Flow Control, and different Protocols

Presented By

Dr. Md. Abir Hossain

Topics Discussed in the Section

- ✓ **Process-to-Process Communication**
- ✓ **Flow Control**
- ✓ **Error Control**
- ✓ **Sliding window**
- ✓ **Simple Protocol**
- ✓ **Stop-and-Wait Protocol**
- ✓ **Go-Back-*N* Protocol**
- ✓ **Selective-Repeat Protocol**

Process to Process Communication

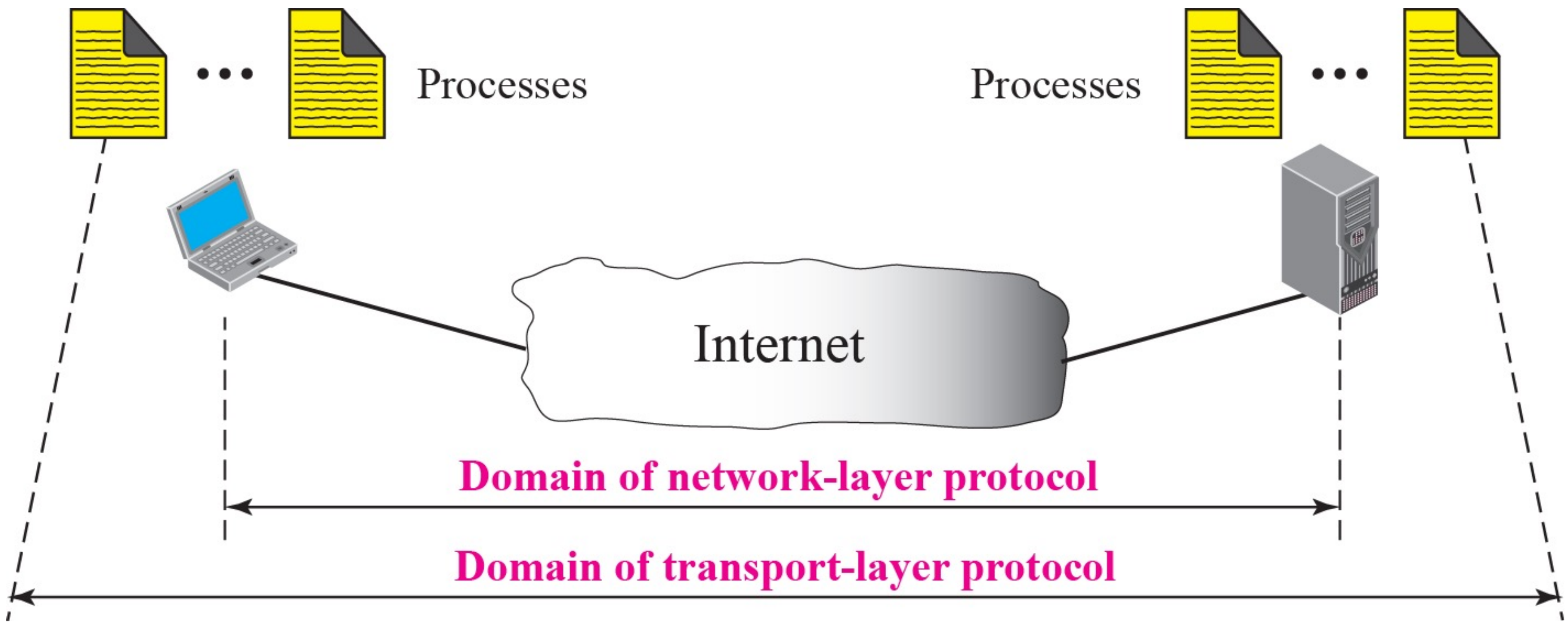


Figure 13.1 *Network layer vs Transport layer*

- In network layer, the message is transmitted but it`s not usable to application layer.
- The transport layer takes over the message and convert it to segment and transfer it for further processing to make it usabe in a process.
- Process is an application layer entity.

Flow Control Mechanisms

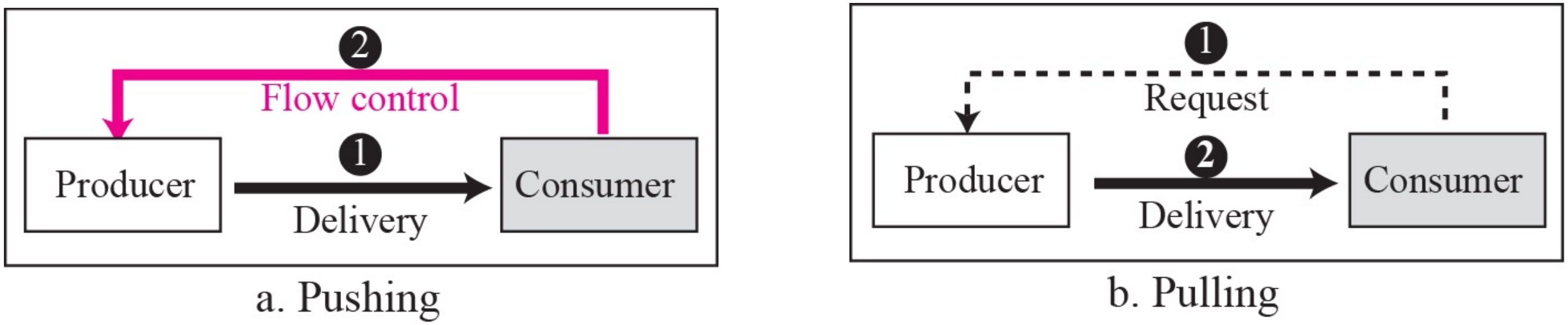


Figure 13.8 ***Pushing or Pulling***

- Delivery of a packet from a sender to a receiver can be occur in two ways: Pushing or Pulling
- If the sender transmits packet wheenever it arrives without any prior request to receiver called pushing.
- If the sender transmits a packet after the receiver`s request called pulling.

Flow Control at transport Layer

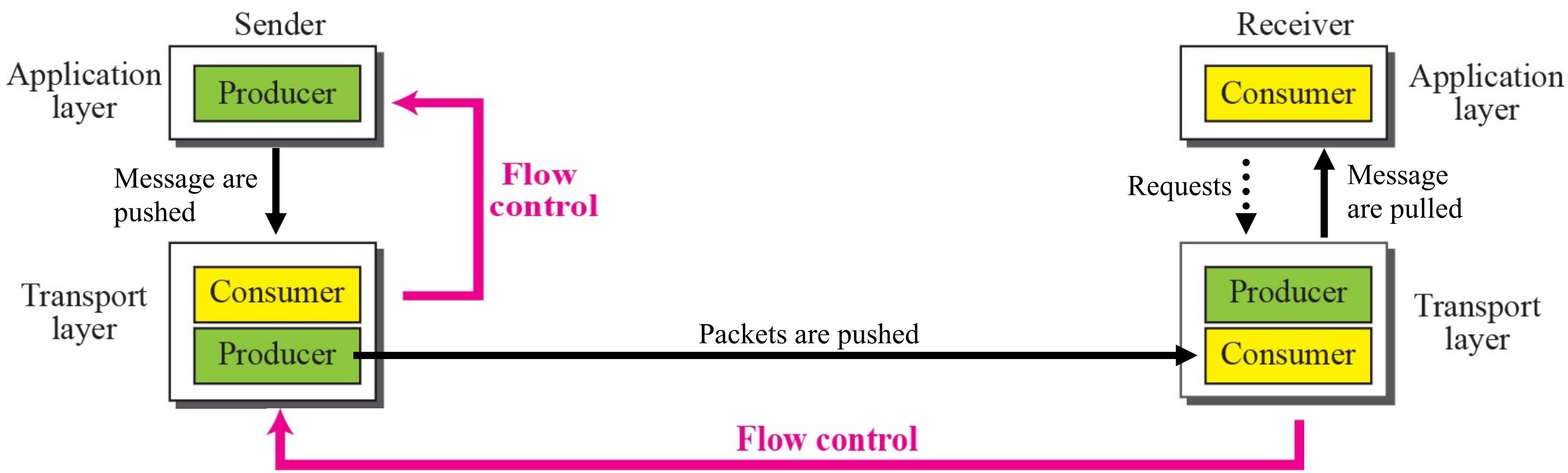


Figure 13.9 *Flow control at Transport Layer*

- The sending process at the application layer is only a producer. It produces message chunks and pushes them to the transport layer.
- The sending transport layer has a **double role**: it is both a consumer and the producer. It consumes the messages pushed by the producer. It encapsulates the messages in packets and pushes them to the receiving transport layer.
- The receiving transport layer has also a **double role**: it is the consumer for the packets received from the sender. It is also a producer; it needs to decapsulate the messages and deliver them to the application layer.
- The last delivery, however, is normally a pulling delivery; the transport layer waits until the application-layer process asks for messages.

