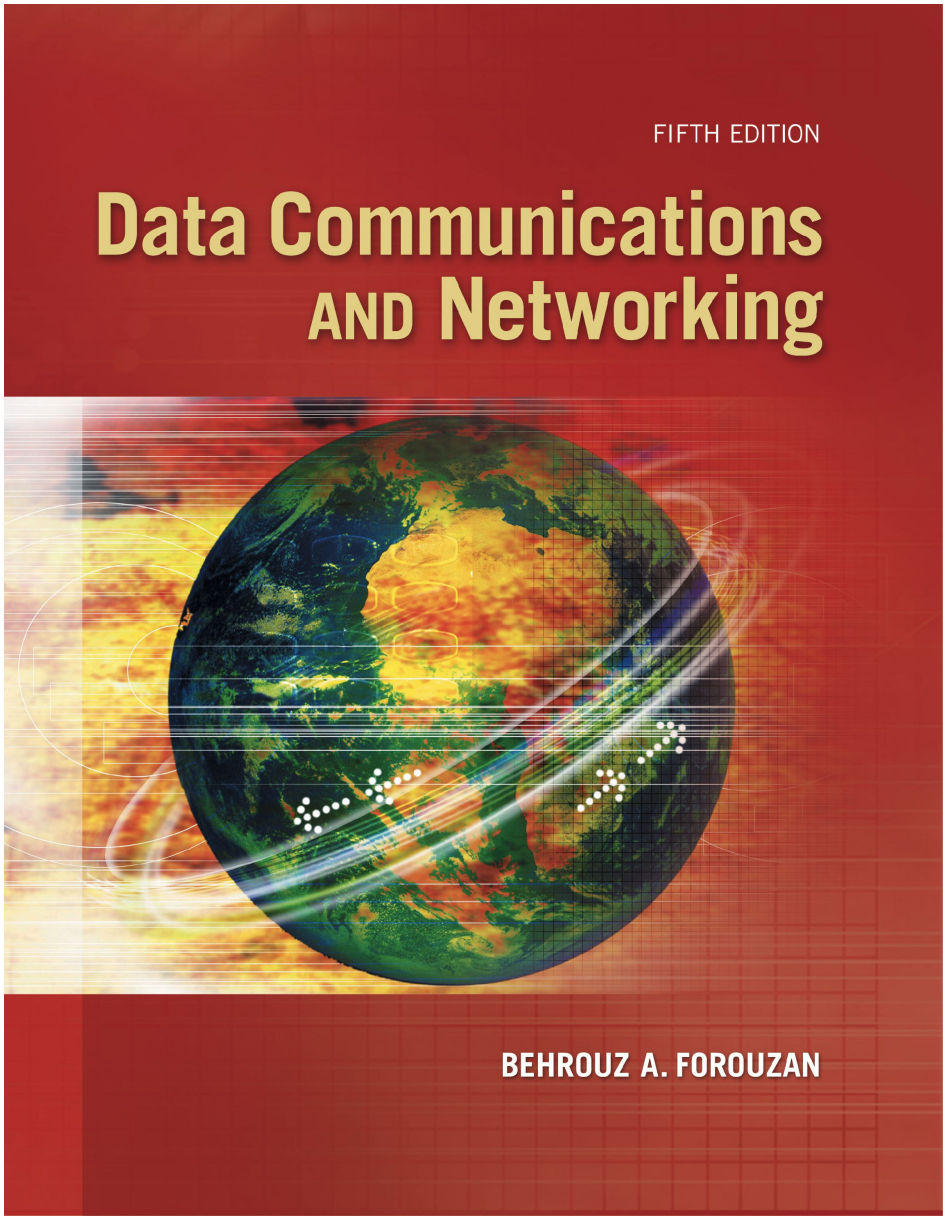
# Chapter 23

# Introduction To Transport Layer



# Chapter 23: Outline

## *23.1 INTRODUCTION*

## *23.2 TRANSPORT-LAYER PROTOCOLS*

# Chapter 23: Objective

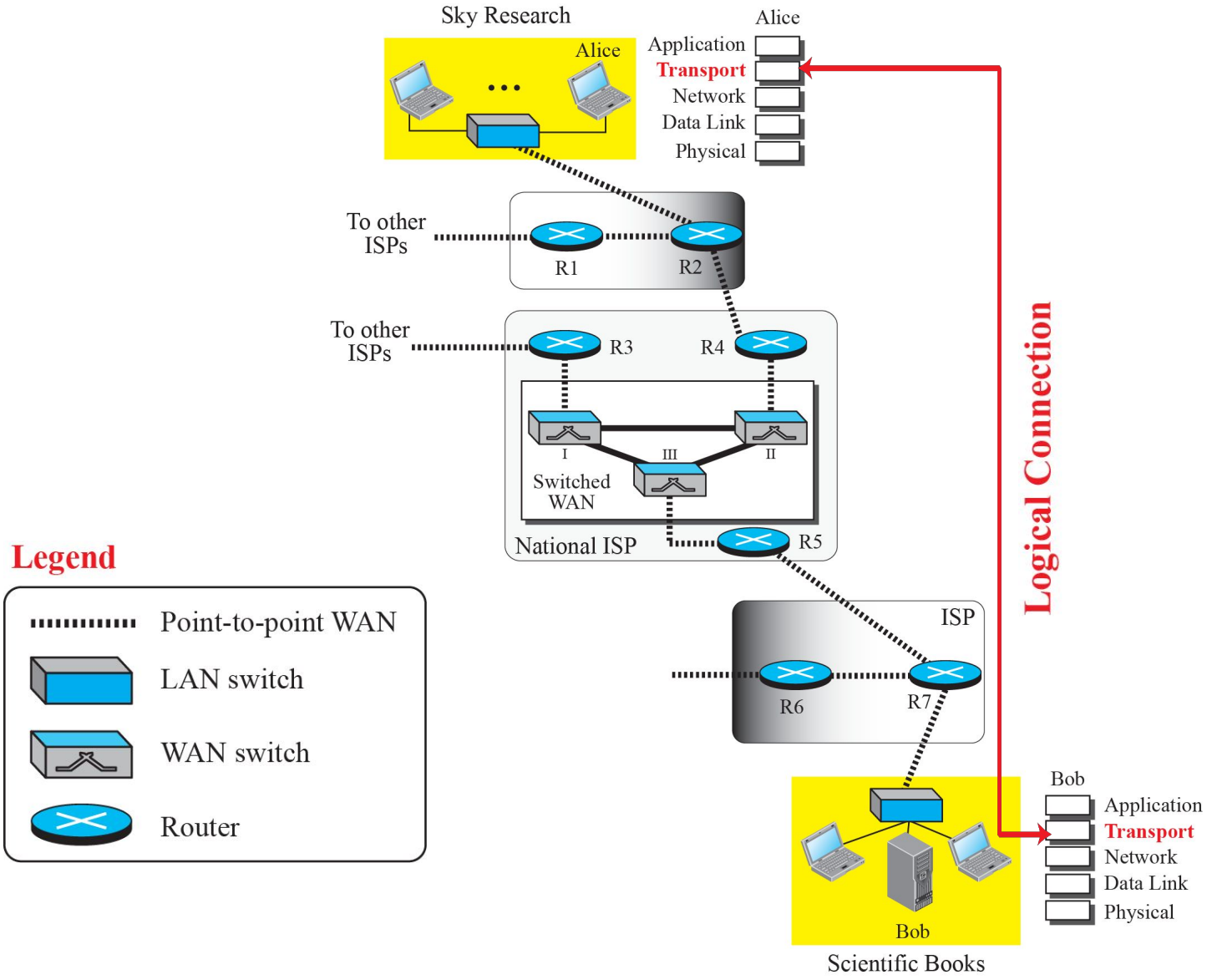
- *The first section introduces the idea behind a transport-layer protocol. We first discuss the general services we normally require from the transport layer, such as process-to-process communication, addressing, multiplexing and de-multiplexing, error, flow, and congestion control. We then show that the transport-layer protocols are divided into two categories: connectionless and connection-oriented.*

- *The second section discusses general transport-layer protocols. These protocols concentrate on flow and error control services provided by an actual transport layer protocol. Understanding these protocols helps us better understand the design of the transport-layer protocols in the Internet, such as UDP, TCP, and SCTP.*

# 23-1 INTRODUCTION

*The transport layer is located between the application layer and the network layer. It provides a process-to-process communication between two application layers, one at the local host and the other at the remote host. Communication is provided using a logical connection. Figure 23.1 shows the idea behind this logical connection.*

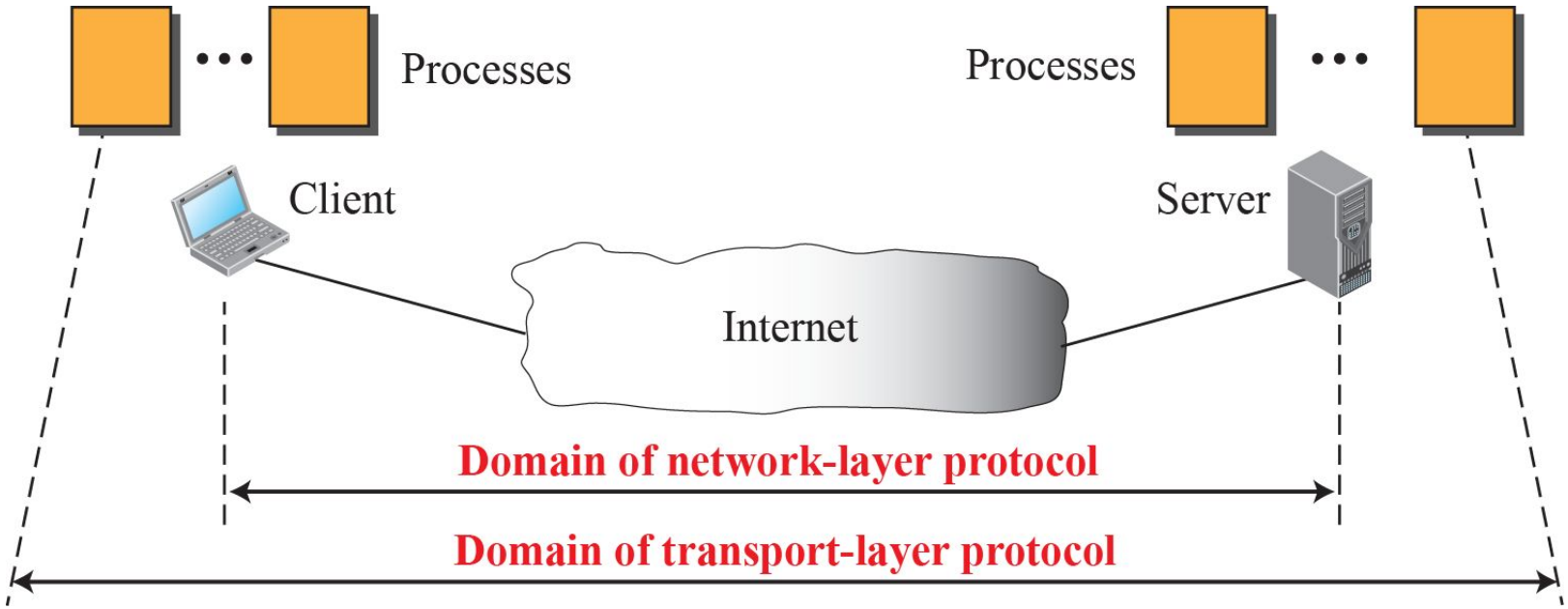
# *Figure 23.1: Logical connection at the transport layer*



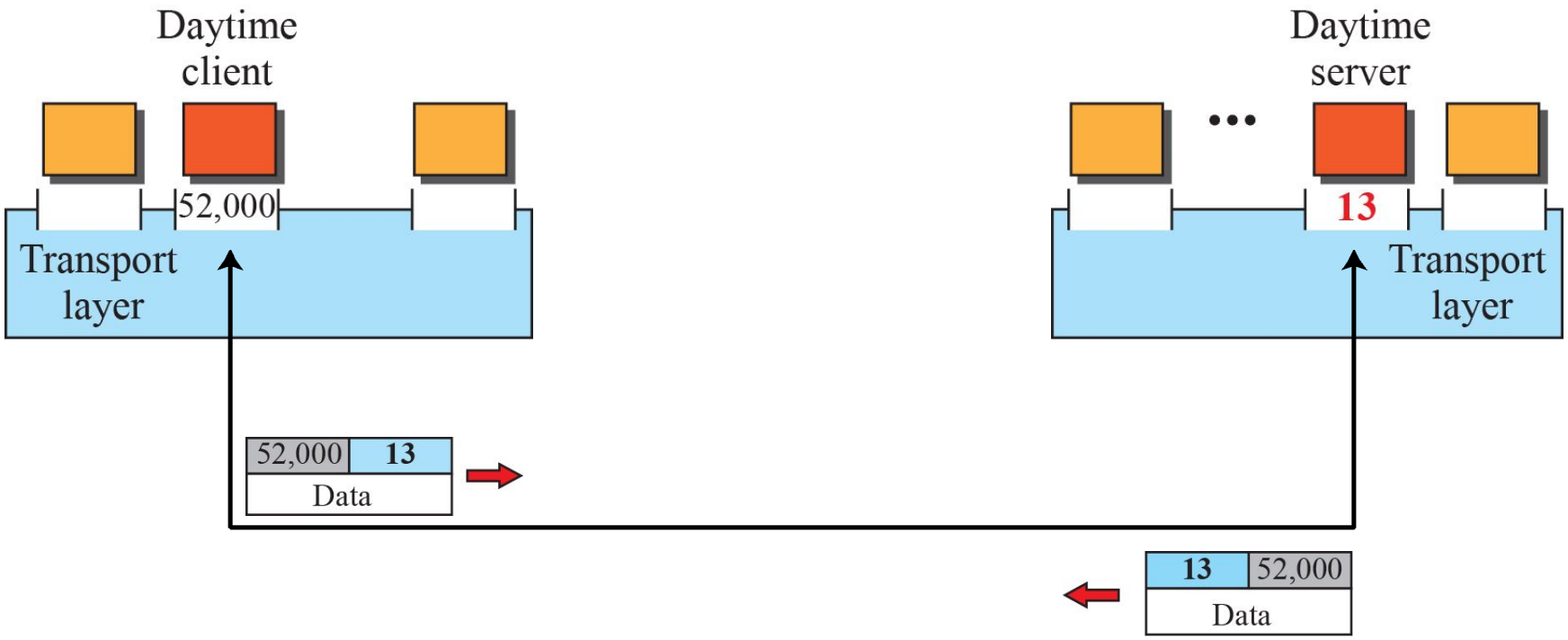
# 23.23.1 Transport-Layer Services

*As we discussed in Chapter 2, the transport layer is located between the network layer and the application layer. The transport layer is responsible for providing services to the application layer; it receives services from the network layer. In this section, we discuss the services that can be provided by the transport layer; in the next section, we discuss several transport-layer protocols.*

