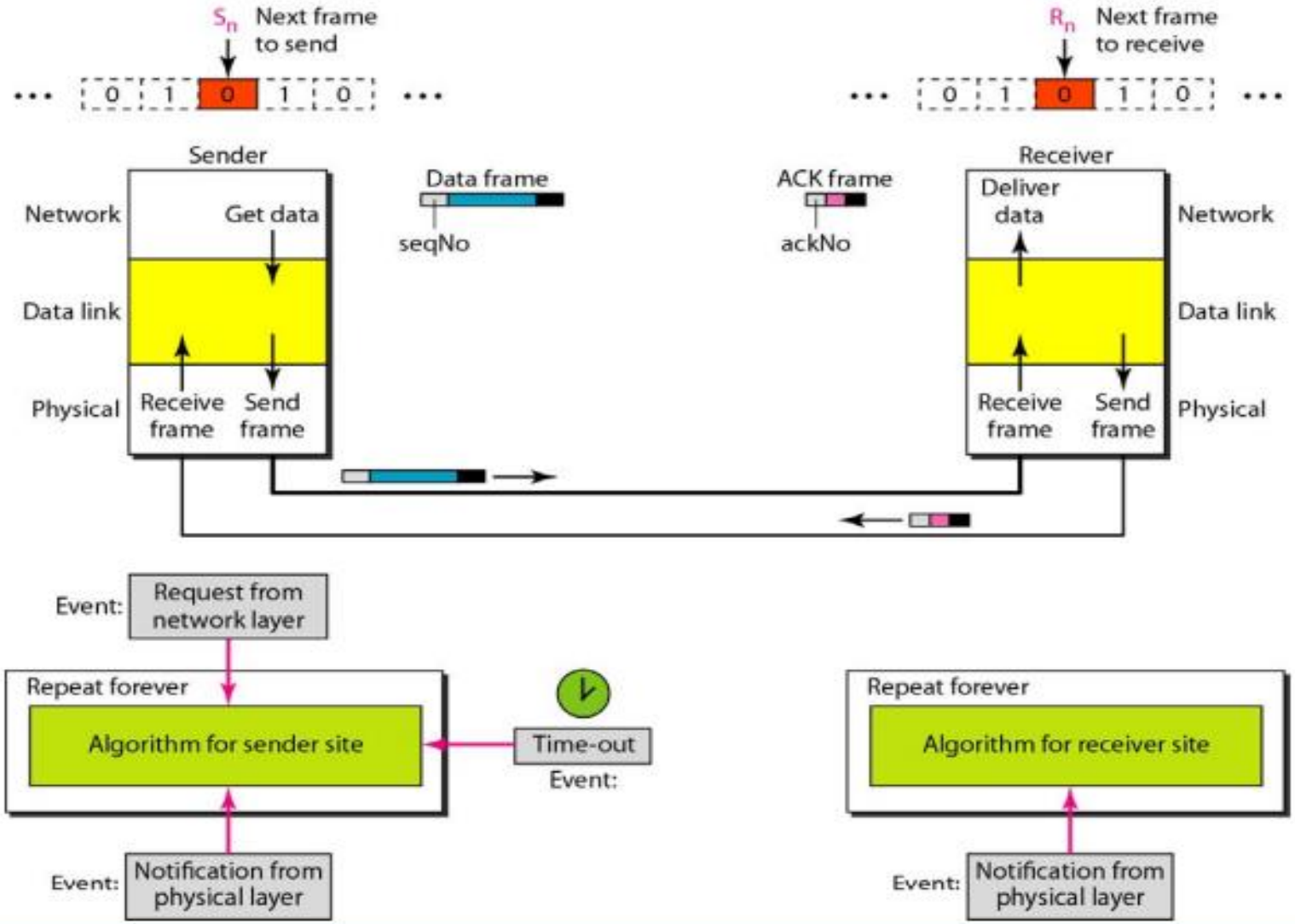
- The ACK no. always announces the sequence number of the next frame expected by the receiver.

Ex: frame 0 arrived safe and sound, the receiver sends ACK frame with ACK 1

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  }
```

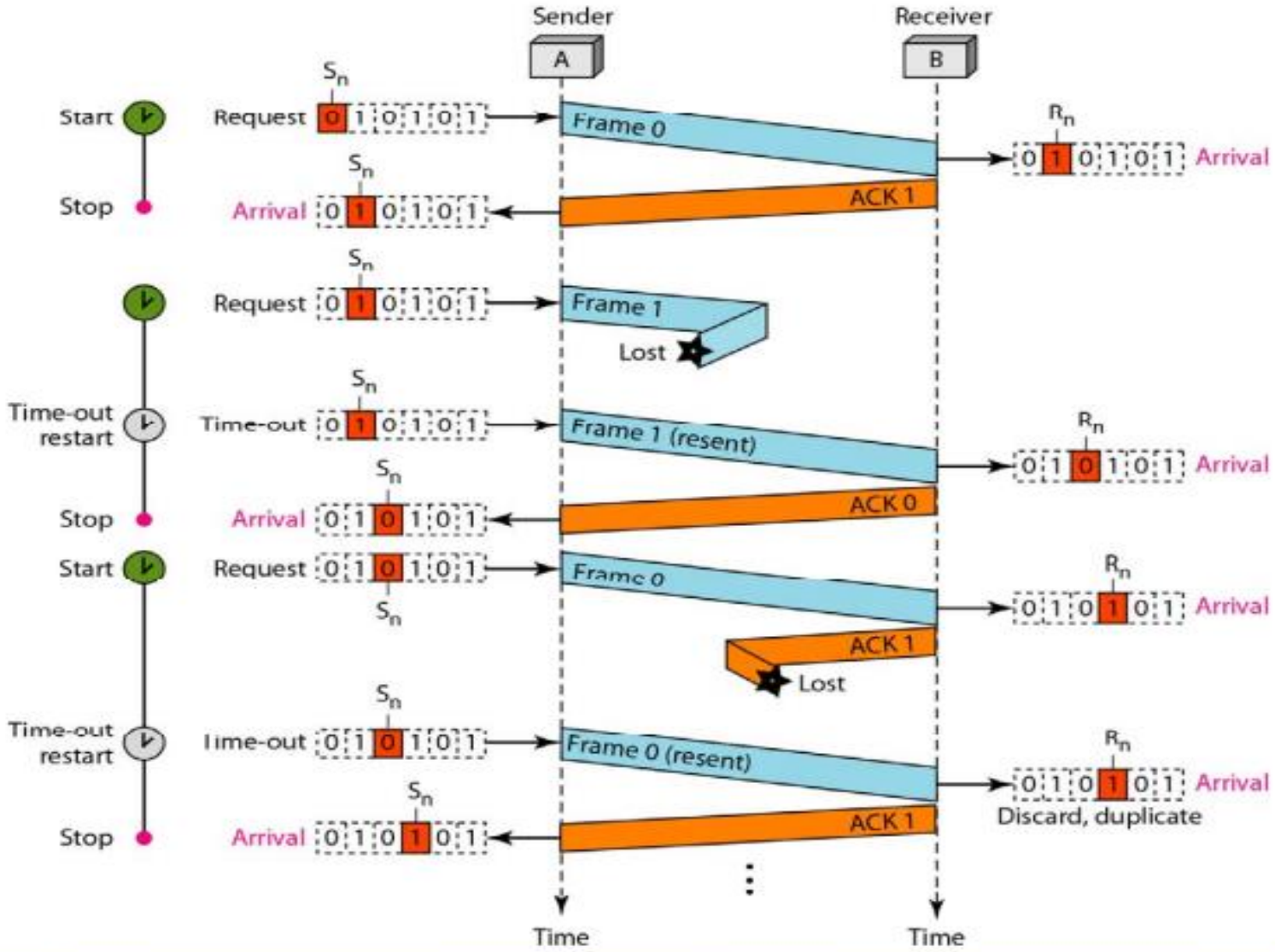
Algorithm 11.6 *Receiver-site algorithm for Stop-and-Wait ARQ Protocol*

```
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*



PIPELINING

- What is pipelining ?
- What are its benefits ?

PIPELINING

- In networking and in other areas, a task is often begun before the previous task has ended.

- Benefits:
- Improves efficiency of the transmission if number of bits in transition is large w.r.t BW-delay product

GO BACK N AUTOMATIC REPEAT REQUEST

- It keeps the channel busy with more than one outstanding frame while the sender is waiting for ACK.
- Procedure
- Send multiple frames before receiving acknowledgement and keep a copy of the frames till the ACK arrives.

SEQUENCE NUMBERS

- Frames from sending station are numbered sequentially
- In the header we include the sequence number
- If there are m bits allowed by the header, then sequence numbers will range from 0 to 2^m -1.
- If m =4 , then ? 0 through 15 inclusive.
- We can repeat sequences:
- 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,0,1,2,3,4,5,6,7

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.

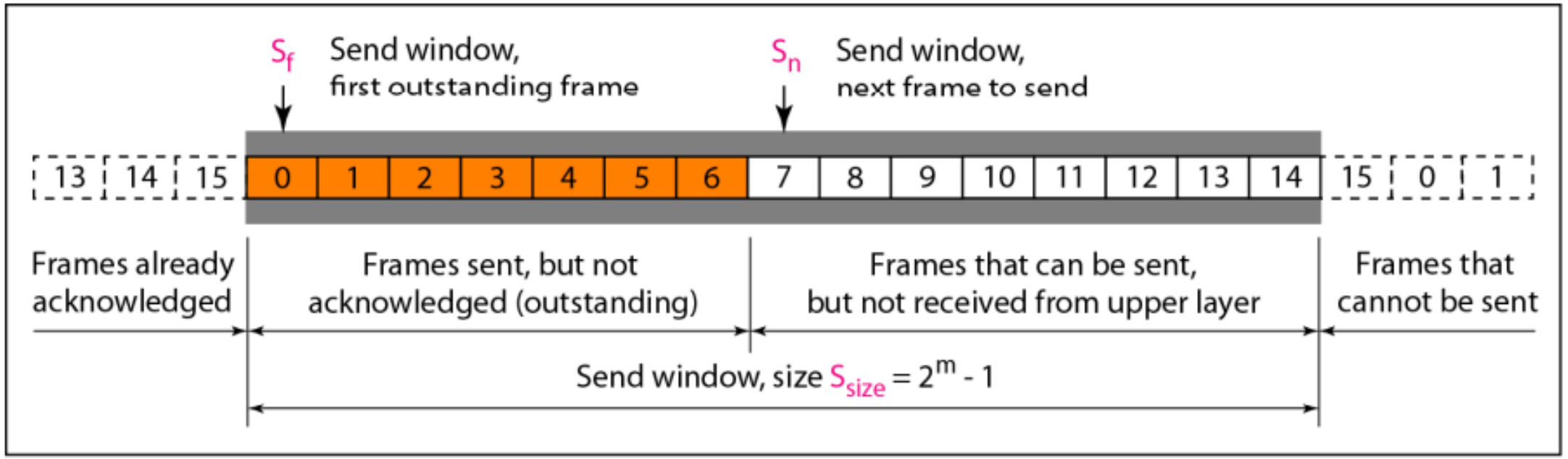
SLIDING WINDOW

- It is an abstract concept that defines the range of sequence numbers that is the concern of the sender and receiver.

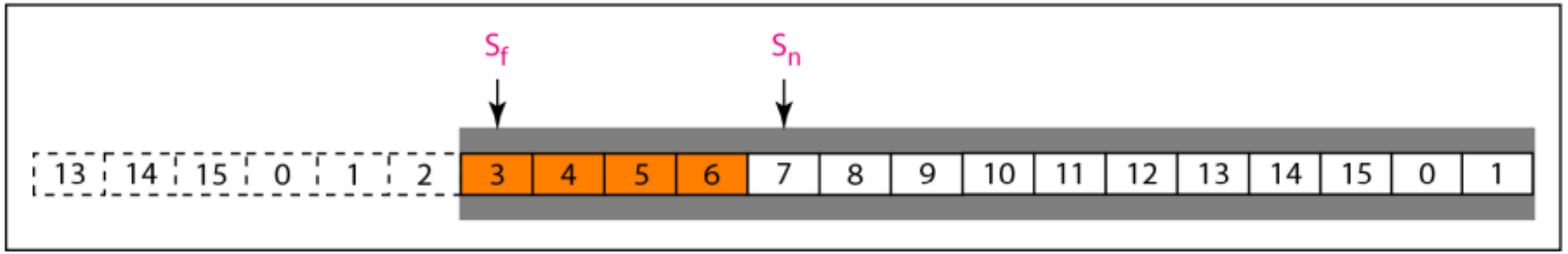
- The range which is in concern with the sender is called SEND SLIDING WINDOW.
- The range which is in concern of the receiver is called RECEIVE SLIDING WINDOW.

