데이터통신시스템

서강대학교
소재우

5. Data Link Control(2/2)

학습에 앞서

▪학습개요

– **데이터링크 제어의 필요성과** ARQ **프로토콜의 동작을 학습한다**.

▪학습목표

– ARQ **프로토콜 동작을 학습하고 장단점을 이해한다**.
– Stop-and-Wait ARQ, Go-Back-N ARQ, Selective Repeat ARQ **프로토콜 동작을 학습한다**.

Go-Back-N ARQ

- To improve the efficiency of transmission, multiple frames must be in transition while waiting for acknowledgment.

- Go-Back-N ARQ
  - We can send several frames before receiving acknowledgments;
  - We keep a copy of these frames until the acknowledgements arrive.

- Sequence Numbers
  - The sequence numbers range from 0 to 2^m-1.
  - For example, if m = 4, SN = {0, 1, 2, …, 15, 0, 1, 2, …}

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.

1. Noisy Channels

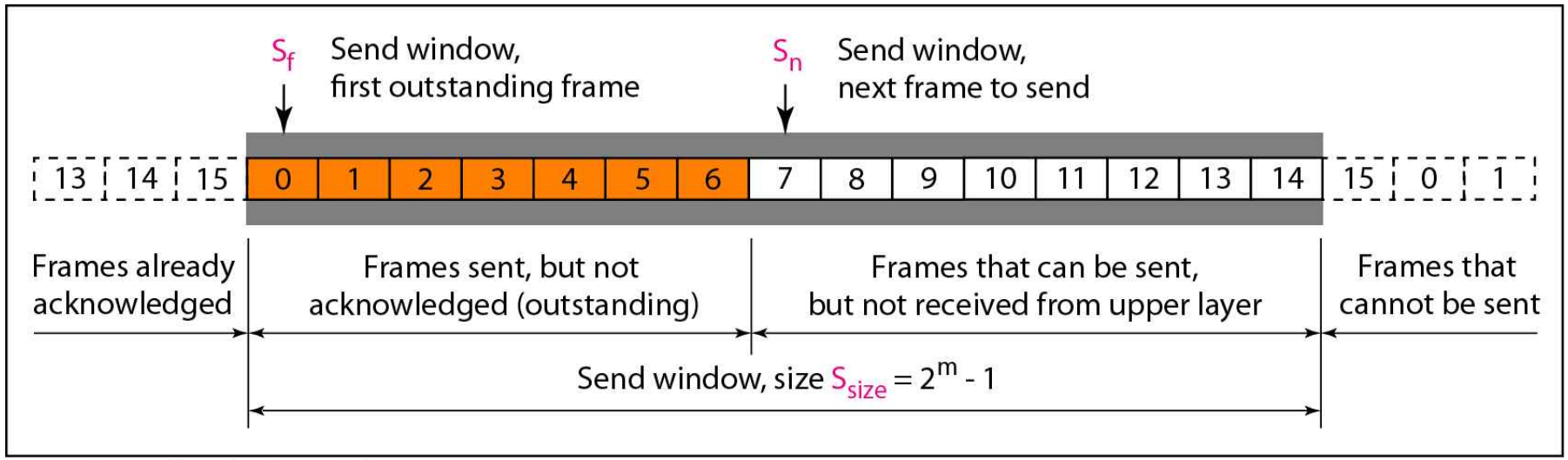
- Sliding Window
  - An abstract concept that defines the range of sequence numbers that is the concern of the sender and receiver.

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}.

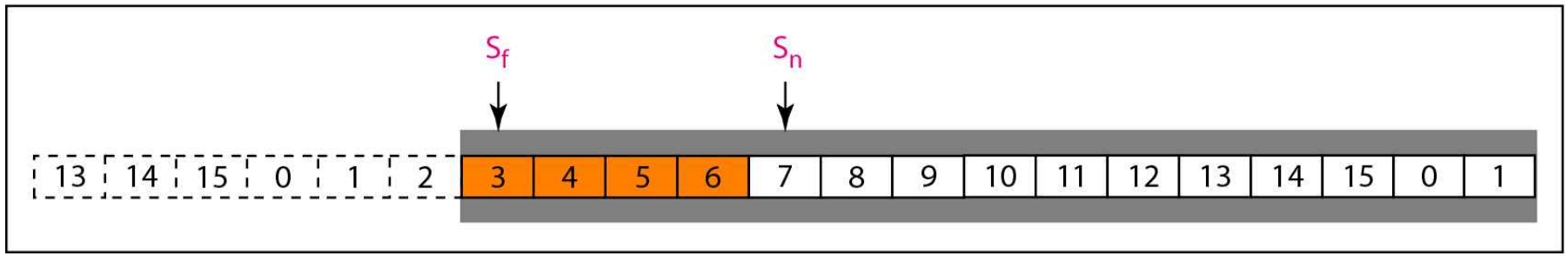
- The acknowledgements are cumulative, meaning that more than one frame can be acknowledged by an ACK frame.

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

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



a. Send window before sliding



b. Send window after sliding

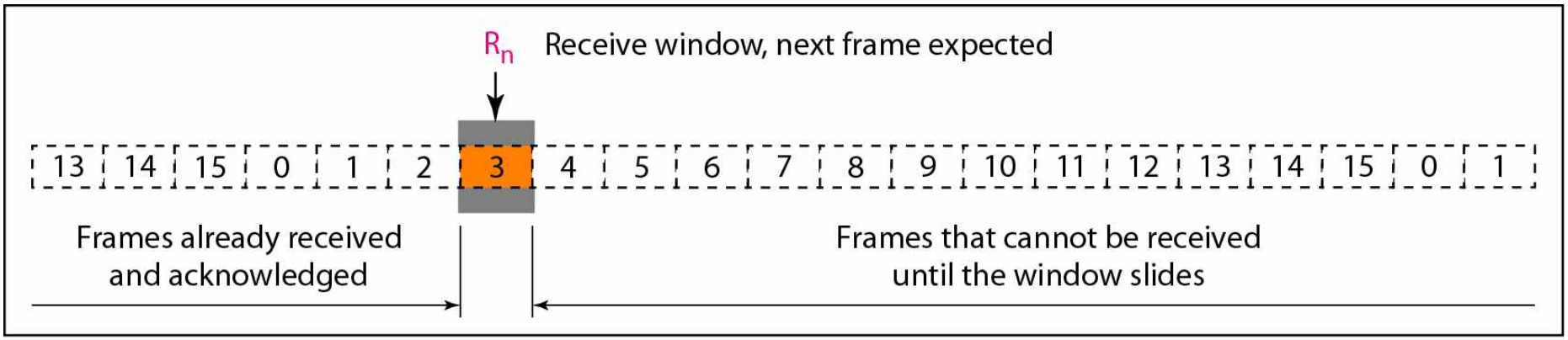
1. Noisy Channels

- Receive Sliding Window
  - The receive window slides only one slot at a time.