Error control

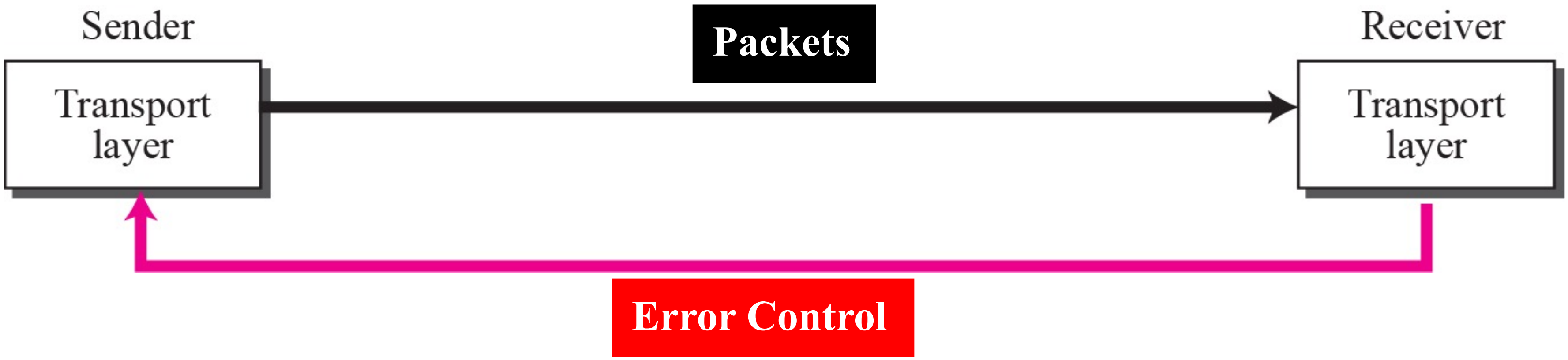


Figure 13.10 *Error control at Transport Layer*

Error control at the transport layer is responsible to

- Detect and discard corrupted packets.
- Keep track of lost and discarded packets and resend them.
- Recognize duplicate packets and discard them.
- Buffer out-of-order packets until the missing packets arrive.

Sliding Window

- The buffer is presented as a set of slice, called sliding window. It follows modulo 2^m (sequence number from 0 to 2^m -1)

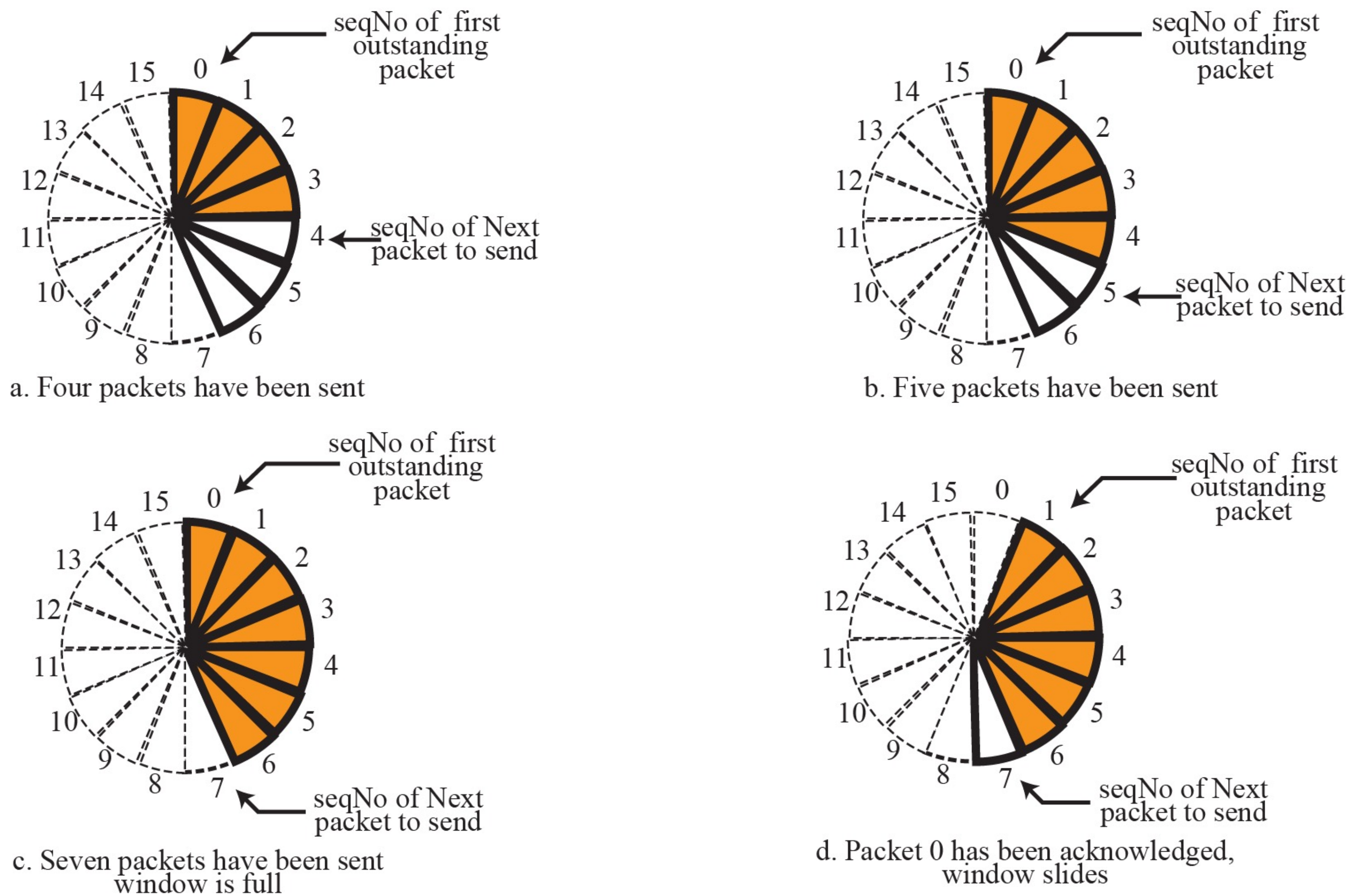


Figure 13.11 *Sliding window in circular format*

Sliding Window

- The sequence number are modulo 16 (m=4) and size of the window is 7.

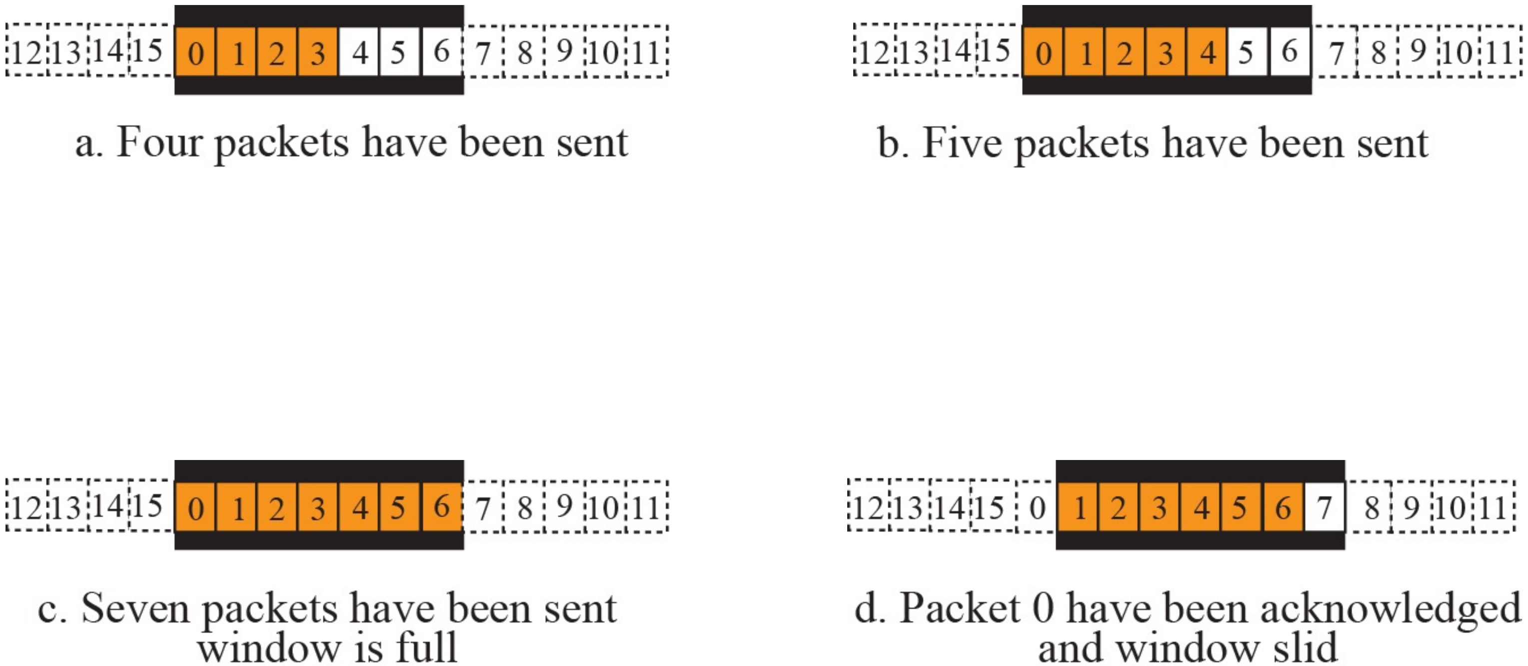


Figure 13.12 ***Sliding window in linear format***

Simple Protocol

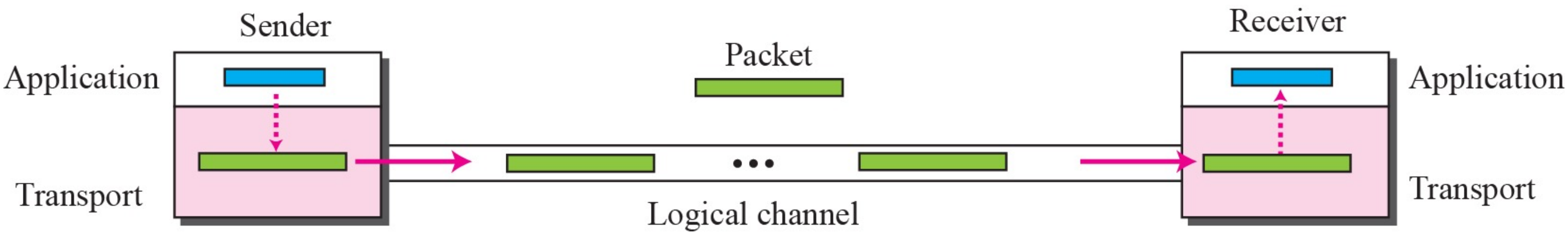
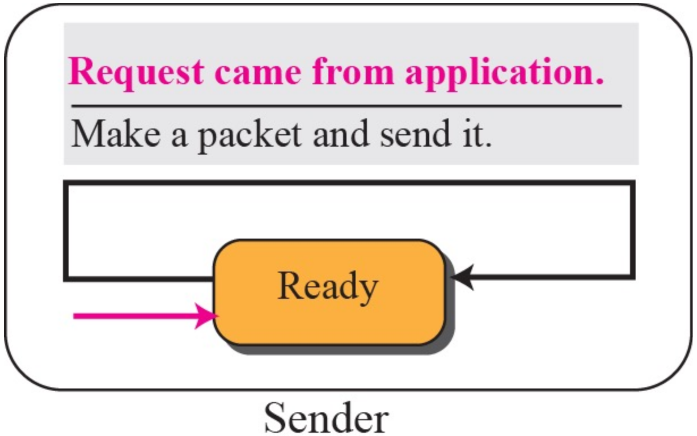


Figure 13.16 ***Simple protocol***

- Simple connectionless protocol with neither flow nor error control.
- The sender gets a message from its application layer, makes a packet out of it, and sends the packet.
- The transport layer at the receiver receives a packet from its network layer, extracts the message from the packet, and delivers the message to its application layer.
- The transport layers of the sender and receiver provide transmission services for their application layers.