- The max. size of the window is 2^m - 1

Figure 11.12 *Send window for Go-Back-N ARQ*



a. Send window before sliding



b. Send window after sliding

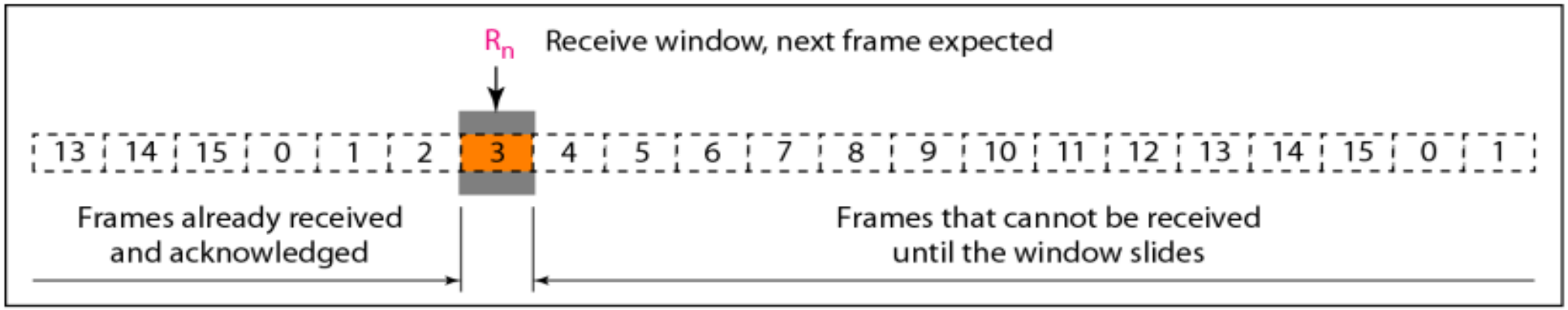
Note

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}.

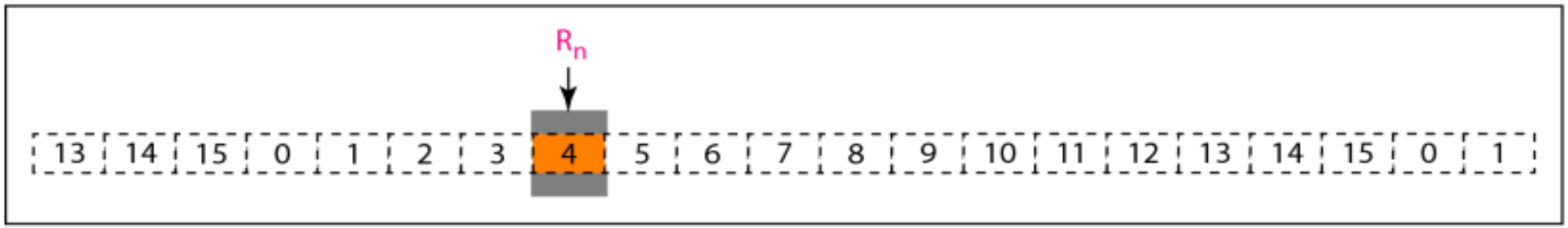
Note

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

Figure 11.13 *Receive window for Go-Back-N ARQ*



a. Receive window

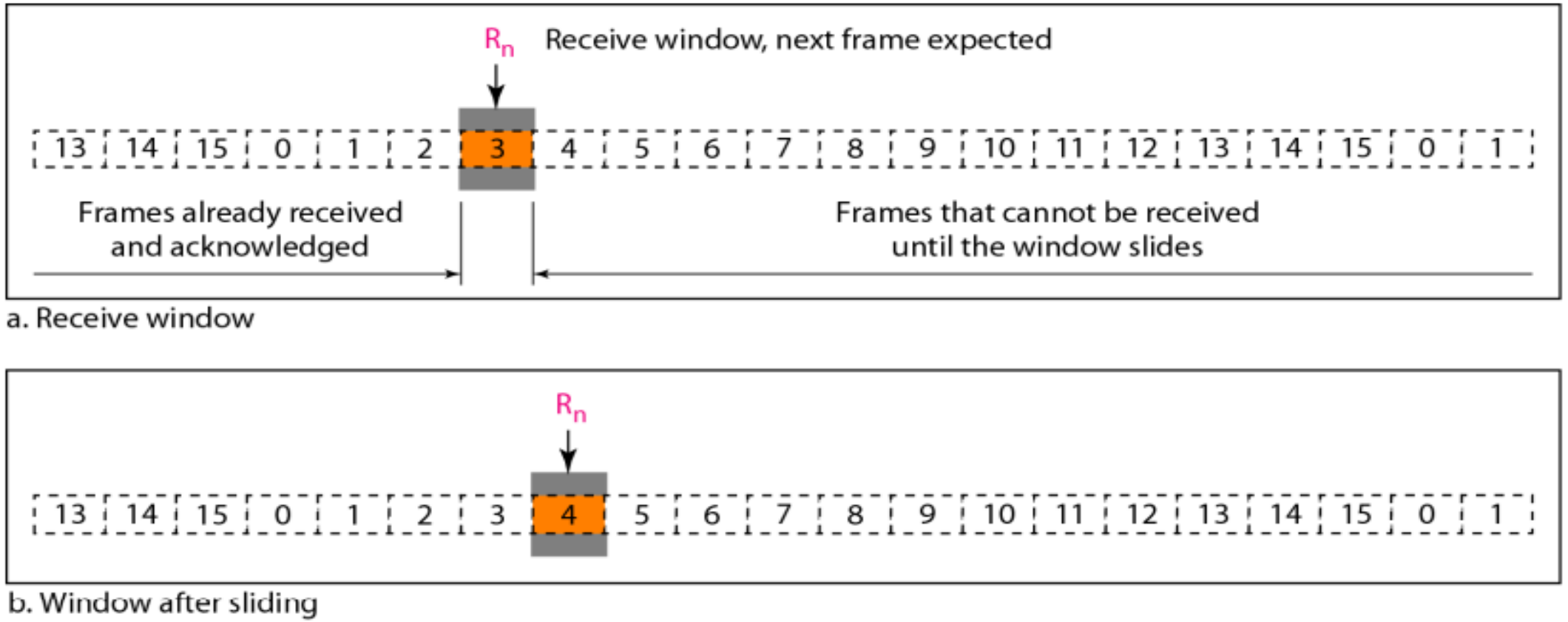


b. Window after sliding

RECEIVE WINDOW

- It makes sure that correct data frames are received and that correct acknowledgements are sent.
- The size of the receive window is always 1

Figure 11.13 *Receive window for Go-Back-N ARQ*



Note

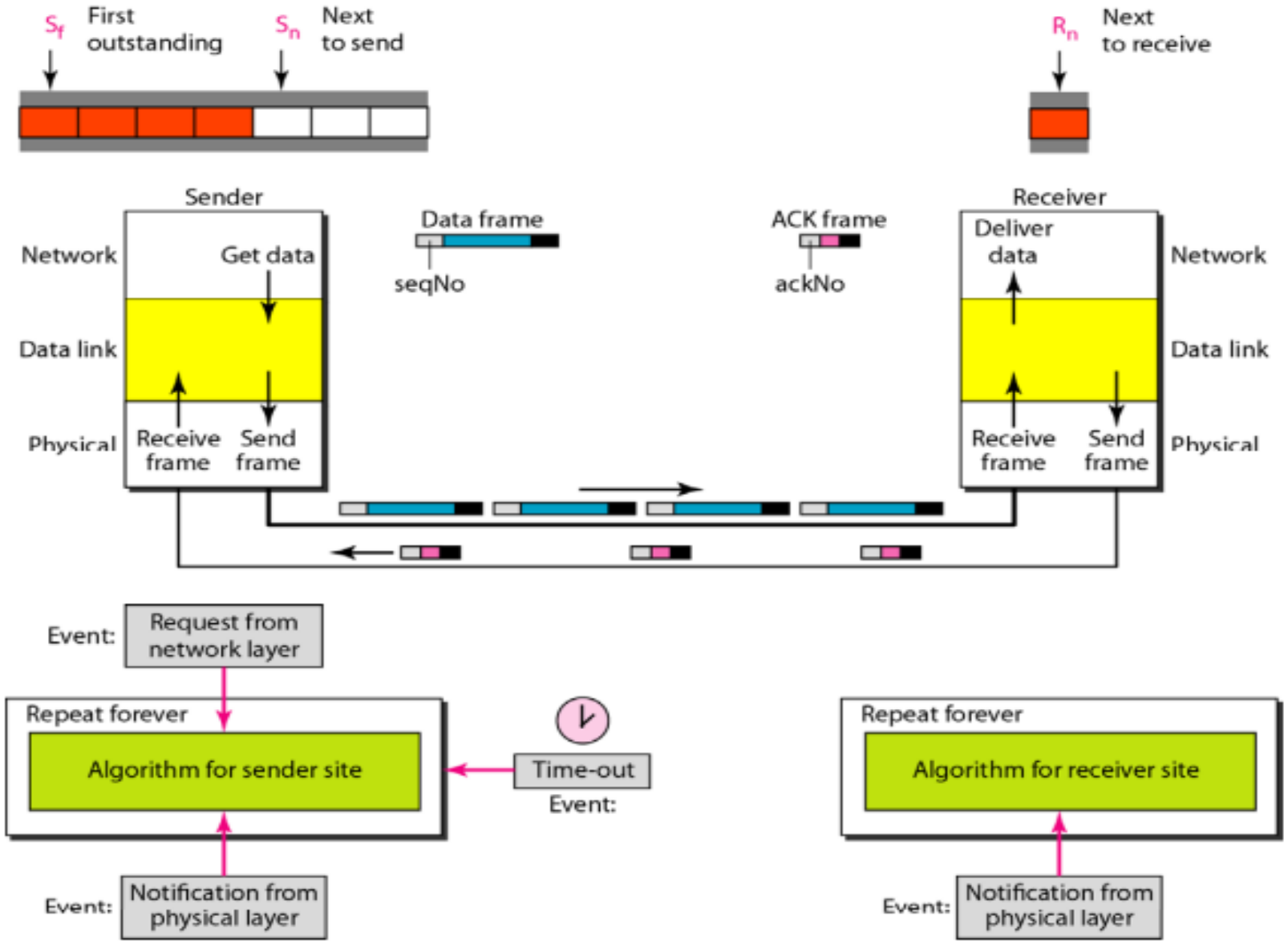
The receive window is an abstract concept defining an imaginary box of size 1 with one single variable R_n. The window slides when a correct frame has arrived; sliding occurs one slot at a time.

ACK

- Rx sends a ACK if frame has arrived safe and sound
- If it is damaged it remains silent
- The silence causes the timer to expire and the sender resends the frames for which it had not got ACKs

- Scenario: Sender has already sent frame 6, but timer for frame 3 expires !! Now which all frame/frames to resend ???