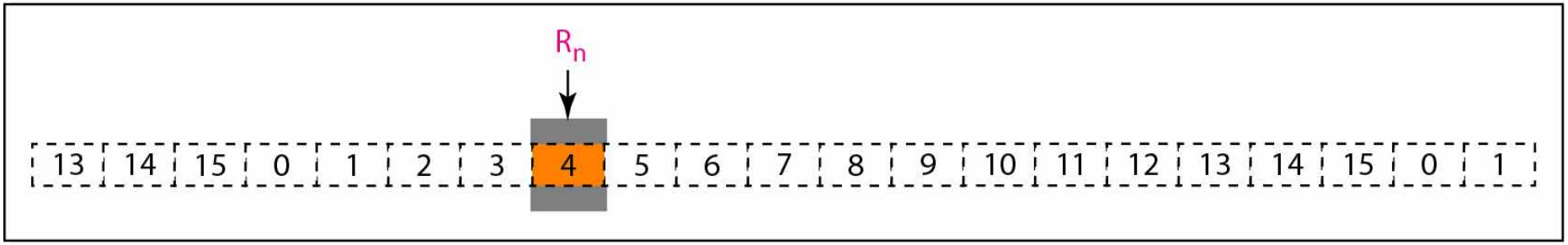
The receive window is an abstract concept defining an imaginary box of size 1 with one single variable R_n.
The window slides when a correct frame has arrived; sliding occurs one slot at a time.

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.

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



a. Receive window



b. Window after sliding

1. Noisy Channels

- Timers
  - We use only one timer.
  - The reason is that the timer for the first outstanding frame always expires first; we send all outstanding frames when this timer expires.

- Acknowledgement
  - The receiver sends a positive acknowledgement.
  - If a frame is damaged or is received out of order,
    - The receiver is silent and will discard all subsequent frames until it receives the one it is expecting.
    - After timer expires, the resends all outstanding frames.
  - The receiver does not have to acknowledge each frame received. → It can send one cumulative acknowledgement for several frames.

- Resending a Frame
  - For example, the sender has already sent frame 6, but the timer for frame 3 expires.
  - The sender goes back and sends frames 3, 4, 5, and 6 again.