FSM for Simple Protocol



The sender site should not send a packet until its application layer has a message to send.

The receiver site cannot deliver a message to its application layer until a packet arrives.

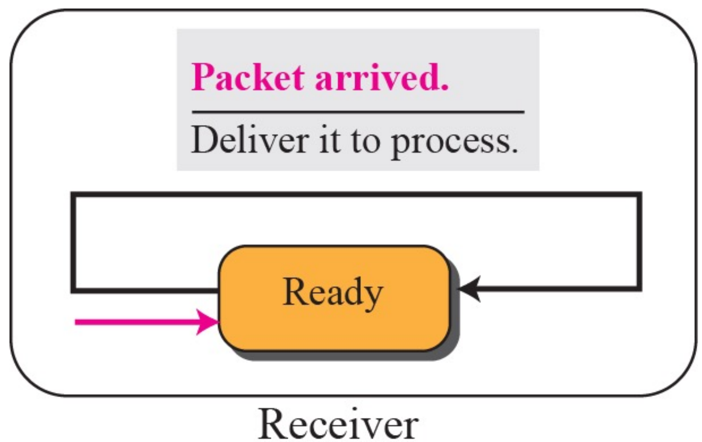


Figure 13.17 *FSMs for simple protocol*

Stop-and-wait Protocol

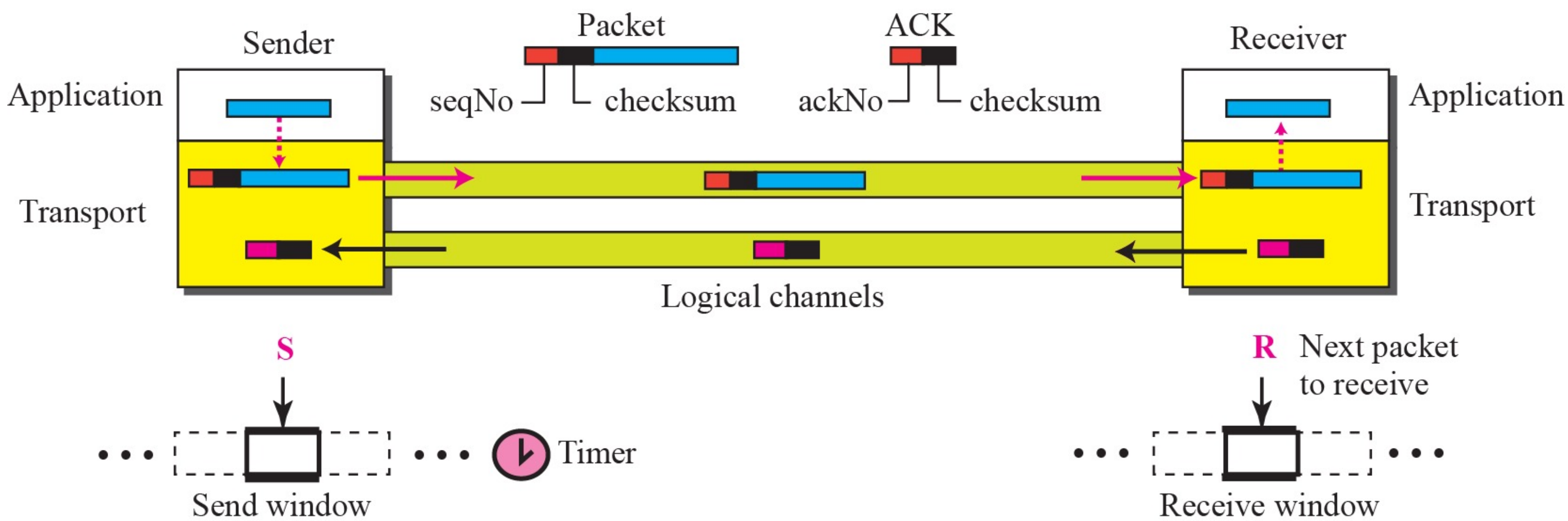


Figure 13.19 ***Stop-and-wait protocol***

- The sender sends one packet at a time and waits for an acknowledgment before sending the next packet.
- Every time the sender sends a packet, it starts a timer. If an acknowledgment arrives before the timer expires, the timer is stopped and the sender sends the next packet.
- When a packet arrives at the receiver site, it is checked. If its checksum is incorrect, the packet is corrupted and silently discarded.
- If the timer expires, the sender resends the previous packet assuming that either the packet was lost or corrupted.

Note

In Stop-and-Wait protocol, flow control is achieved by forcing the sender to wait for an acknowledgment, and error control is achieved by discarding corrupted packets and letting the sender resend unacknowledged packets when the timer expires.

FSM for Stop-and-wait Protocol

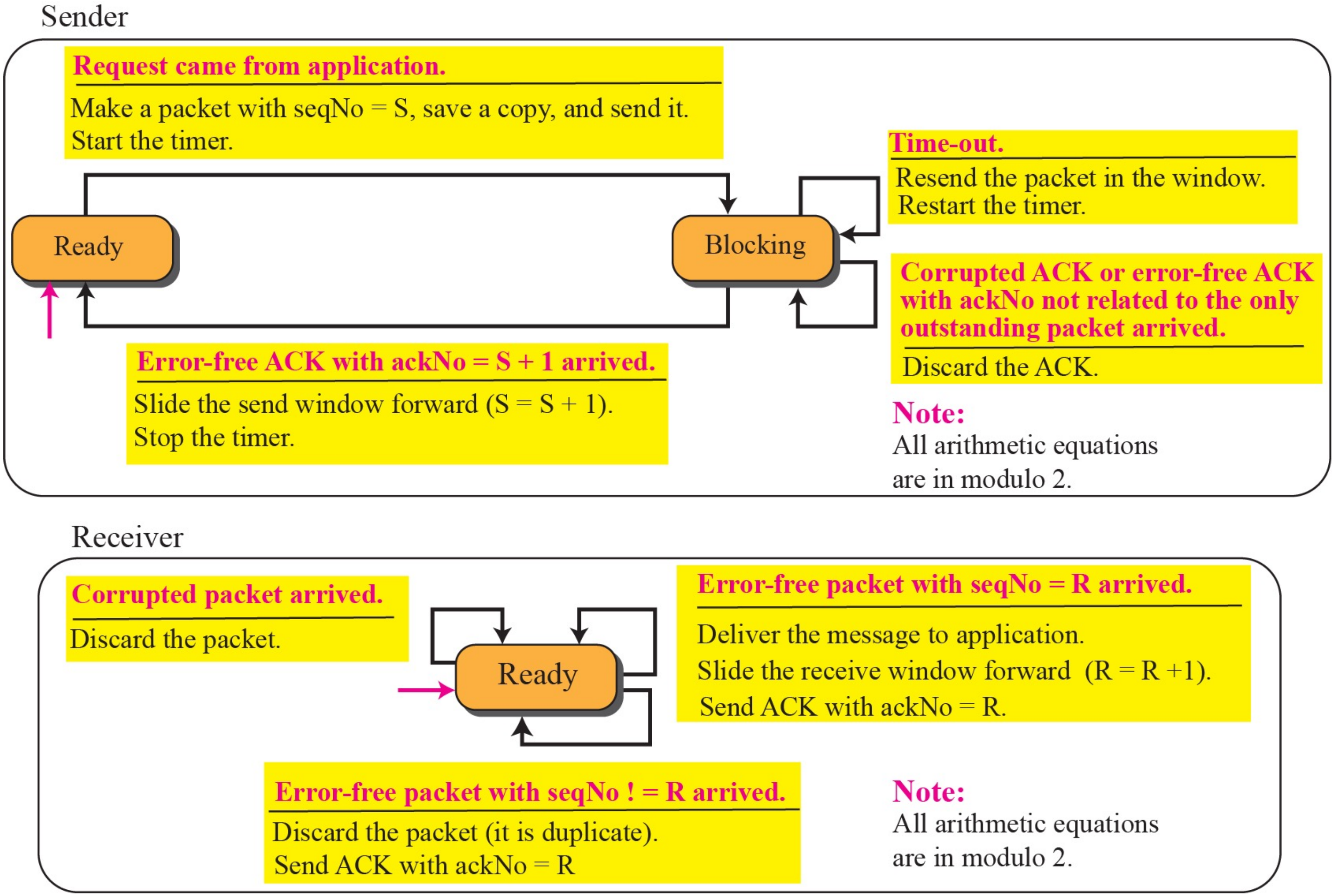


Figure 13.20 *FSMs for stop-and-wait protocol*

Example 13.4

Figure 13.21 shows an example of Stop-and-Wait protocol. Packet 0 is sent and acknowledged. Packet 1 is lost and resent after the time-out. The resent packet 1 is acknowledged and the timer stops. Packet 0 is sent and acknowledged, but the acknowledgment is lost. The sender has no idea if the packet or the acknowledgment is lost, so after the time-out, it resends packet 0, which is acknowledged.