Figure 11.14 *Design of Go-Back-N ARQ*



COMPARE

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

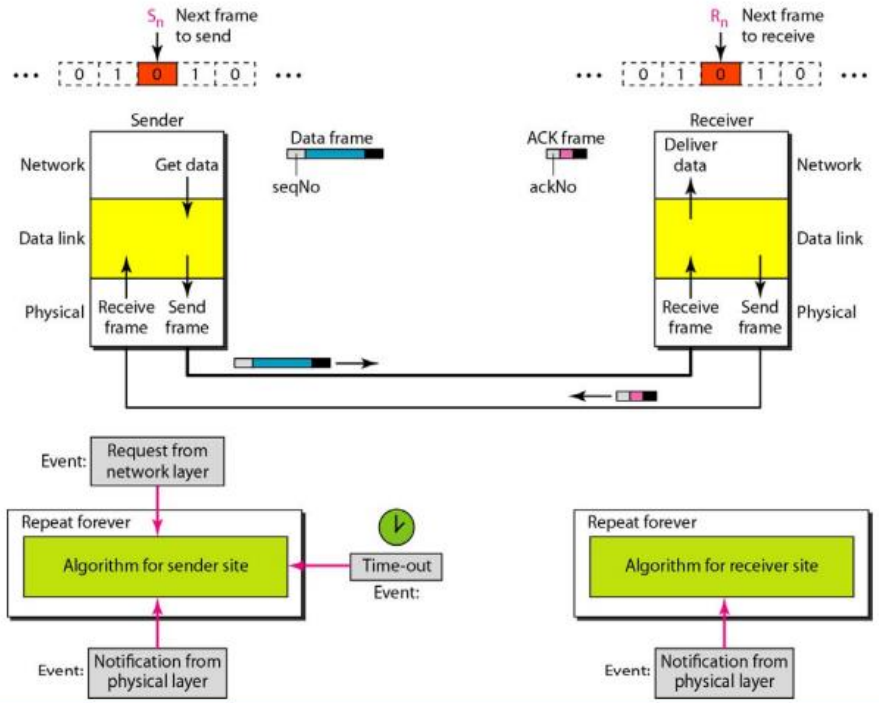
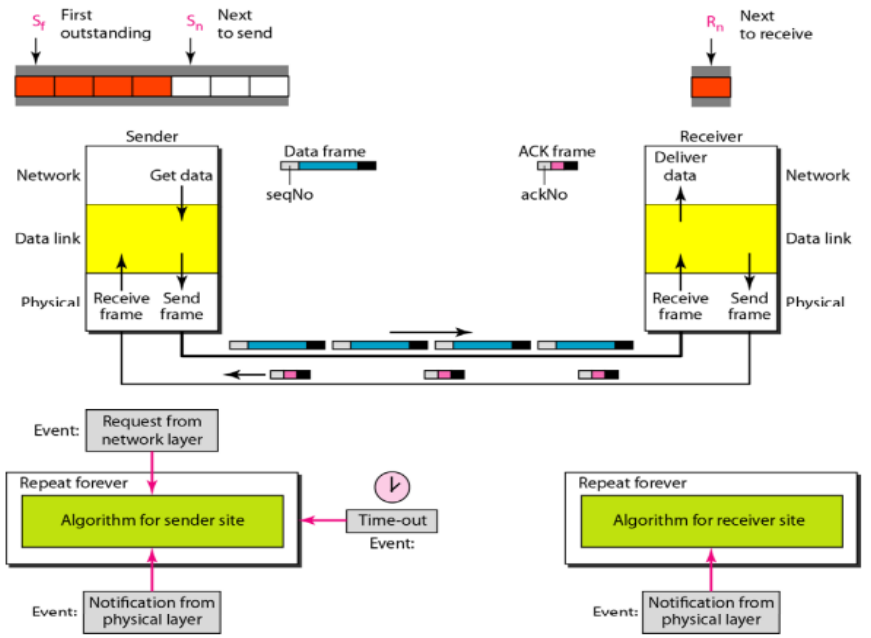
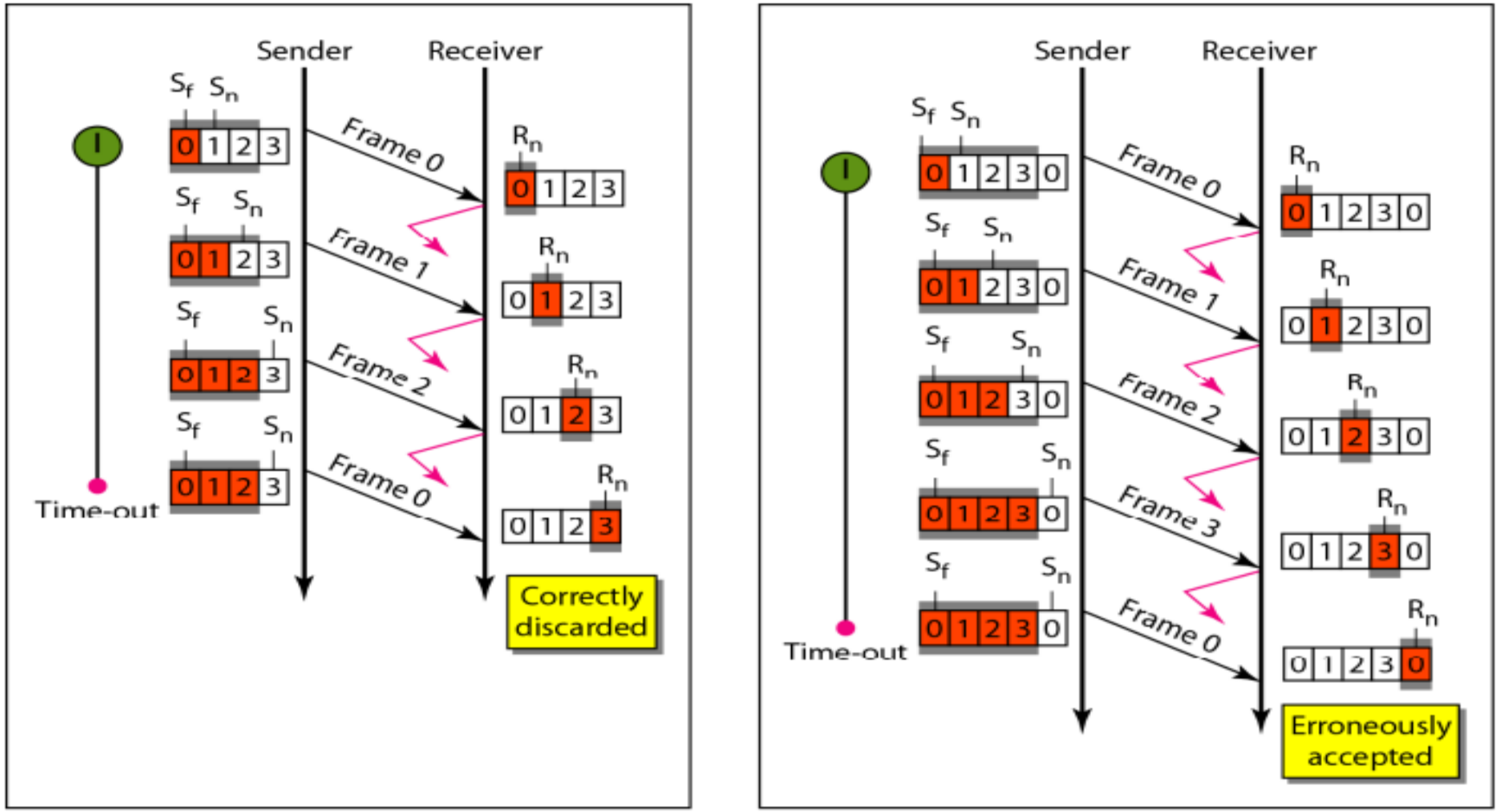


Figure 11.14 *Design of Go-Back-N ARQ*



SEND WINDOW SIZE

Figure 11.15 *Window size for Go-Back-N ARQ*



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 = 2^m - 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  }
```

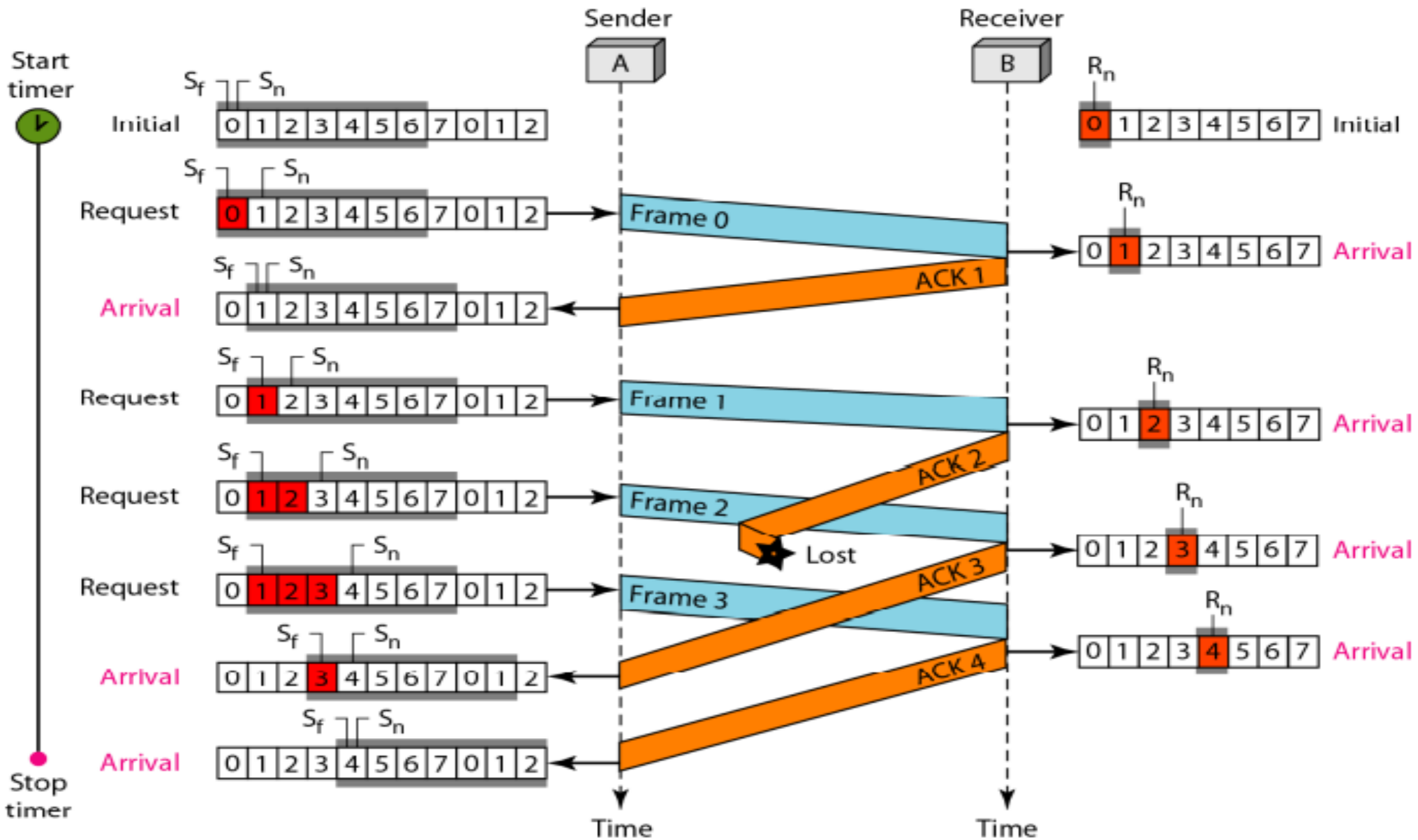
Algorithm 11.8 *Go-Back-N receiver algorithm*

```
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*



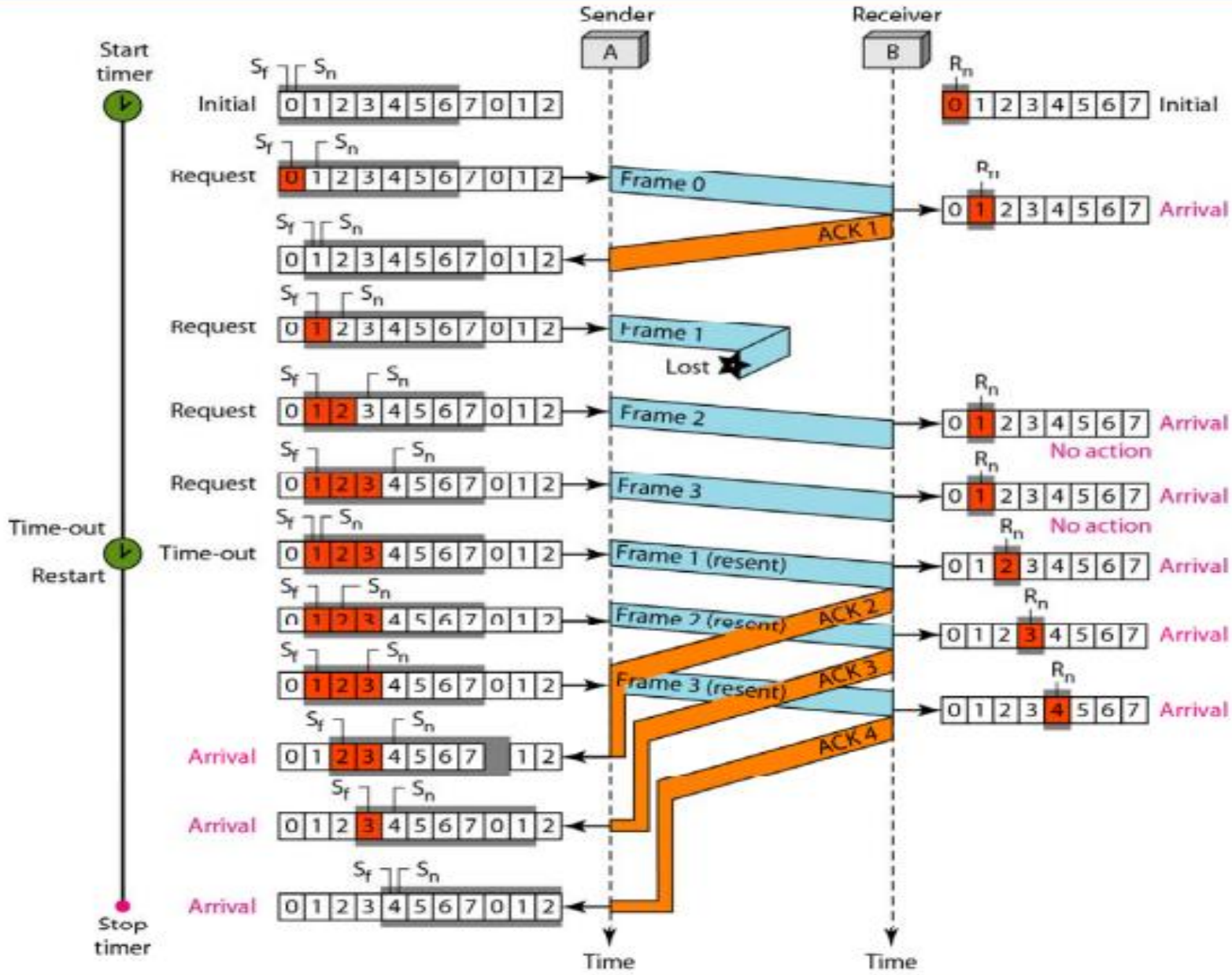
Example 11.7

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*



SELECTIVE REPEAT AUTOMATIC REPEAT REQUEST

- Go-Back N ARQ process is simple at the Receiver side - HOW ?
- It is INEFFICIENT – HOW ?