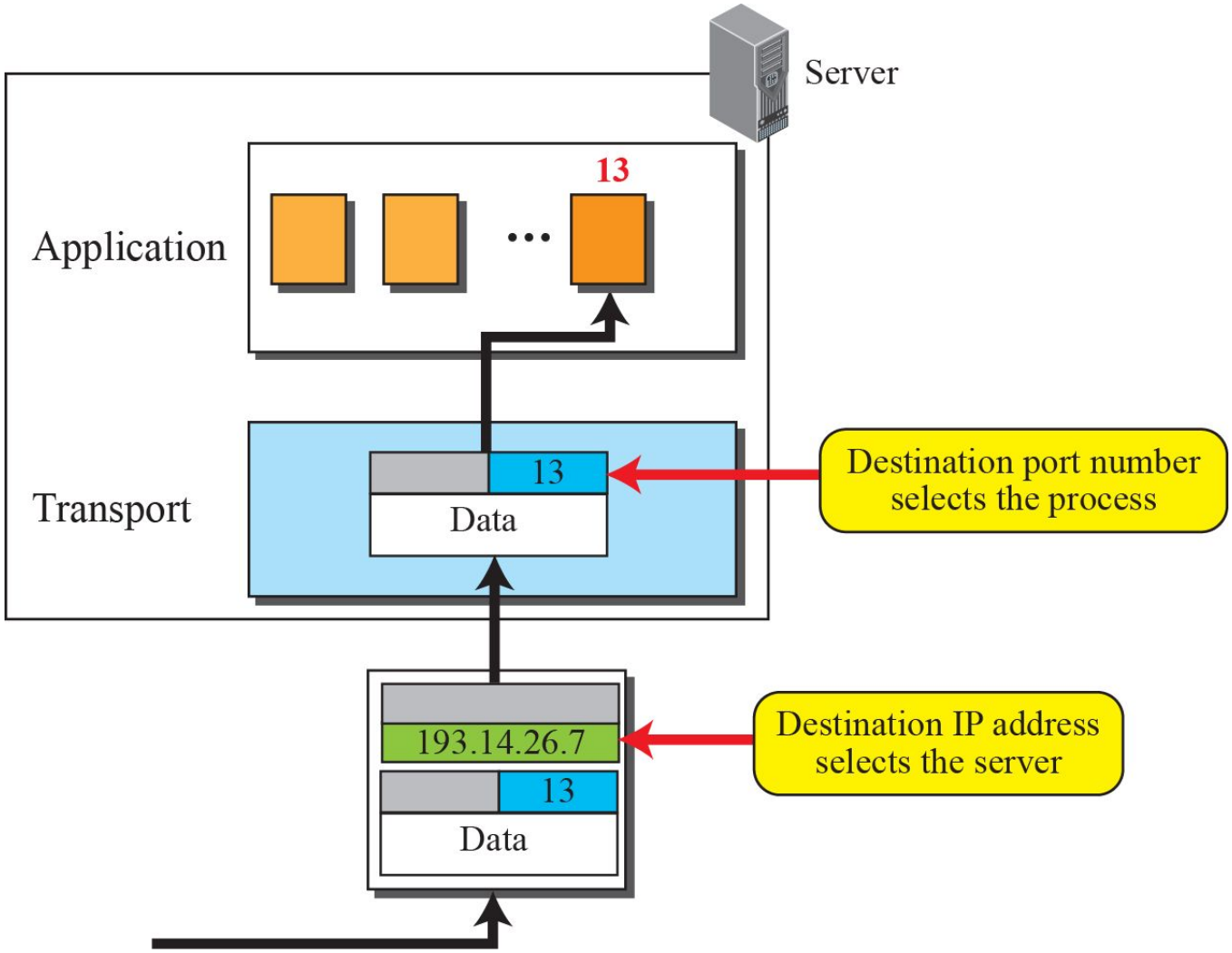
*Figure 23.2: Network layer versus transport layer*



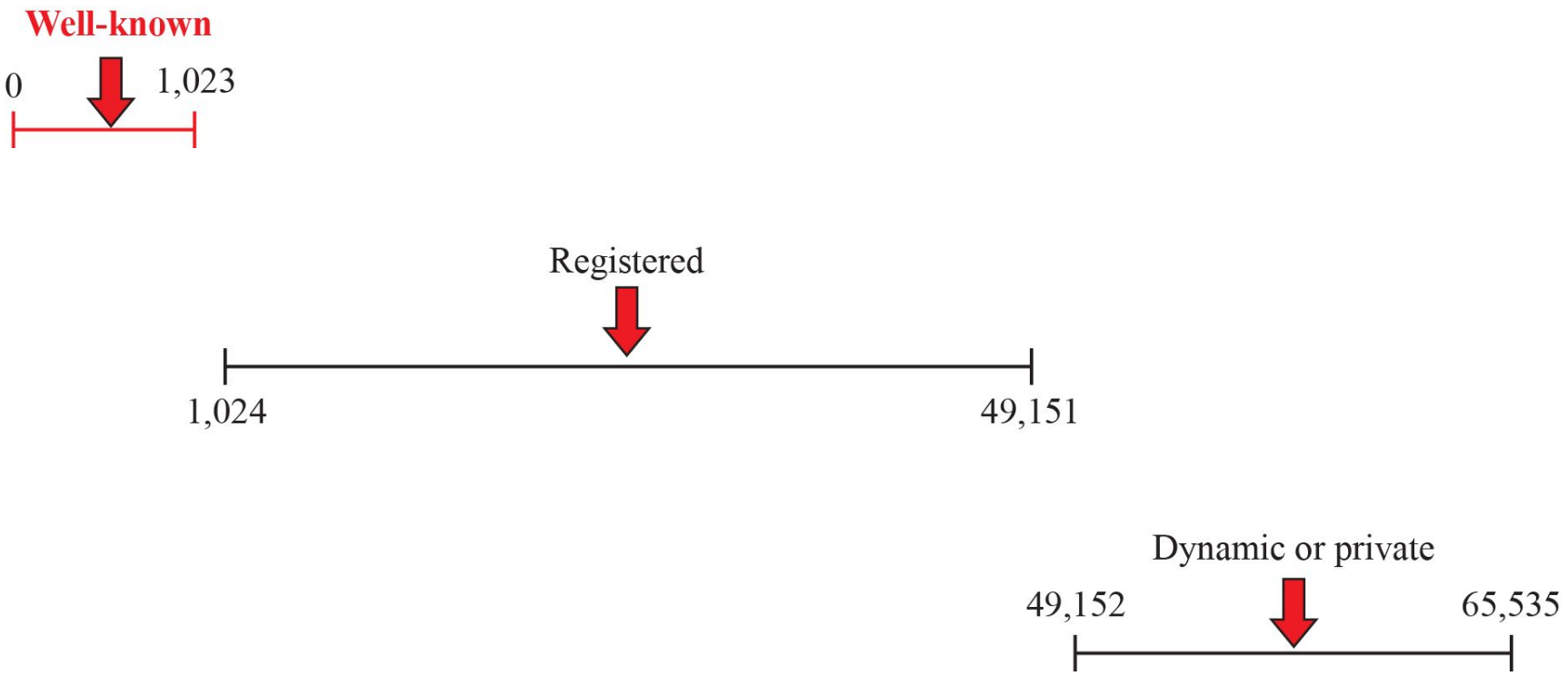
## *Figure 23.3: Port numbers*



## *Figure 23.4: IP addresses versus port numbers*



## *Figure 23.5: ICANN ranges*



# Example 23.1

In UNIX, the well-known ports are stored in a file called /etc/services. We can use the *grep* utility to extract the line corresponding to the desired application.

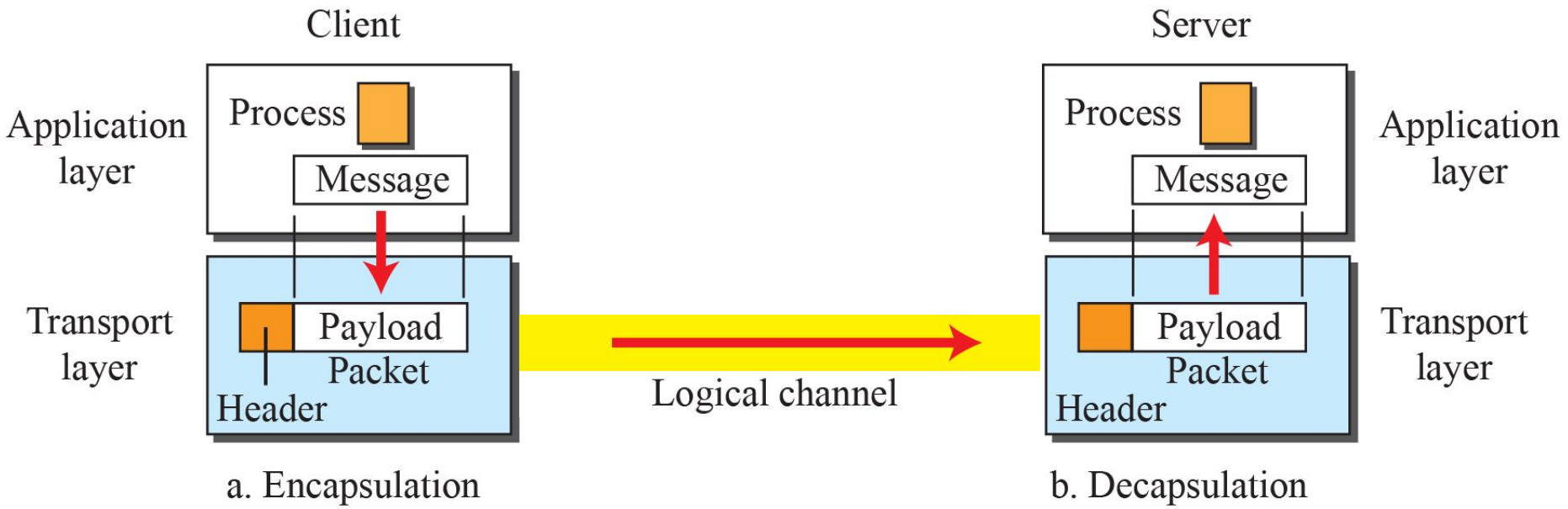
```
$grep     tftp/etc/services
tftp 69/tcp
tftp 69/udp
```

SNMP (see Chapter 27) uses two port numbers (161 and 162), each for a different purpose.

```
$grep     snmp/etc/services
snmp161/tcp#Simple Net Mgmt Proto
snmp161/udp#Simple Net Mgmt Proto
snmptrap162/udp#Traps for SNMP
```

*Figure 23.6:* ***Socket address***

## *Figure 23.7: Encapsulation and decapsulation*



## *Figure 23.8: Multiplexing and demultiplexing*

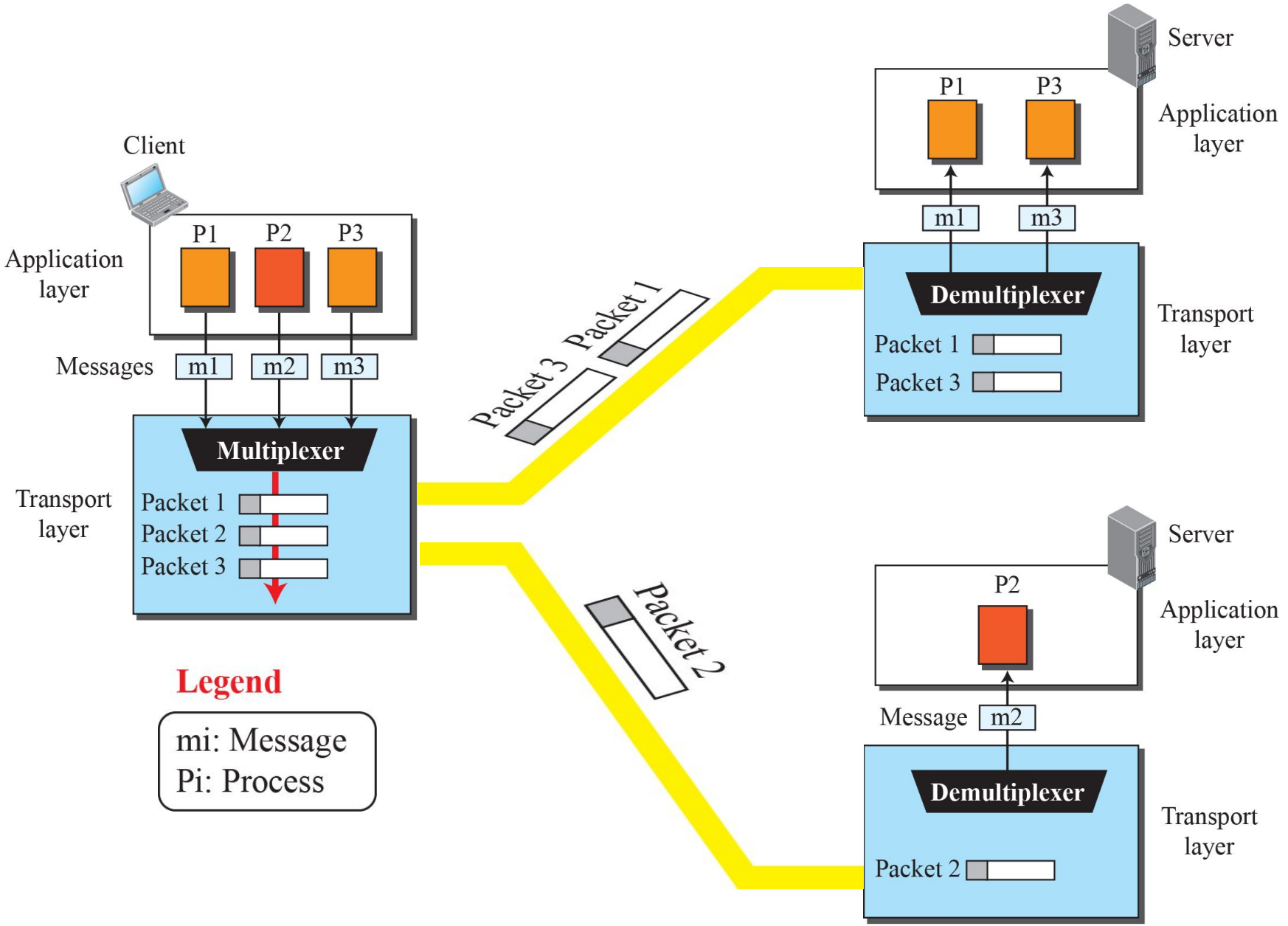
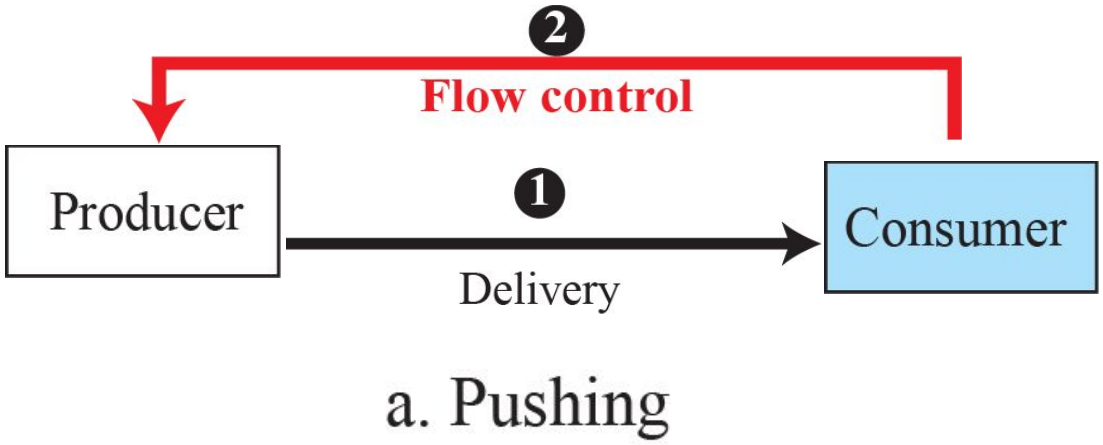
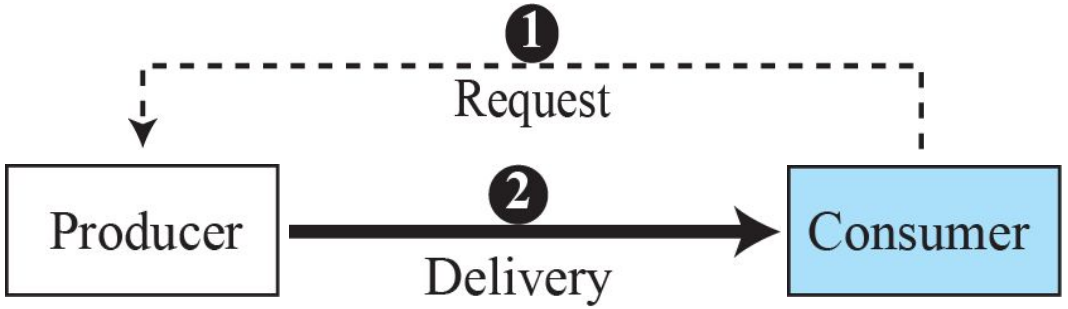