Flow diagram of Stop-and-wait Protocol

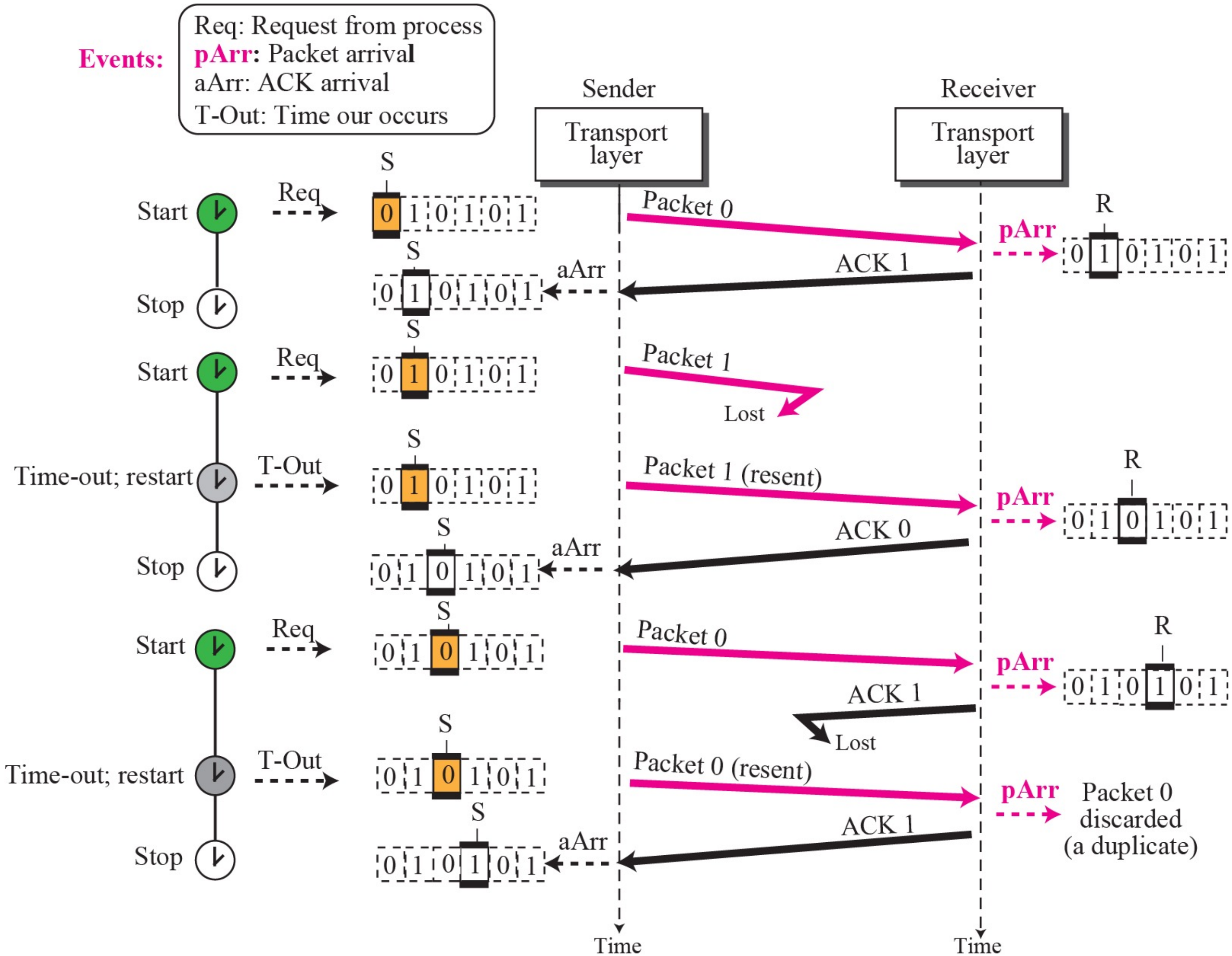


Figure 13.21 *Flow diagram of stop-and-wait protocol*

Example 13.5

Assume that, in a Stop-and-Wait system, the bandwidth of the line is 1 Mbps, and 1 bit takes 20 milliseconds to make a round trip. What is the bandwidth-delay product? If the system data packets are 1,000 bits in length, what is the utilization percentage of the link?

Solution

Channel Utilization= (Packet length/Link Rate)/ RTT+ (Packet length/Link Rate)

The bandwidth-delay product is $(1 \times 10^6) \times (20 \times 10^{-3}) = 20{,}000$ bits. The system can send 20,000 bits during the time it takes for the data to go from the sender to the receiver and the acknowledgment to come back. However, the system sends only 1,000 bits. We can say that the link utilization is only 1,000/20,000, or 5 percent. For this reason, for a link with a high bandwidth or long delay, the use of Stop-and-Wait wastes the capacity of the link.

Example 13.6

What is the utilization percentage of the link in Example 13.5 if we have a protocol that can send up to 15 packets before stopping and worrying about the acknowledgments?

Solution

The bandwidth-delay product is still 20,000 bits. The system can send up to 15 packets 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 packets, the utilization percentage is much less because packets have to be resent.

Go-Back-N Protocol

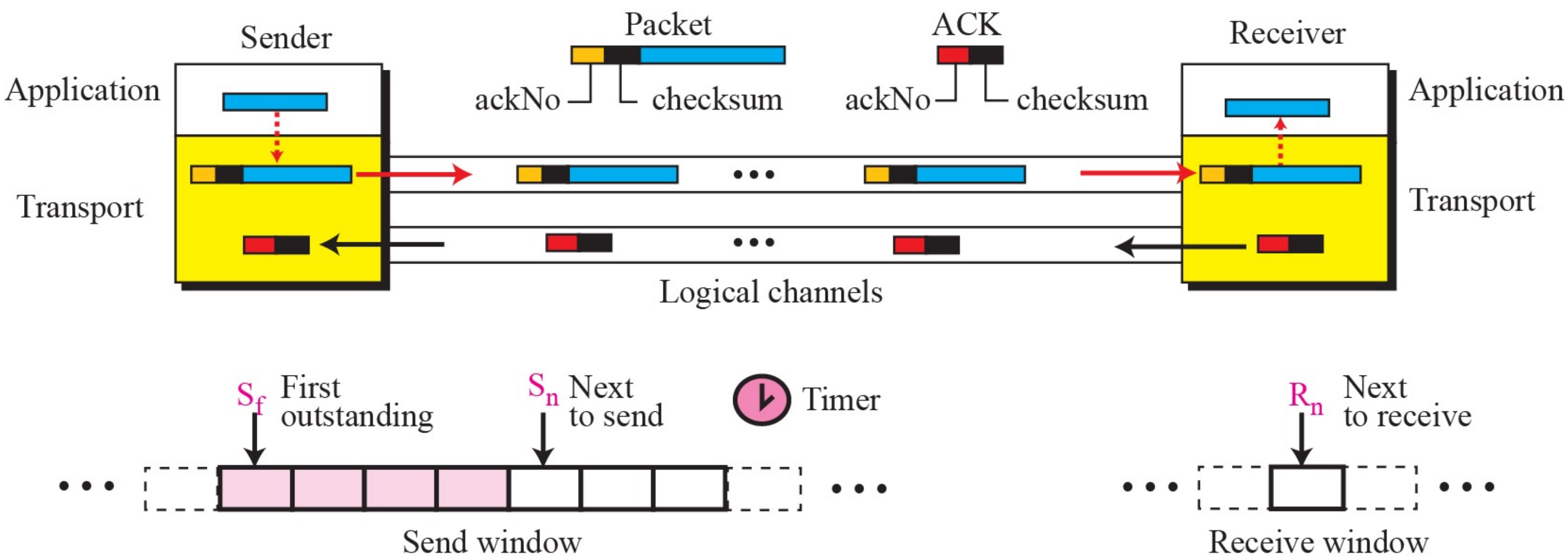


Figure 13.22 ***Go-Back-N protocol***

- The key concept of the Go-back-*N(GBN)* is to send several packets before receiving acknowledgments.
- The receiver can only buffer one packet at a time.
- The sender keep a copy of the sent packets until the acknowledgments arrive.
- The Go-back-*N(GBN)* considers the sliding window of size 7(m=3)
- 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.