SELECTIVE REPEAT ARQ

- It is more efficient for noisy links.
- But the processing at the receiver is more complex

WINDOWS

Figure 11.18 *Send window for Selective Repeat ARQ*

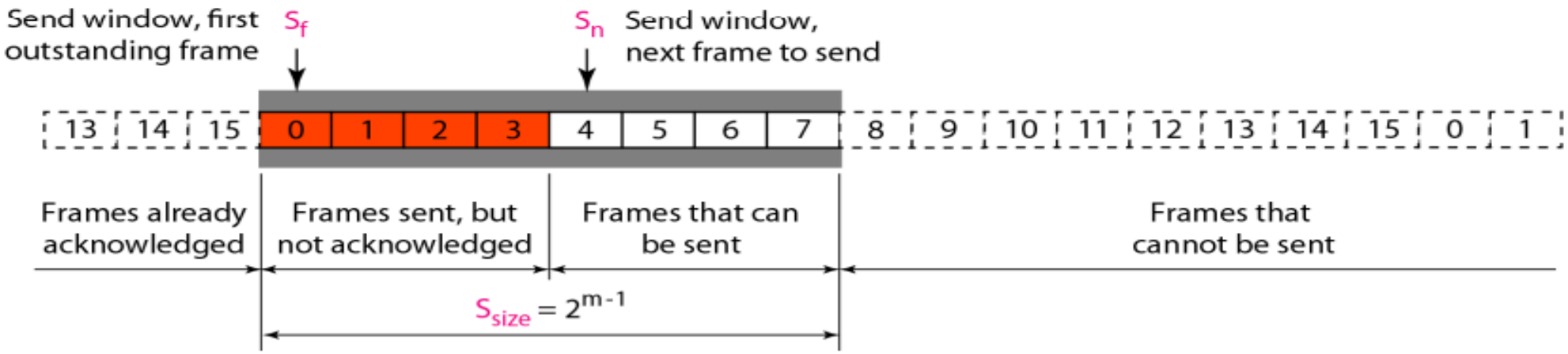
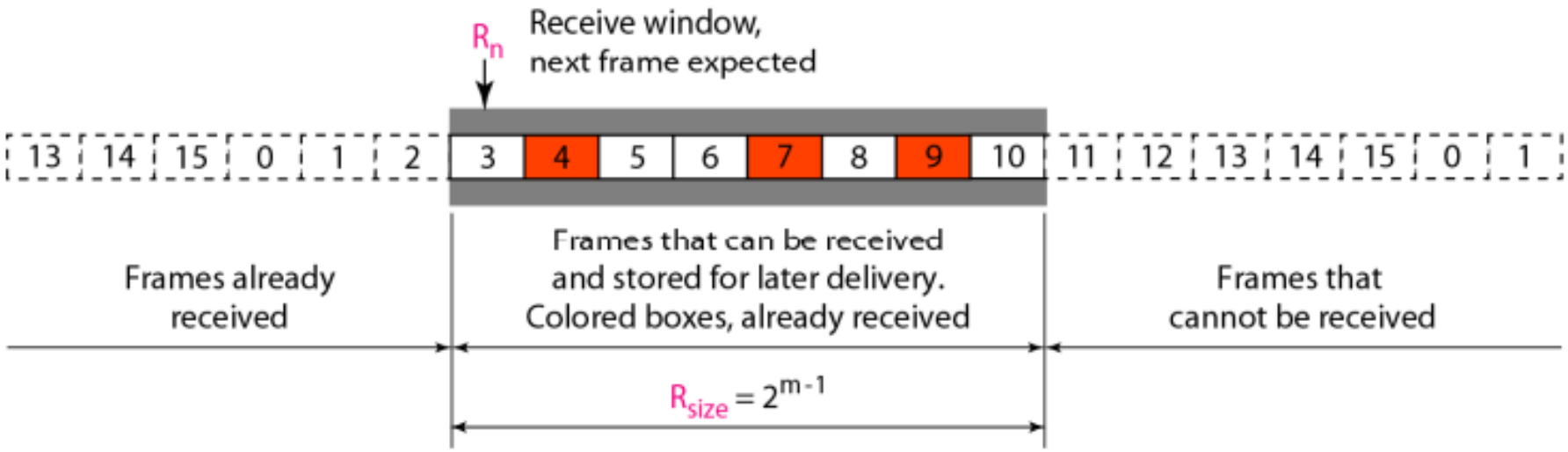


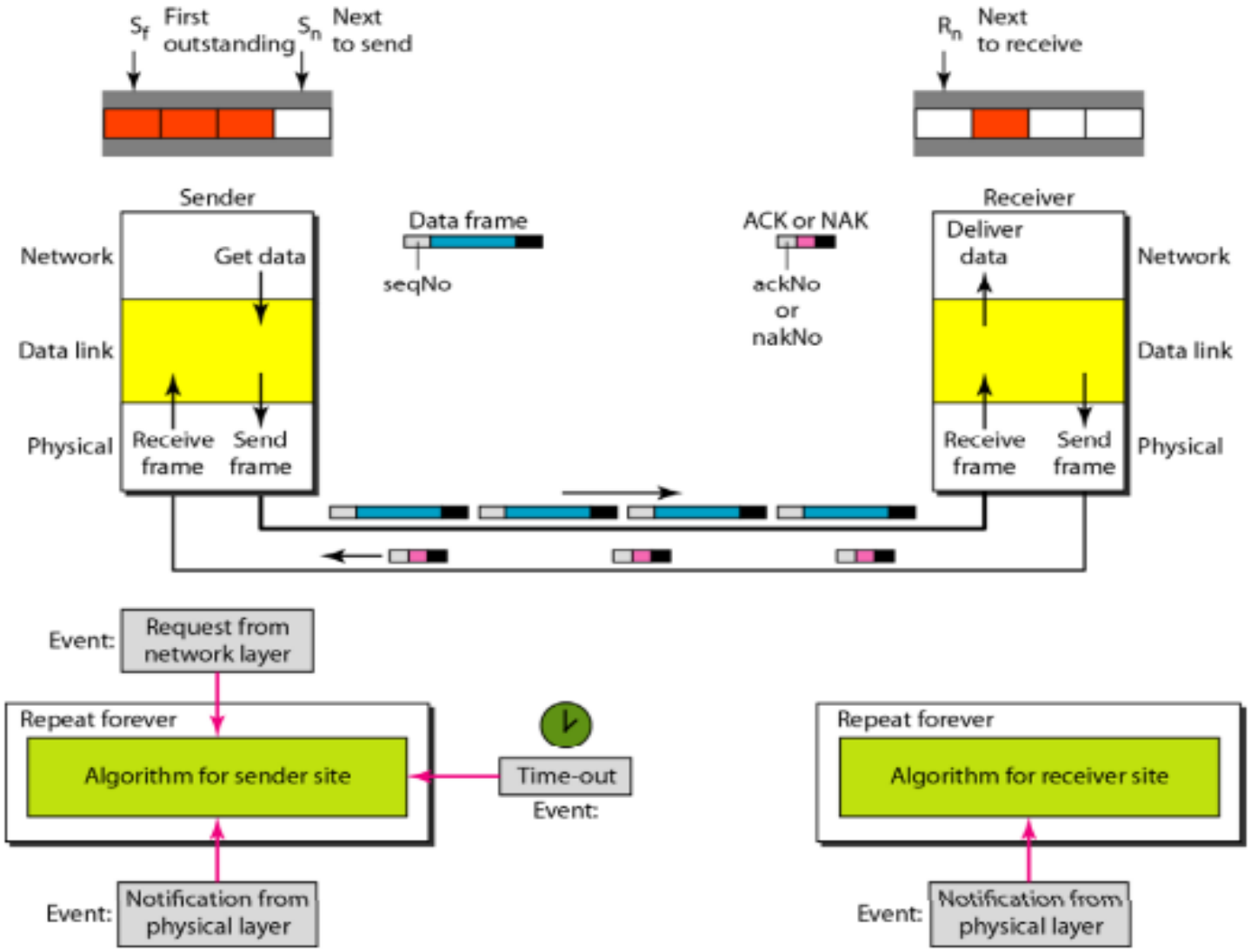
Figure 11.19 *Receive window for Selective Repeat ARQ*



DETAILS

- 2 windows: Send and Receive
- Size of Send Window = Size of Receive Window = 2^(m-1)
- If m=4, size will be 8 !!
- The smaller the window size means less efficiency in filling the pipe.
- The protocol allows as many frames as the size of the receive window to arrive out of order and be kept until there is a set of in-order frames to be delivered to the network layer.
- The frames in the send frame can arrive out of order and be stored until they can be delivered.
- The receiver never delivers packets out of order to the network layer.

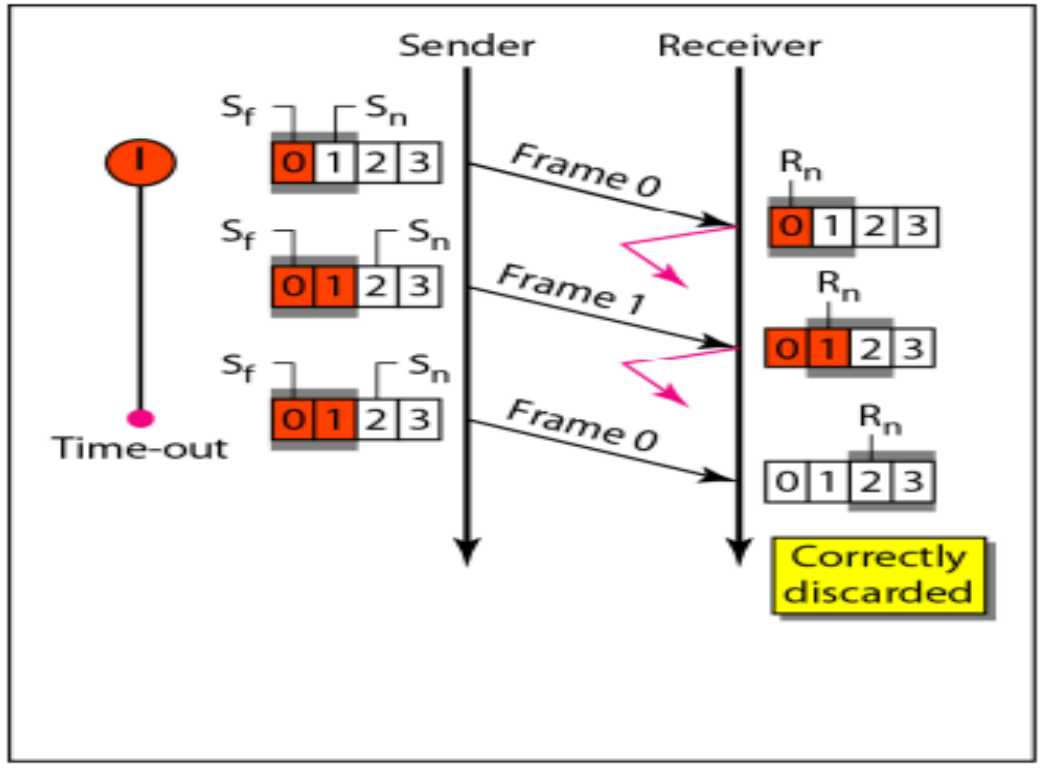
Figure 11.20 *Design of Selective Repeat ARQ*



