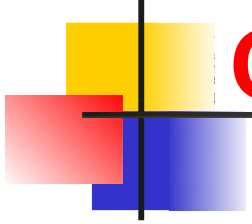
Chapter 3

Transport Layer



Chapter 3: Outline

1. *INTRODUCTION*
2. *TRANSPORT-LAYER PROTOCOLS*
3. *USER DATAGRAM PROTOCOL*
4. *TRANSMISSION CONTROL PROTOCOL*

Chapter 3: Objective

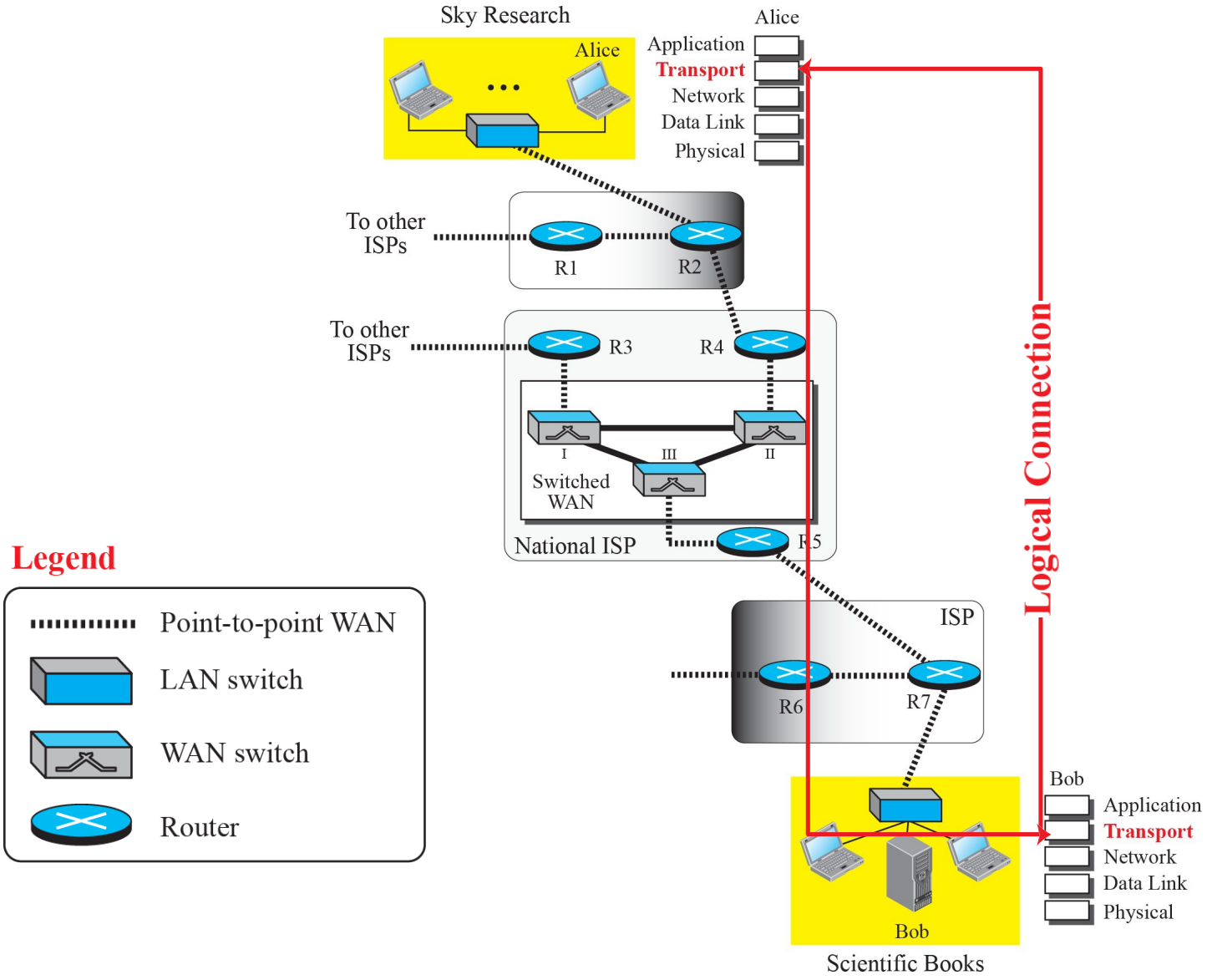
- *We introduce general services we normally require from the transport layer, such as process-to-process communication, addressing, multiplexing, error, flow, and congestion control.*
- *We discuss general transport-layer protocols such as Stop-and-Wait, Go-Back-N, and Selective-Repeat.*
- *We discuss UDP, which is the simpler of the two protocols we discuss in this chapter.*
- *We discuss TCP, which provides a connection-oriented service using a transition diagram. Finally, we show how TCP controls error and congestion control in TCP.*

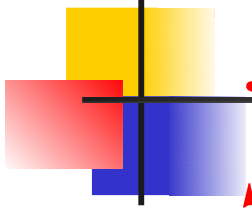
3-1
INTRODUCTION

The transport layer provides a process-to-process communication between two application layers.

Communication is provided using a logical connection, which means that the two application layers assume that there is an imaginary direct connection through which they can send and receive messages.

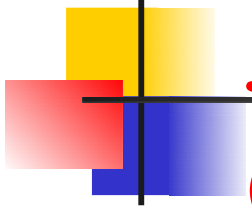
Figure 3.1: Logical connection at the transport layer





3.1.1 Transport-Layer Services

As we have already discussed , 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.