Window for Go-Back-N Protocol

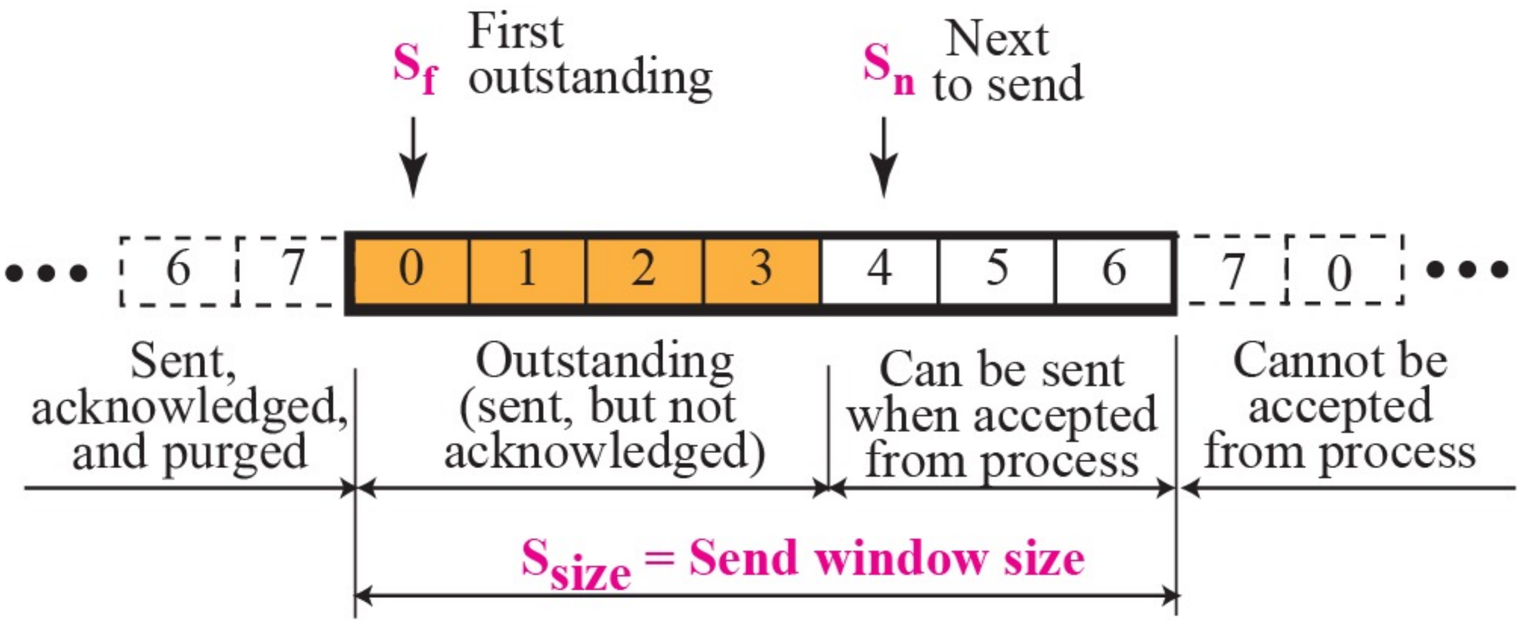


Figure 13.23 *Send window for Go-Back-N*

Note

The send window is an abstract concept defining an imaginary box of maximum 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 an error-free ACK with ackNo between S_f and S_n (in modular arithmetic) arrives.

Sliding Window for Go-Back-N Protocol

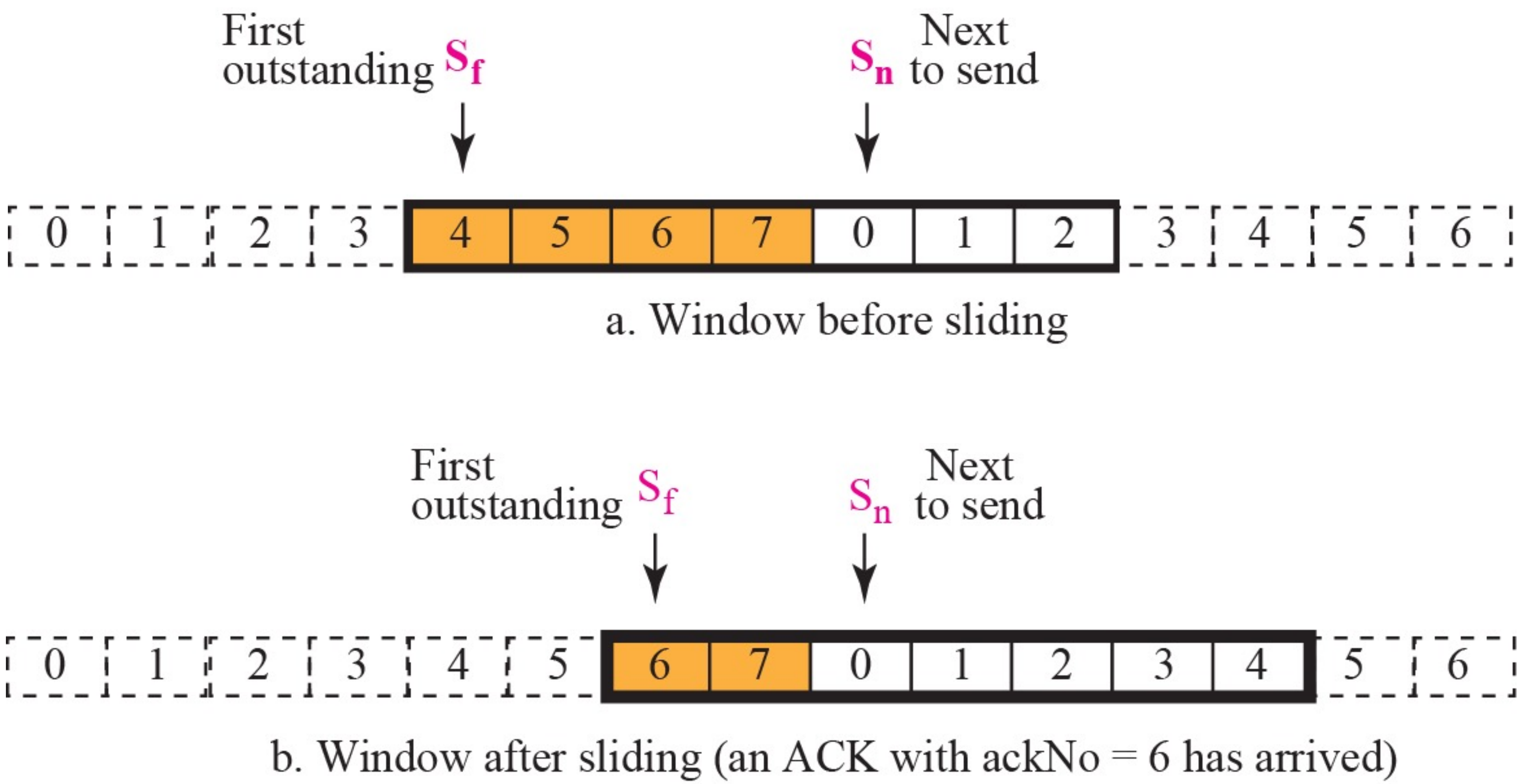


Figure 13.24 ***Sliding the send window***

Receiver Sliding Window for Go-Back-N

- In Go-back-N, the size of the receive window is always 1.
- The receiver is always looking for the arrival of a specific packet.
- Any packet arriving out of order is discarded and needs to be resent.
- Only a packet with a sequence number matching the value of R_n is accepted and acknowledged.
- The receive window also slides, but only one slot at a time. When a correct packet is received, the window slides, $R_n = (R_n + 1)$.

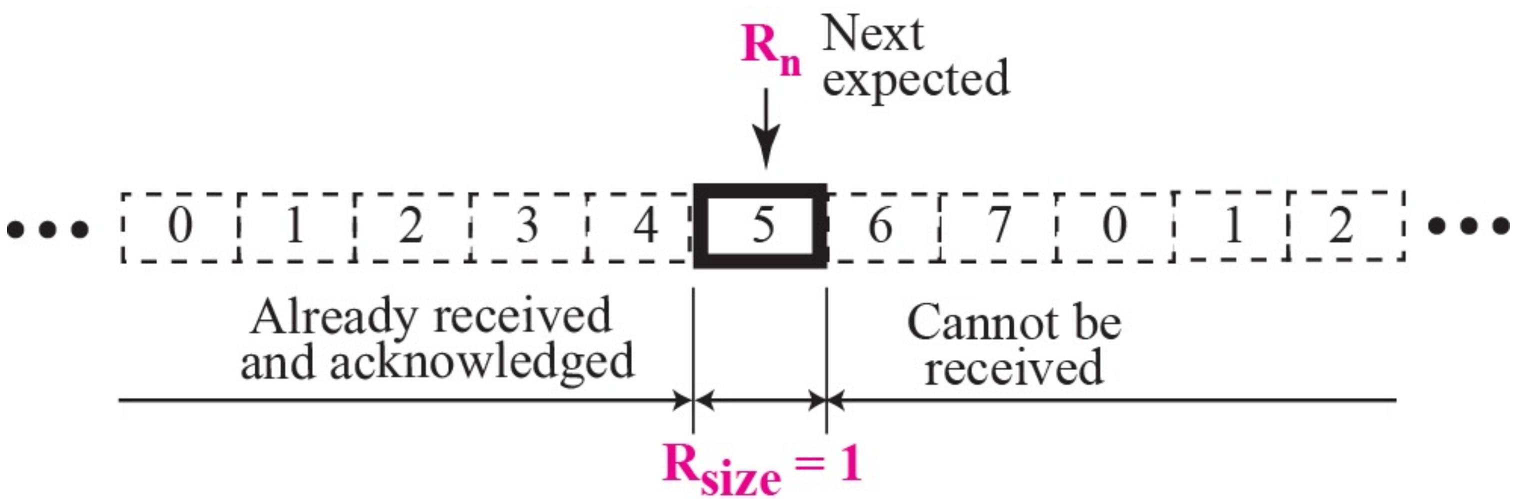
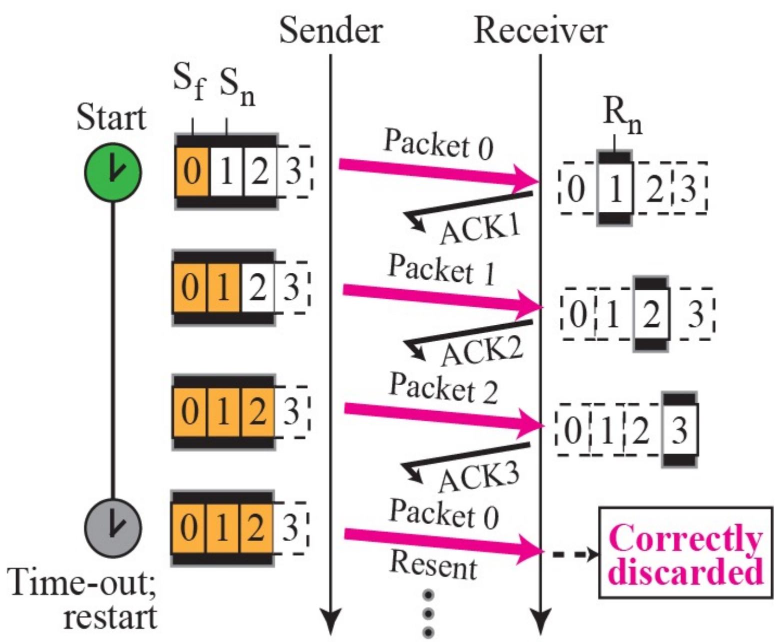