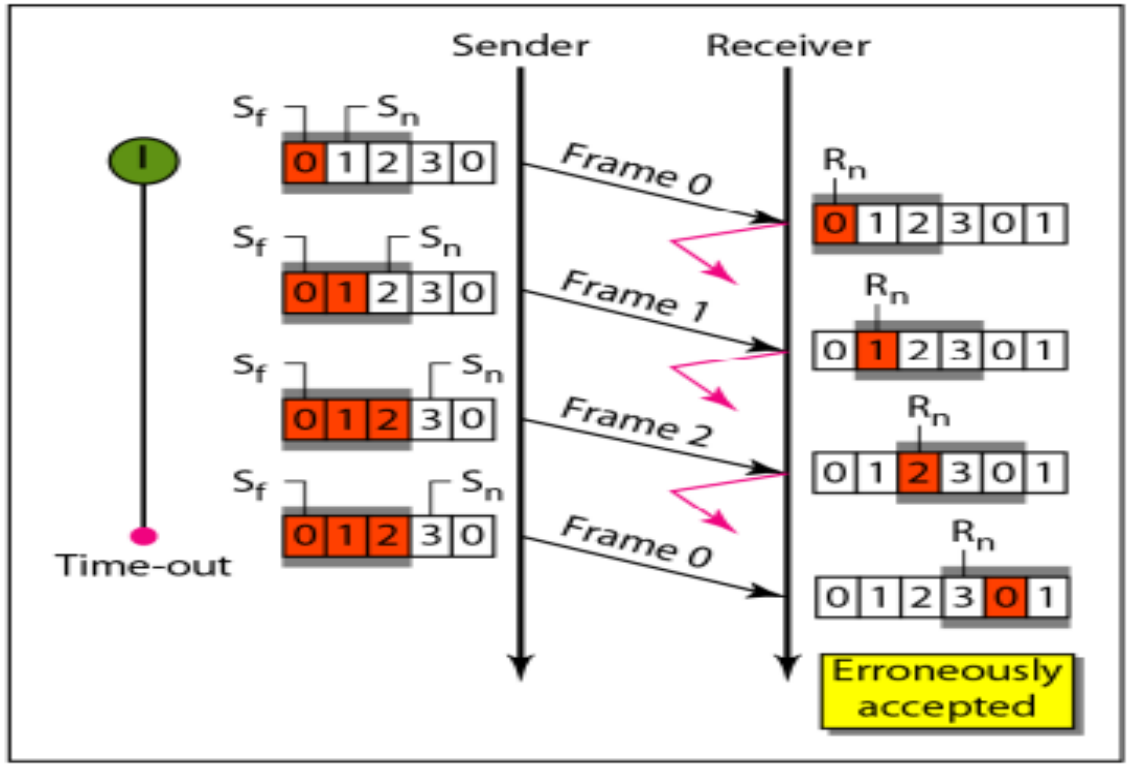
WINDOW SIZES

The size of the sender & RECEIVER WINDOWS must be at most one-half of 2^m

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 = 2^(m-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        ScndNAK(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  }
```

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.

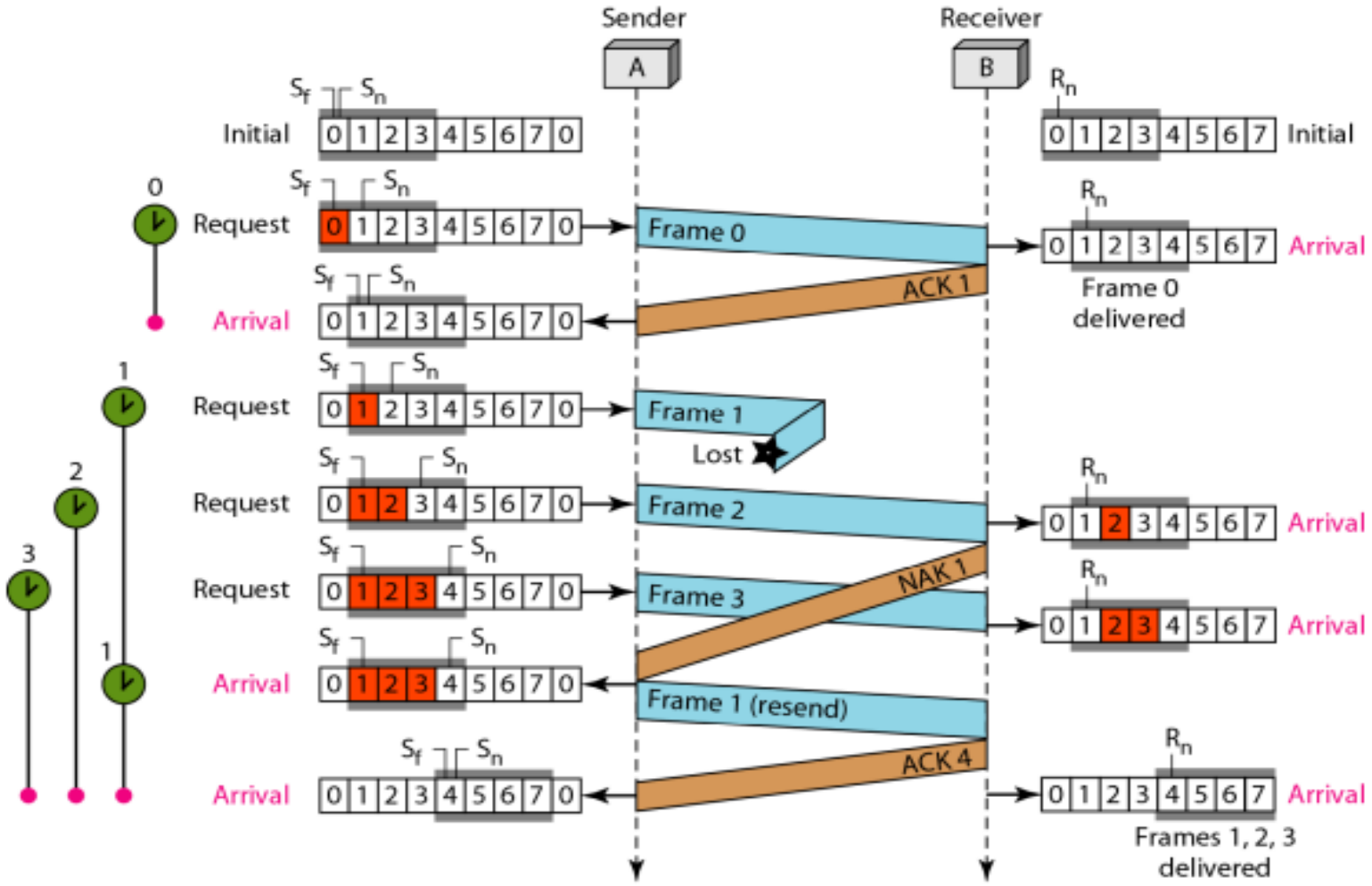
Example 11.8 (continued)

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.

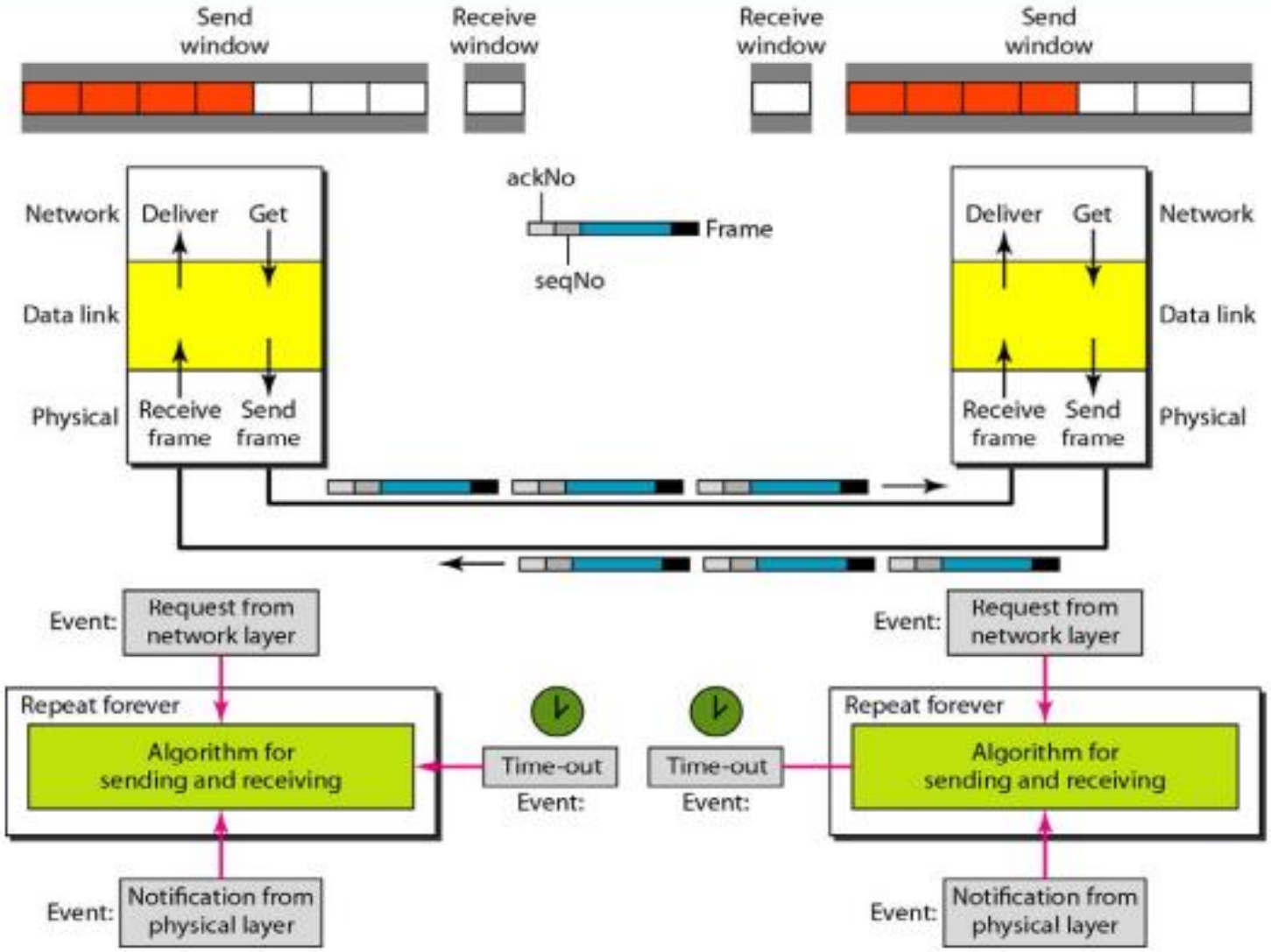
Figure 11.23 *Flow diagram for Example 11.8*



PIGGYBACKING

- As discussed in previous protocols, Data frames flow in only one direction although control information such as ACK & NAK frames can travel in the other direction.
- In real life, data frames are normally flowing in both the directions. This means even control information needs to flow in both the directions.
- A technique called Piggybacking is used to improve the efficiency of bidirectional protocols.
- When a frame is carrying control data from A to B , it can also carry the control info and vice versa.

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



DETAILS

- Each node has 2 windows (Send & Receive) with a timer for each.
- 3 types of events: request , arrival and time-out.
- Arrival event = handle control info + frame
- Request event uses only the send window at each site
- Arrival event needs to use both windows
- In Piggybacking, both sides must use same algorithm

HDLC- HIGH LEVEL DATA LINK

- It is a bit-oriented protocol for communication over point-to-point and multipoint links.

CONFIGURATIONS AND TRANSFER MODES

2 common transfer modes that can be used in different configurations are:

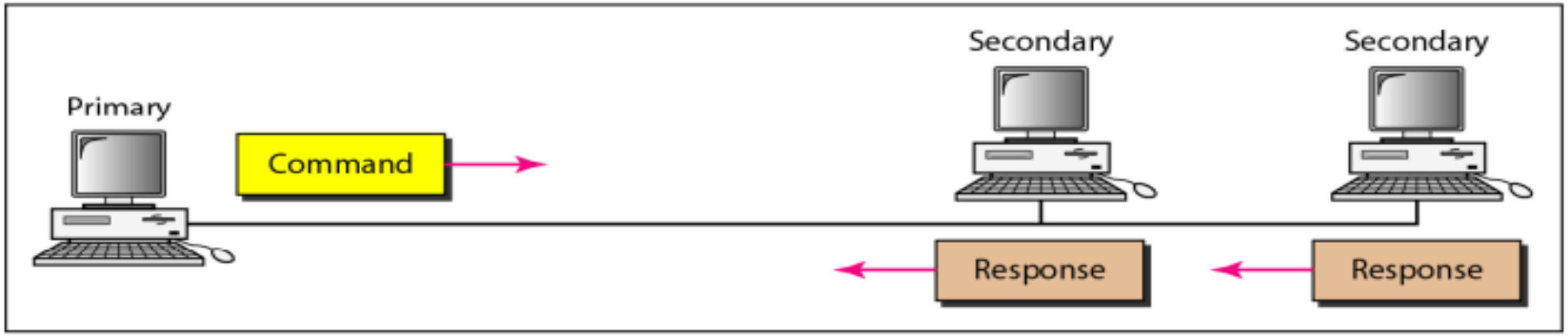
- Normal Response Mode (NRM)
- Asynchronous Balanced Mode (ABM)

NORMAL RESPONSE MODE

- Here the station configuration is unbalanced
- It has one primary station and multiple secondary stations
- A primary station can send commands and a secondary station can only respond
- NRM is used for both point-to-point and multi-point links

Figure 11.25 *Normal response mode*

a. Point-to-point



b. Multipoint

ASYNCHRONOUS BALANCED MODE

- The configuration is balanced
- The link is point-to-point and each station can function as primary and secondary.
- This is the common mode today

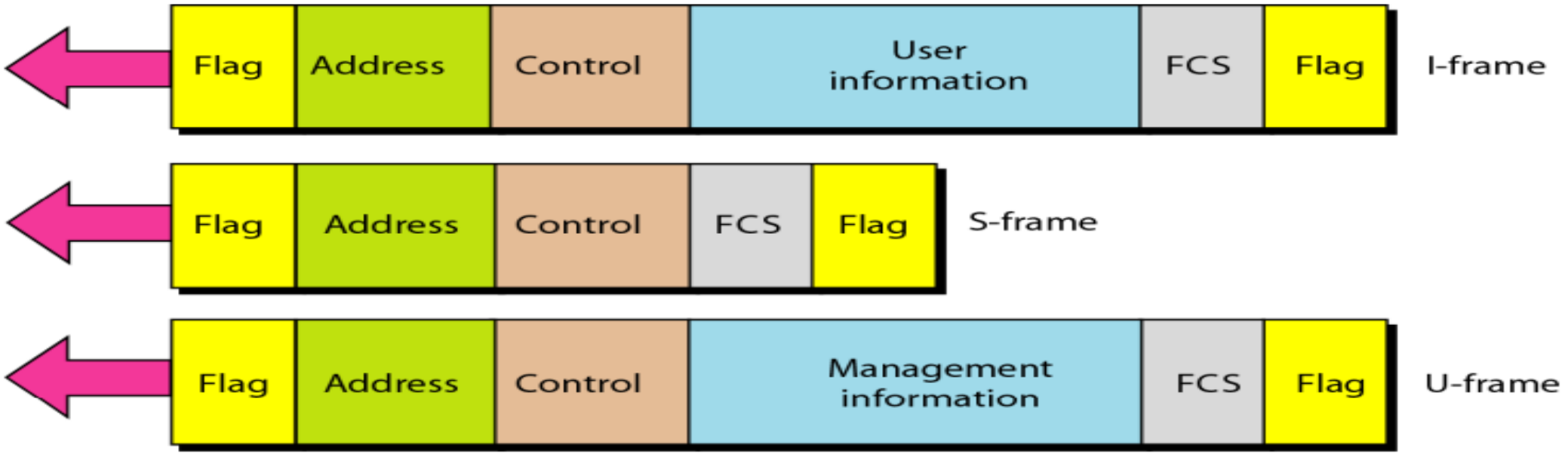