3.1.1 (continued)

- *Process-to-Process Communication*
- *Addressing: Port Numbers*
- *ICANN Ranges*
  - *Well-known ports*
  - *Registered ports*
  - *Dynamic ports*
- *Encapsulation and Decapsulation*
- *Multiplexing and Demultiplexing*

3.1.1 (continued)

θ Flow Control

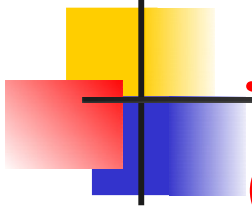
- ϖ Pushing or Pulling
- ϖ Flow Control at Transport Layer
- ϖ Buffers

θ Error Control

- ϖ Sequence Numbers
- ϖ Acknowledgment

θ Combination of Flow and Error Control

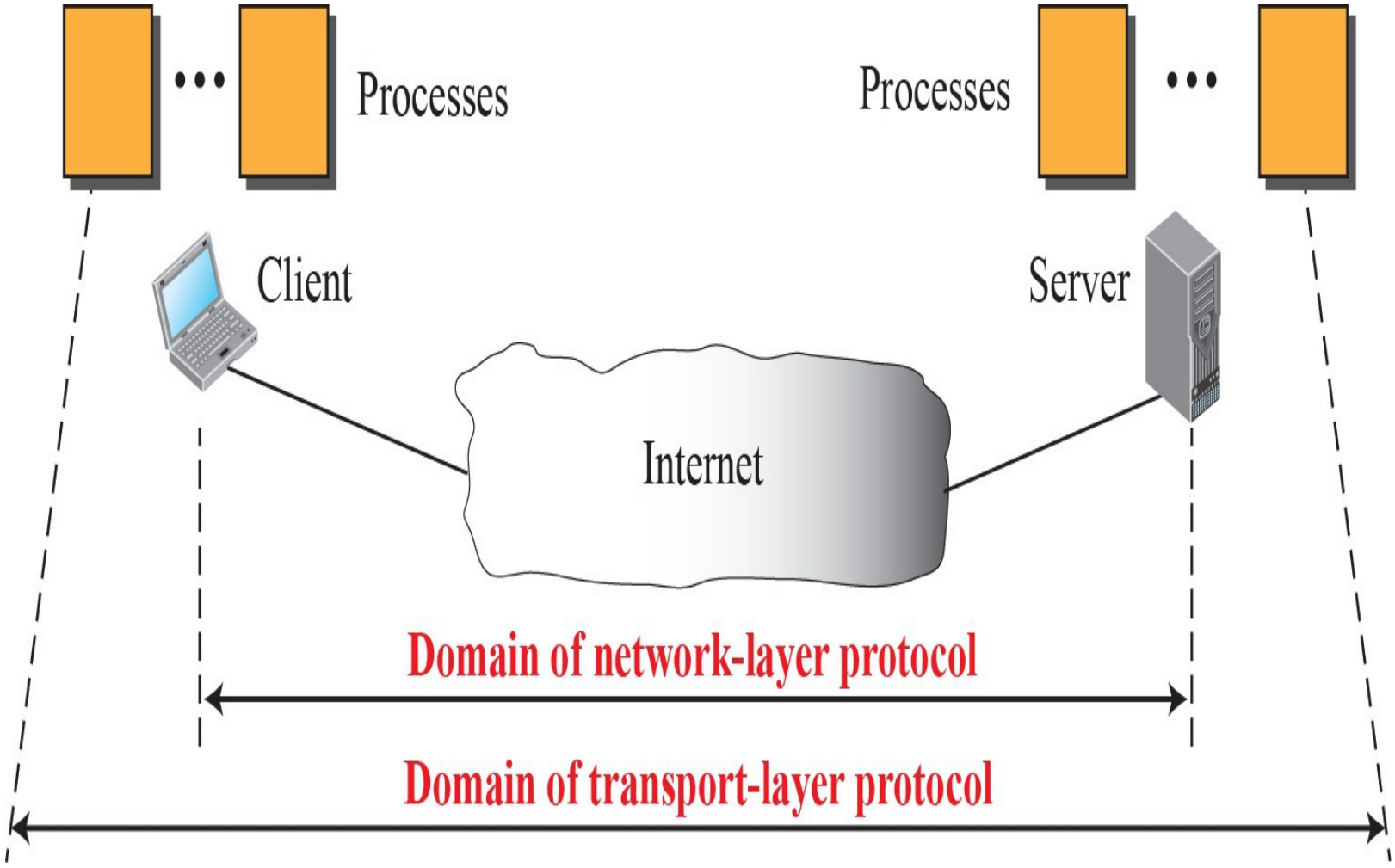
- ϖ Sliding Window



3.1.1 (continued)