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

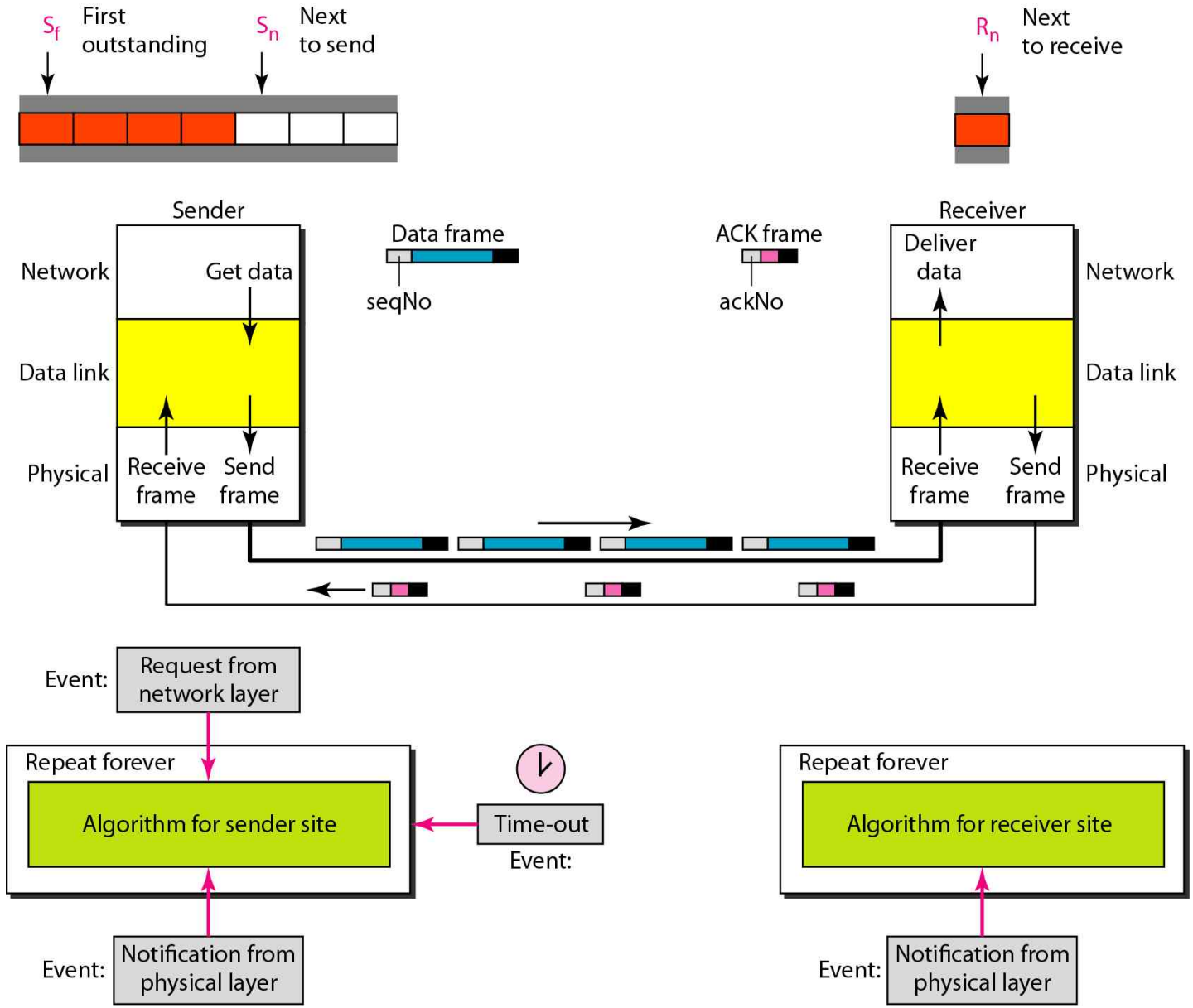
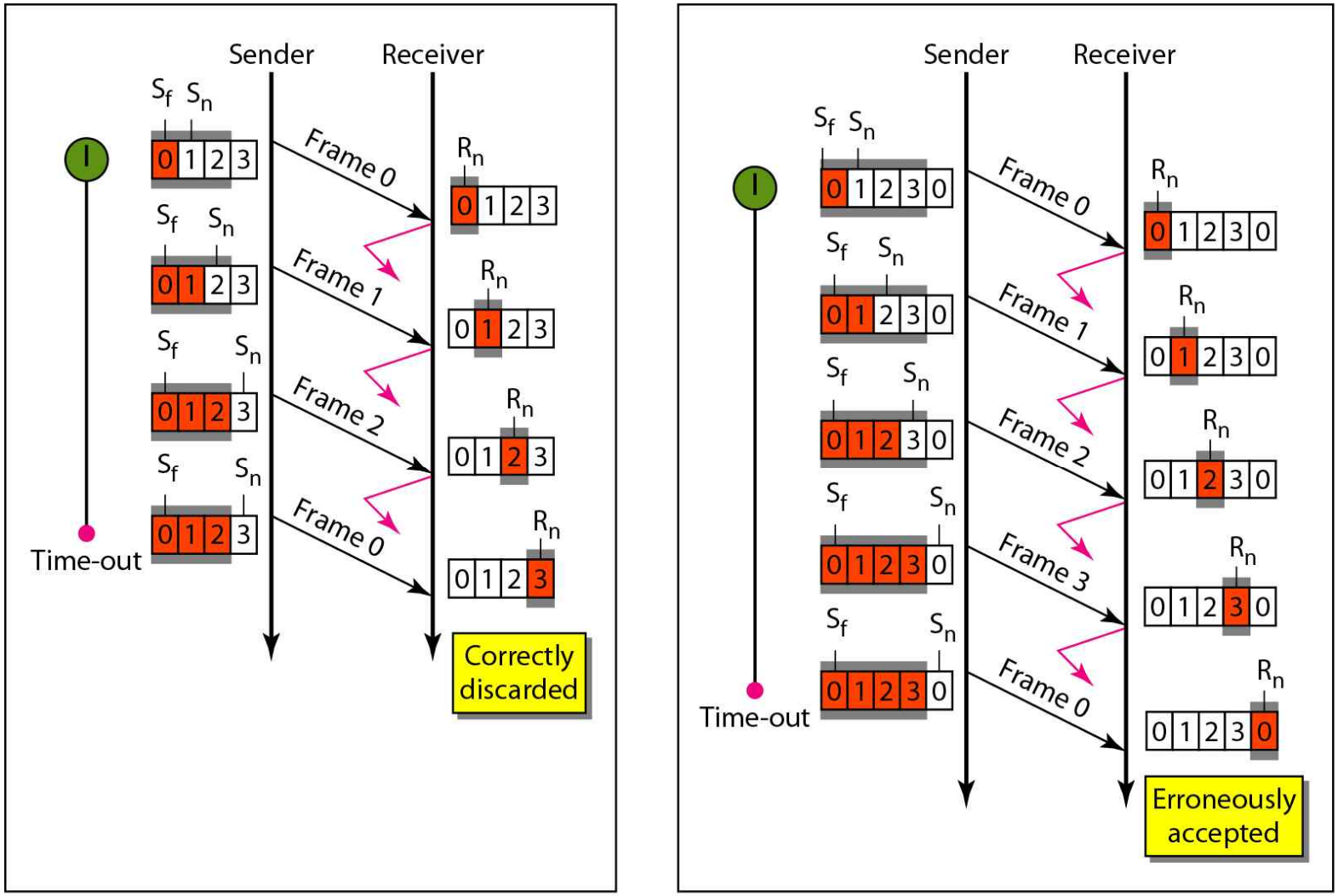


Figure 11.15 *Window size for Go-Back-N ARQ*

As an example, m = 2 and all acknowledgments are lost



a. Window size < 2^m

b. Window size = 2^m

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
```

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 }
```