Figure 11.26 *Asynchronous balanced mode*

HDLC DEFINES 3 TYPES OF FRAMES

- Information Frames : I-Frames
  - ✓ Used to transport user data and control information relating to user data
- Supervisory Frames: S-Frames
  - ✓ Used only to transport control information
- Unnumbered Frames: U-Frames
  - ✓ They are reserved for system management.
  - ✓ They are used for management of the link itself.

FRAME FORMAT

Figure 11.27 *HDLC frames*



6 FIELDS

- A beginning flag field
- An address field
- A control field
- An information field
- A frame check sequence (FCS) field
- An ending flag field

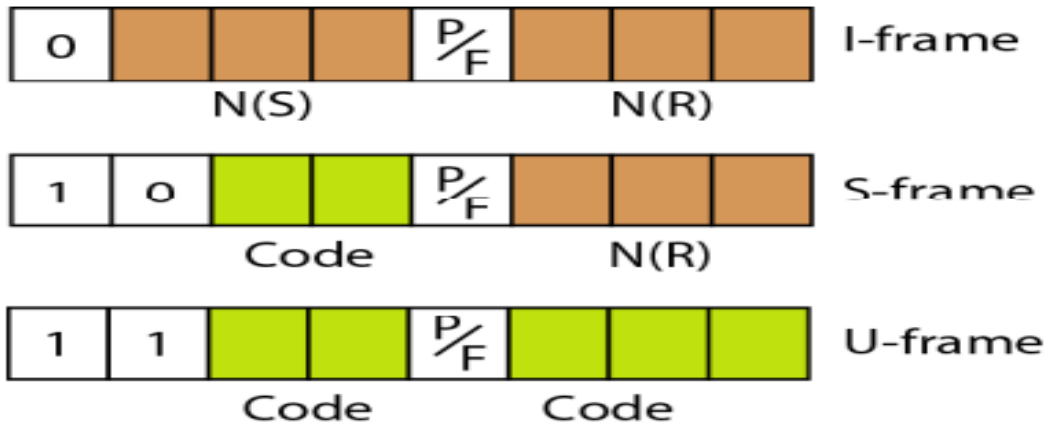
DETAILS

- **Flag field:** It is an 8-bit sequence with bit pattern 01111110 that identifies the beginning and end of frame and helps in synchronization.
- **Address Field:** Address of the secondary station.
  - ✓ If a primary station created a frame, it contains a ***to address***
  - ✓ *If a secondary creates the frame, it contains a* ***from address***
  - ✓ It can be 1 byte or several bytes long, depending on the needs of the network**.**
- **Control Field:** It can be 1- or 2- byte segment of the frame used for flow and error control.
- **Information Field:** It contains the user's data from the network layer or management information.
- **FCS field:** It is error detection field. It can be either 2- or 4-byte ITU-T CRC.

CONTROL FIELD

- It determines the type of frame and defines its functionality.

Figure 11.28 *Control field format for the different frame types*



CONTROL FIELD FOR I-FRAMES

- I-Frames are designed to carry user data from the network layer.
- The first bit defines the type.
- If first bit of the control field is 0, this means it is I-Frame
- Next three bits, called N(S) defines the sequence no. of the frame.
- Last three bits, called N(R) correspond to acknowledgement number when piggybacking is used.
- P/F field is a single bit used for dual purpose
- It has a meaning only when it is set.
- P- Poll / F-Final
- It means Poll when the frame is sent by a primary station to a secondary.
- It means Final when the frame is sent by a secondary to a primary.

CONTROL FIELD FOR S-FRAME

- Supervisory frames are used for flow and error control.
- S-frames do not have information fields.
- If the first 2 bits of the control field is 10, then it is a S-frame