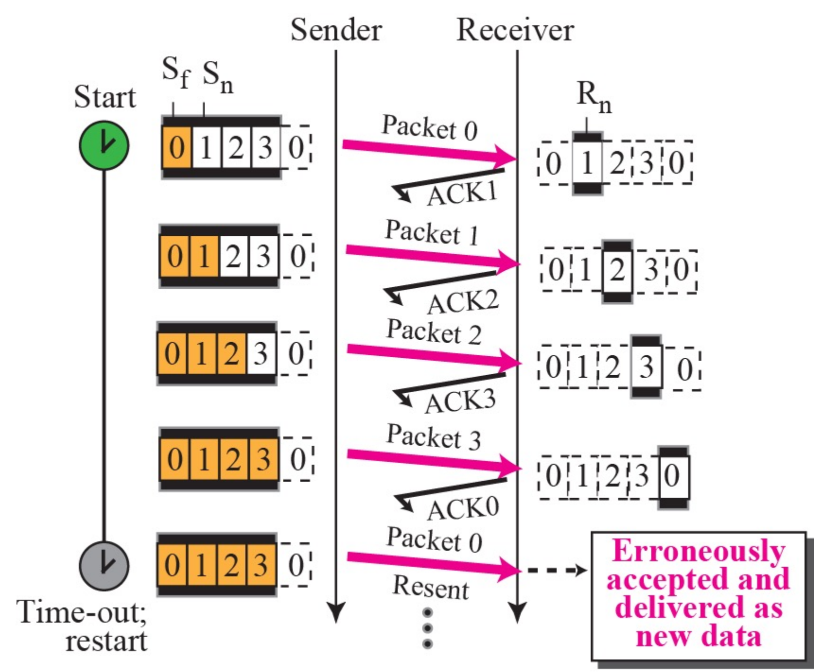
Figure 13.25 ***Receive window for Go-Back-N***

Why window size must be < 2^m in GBN?



a. Send window of size < 2^m

window size 3(2^m-1)



b. Send window of size = 2^m

window size 4(2^m)

Figure 13.27 ***Send window size for Go-Back-N***

Example 13.7

Figure 13.28 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 packets 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.

Cumulative Acknowledgment

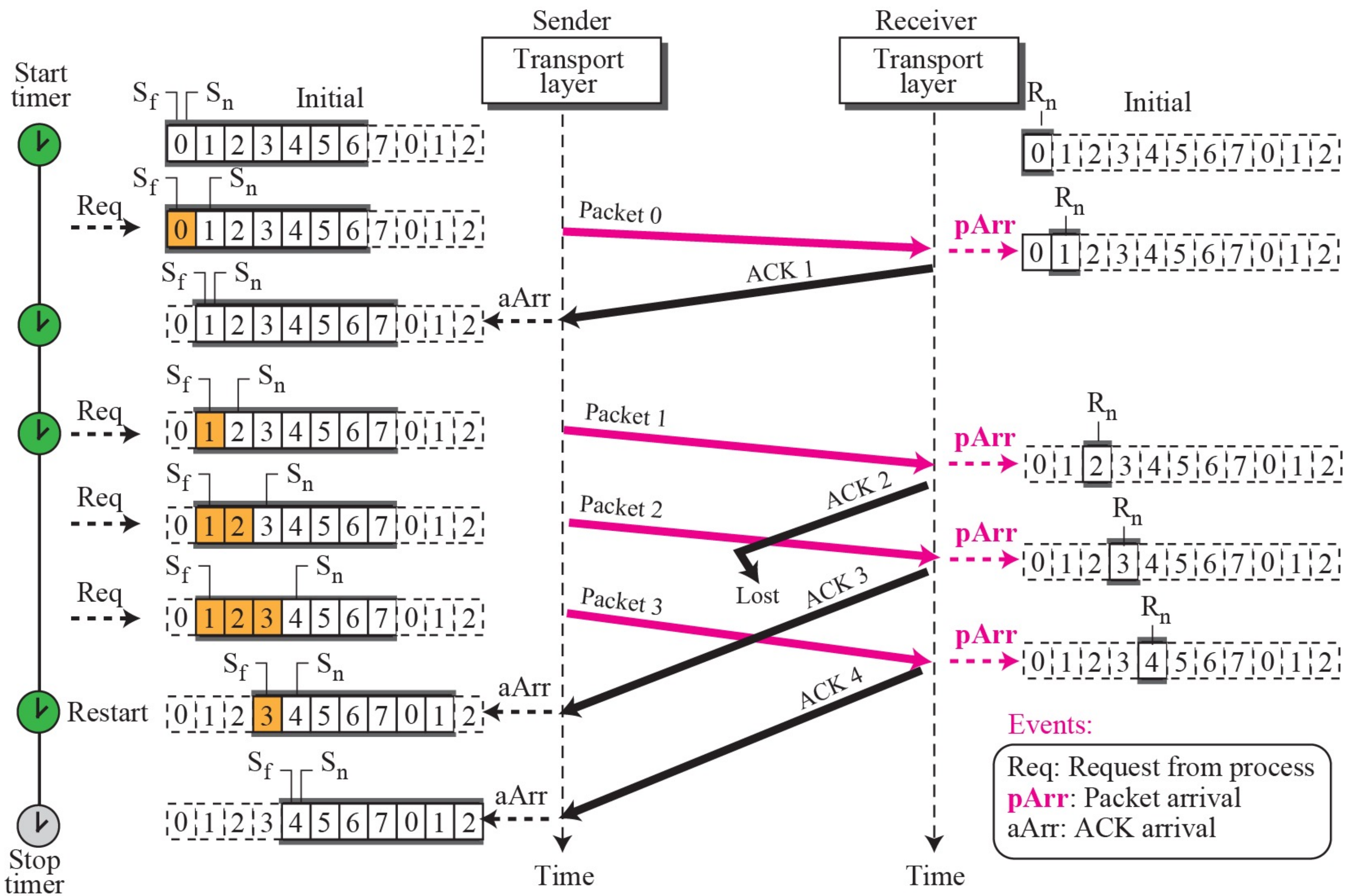


Figure 13.28 *Example 13.7*

Example 13.8

Figure 13.29 shows what happens when a packet is lost. Packets 0, 1, 2, and 3 are sent. However, packet 1 is lost. The receiver receives packets 2 and 3, but they are discarded because they are received out of order (packet 1 is expected). When the receiver receives packets 2 and 3, it sends ACK1 to show that it expects to receive packet 1. However, these ACKs are not useful for the sender because the ackNo is equal S_f , not greater that S_f . So the sender discards them. When the time-out occurs, the sender resends packets 1, 2, and 3, which are acknowledged..