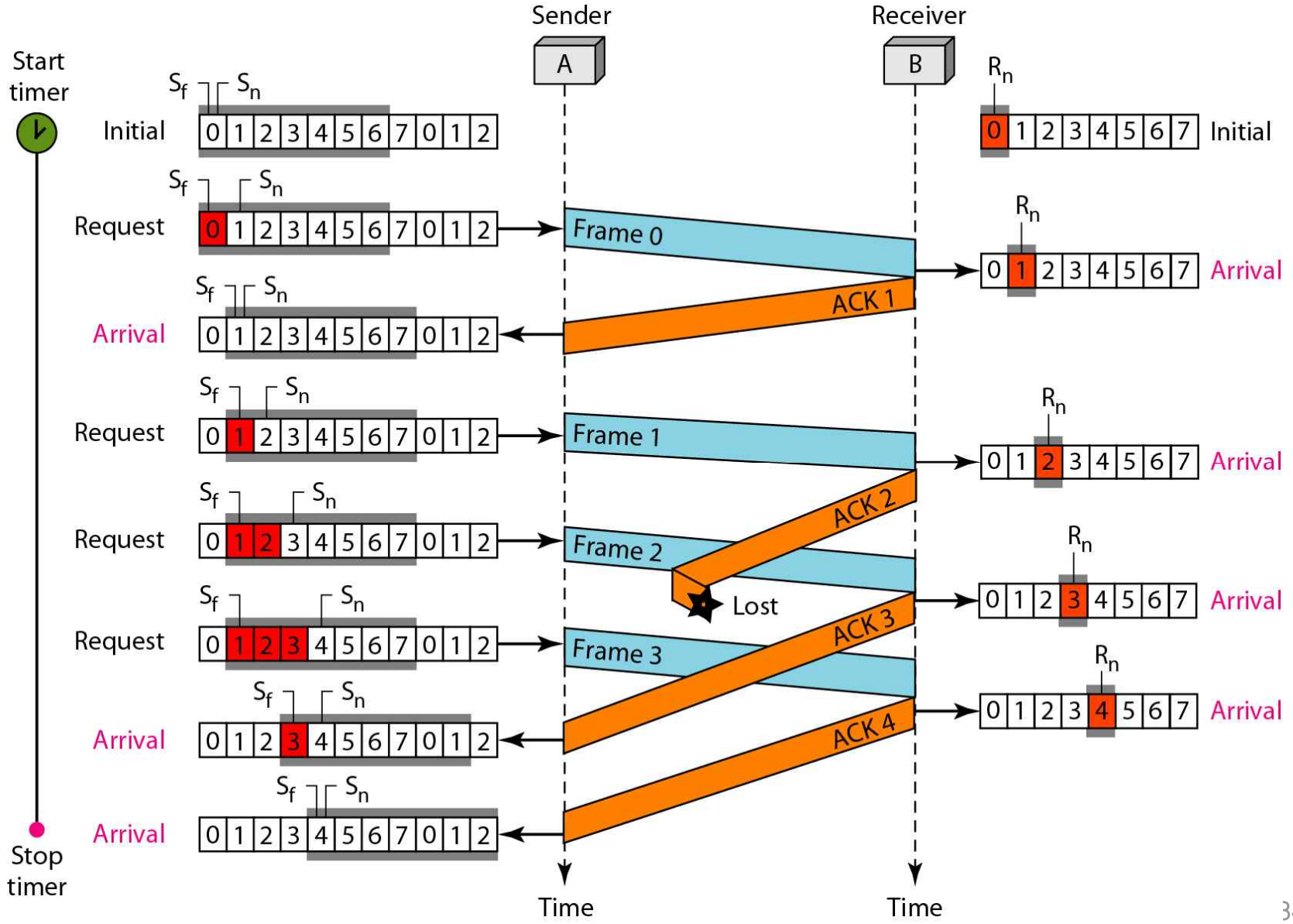
Algorithm 11.8 *Go-Back-N receiver algorithm*

```
 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*

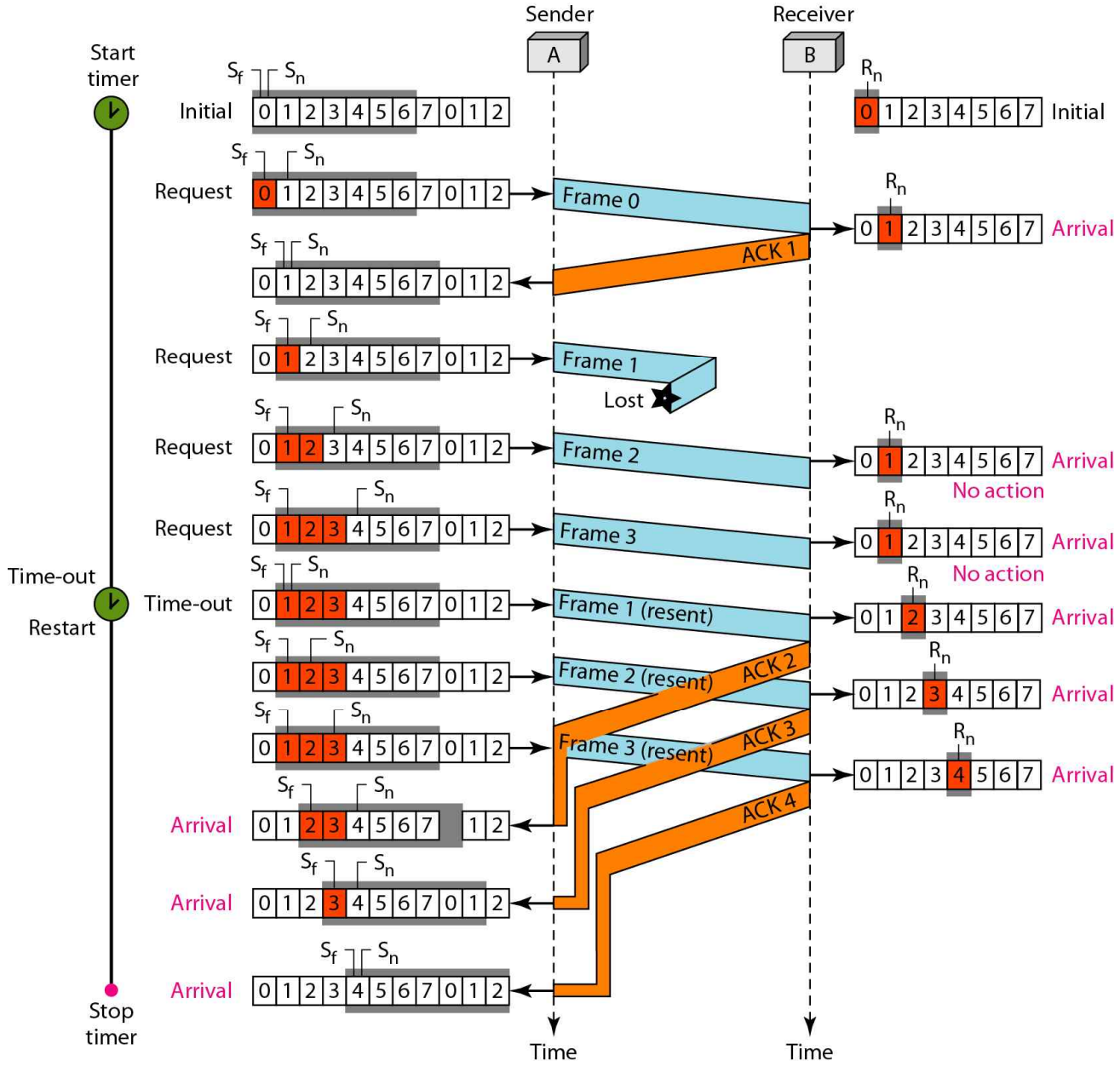


Example 11.7

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.

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*



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

Selective Repeat ARQ

- Selective Repeat ARQ
  - Does not resend N frames when just one frame is damaged;