- The last three bits N(R) corresponds to acknowledgement number (ACK or NAK)
- The two bits called Code is used to define the type of S-frame itself.

- Types of code
  - Receive ready (RR): Code 00. It acknowledges the receipt of a safe and sound frame/s.
  - Receive not ready (RNR): Code 10. It acknowledges the receipt of frame/s and announces the receiver is busy and cannot receive more frames. It helps in congestion control.
  - Reject (REJ): Code 01.This is a NAK frame. It can be used in Go-back N ARQ to improve the efficiency of the process by informing the sender, before the sender timer expires, that the last frame is lost or damaged.
  - Selective reject: Code 11. This is a NAK frame used in Selective Repeat ARQ. HDLC protocol uses the term selective reject instead of elective repeat.

CONTROL FIELD FOR U-FRAMES

- They are used to exchange session management and control information between connected devices.
- It contains an information field but one used for system management information and not user data.
- U-Frame codes are divided into two sections : a 2-bit prefix before the P/F bit and a 3-bit suffix after the P/F bit.
- Together these two segments (5 bits) can be used to create up to 32 different types of U-frames.

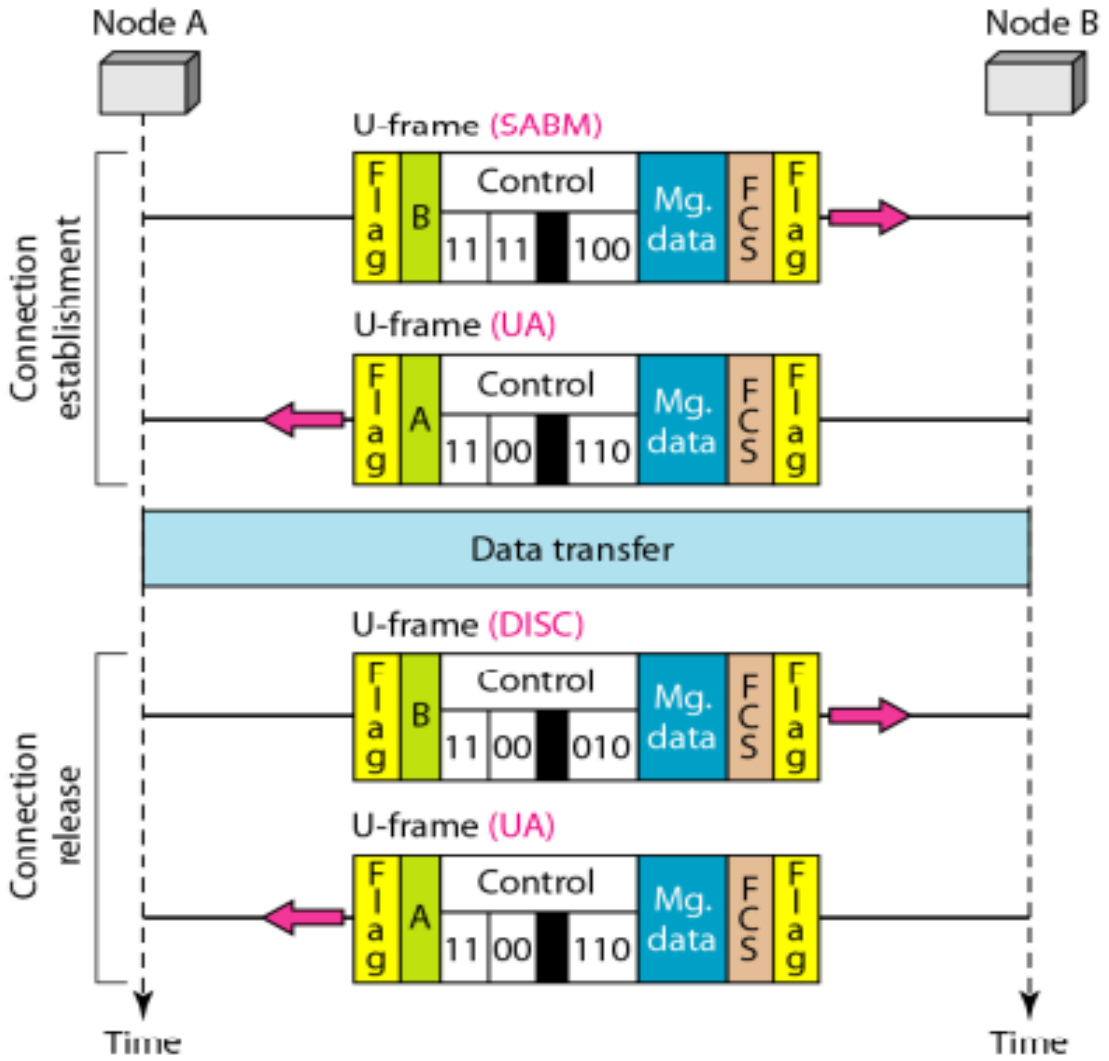
Table 11.1 *U-frame control command and response*

Code	*Command*	*Response*	*Meaning*
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*



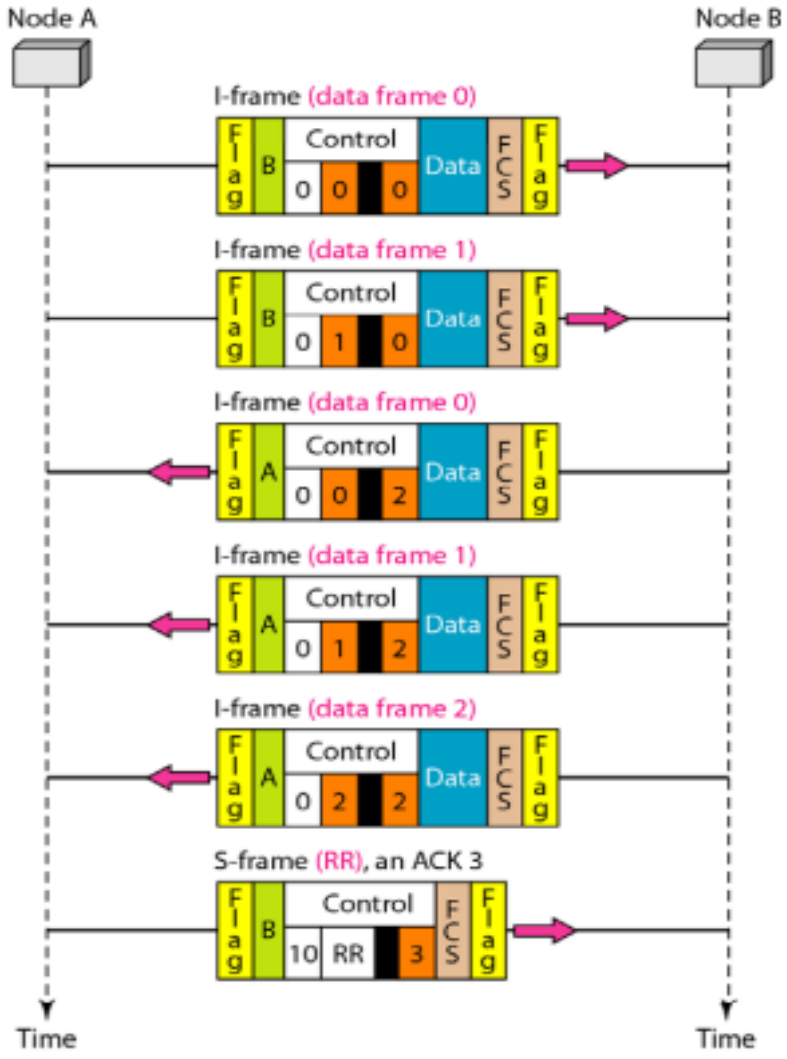
Example 11.10

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.

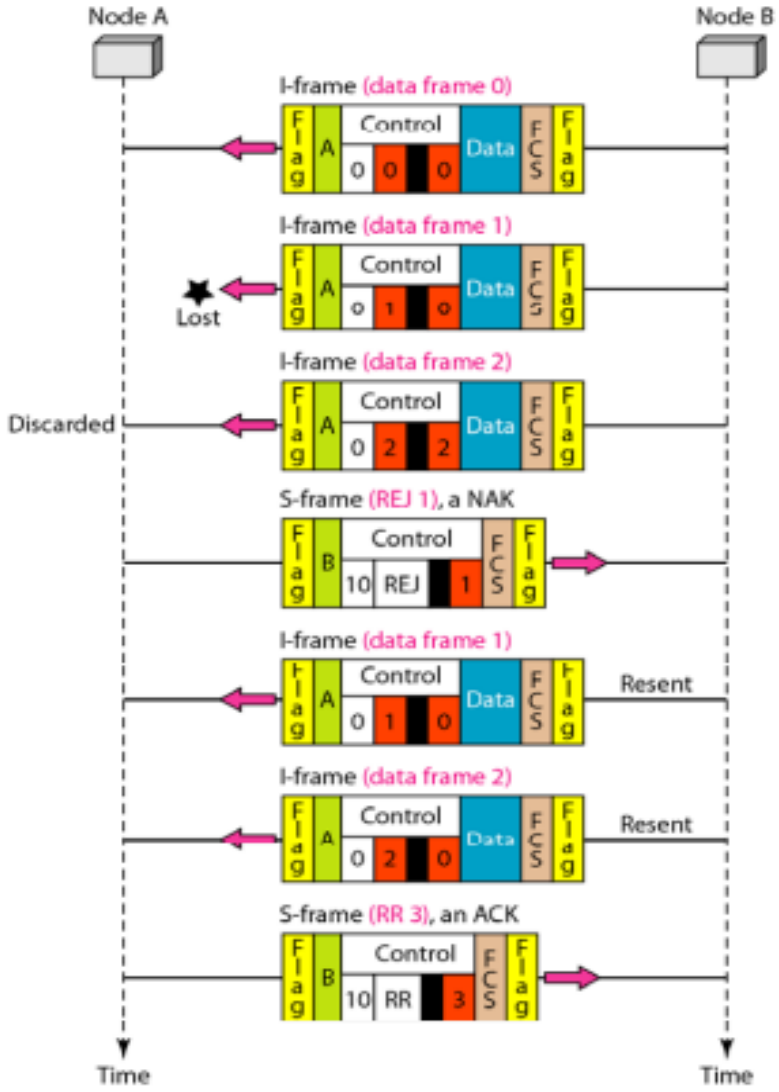
Figure 11.30 *Example of piggybacking without error*



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*



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

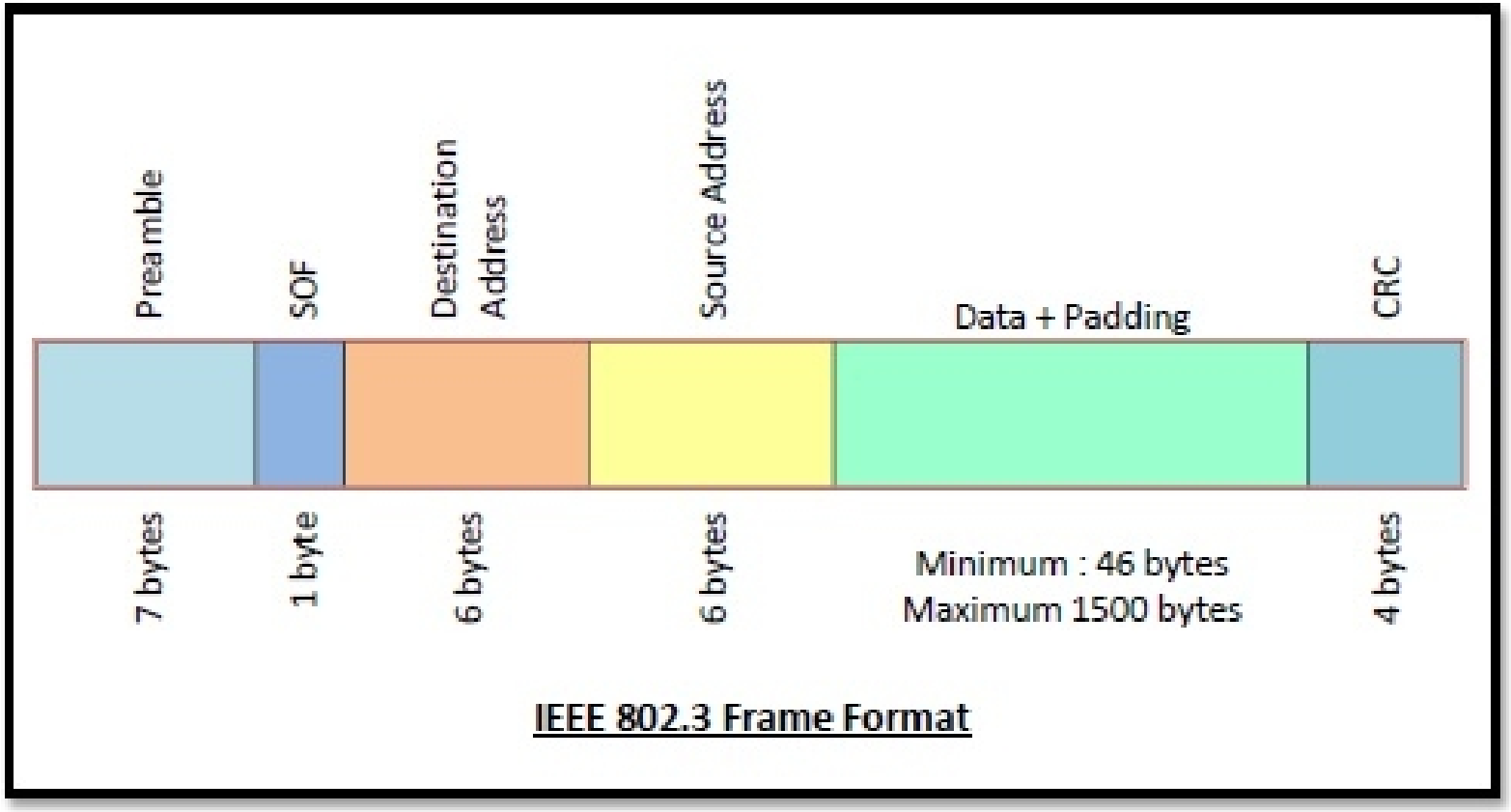
ETHERNET (IEEE 802.3)

- 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 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 (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

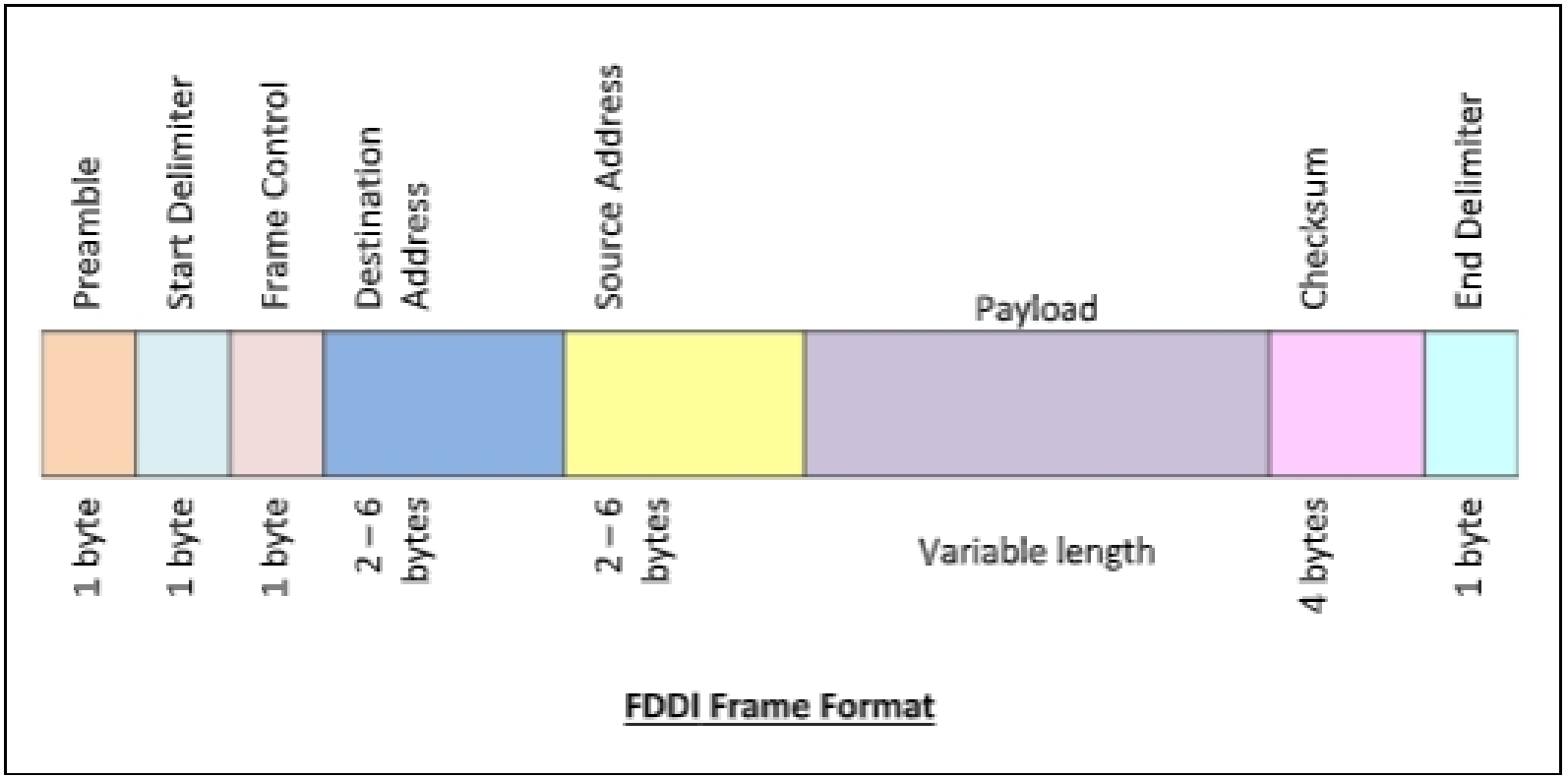
ETHERNET (IEEE 802.3) Frame Format

- ✓ **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



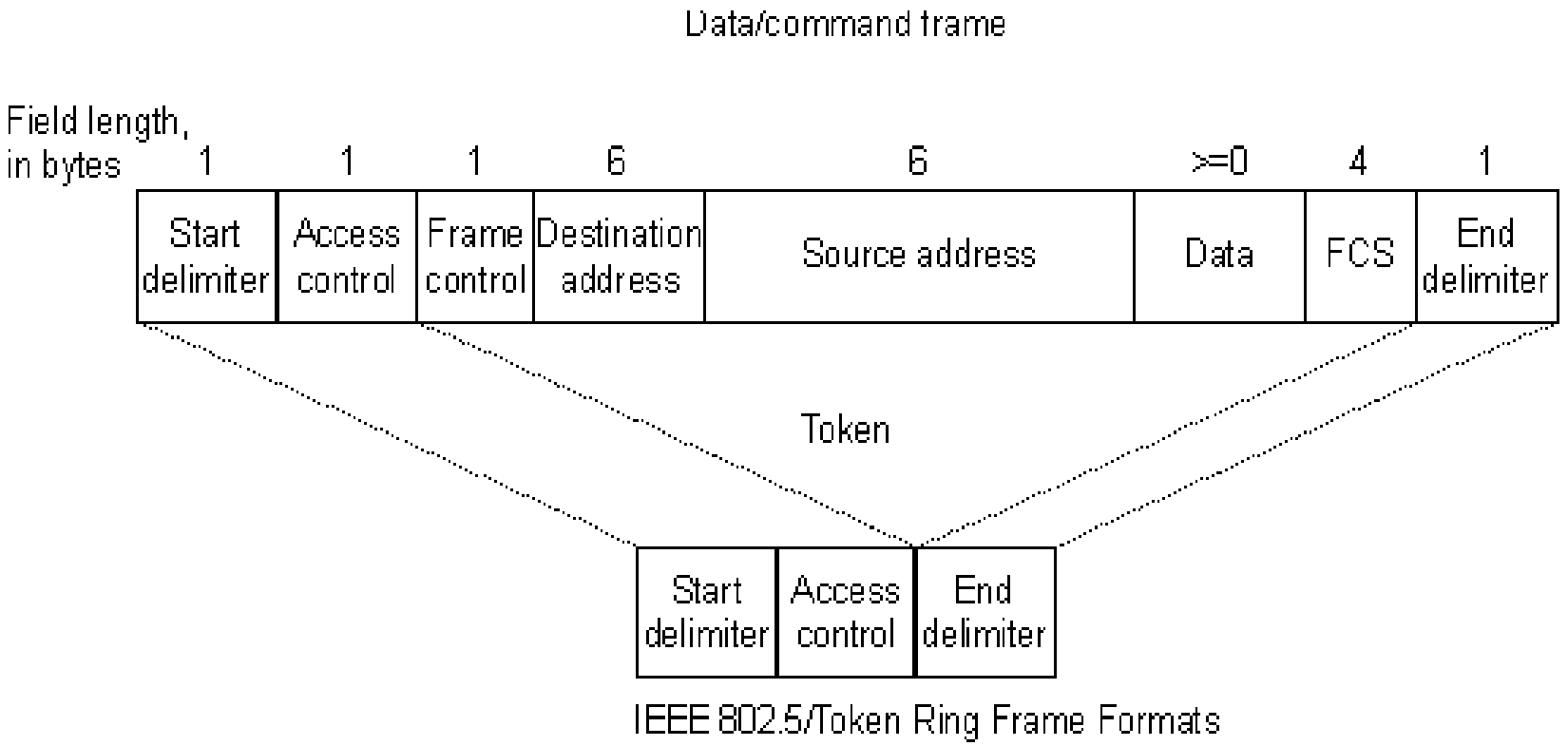
TOKEN BUS (IEEE 802.4) Frame Format

- ✓ **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



IEEE 802.5/Token Ring Frame Formats

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.

Module 2

IEEE 802.11

14-1 IEEE 802.11