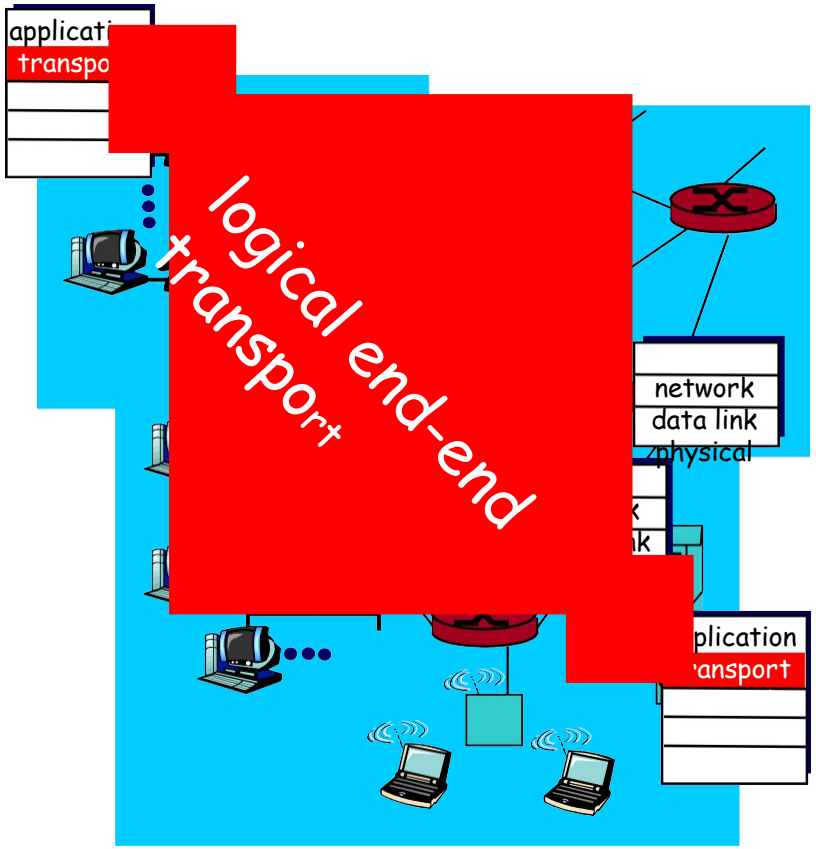
- *Congestion Control*
- *Connectionless and Connection-Oriented*
  - *Connectionless Service*
  - *Connection-Oriented Service*
  - *Finite State Machine*
- *Multiplexing and Demultiplexing*

Figure 3.2: Network layer versus transport layer

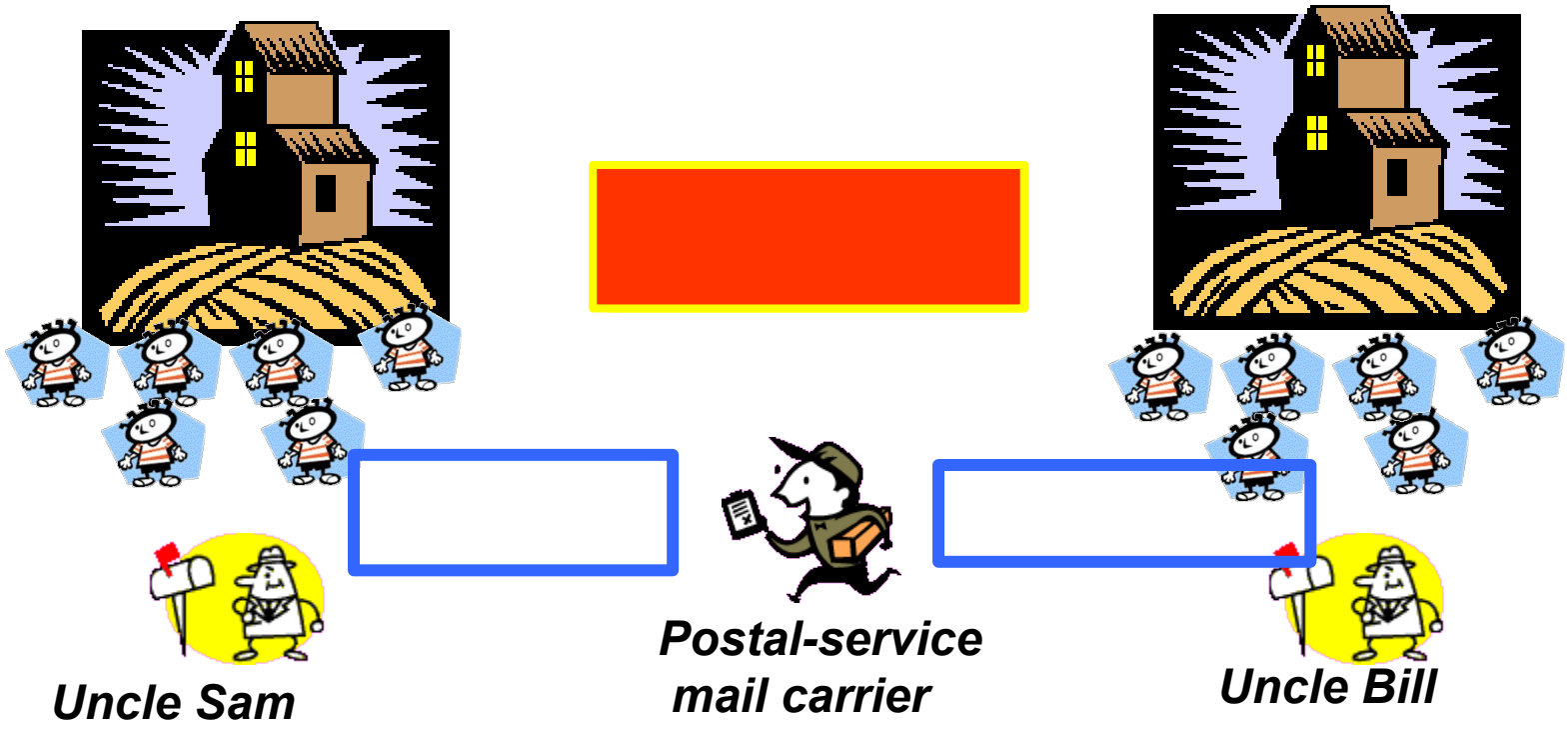


Transport Protocols

- Provide *logical communication* between application processes running on different hosts
- Run on end hosts
- Sender: breaks application messages into segments, and passes to network layer
- Receiver: reassembles segments into messages, passes to application layer
- Multiple transport protocol available to applications
  - Internet: TCP and UDP