  - Only the damaged frame is resent.

- Windows
  - Two windows: a send window and a receive window
  - Size: 2^{m-1} (the two windows has the same size)
  - The smaller window size means less efficiency but fewer duplicate frames

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

Figure 11.18 *Send window for Selective Repeat ARQ*

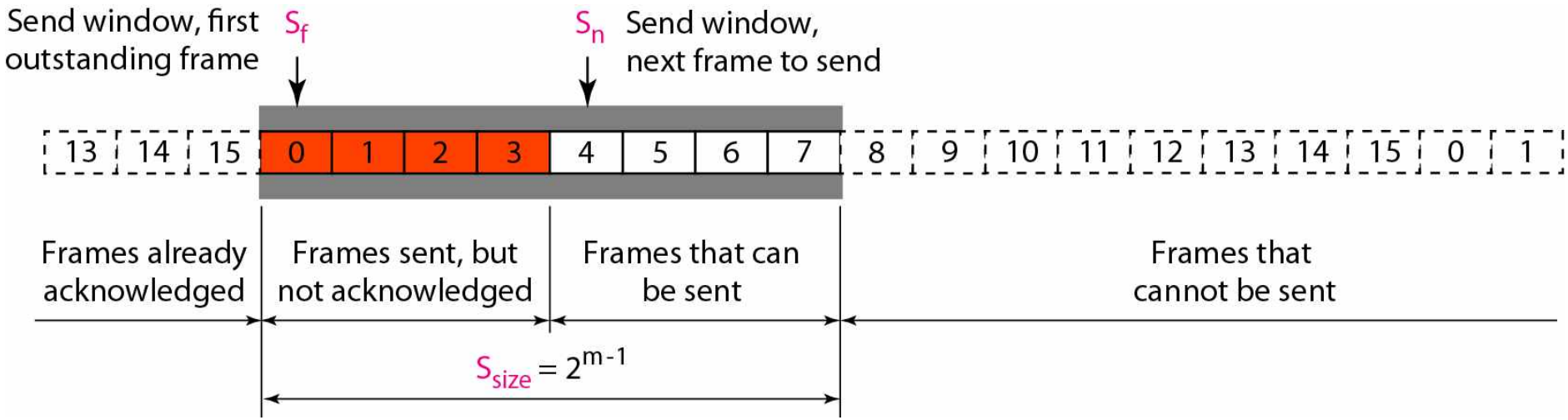


Figure 11.19 *Receive window for Selective Repeat ARQ*

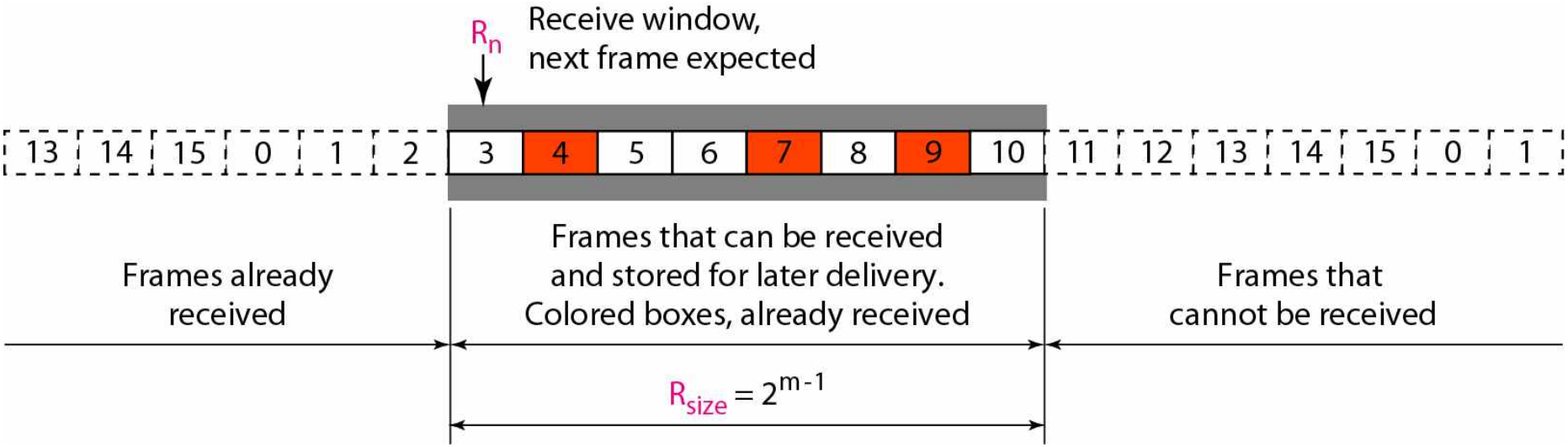