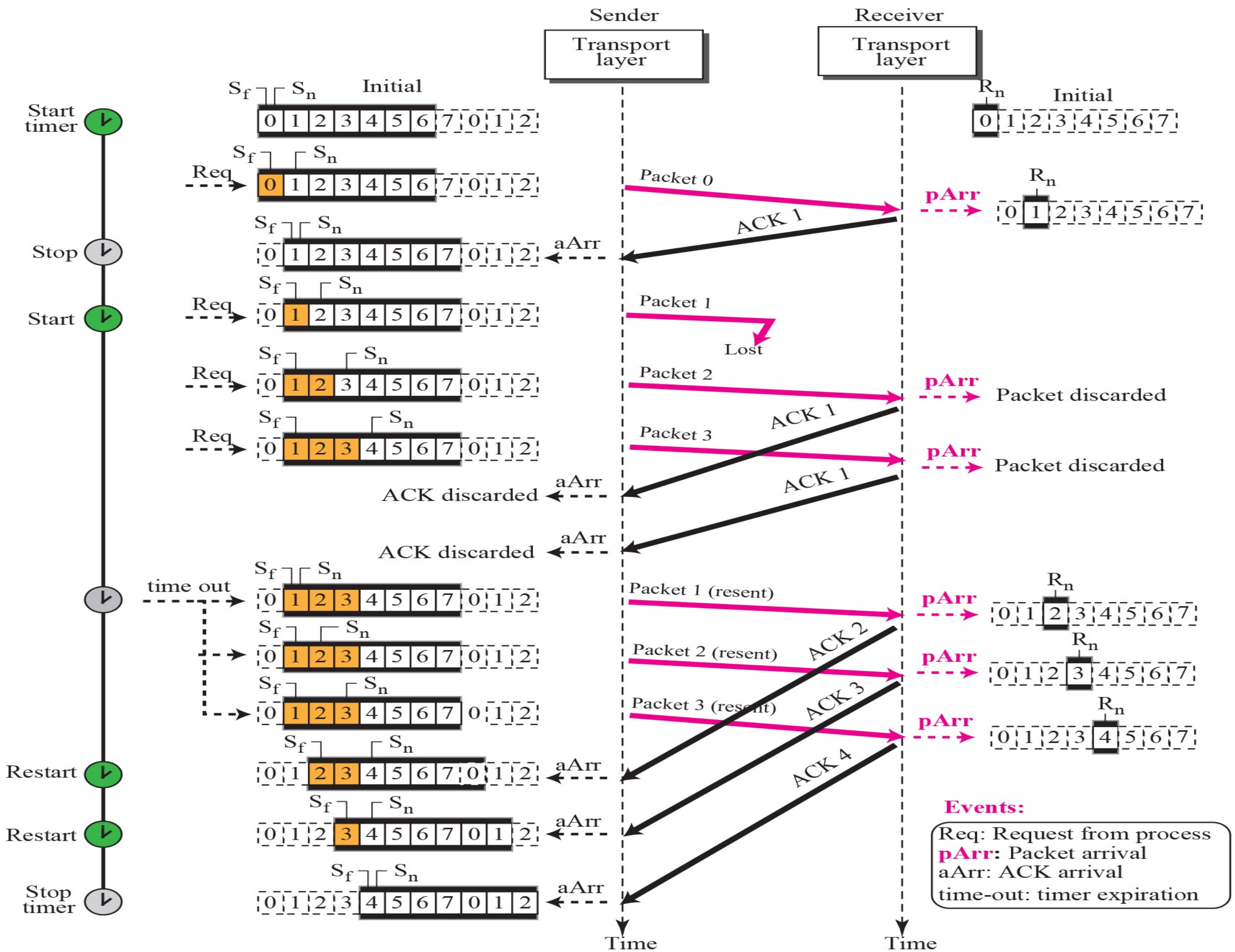


Figure 13.29 ***Example 13.8***

Selective-Repeat Protocol

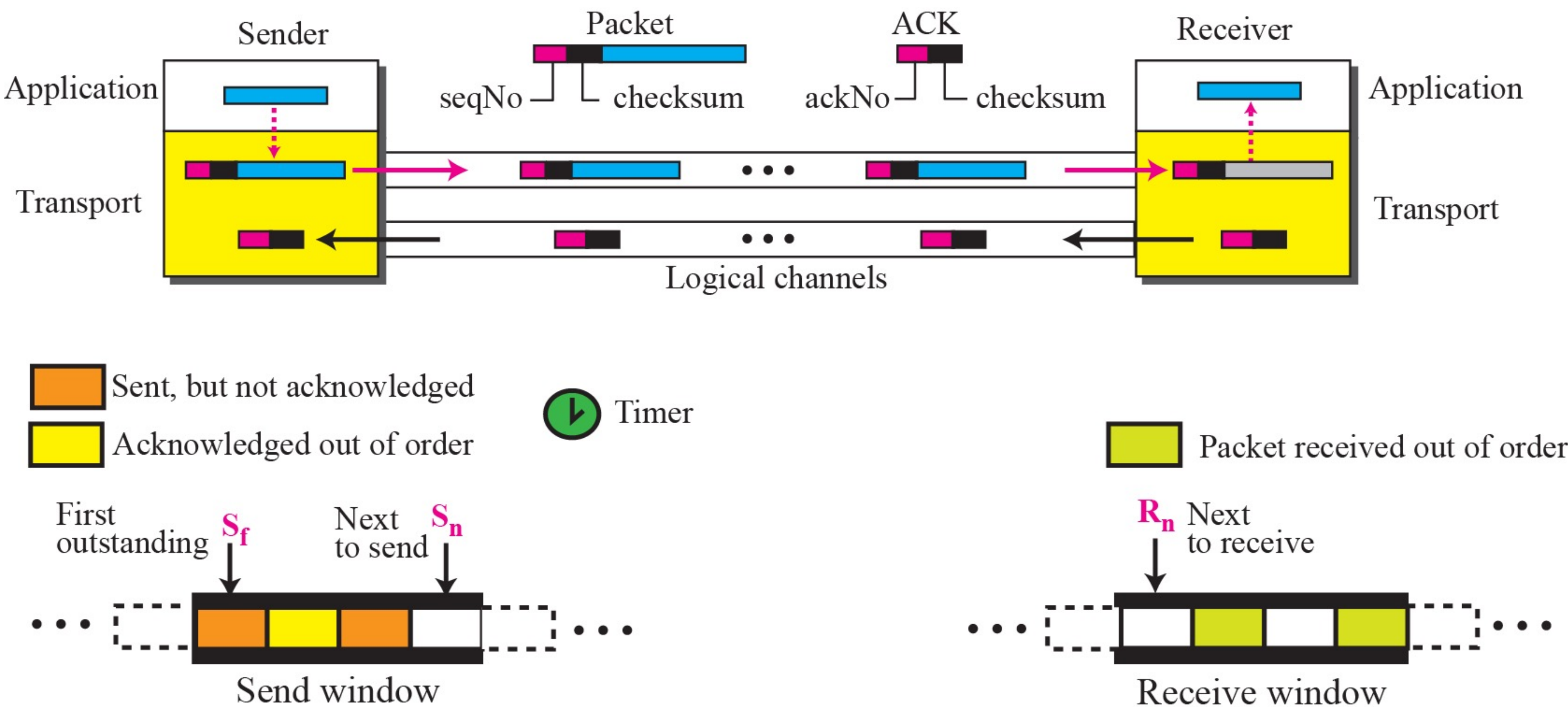


Figure 13.30 ***Selective-Repeat Protocol***

- In Go-back-*N*, if a single packet is lost or corrupted, the sender needs to resends all outstanding packets that significantly decrease the network performance.
- The Selective-Repeat resends only selective packets, those that are actually lost.
- The Selective-Repeat protocol also uses two windows: a send window and a receive window. The maximum size of the both sender and receiver window is 2^{m-1}.

Sender window for Selective-Repeat Protocol

- The send window maximum size can be 2^{m-1}. For example, if $m = 4$, the sequence numbers go from 0 to 15, but the maximum size of the window is just 8 (it is 15 in the Go-Back-N Protocol).

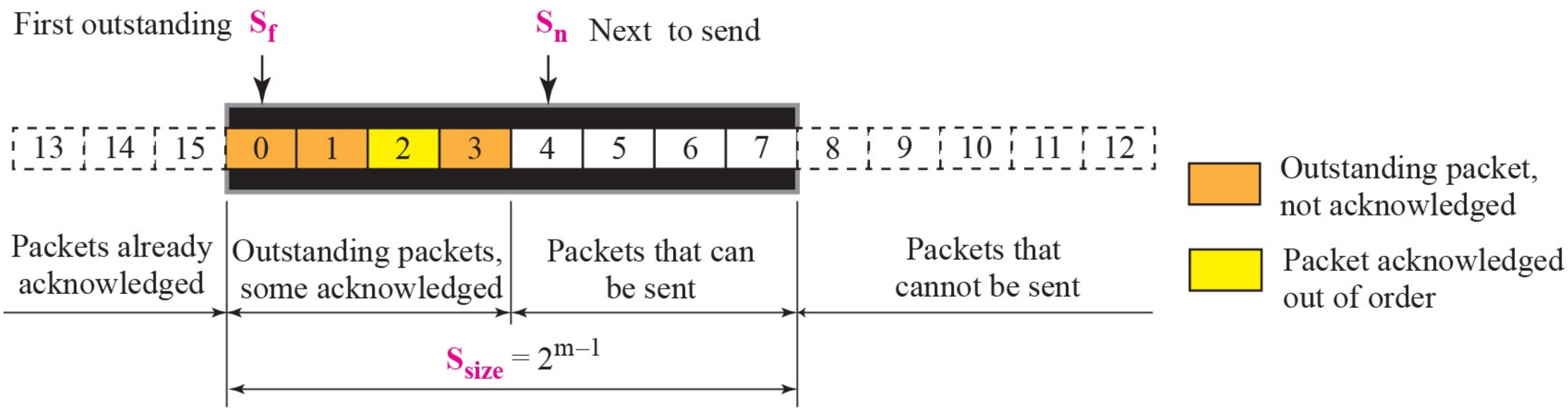


Figure 13.31 ***Send window for Selective-Repeat protocol***

Receiver window for Selective-Repeat Protocol

- The receive window in Selective-Repeat is totally different from the one in Go- Back-*N*.
- The receive window maximum size can be 2^{m-1}.
- The Selective-Repeat protocol allows as many packets as the size of the receive window.
- The shaded box presents packets that are received out of order.

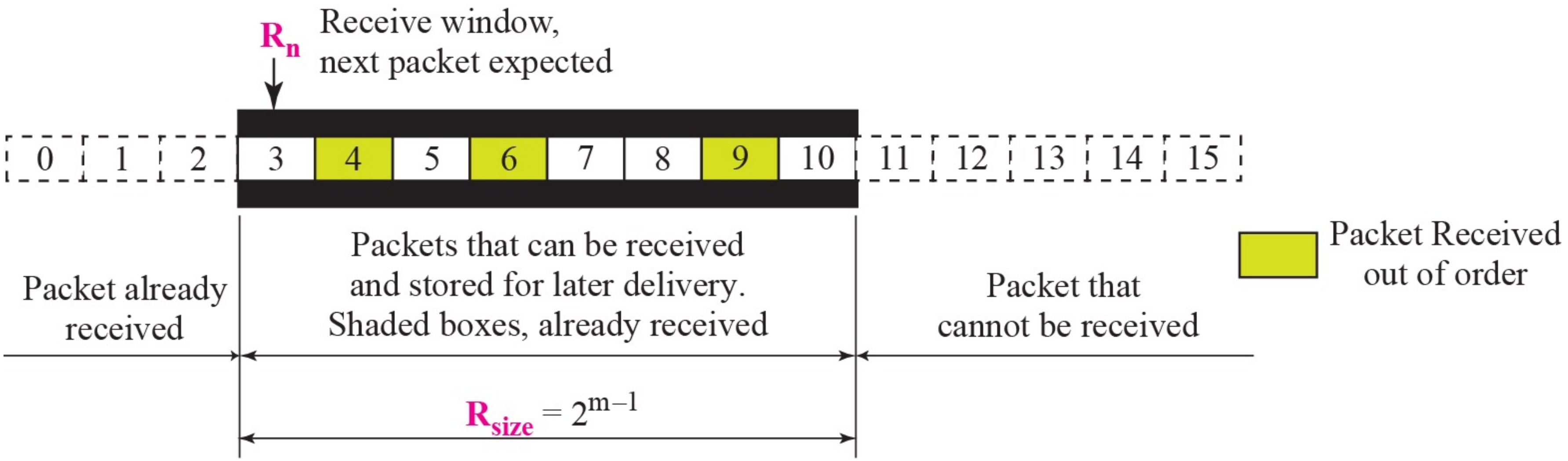


Figure 13.32 ***Receive window for Selective-Repeat protocol***

Note

Selective-Repeat uses one timer for each outstanding packet. When a timer expires, only the corresponding packet is resent. In other words, GBN treats outstanding packets as a group.