*Figure 23.9:* ***Pushing or pulling***

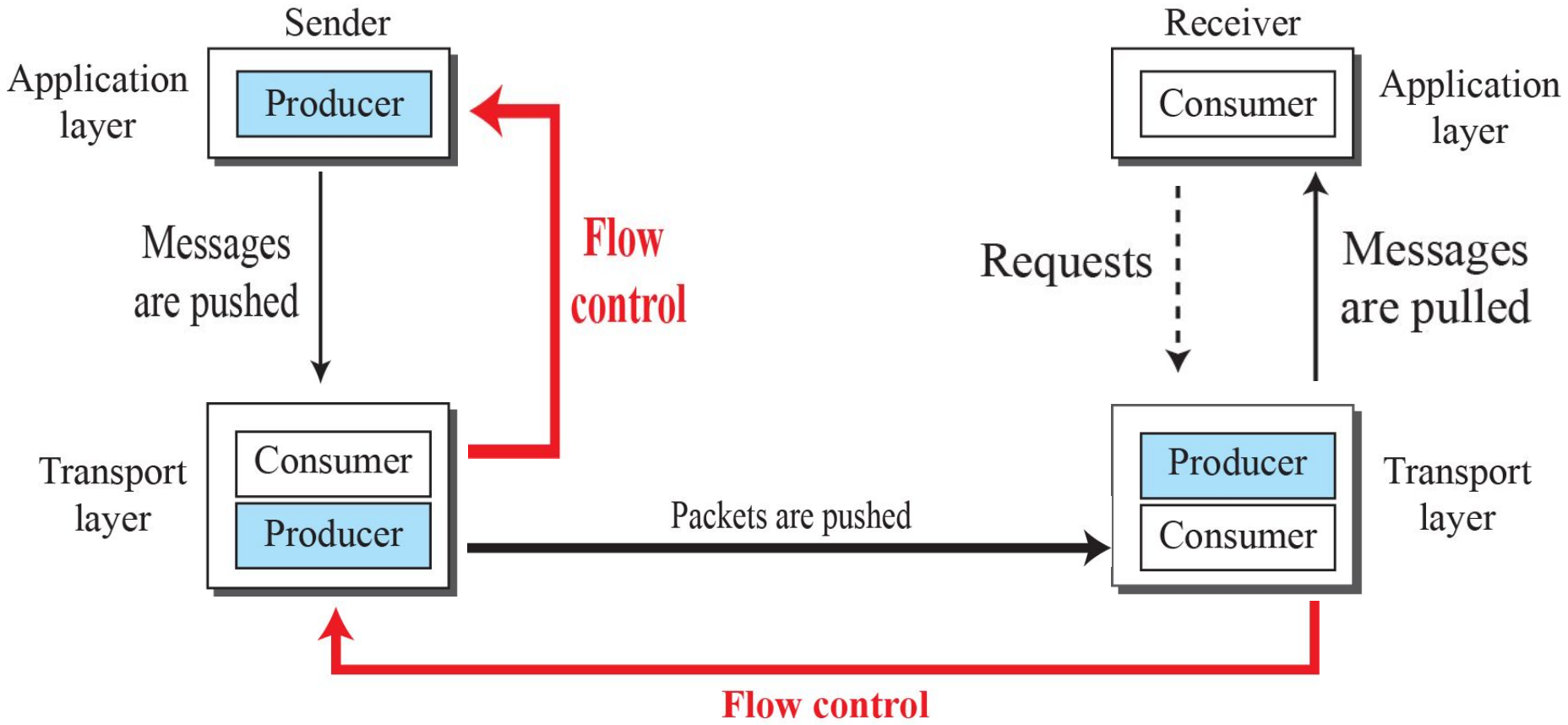



a. Pushing




b. Pulling

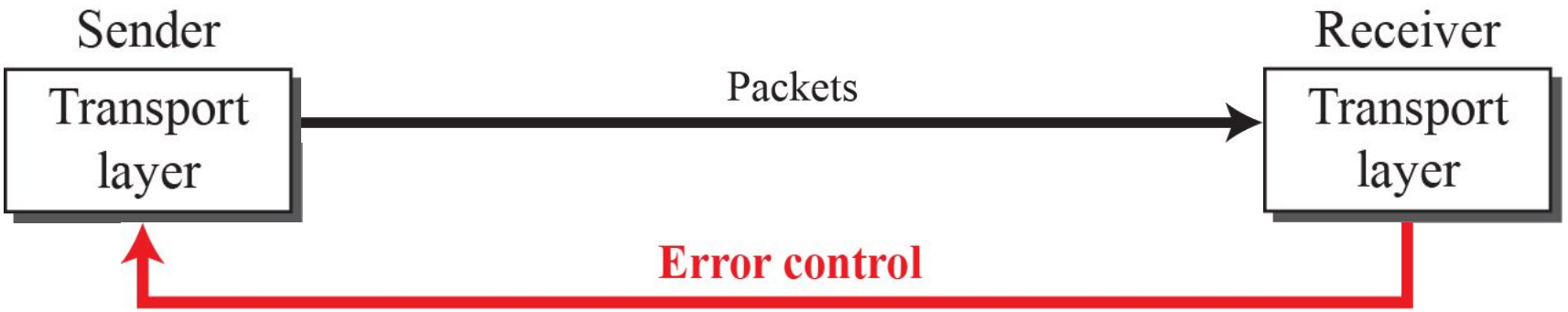
*Figure 23.10:* ***Flow control at the transport layer***



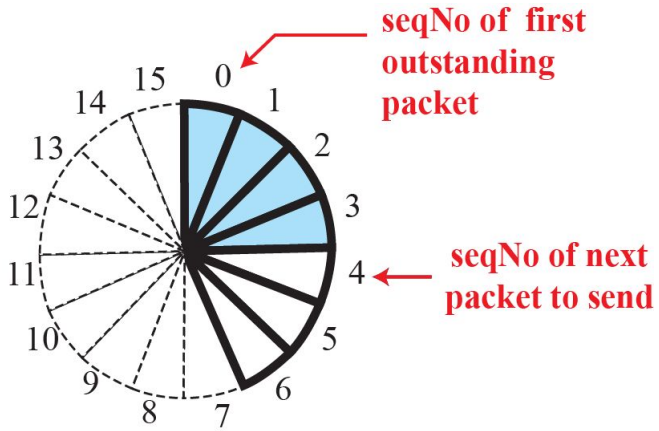
# Example 23.2

The above discussion requires that the consumers communicate with the producers on two occasions: when the buffer is full and when there are vacancies. If the two parties use a buffer with only one slot, the communication can be easier. Assume that each transport layer uses one single memory location to hold a packet. When this single slot in the sending transport layer is empty, the sending transport layer sends a note to the application layer to send its next chunk; when this single slot in the receiving transport layer is empty, it sends an acknowledgment to the sending transport layer to send its next packet. As we will see later, however, this type of flow control, using a single-slot buffer at the sender and the receiver, is inefficient.

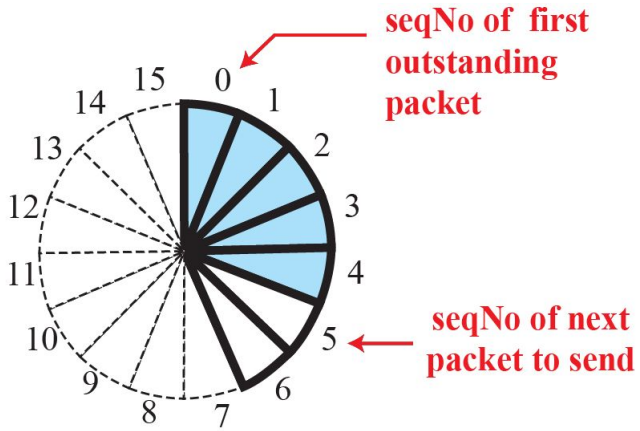
*Figure 23.11:* ***Error control at the transport layer***



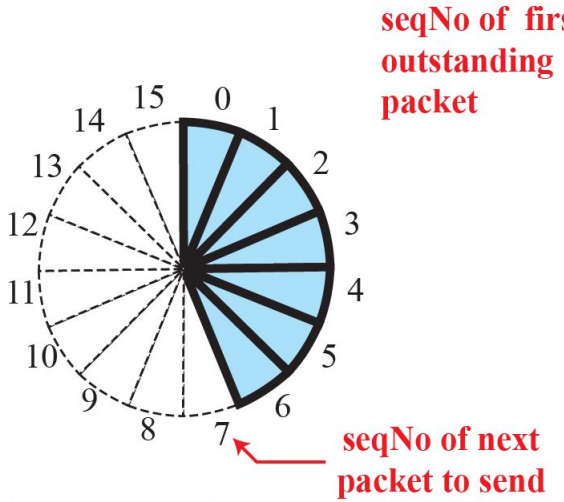
## *Figure 23.12: Sliding window in circular format*



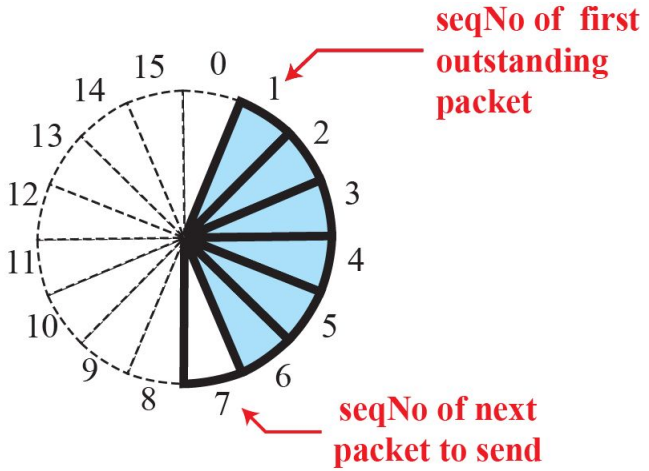

a. Four packets have been sent.




b. Five packets have been sent.

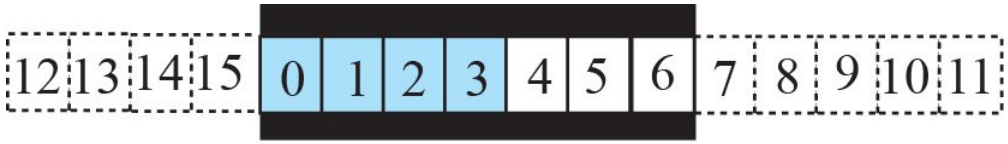



c. Seven packets have been sent; window is full.

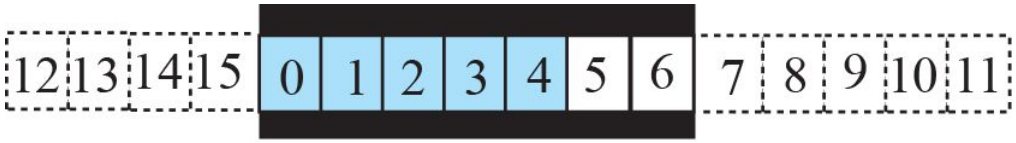



d. Packet 0 has been acknowledged; window slides.

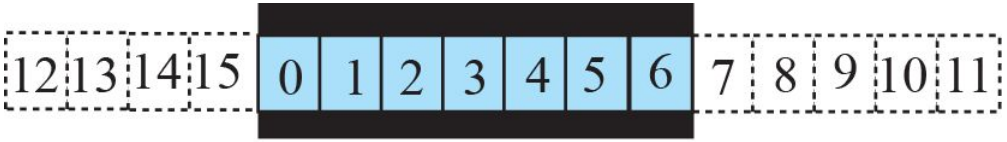
## *Figure 23.13: Sliding window in linear format*



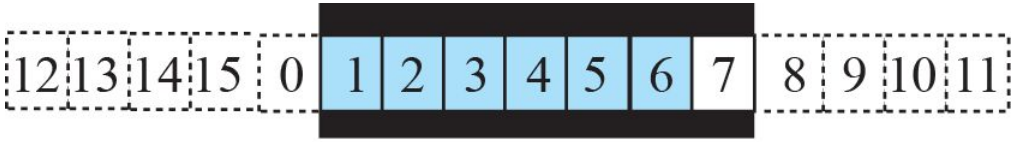

a. Four packets have been sent.




b. Five packets have been sent.




c. Seven packets have been sent;
window is full.

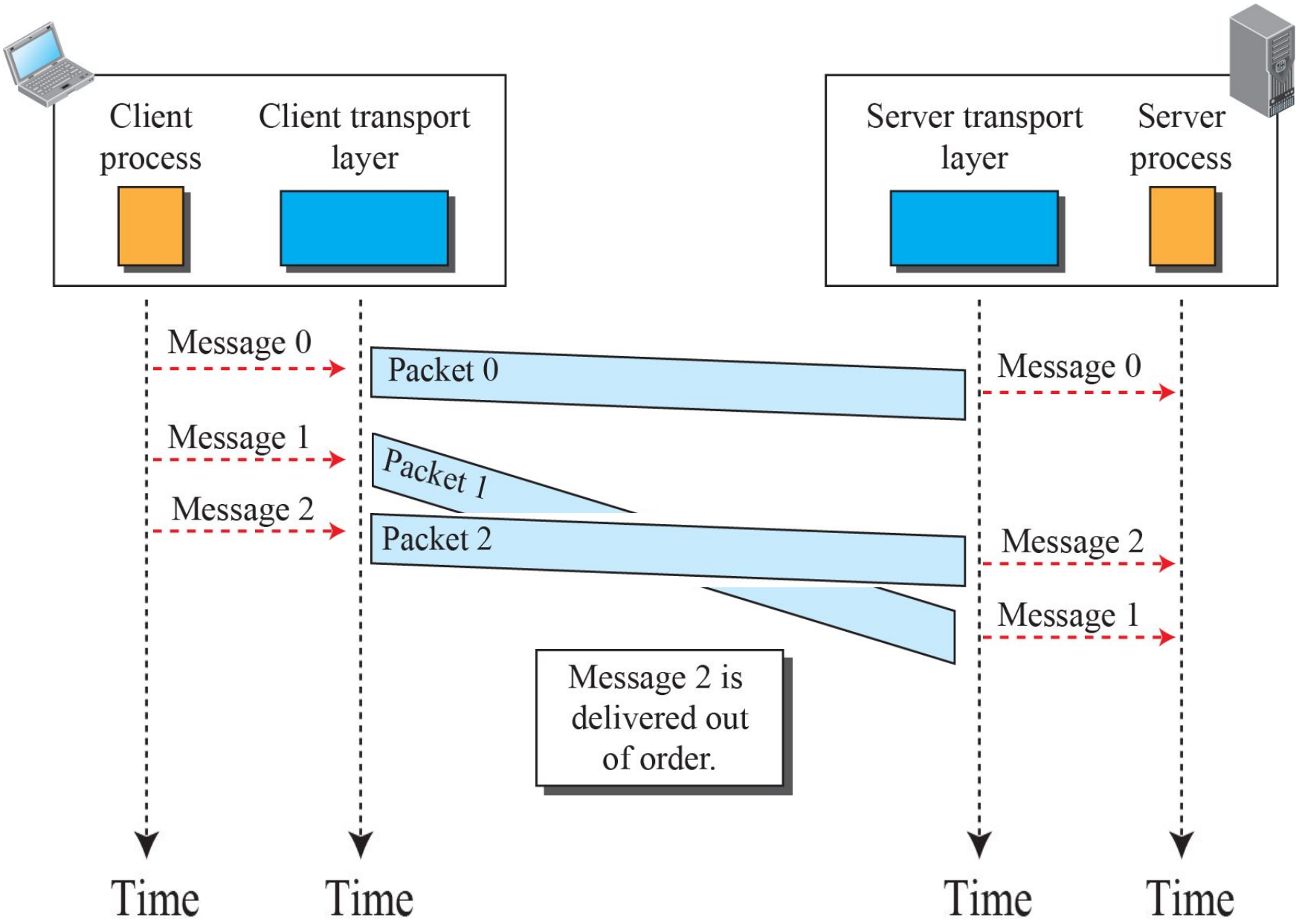



d. Packet 0 has been acknowledged;
window slides.