Figure 11.20 *Design of Selective Repeat ARQ*

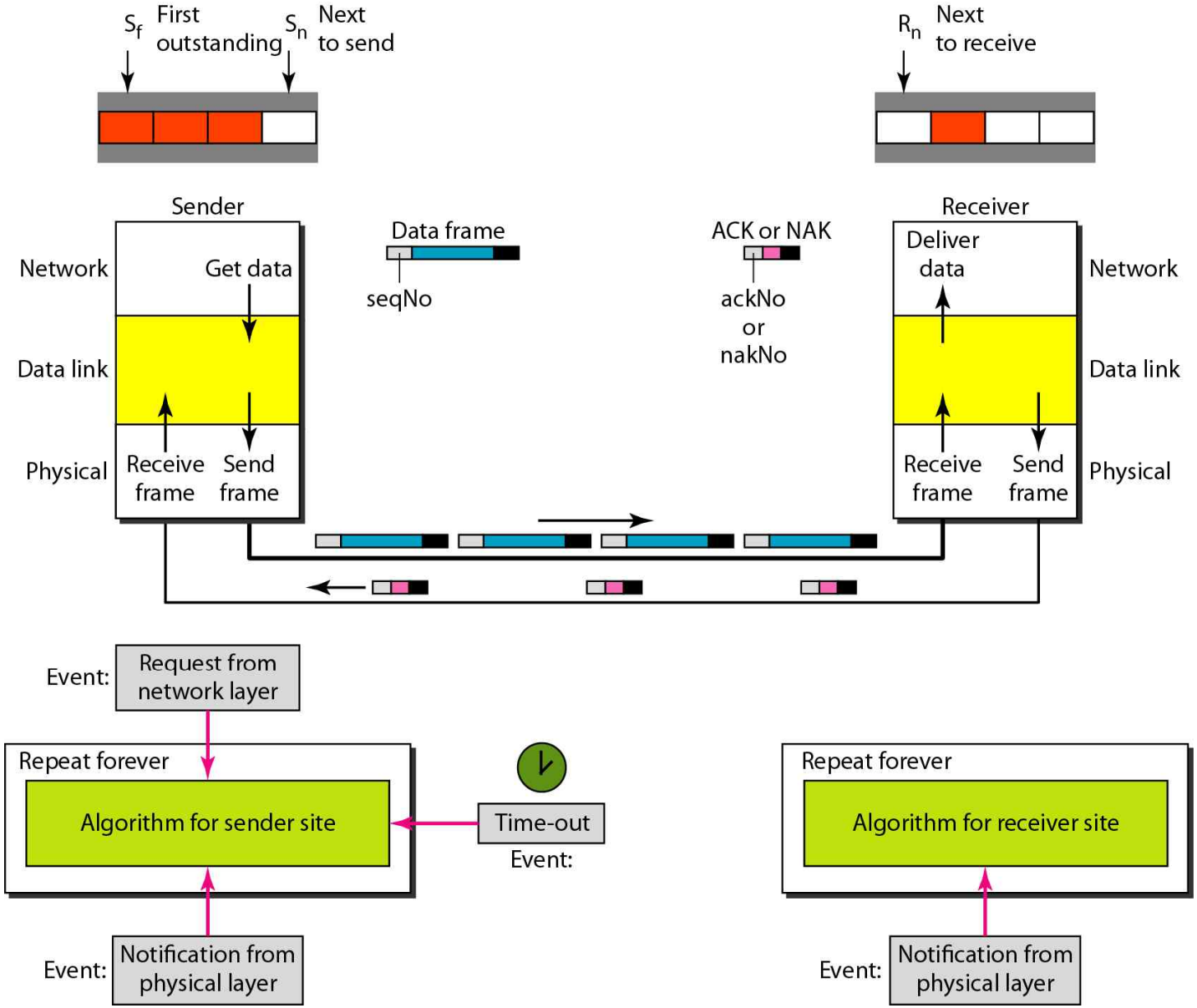
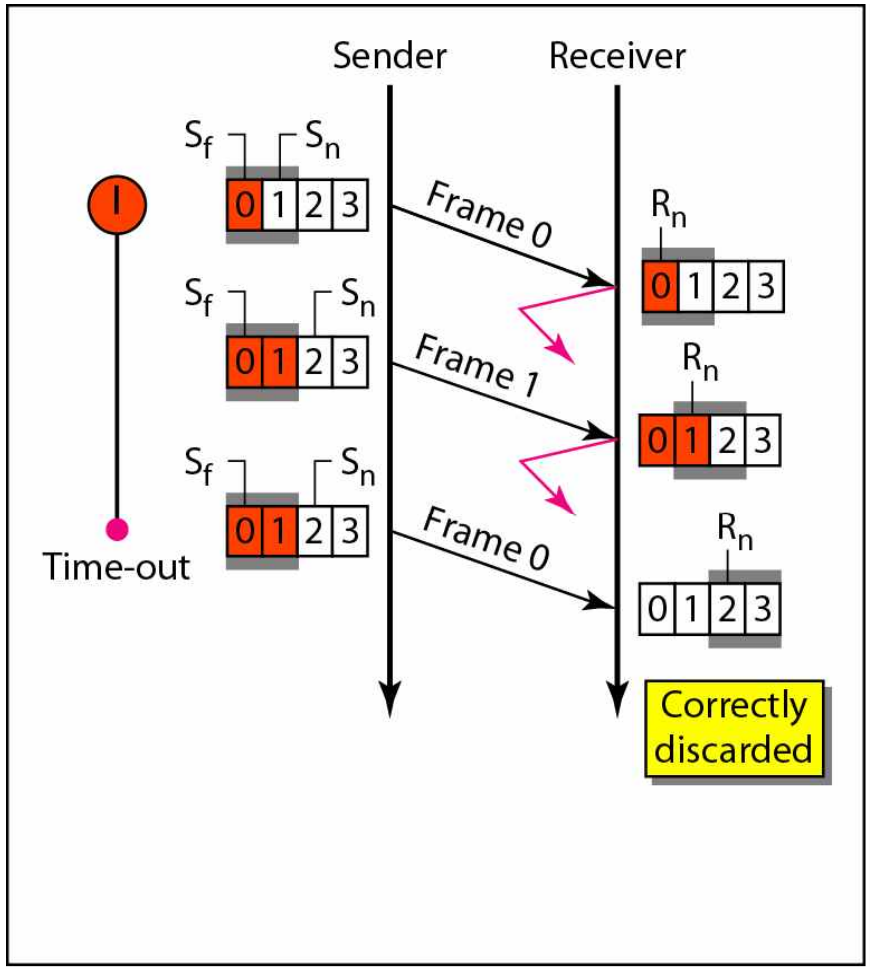
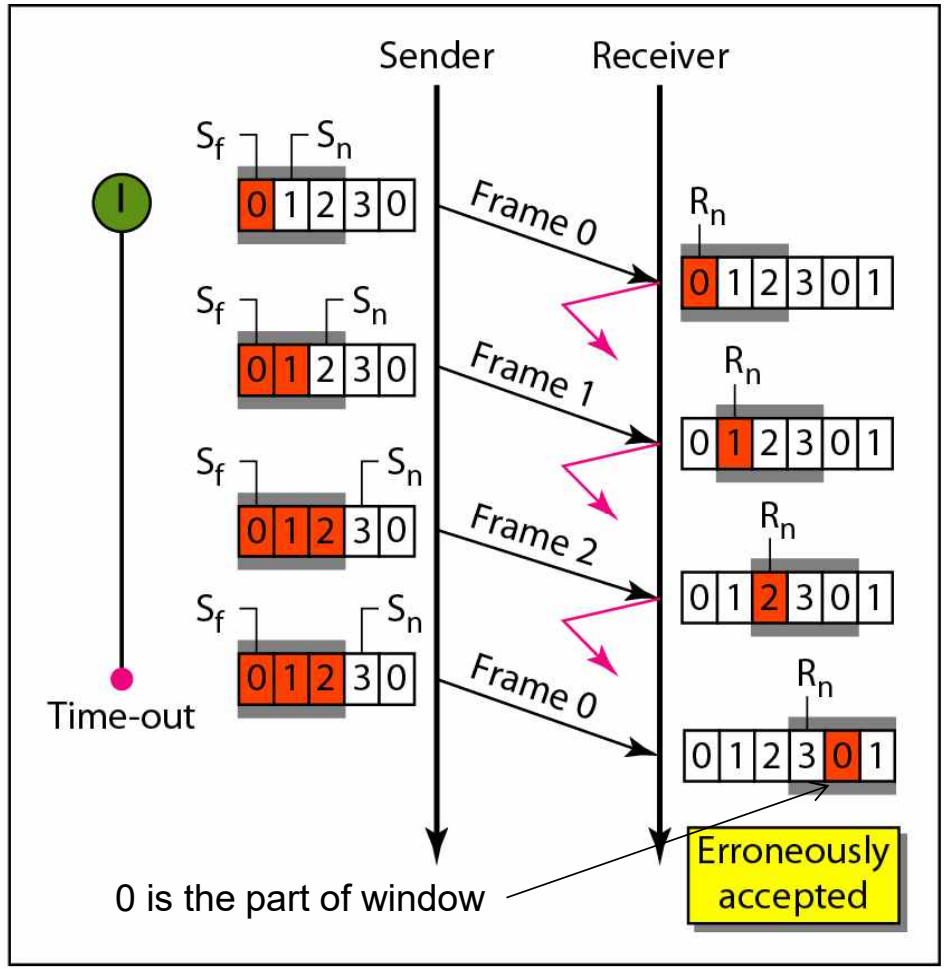


Figure 11.21 *Selective Repeat ARQ, window size*

As an example, m = 2 and all acknowledgments are lost



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



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

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
```