IEEE has defined the specifications for a wireless LAN, called IEEE 802.11, which covers the physical and data link layers.

Architecture

The standard defines two kinds of services

1. the basic service set (BSS)

2. the extended service set (ESS)

Figure 14.1 *Basic service sets (BSSs)*

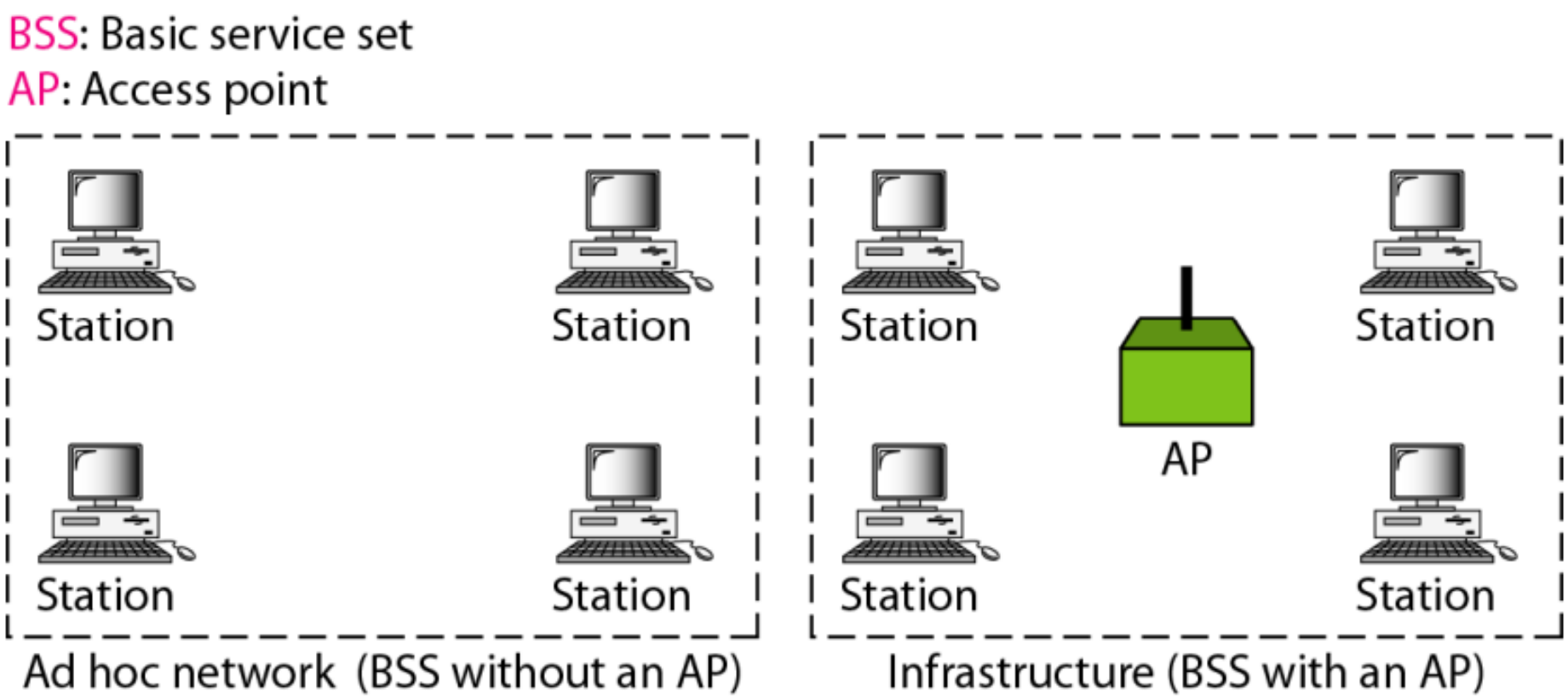


Figure 14.2 *Extended service sets (ESSs)*

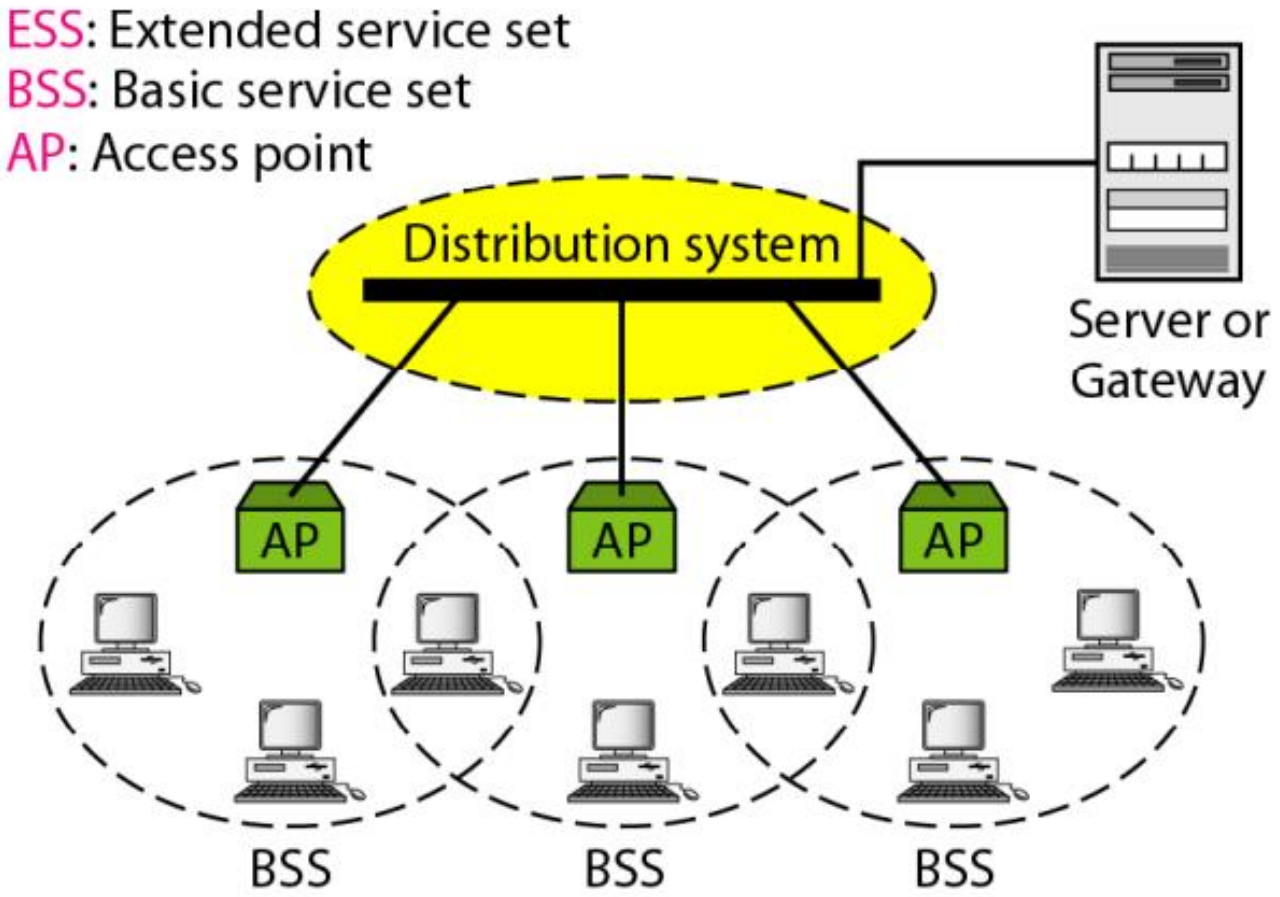


Figure 14.7 *Frame format*

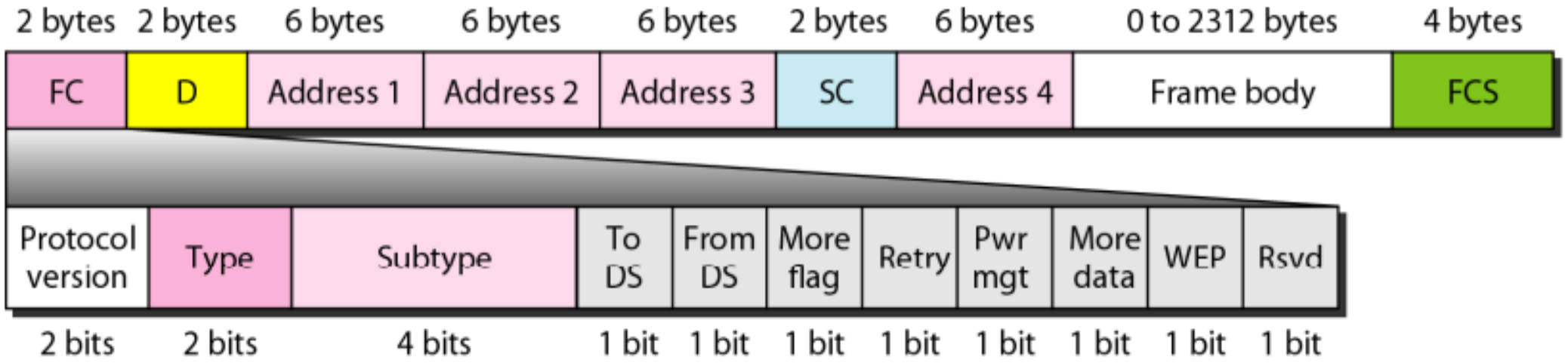


Table 14.1 *Subfields in FC field*

Field	*Explanation*
Version	Current version is 0
Type	Type of information: management (00), control (01), or data (10)
Subtype	Subtype of each type (see Table 14.2)
To DS	Defined later
From DS	Defined later
More flag	When set to 1, means more fragments
Retry	When set to 1, means retransmitted frame
Pwr mgt	When set to 1, means station is in power management mode
More data	When set to 1, means station has more data to send
WEP	Wired equivalent privacy (encryption implemented)
Rsvd	Reserved

Figure 14.8 *Control frames*

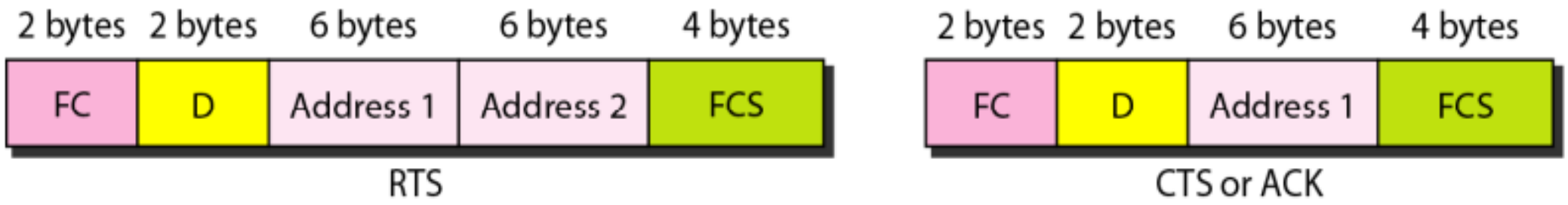


Table 14.2 ***Values of subfields in control frames***

Subtype	*Meaning*
1011	Request to send (RTS)
1100	Clear to send (CTS)
1101	Acknowledgment (ACK)

Table 14.3 *Addresses*

To DS	*From DS*	*Address 1*	*Address 2*	*Address 3*	*Address 4*
0	0	Destination	Source	BSS ID	N/A
0	1	Destination	Sending AP	Source	N/A
1	0	Receiving AP	Source	Destination	N/A
1	1	Receiving AP	Sending AP	Destination	Source

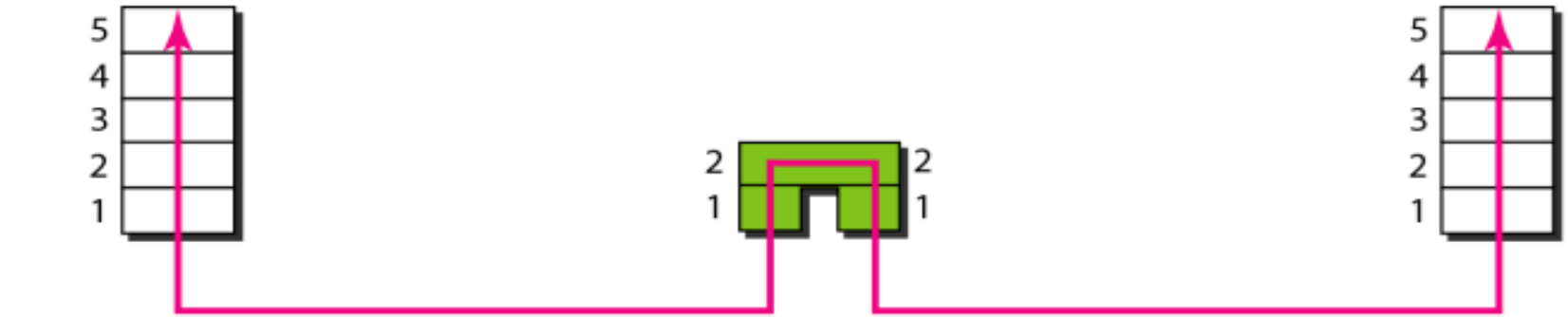
FDDI - Fiber Distributed Data Interface

- It is a network standard that uses fiber optic connections in a local area network (LAN)
- It can extend in range up to 200 kilometers (124 miles).
- The FDDI protocol is based on the token bus protocol.
- A FDDI LAN can support thousands of users.

Bridges

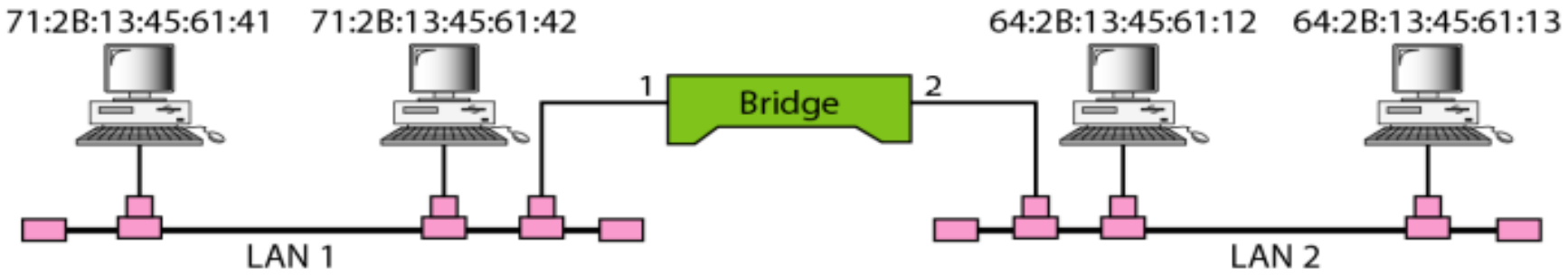
- A bridge operates in both the physical and the data link layer.
- As a physical device, it regenerates the signal it receives.
- As a data link layer device, the bridge can check the physical (MAC) addresses contained in the frame

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



MODULE 2:

Flow Control and Error control – Stop and Wait – Go Back N ARQ – Selective Repeat ARQ – Sliding Window Techniques – HDLC – LAN – Ethernet IEEE 802.3 (Only Description) IEEE 802.4 and IEEE 802.5 – IEEE 802.11–FDDI – Bridges