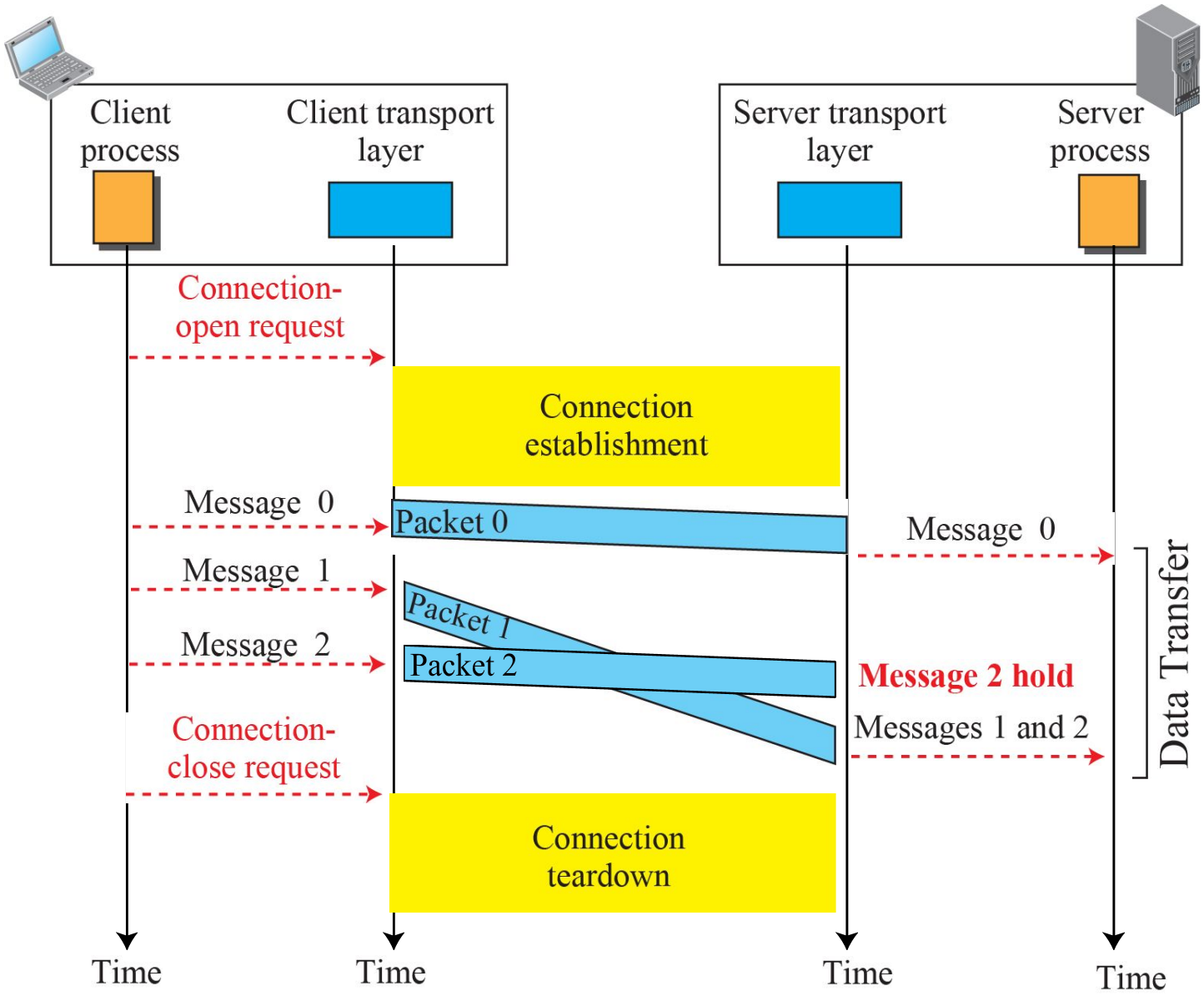
# 23.23.2 Connection

*A transport-layer protocol, like a network-layer protocol, can provide two types of services: connectionless and connection-oriented. The nature of these services at the transport layer, however, is different from the ones at the network layer. At the network layer, a connectionless service may mean different paths for different datagrams belonging to the same message. Connectionless service at the transport layer means independency between packets; connection-oriented means dependency. Let us elaborate on these two services.*

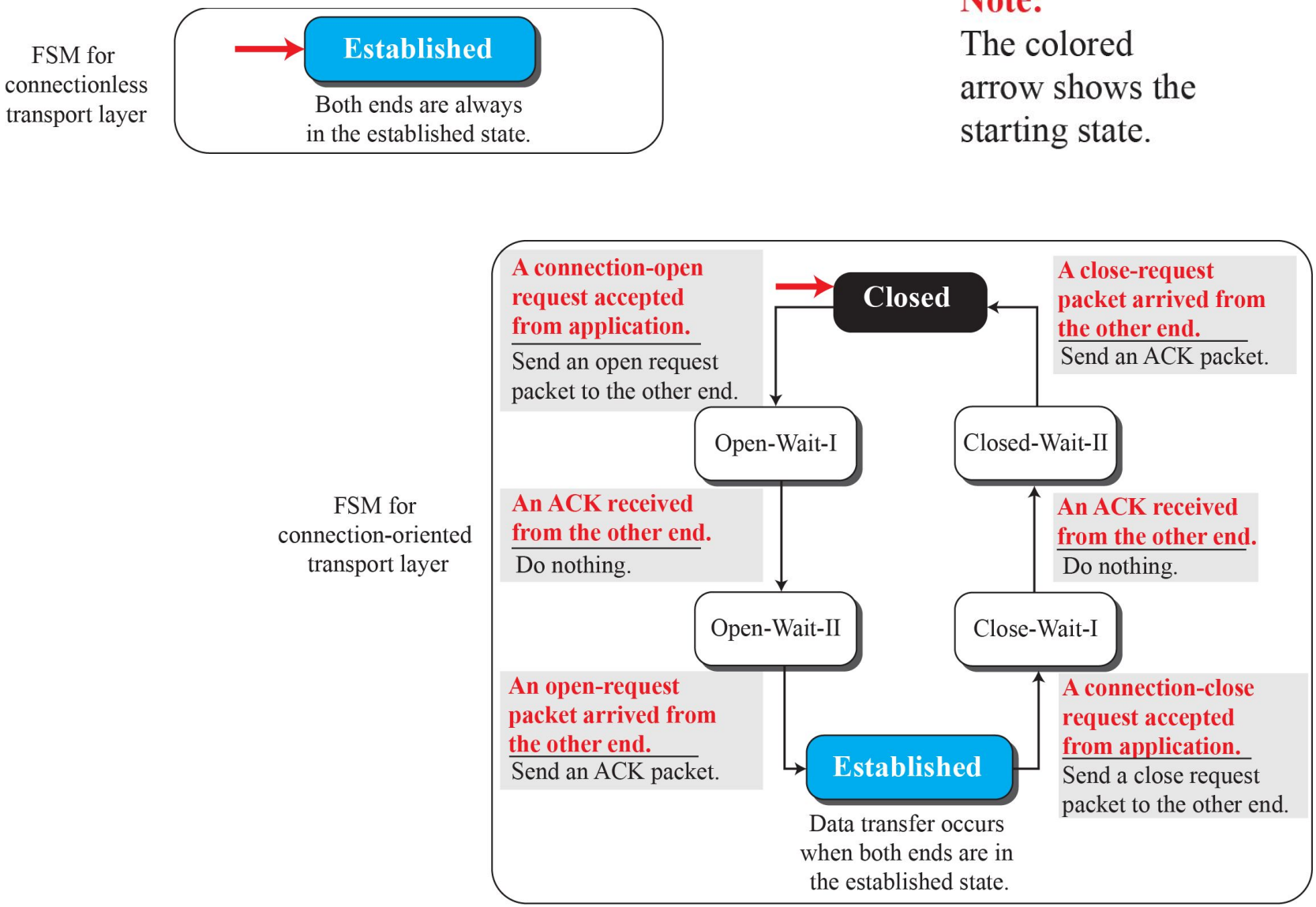
*Figure 3.14:* ***Connectionless service***



## *Figure 23.15: Connection-oriented service*



## *Figure 23.16: Connectionless and connection-oriented service represented as FSMs*



# 23-2 TRANSPORT-LAYER PROTOCOLS

*We can create a transport-layer protocol by combining a set of services described in the previous sections. To better understand the behavior of these protocols, we start with the simplest one and gradually add more complexity. The TCP/IP protocol uses a transport-layer protocol that is either a modification or a combination of some of these protocols.*

# 23.2.1 Simple Protocol

*Our first protocol is a simple connectionless protocol with neither flow nor error control. We assume that the receiver can immediately handle any packet it receives. In other words, the receiver can never be overwhelmed with incoming packets. Figure 23.17 shows the layout for this protocol.*

## *Figure 23.17: Simple protocol*

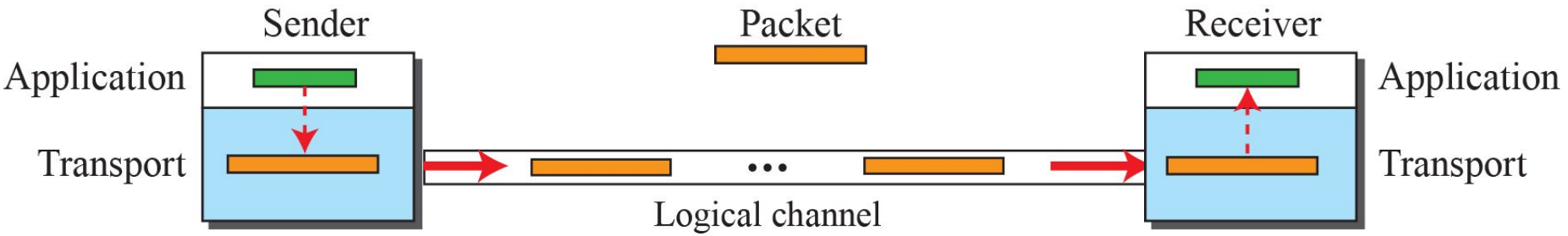
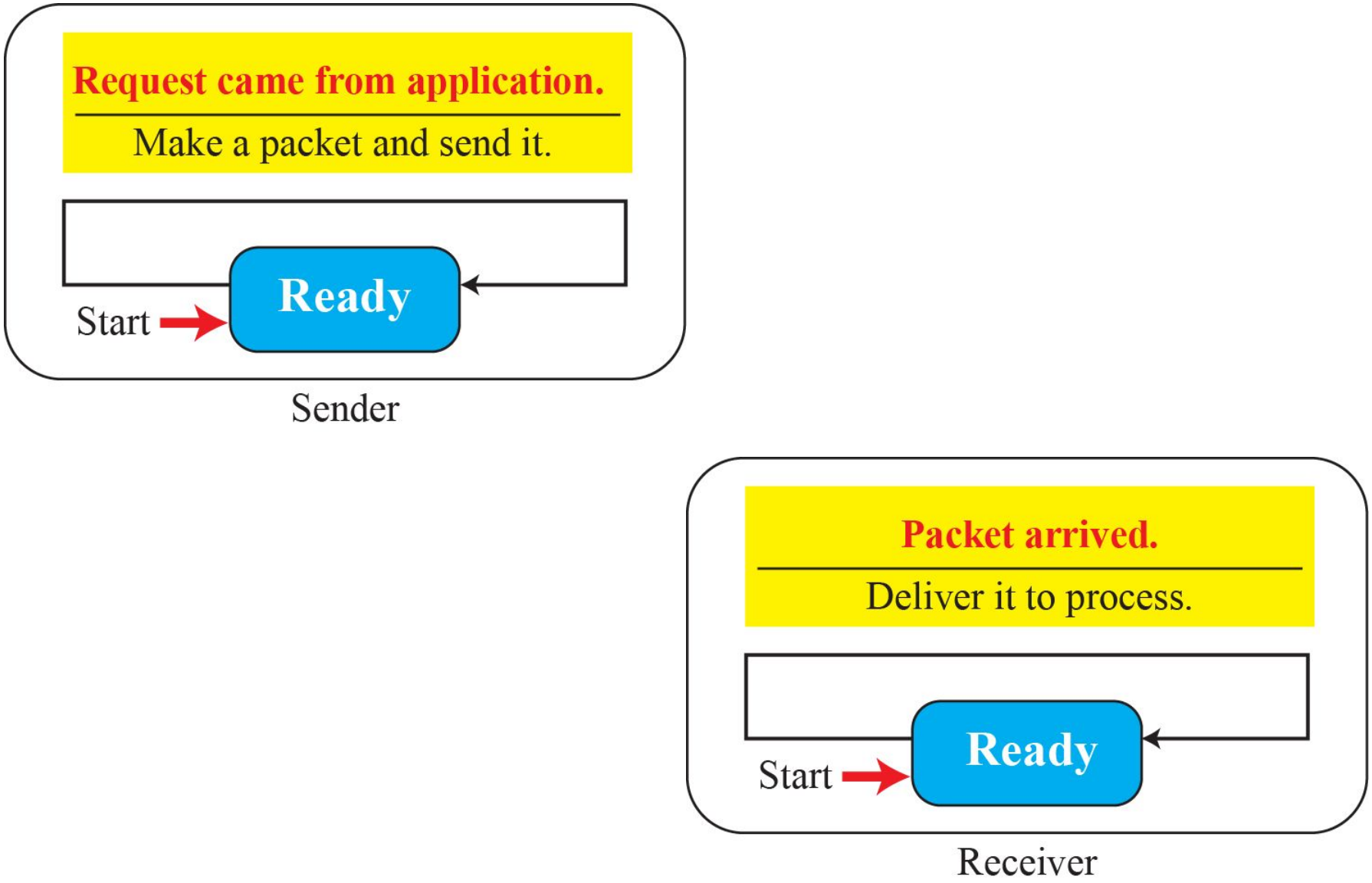


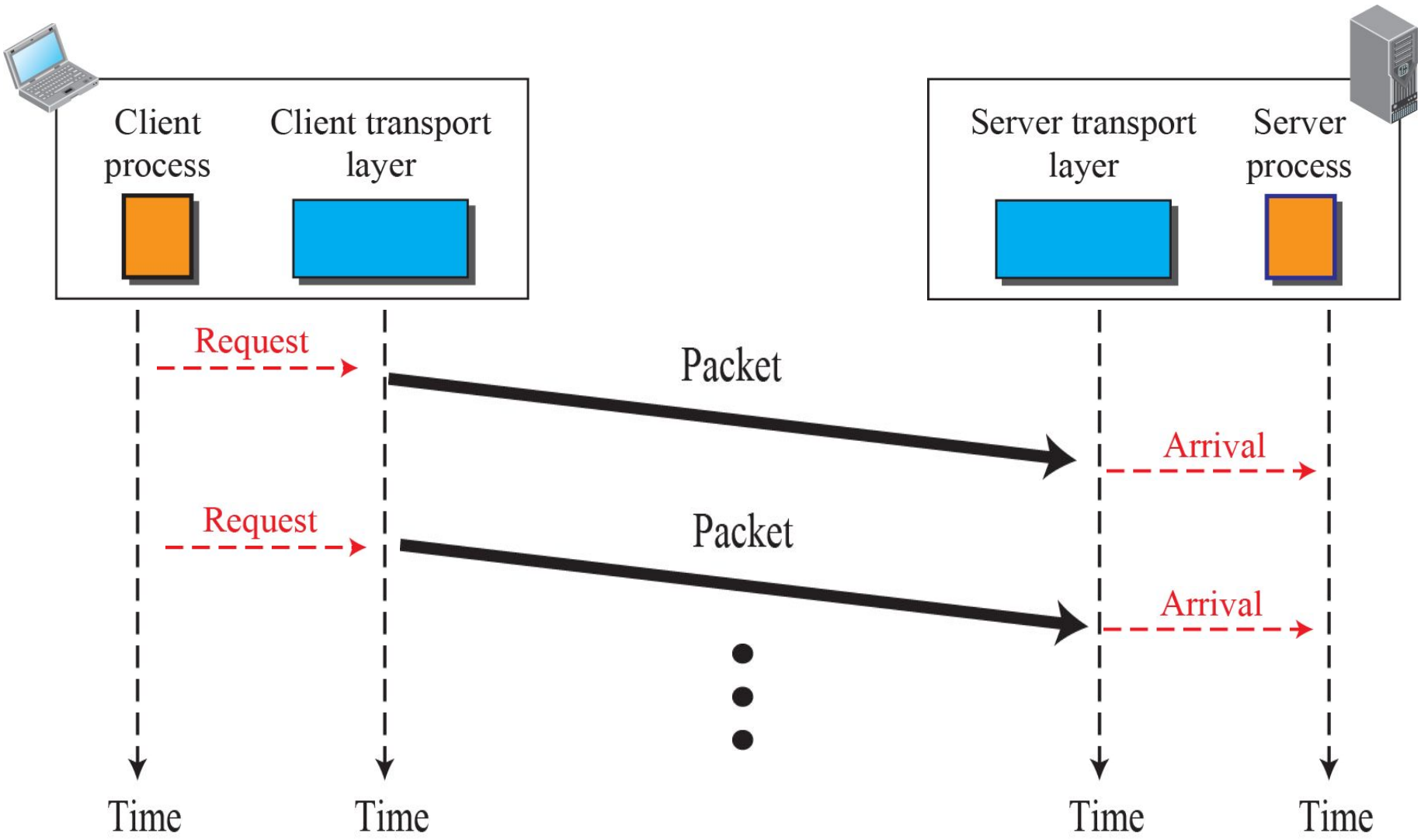
*Figure 23.18:* ***FSMs for the simple protocol***



# Example 23.3

Figure 23.19 shows an example of communication using this protocol. It is very simple. The sender sends packets one after another without even thinking about the receiver.