An Analogy: Cousins sending letters



θ *Uncle Sam & Uncle Bill - responsible for mail collection, distribution, and communicating with postal service*

θ **Postal service – carries the mails**

***hosts* (also called end systems) = ?**
***processes* = ?**
***application messages* = ? *network layer protocol* = ? *transport layer protocol* = ?**

Their services are constrained by the possible services that the postal service provides

hosts *(also called end systems)* = houses

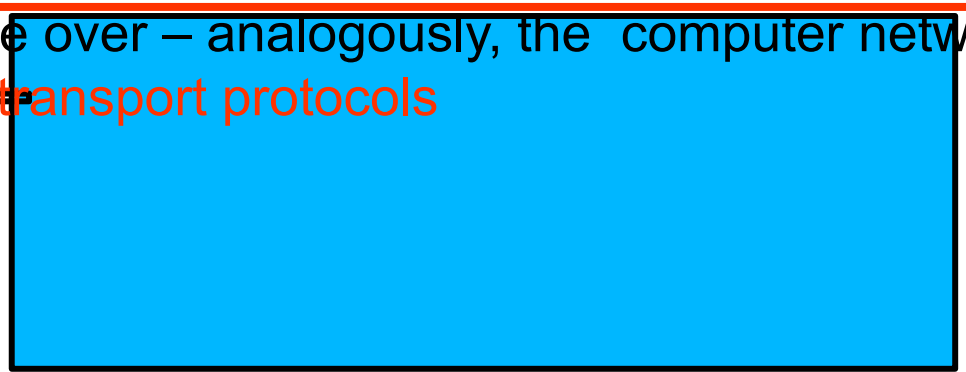
processes = cousins

application messages = letters in envelope

transport layer protocol = Uncle Sam and Uncle Bill

network layer protocol = postal service(including mail persons)

It may so happen that their uncles could get sick, and so other people may take over – analogously, the computer network may provide multiple transport protocols



Transport Layer

θ *logical communication between processes*

θ *moves messages from application process to the network layer and vice-versa: Sending & Receiving sides*