Example 13.9

Assume a sender sends 6 packets: packets 0, 1, 2, 3, 4, and 5. The sender receives an ACK with ackNo = 3. What is the interpretation if the system is using GBN or SR?

Solution

If the system is using GBN, it means that packets 0, 1, and 2 have been received uncorrupted and the receiver is expecting packet 3. If the system is using SR, it means that packet 3 has been received uncorrupted; the ACK does not say anything about other packets.

Example 13.10

This example is similar to Example 3.8 (Figure 13.29) in which packet 1 is lost. We show how Selective-Repeat behaves in this case. Figure 13.34 shows the situation. At the sender, packet 0 is transmitted and acknowledged. Packet 1 is lost. Packets 2 and 3 arrive out of order and are acknowledged. When the timer times out, packet 1 (the only unacknowledged packet) is resent and is acknowledged. The send window then slides.

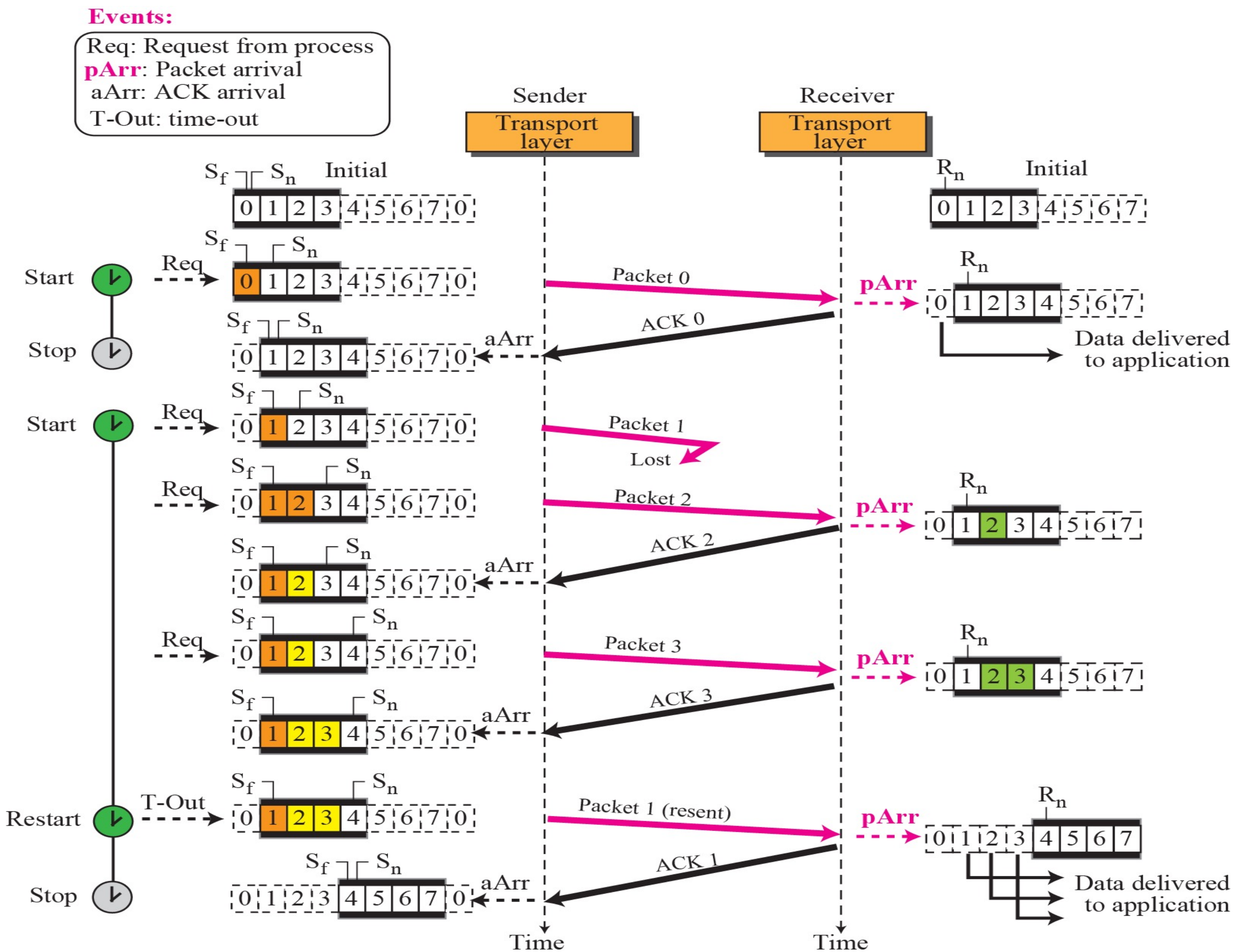


Figure 13.34 *Example 13.10*

Why window size must be < 2^{m-1} in Selective-Repeat?

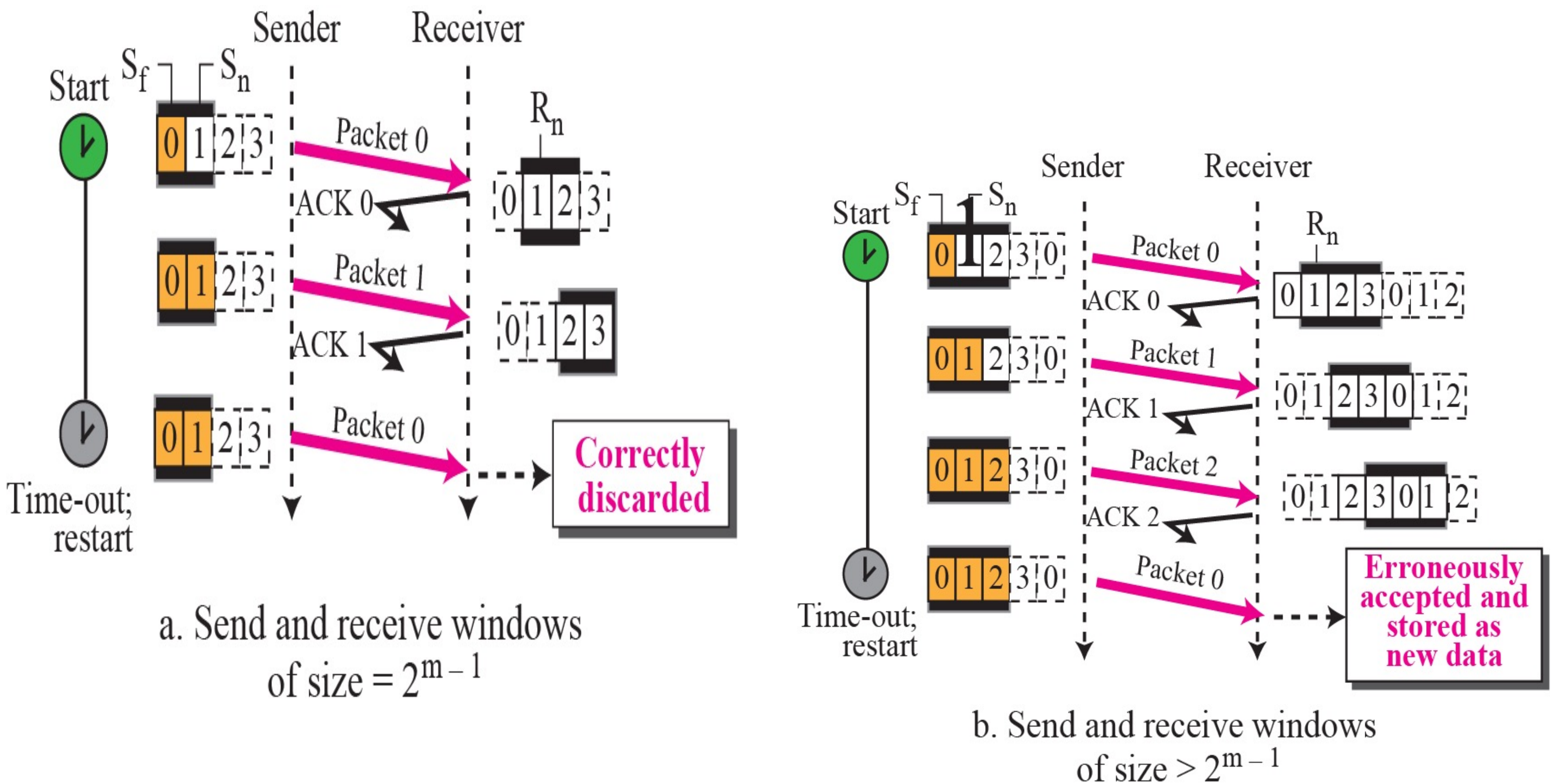


Figure 13.35 ***Selective-Repeat window size***

Note

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