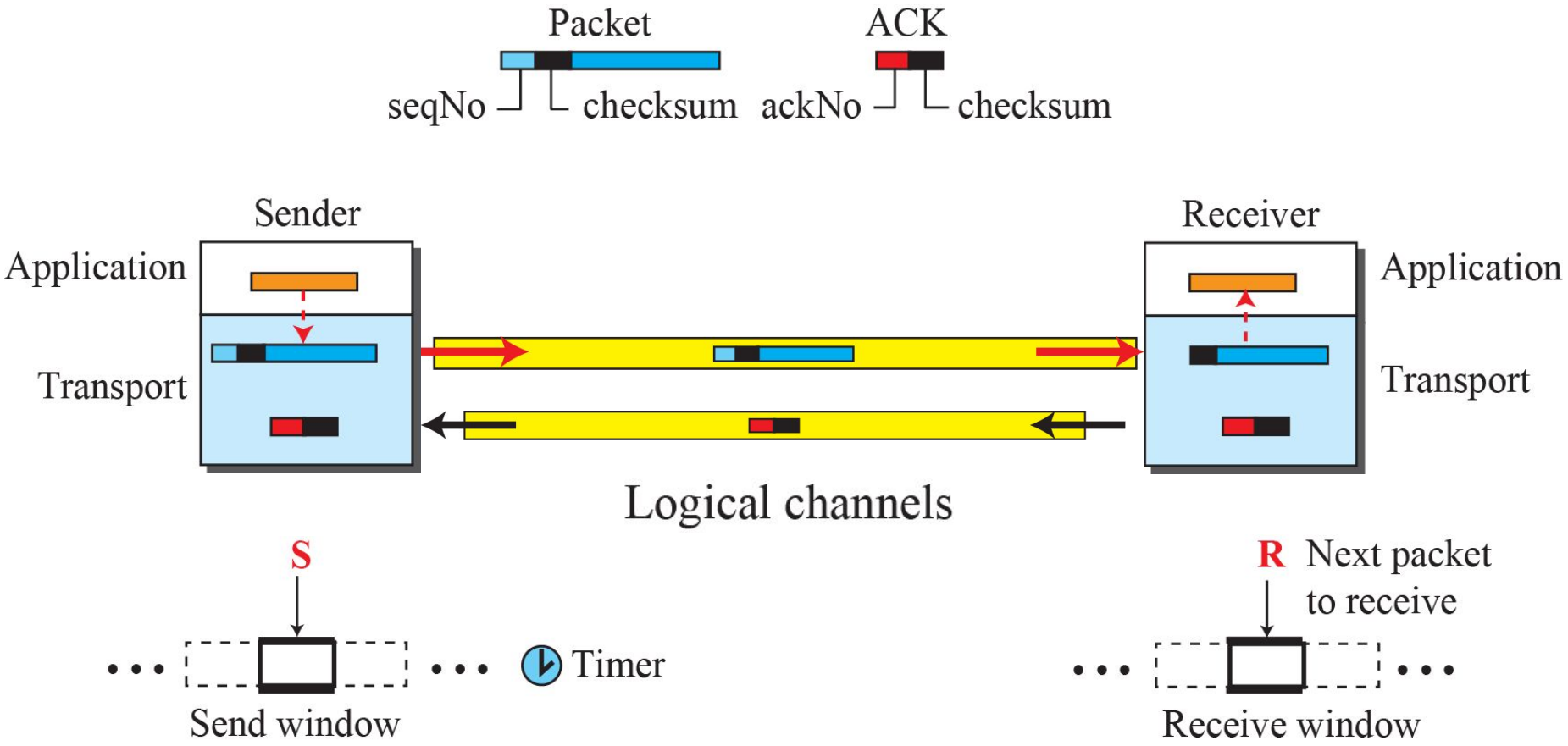
*Figure 23.19:* ***Flow diagram for Example 3.3***



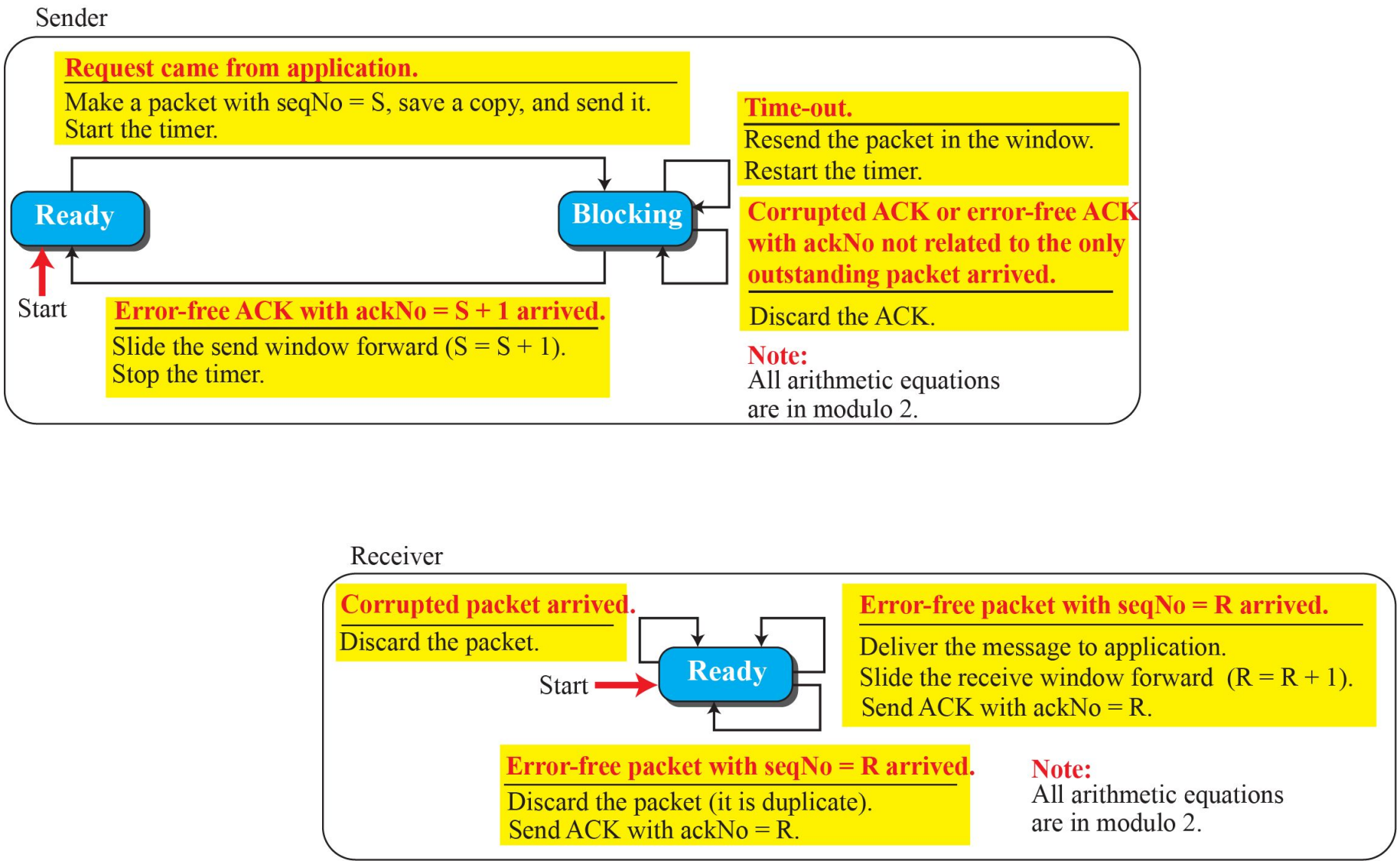
# 23.2.2 Stop-and-Wait Protocol

*Our second protocol is a connection-oriented protocol called the Stop-and-Wait protocol, which uses both flow and error control. Both the sender and the receiver use a sliding window of size 23. The sender sends one packet at a time and waits for an acknowledgment before sending the next one. To detect corrupted packets, we need to add a checksum to each data packet. When a packet arrives at the receiver site, it is checked. If its checksum is incorrect, the packet is corrupted and silently discarded.*

## *Figure 23.20: Stop-and-Wait protocol*



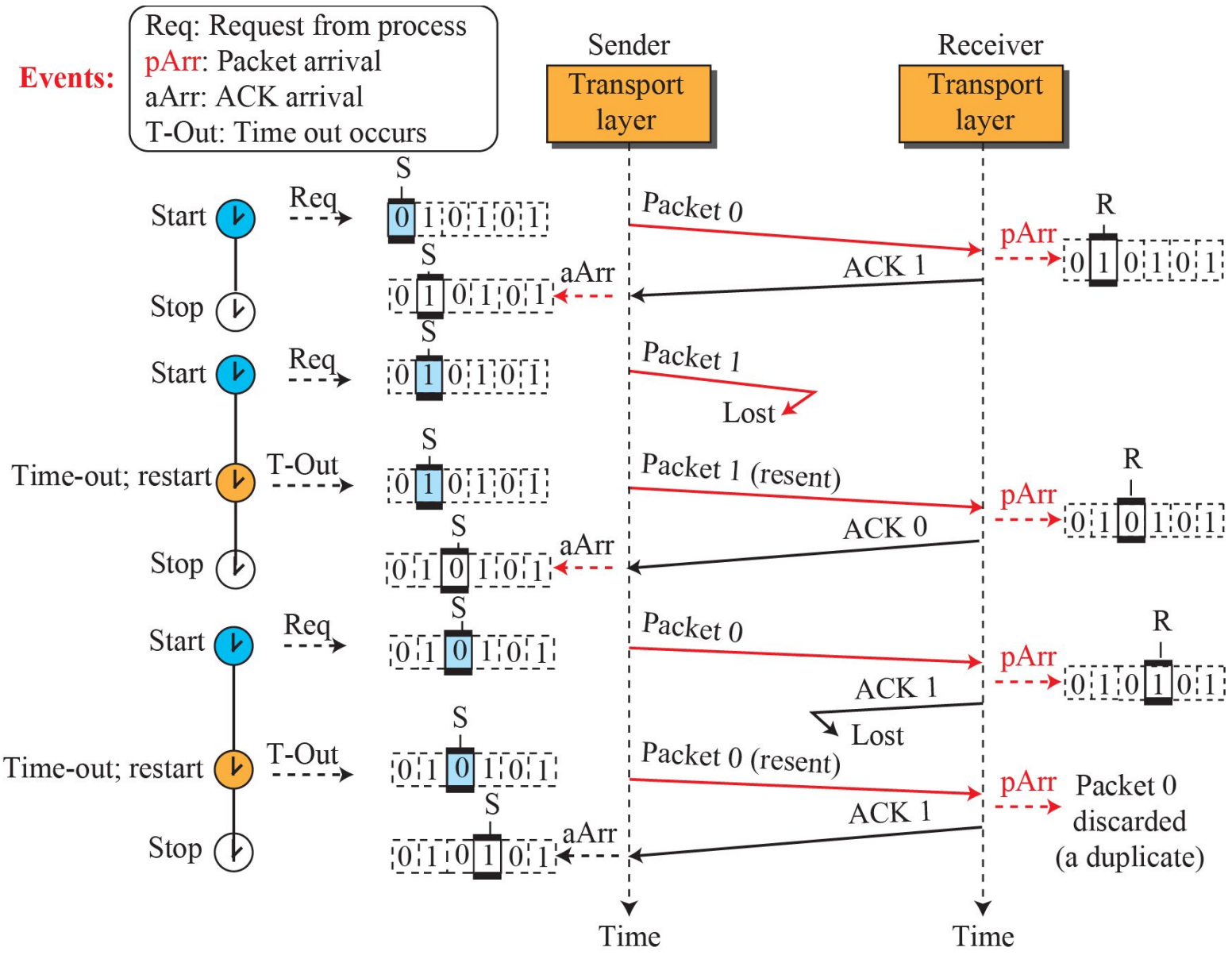
# *Figure 23.21: FSM for the Stop-and-Wait protocol*



# Example 23.4

Figure 23.22 shows an example of the 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.

## *Figure 23.22: Flow diagram for Example 3.4*



# Example 23.5

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**

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. The link utilization is only 1,000/20,000, or 5 percent.

# Example 23.6

What is the utilization percentage of the link in Example 23.5 if we have a protocol that can send up to 15 packets before stopping and worrying about the acknowledgments?

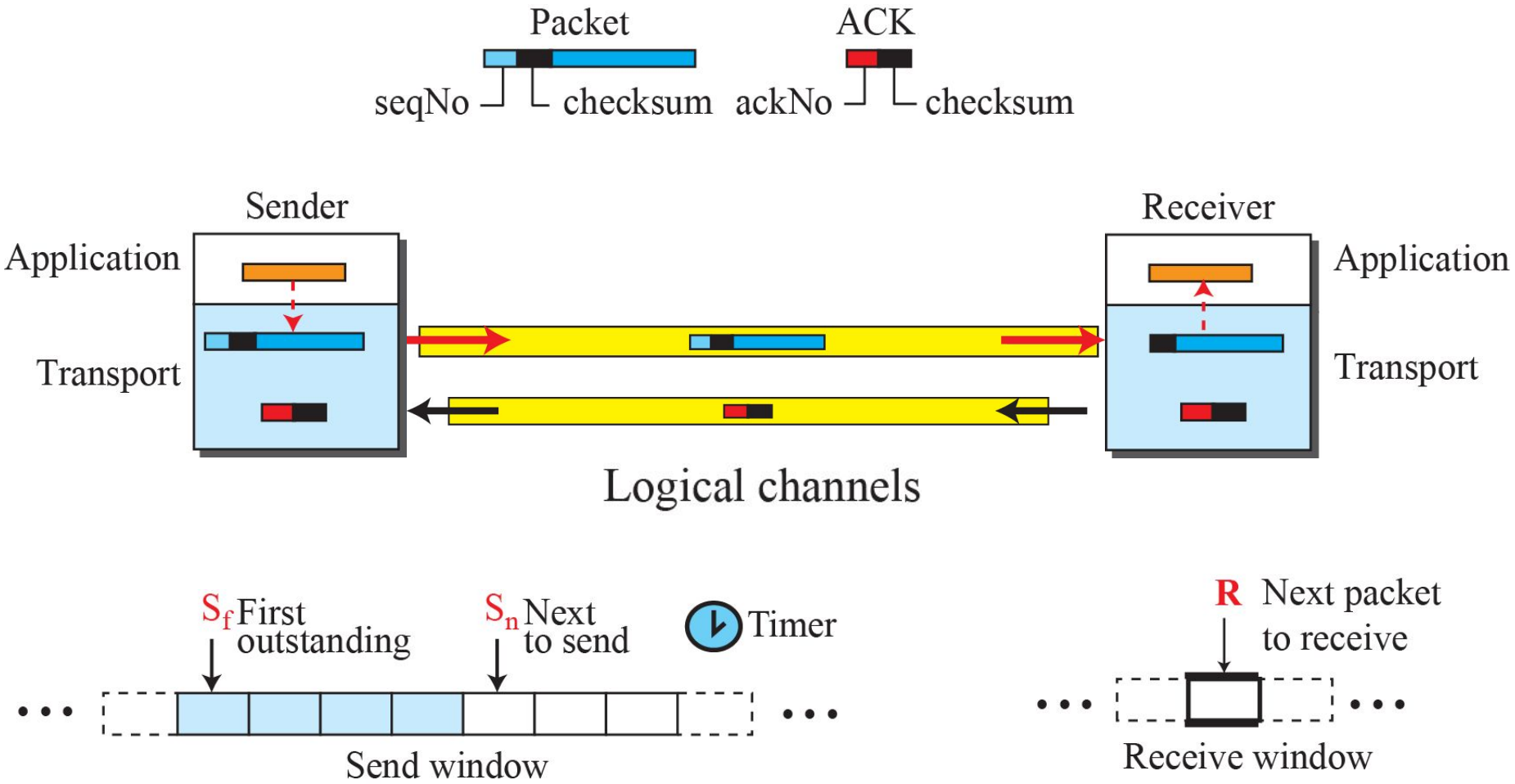
**Solution**

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.