θ *computer network can make multiple transport layer protocols available*

- TCP
- UDP

θ *process-to-process communication*

Network Layer

θ *logical communication between end systems*

θ *host-to-host communication*

Port numbers

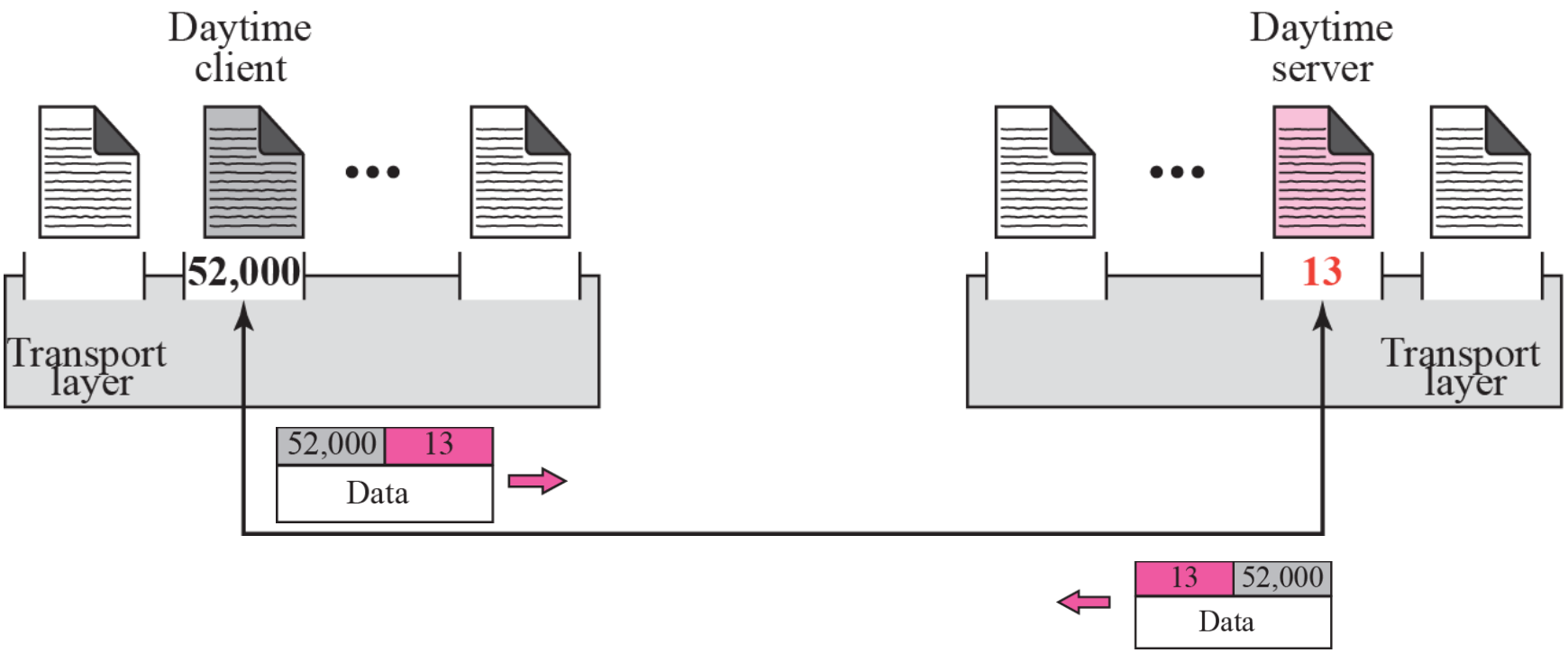
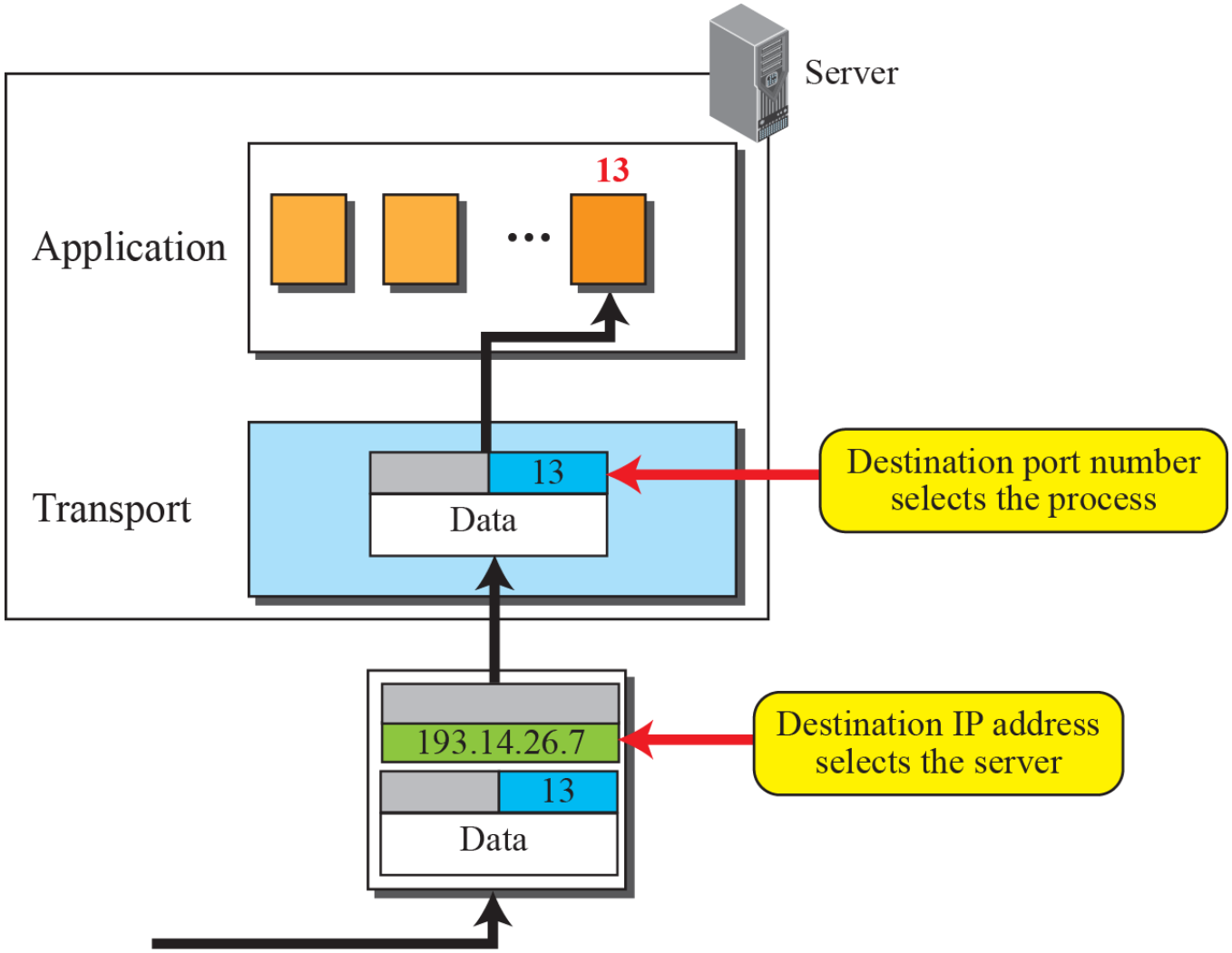