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  }
```

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  }
```

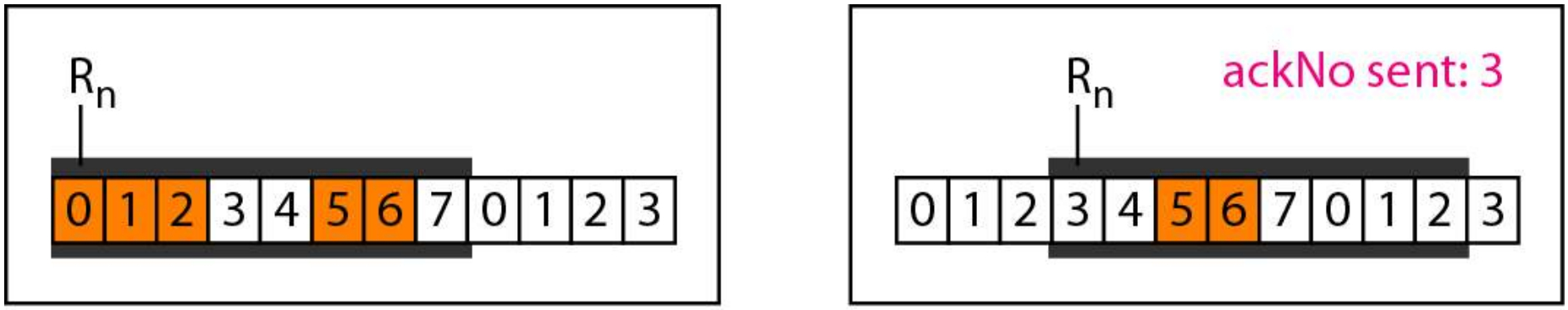
Algorithm 11.10 *Receiver-site Selective Repeat algorithm*

```
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*



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.

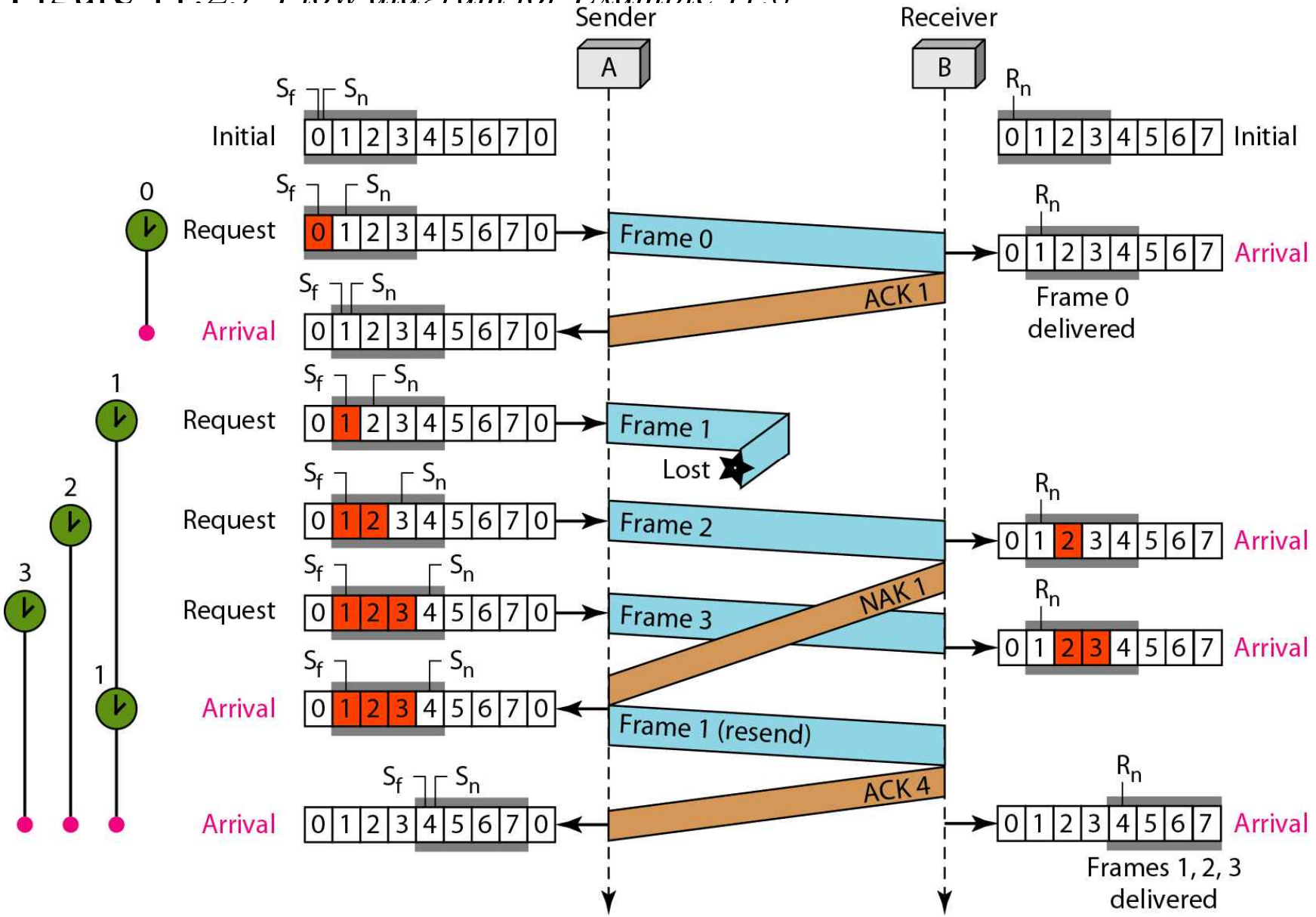
At the receiver site we need to distinguish between the acceptance of a frame and its delivery to the network layer. At the second arrival, frame 2 arrives and is stored and marked, but it cannot be delivered because frame 1 is missing. At the next arrival, frame 3 arrives and is marked and stored, but 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

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.

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.

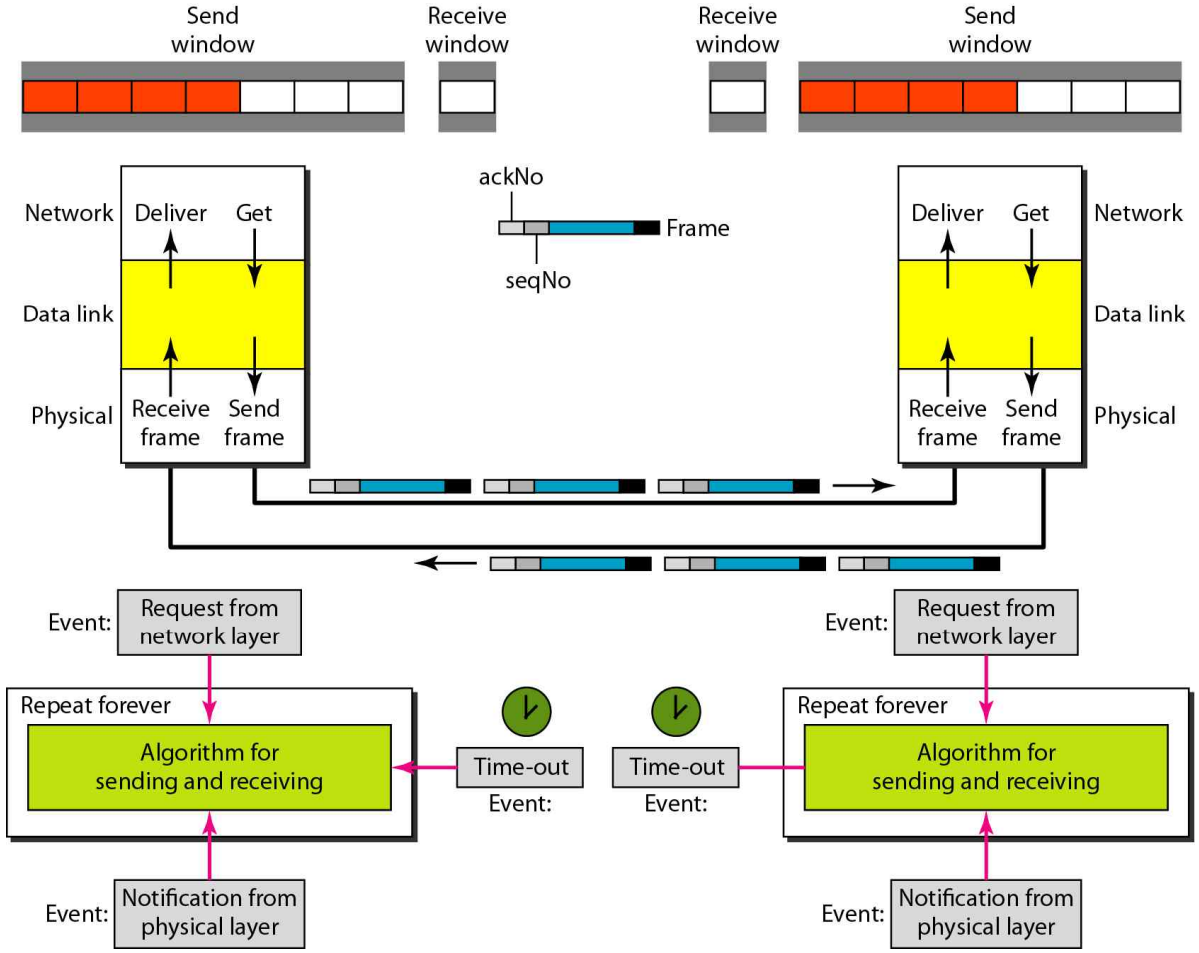
Figure 11.23 *Flow diagram for Example 11.8*



Piggybacking

- When a frame is carrying data from A to B, it can also carry control information about arrived (or lost) frames from B.

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



Summary

- Error control is both error detection and error correction.
- In Stop-and-Wait ARQ, the sender sends a frame and waits for an acknowledgment from the receiver before sending the next frame.
- In Go-Back-N ARQ, multiple frames can be in transit at the same time. If there is an error, retransmission begins with the last unacknowledged frame even if subsequent frames have arrived correctly. Duplicate frames are discarded.
- In Selective Repeat ARQ, multiple frames can be in transit at the same time. If there is an error, only the unacknowledged frame is retransmitted.
- Flow control mechanisms with sliding windows have control variables at both sender and receiver sites.