# 23.2.3 Go-Back-N Protocol (GBN)

*To improve the efficiency of transmission (to fill the pipe), multiple packets must be in transition while the sender is waiting for acknowledgment. In other words, we need to let more than one packet be outstanding to keep the channel busy while the sender is waiting for acknowledgment. In this section, we discuss one protocol that can achieve this goal; in the next section, we discuss a second. The first is called Go-Back-N (GBN).*

## *Figure 23.23: Go-Back-N protocol*



## *Figure 23.24: Send window for Go-Back-N*

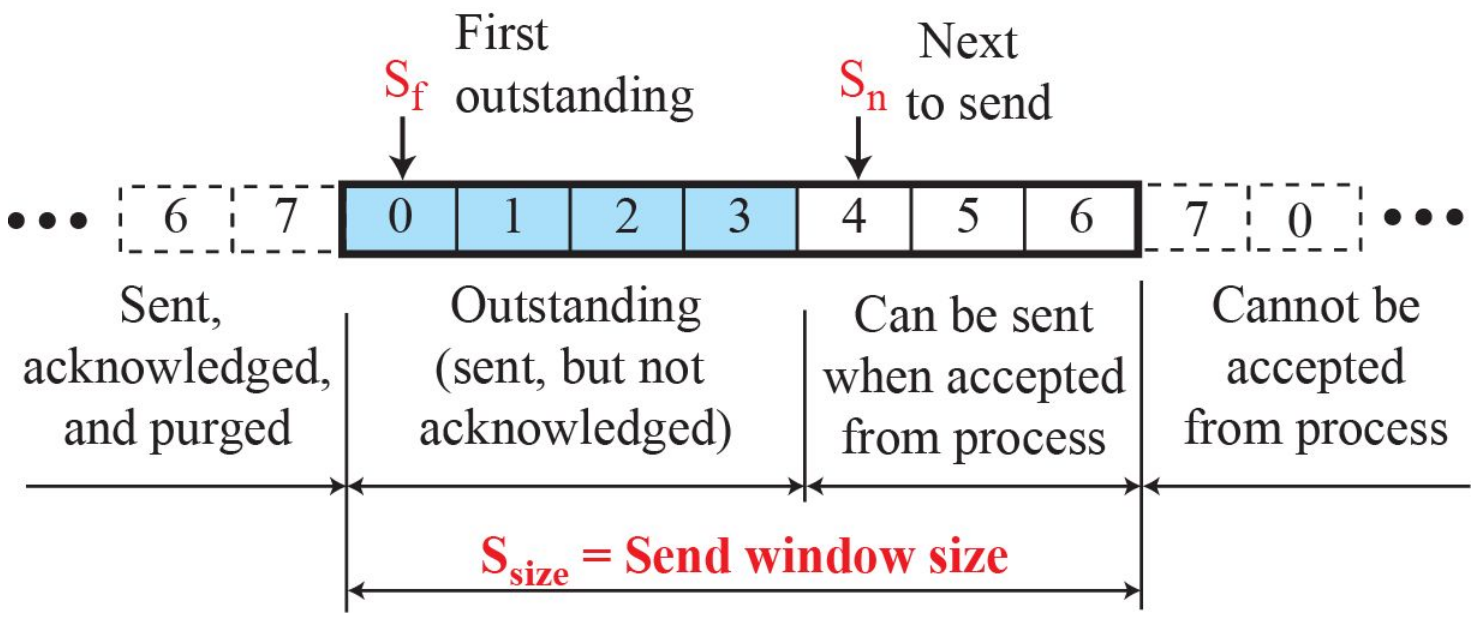


*Figure 23.25:* ***Sliding the send window***

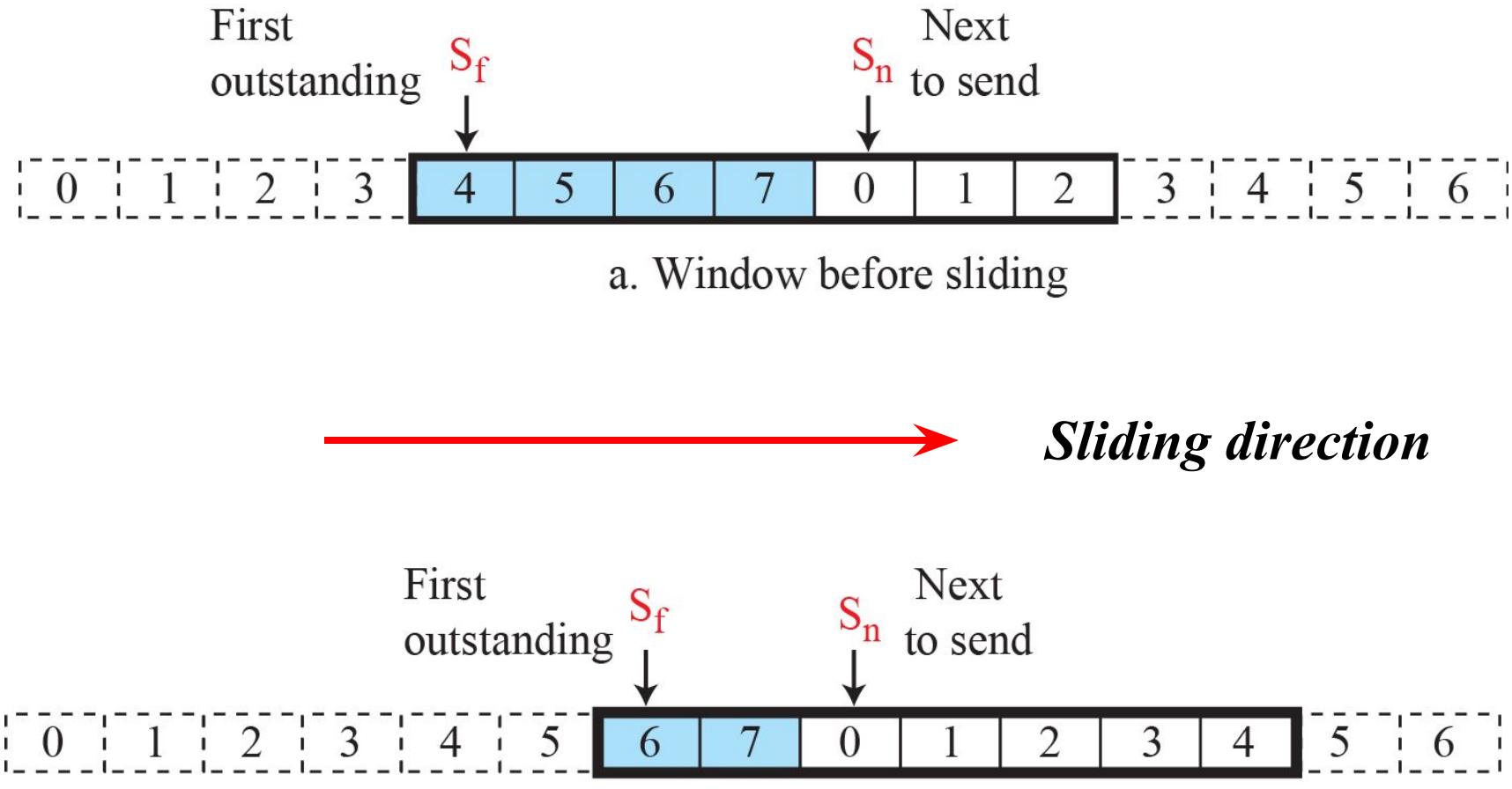
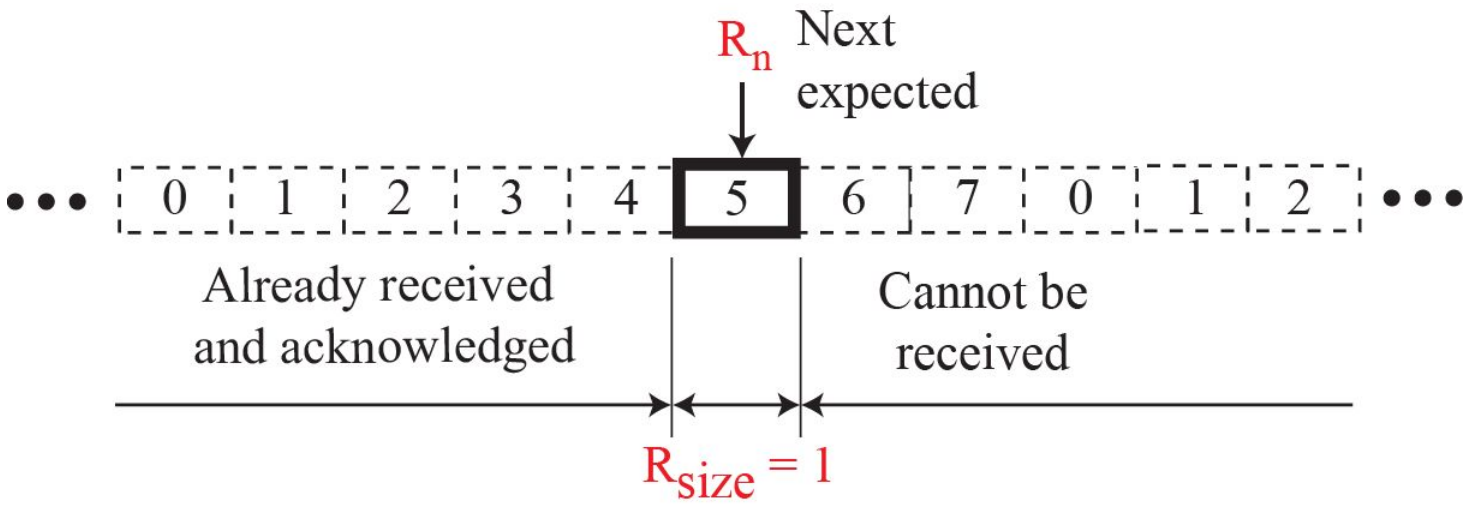
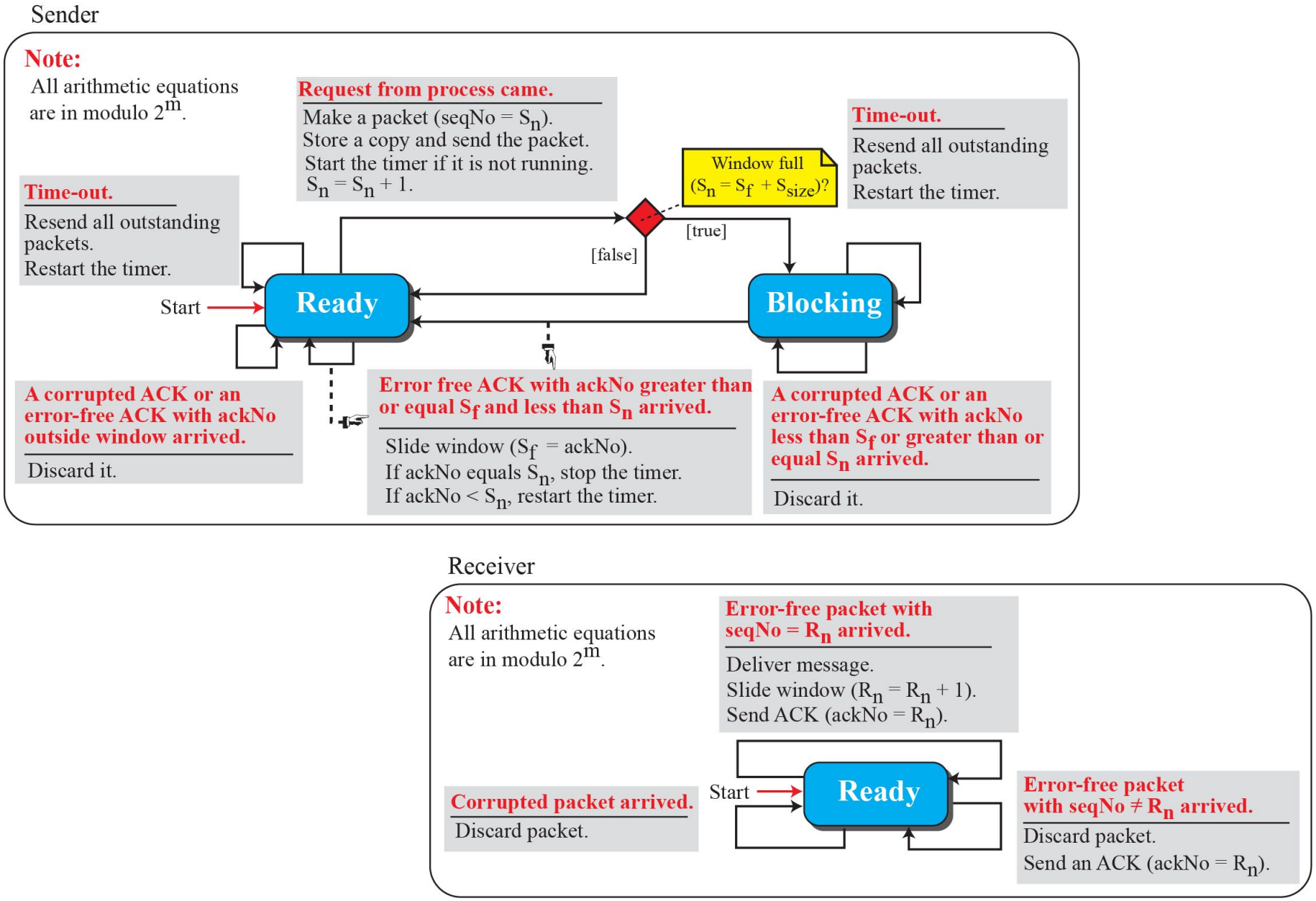


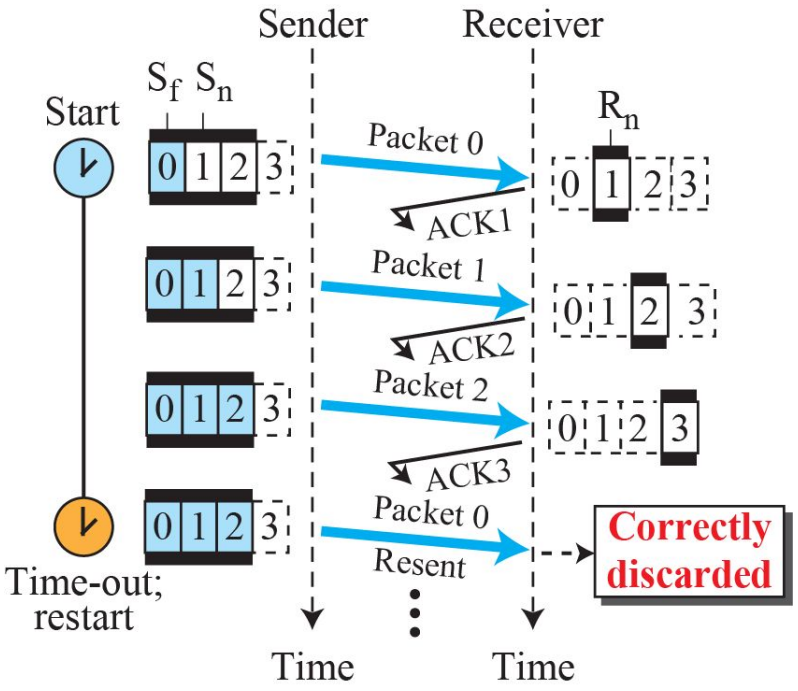
*Figure 23.26:* ***Receive window for Go-Back-N***



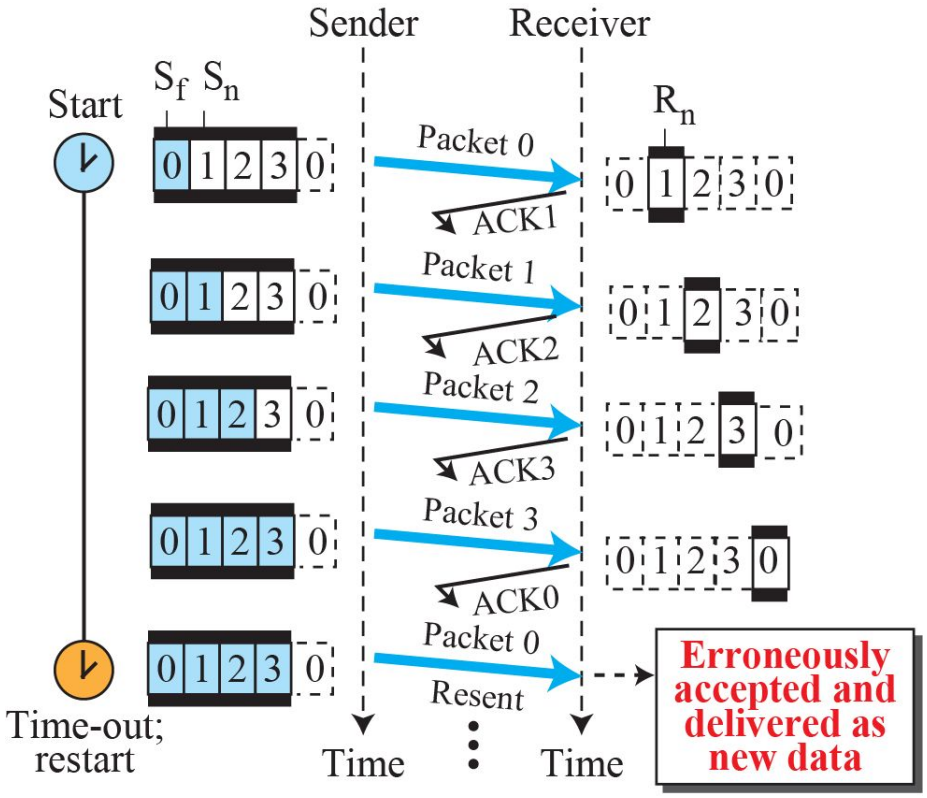
# *Figure 23.27: FSMs for the Go-Back-N protocol*