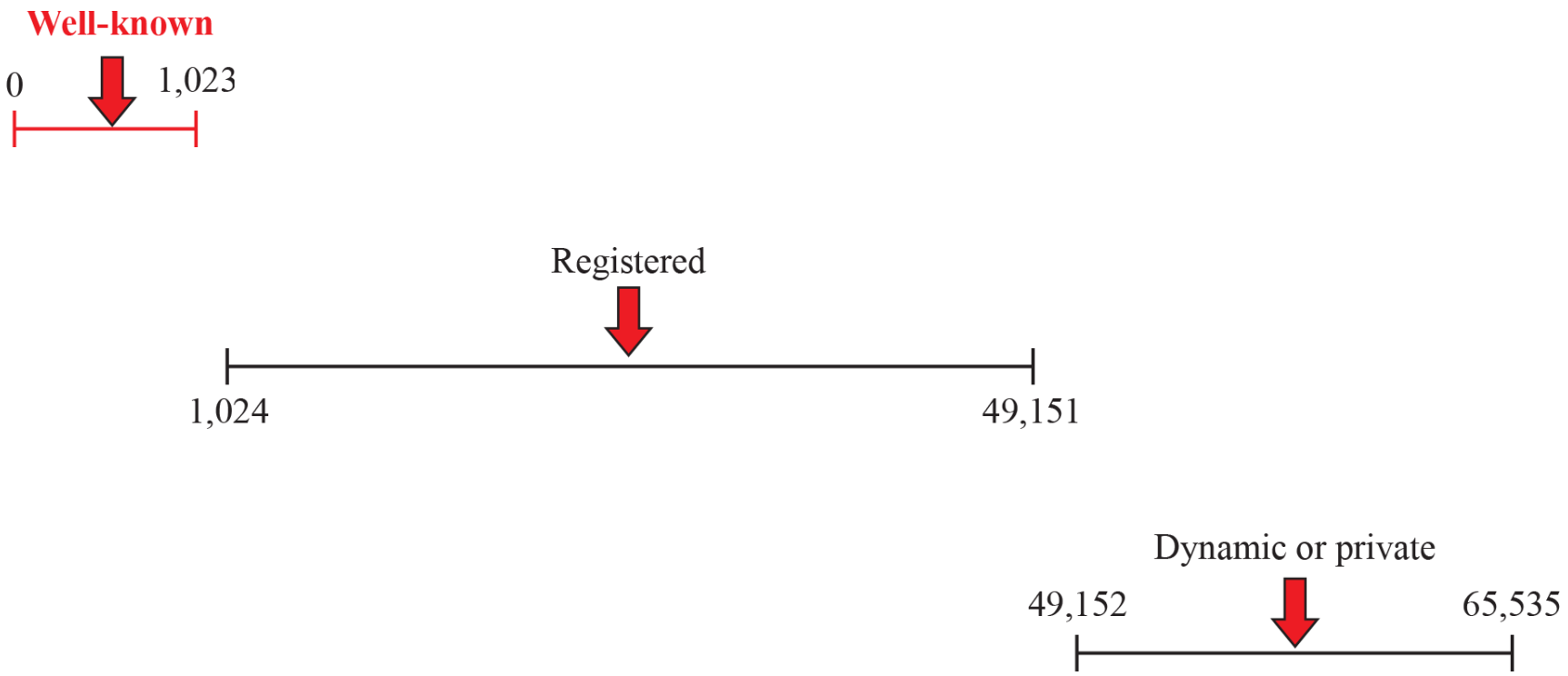
Figure 3.4: IP addresses versus port numbers



Port numbers

- ICANN(Internet Corporation for Assigned Names and Numbers) has divide the port number into three ranges:-
- Well Known-assigned and controlled by ICANN.
- Registered-They can be registered with ICANN to avoid duplication.
- Dynamic-They can be used as temporary or private port numbers.

Figure 3.5: ICANN ranges



Note

The well-known port numbers are less than 1,024.

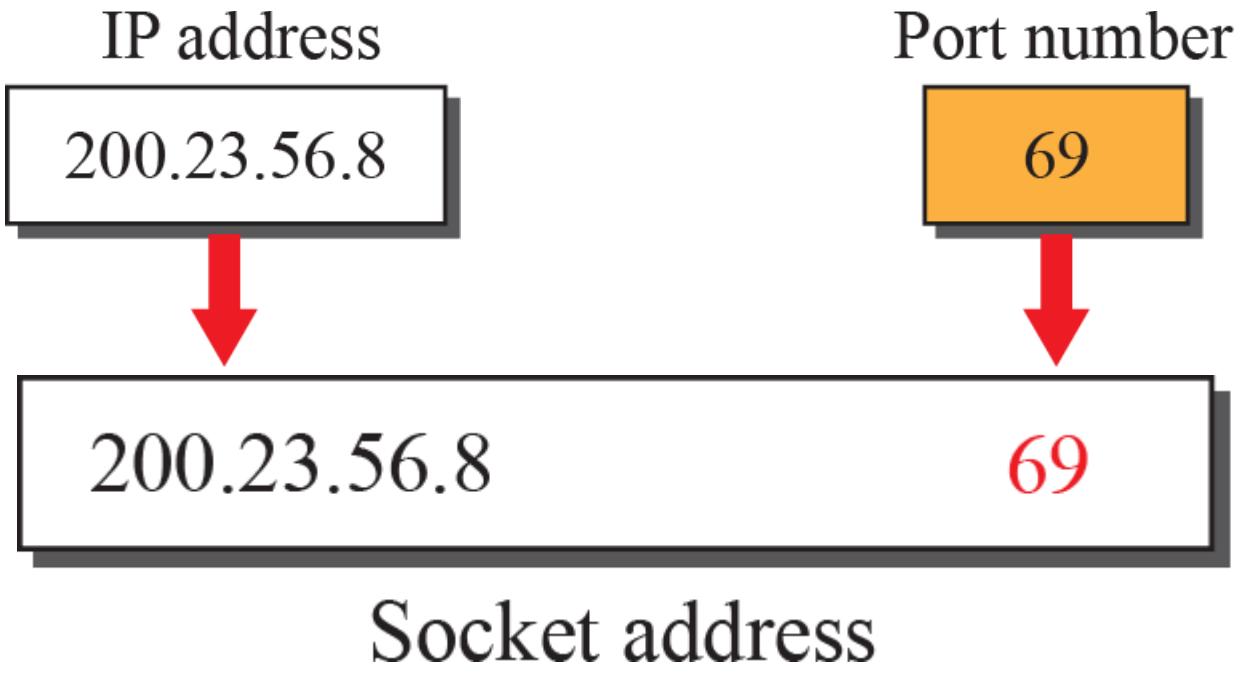
In UNIX, the well-known ports are stored in a file called /etc/services. Each line in this file gives the name of the server and the well-known port number. We can use the grep utility to extract the line corresponding to the desired application. The following shows the port for TFTP. Note that TFTP can use port 69 on either UDP or TCP. SNMP uses two port numbers (161 and 162), each for a different purpose.