## *Figure 23.28: Send window size for Go-Back-N*




a. Send window of size < $2^m$

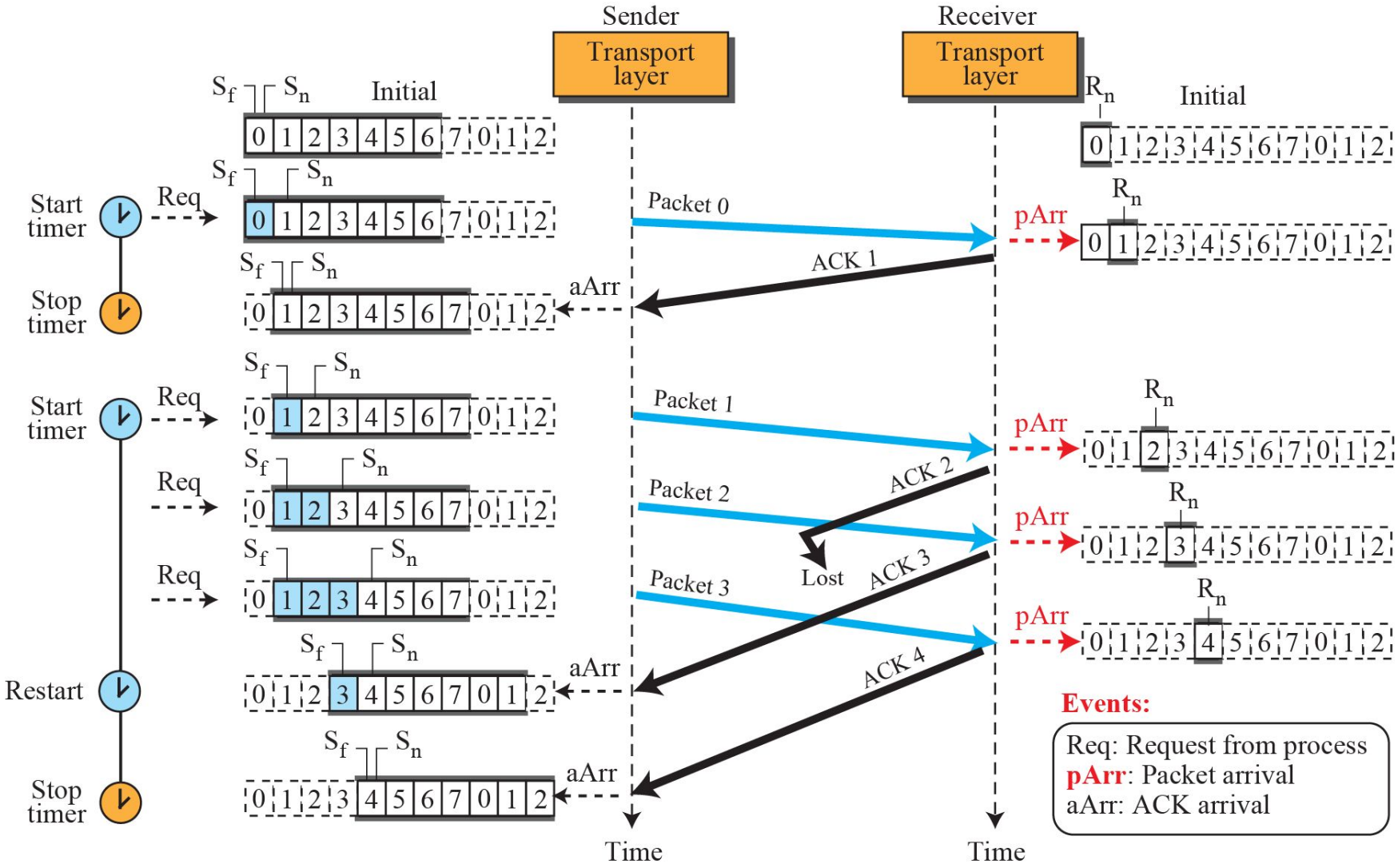



b. Send window of size = $2^m$

# Example 23.7

Figure 23.29 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.

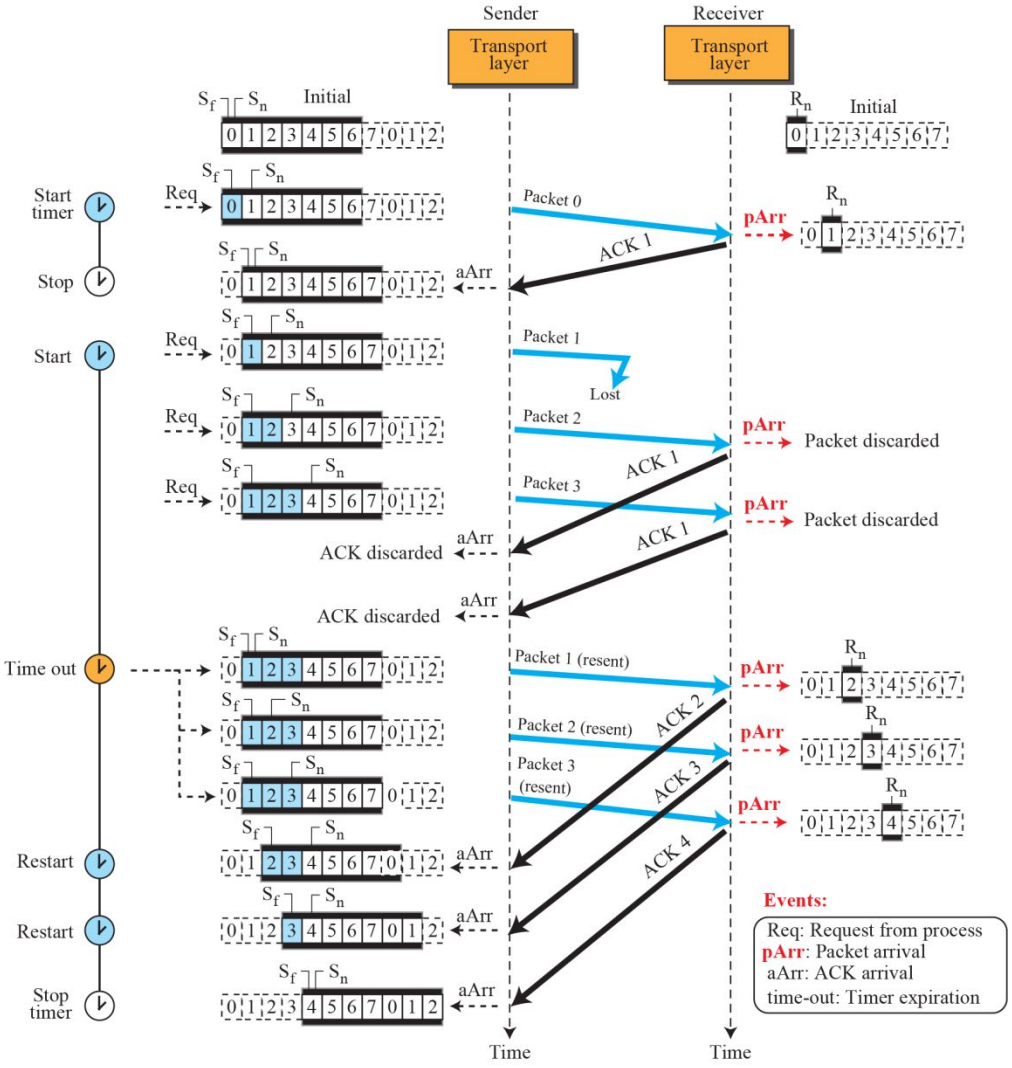
## *Figure 23.29: Flow diagram for Example 3.7*



# Example 23.8

Figure 23.30 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 23. However, these ACKs are not useful for the sender because the ackNo is equal to $S_f$, 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.

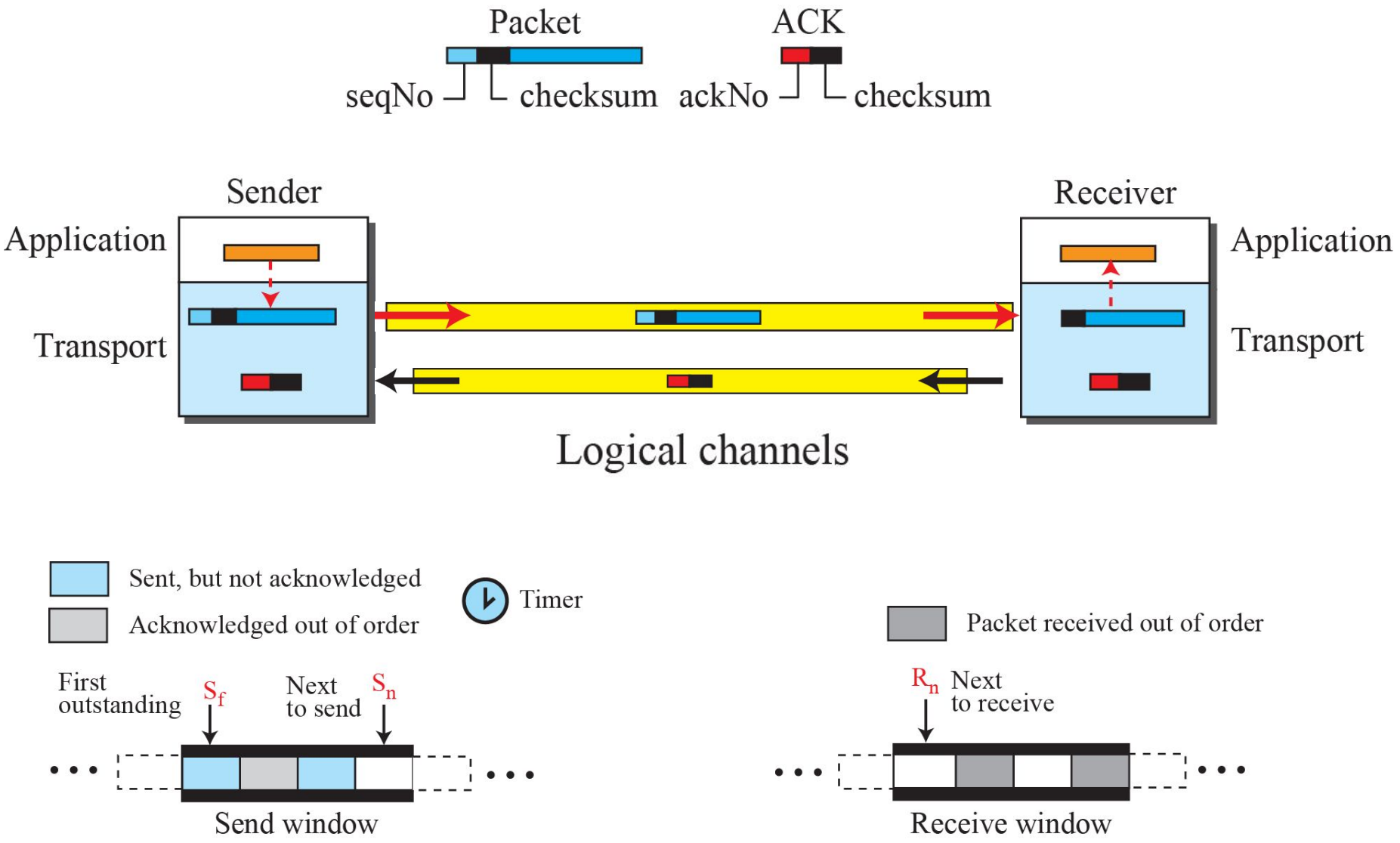
# *Figure 23.30: Flow diagram for Example 3.8*



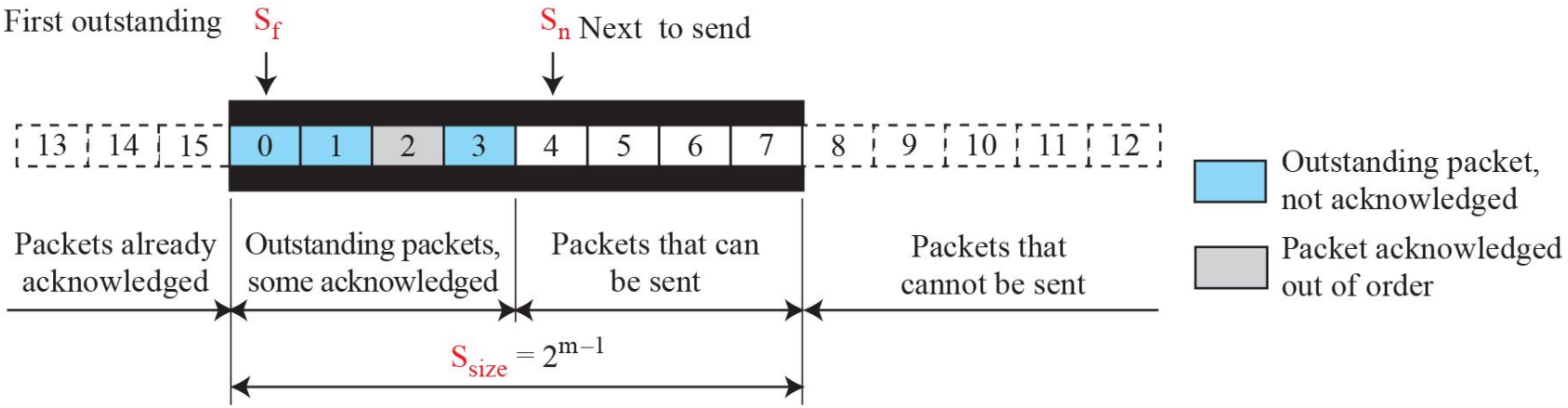
# 23.2.4 Selective-Repeat Protocol

*The Go-Back-N protocol simplifies the process at the receiver. The receiver keeps track of only one variable, and there is no need to buffer out-of-order packets; they are simply discarded. However, this protocol is inefficient if the underlying network protocol loses a lot of packets. Each time a single packet is lost or corrupted, the sender resends all outstanding packets, even though some of these packets may have been received safe and sound but out of order.*

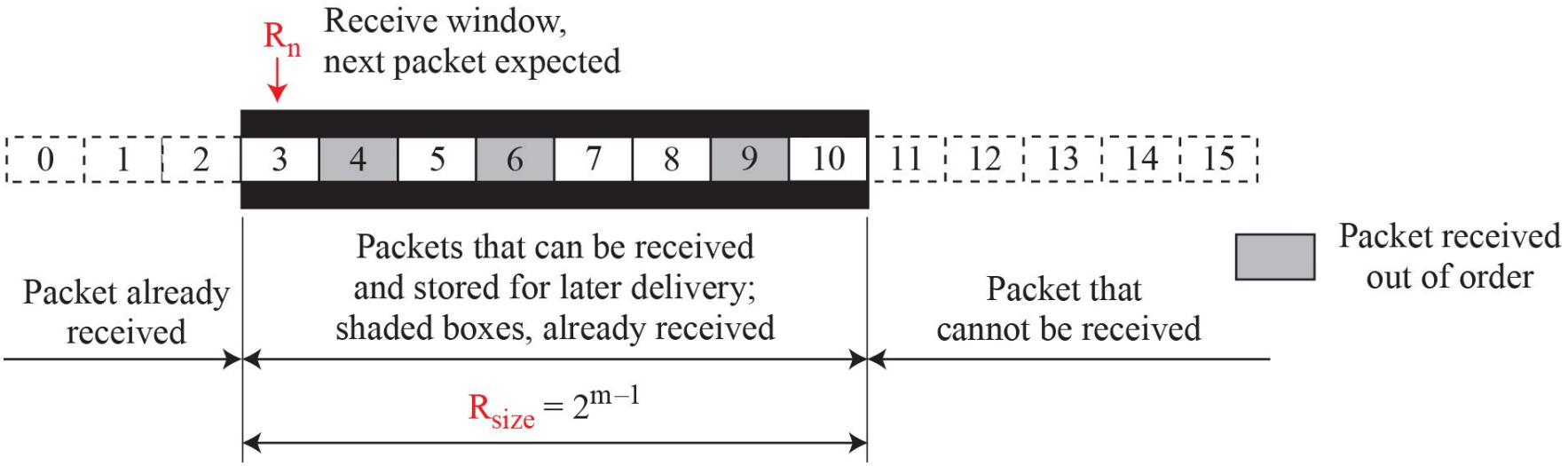
## *Figure 23.31: Outline of Selective-Repeat*



## *Figure 23.32: Send window for Selective-Repeat protocol*



## *Figure 23.33: Receive window for Selective-Repeat protocol*



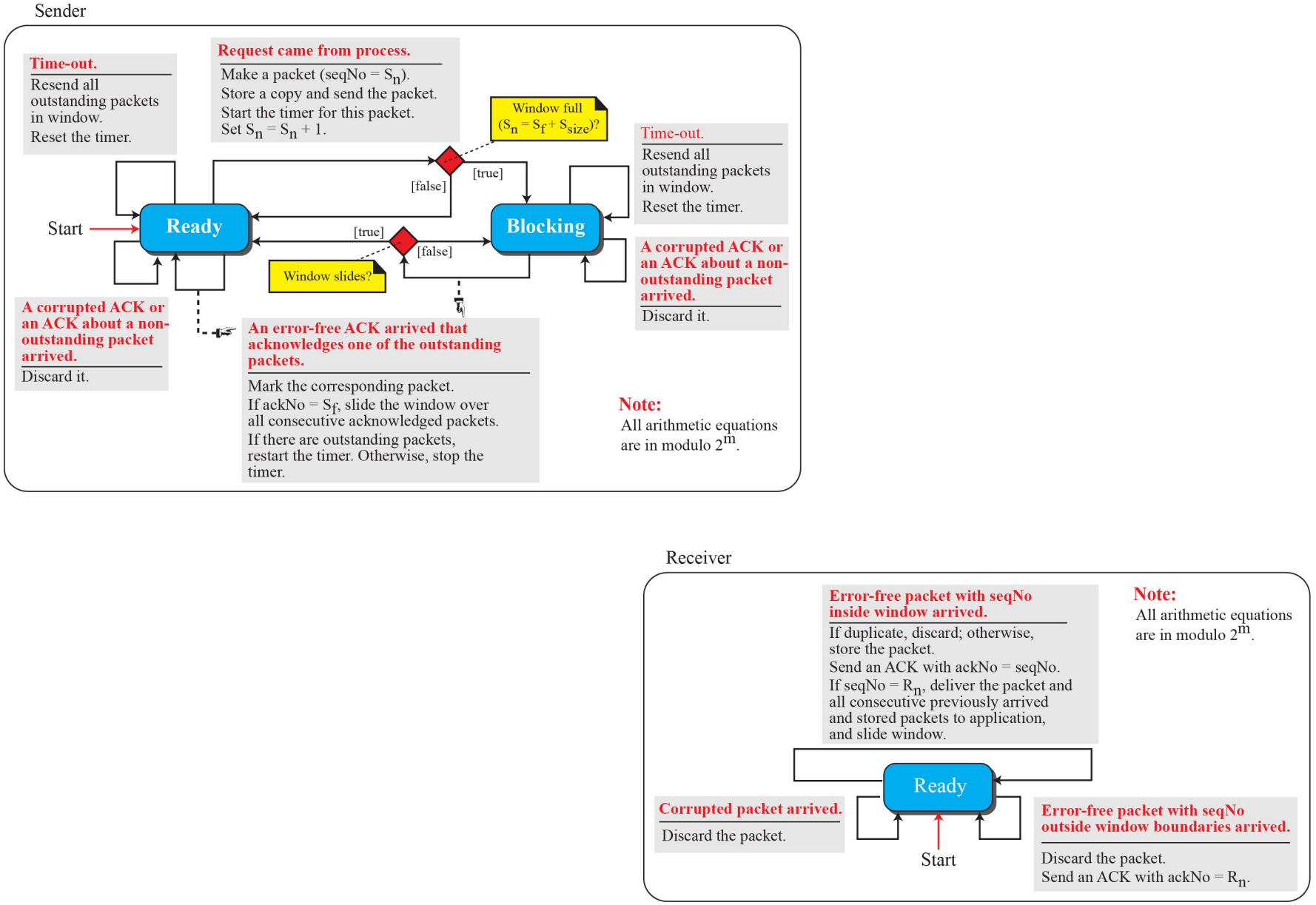
# Example 23.9

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.

# *Figure 23.34: FSMs for SR protocol*



# Example 23.10

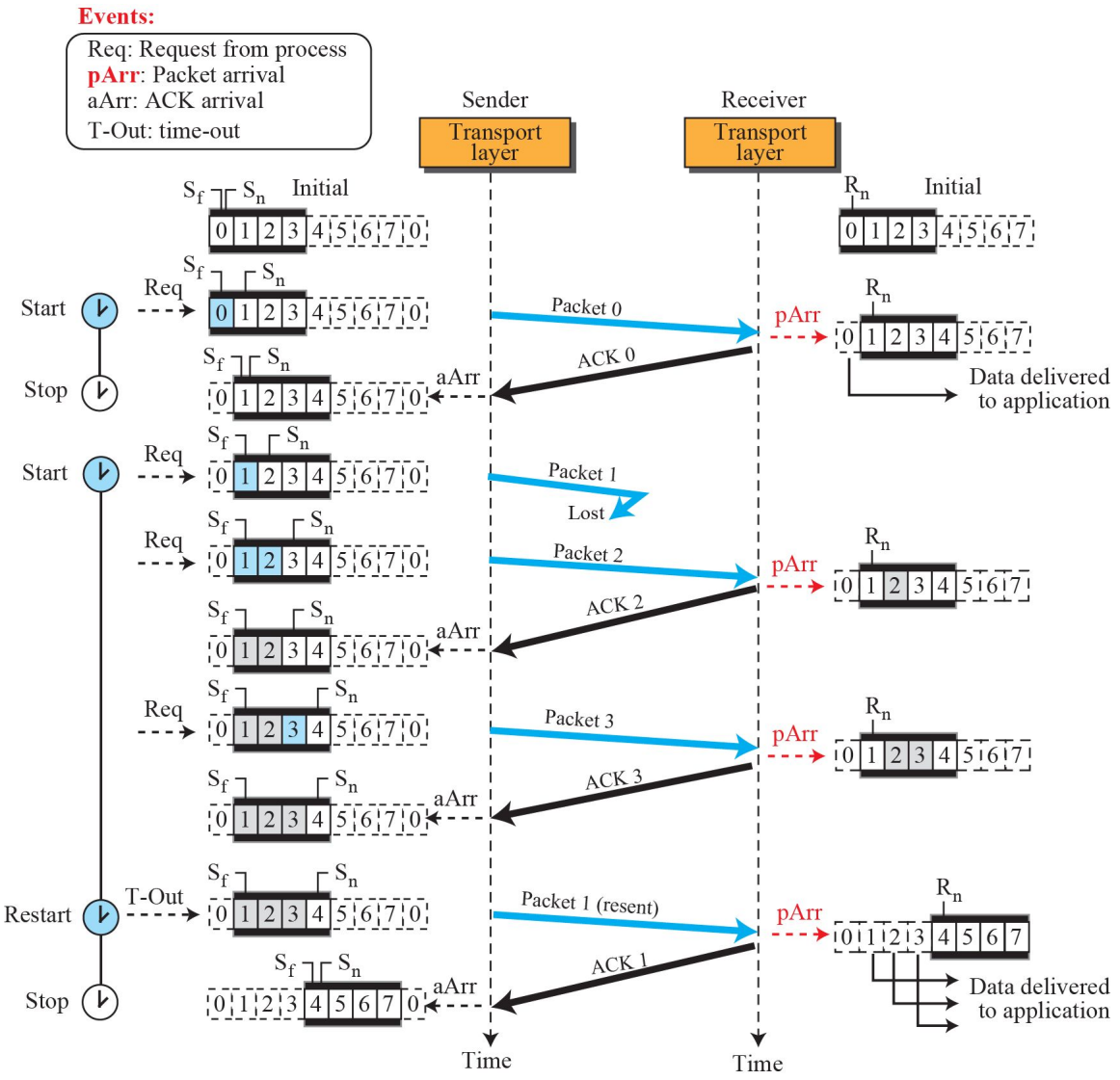
This example is similar to Example 23.8 (Figure 23.30) in which packet 1 is lost. We show how Selective-Repeat behaves in this case. Figure 3.35 shows the situation.

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.

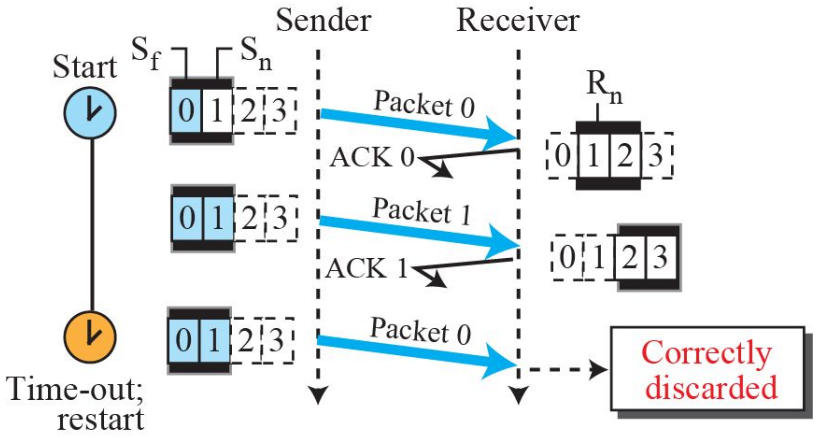
# Example 23.10 (continued)

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

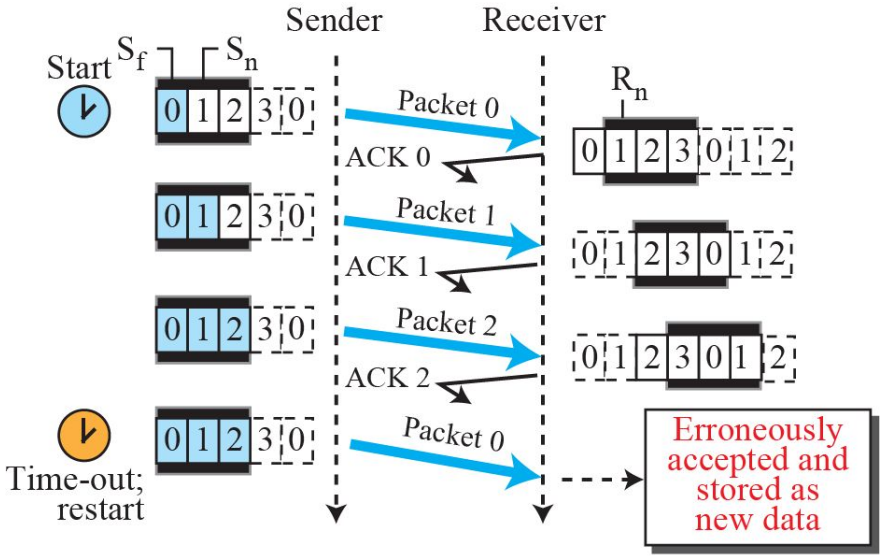
# *Figure 23.35: Flow diagram for Example 3.10*



## *Figure 23.36: Selective-Repeat, window size*




a. Send and receive windows of size = $2^{m-1}$

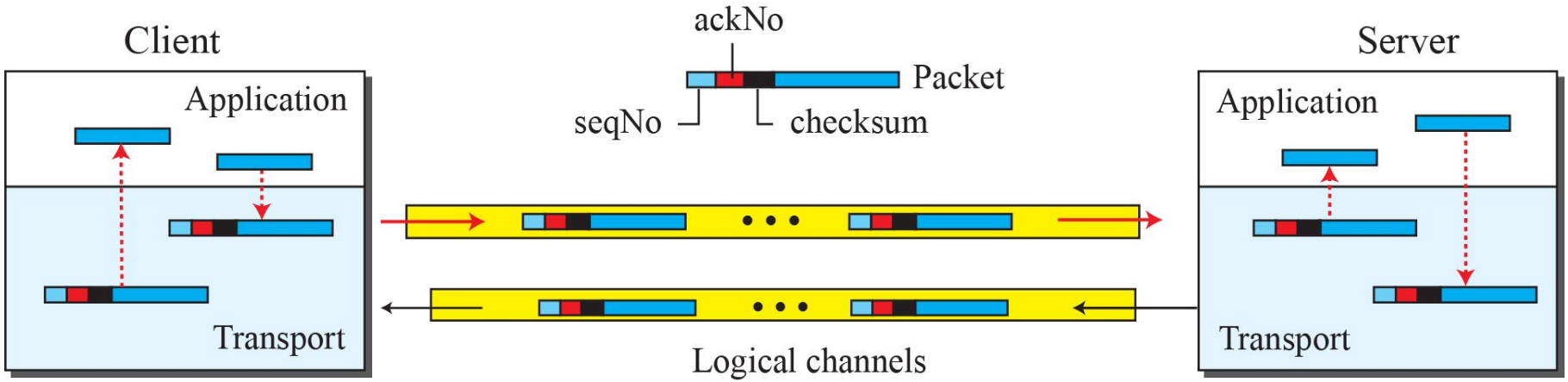


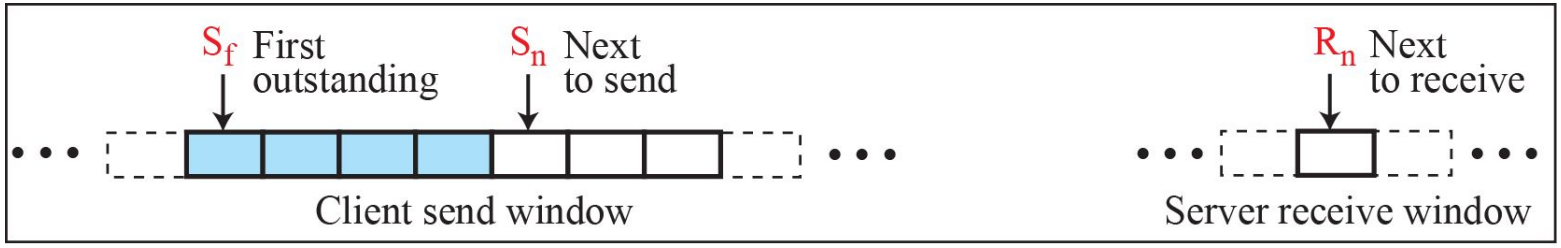

b. Send and receive windows of size > $2^{m-1}$

# 23.2.5 Bidirectional Protocols

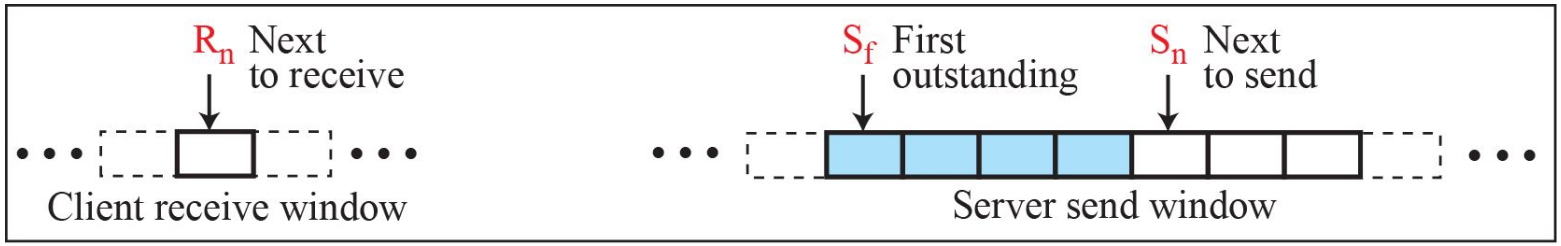
*The four protocols we discussed earlier in this section are all unidirectional: data packets flow in only one direction and acknowledgments travel in the other direction. In real life, data packets are normally flowing in both directions: from client to server and from server to client. This means that acknowledgments also need to flow in both directions. A technique called piggybacking is used to improve the efficiency of the bidirectional protocols. When a packet is carrying data from A to B, it can also carry acknowledgment feedback about arrived packets from B*

## *Figure 23.37: Design of piggybacking in Go-Back-N*




Windows for communication from client to server




Windows for communication from server to client