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

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

Figure 3.6: Socket address

The combination of IP Address and a port number is called Socket address



Encapsulation and Decapsulation

- Encapsulation happens at sender site.When a process has message to send it passes the a message to the transport layer along with a pair of socket address and some other pieces of information ,which depend on transport layer protocol.

Figure 3.7: Encapsulation and decapsulation

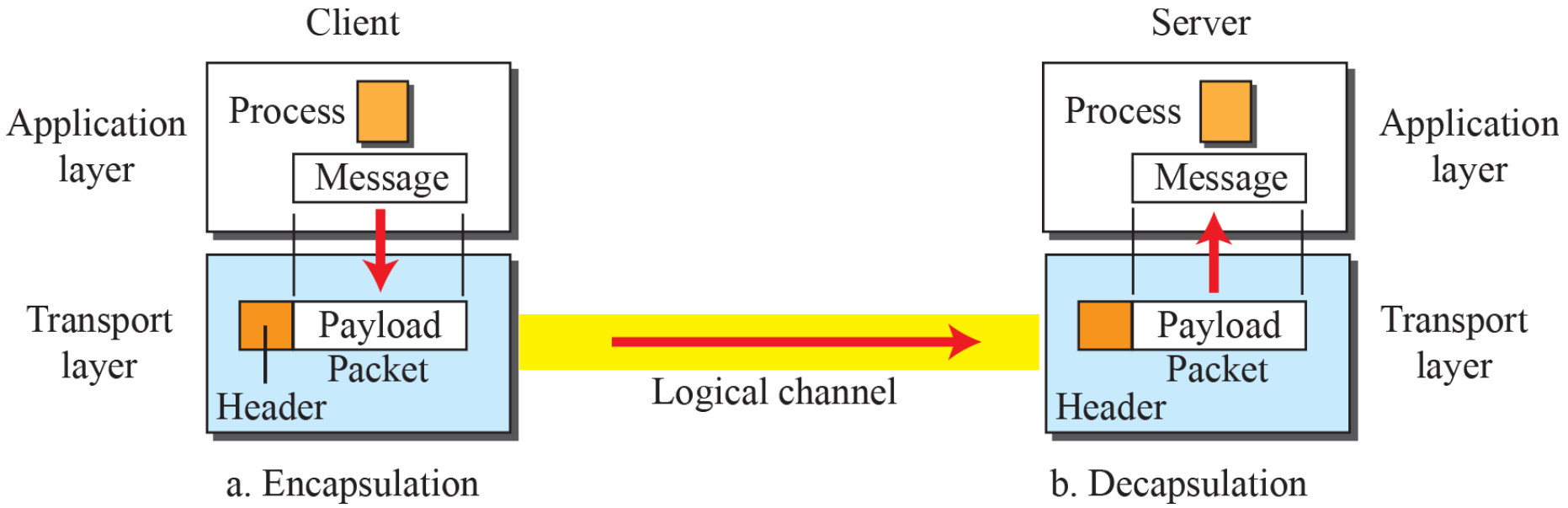


Figure 3.8: Multiplexing and demultiplexing

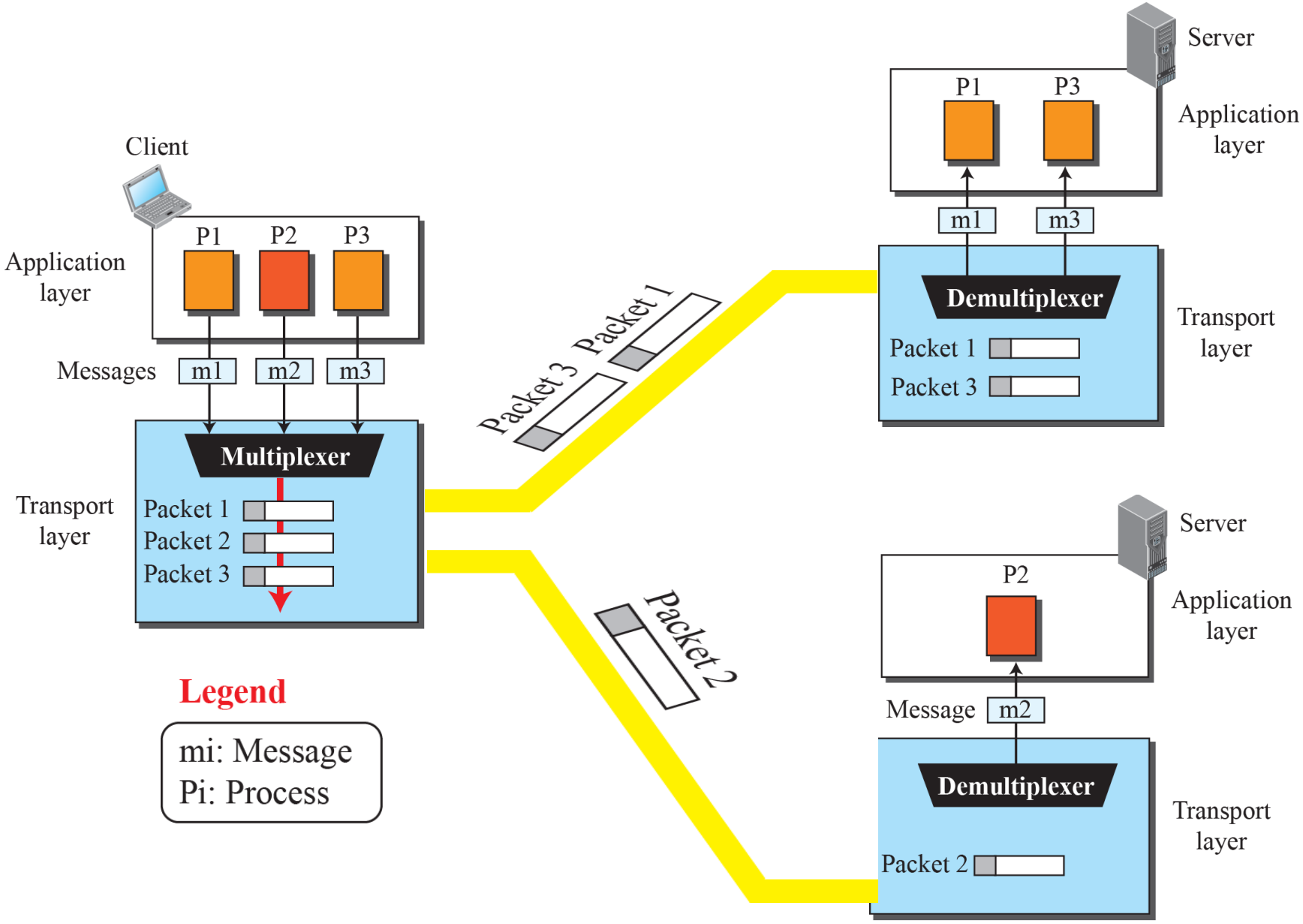
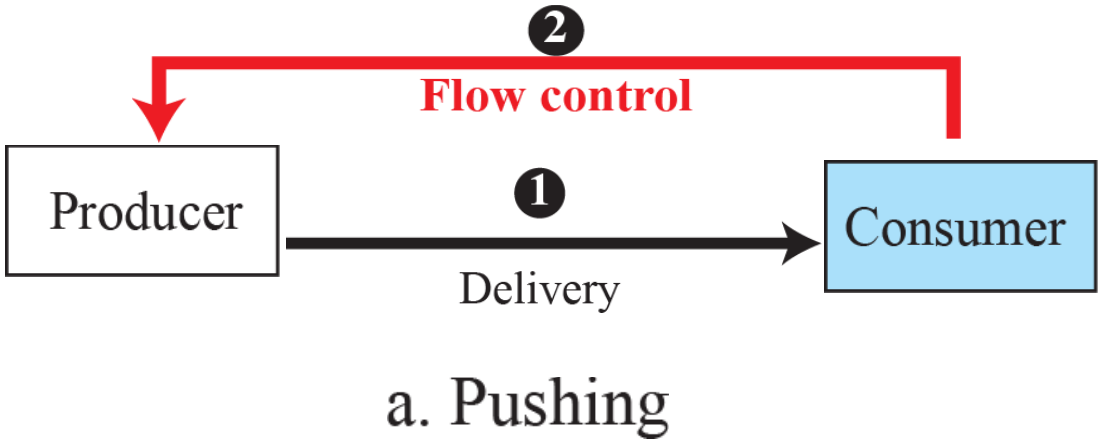
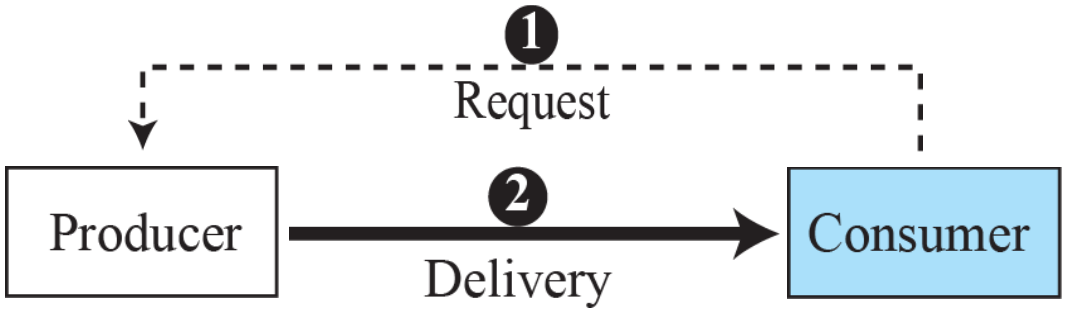


Figure 3.9: Flow Control(Pushing or pulling)

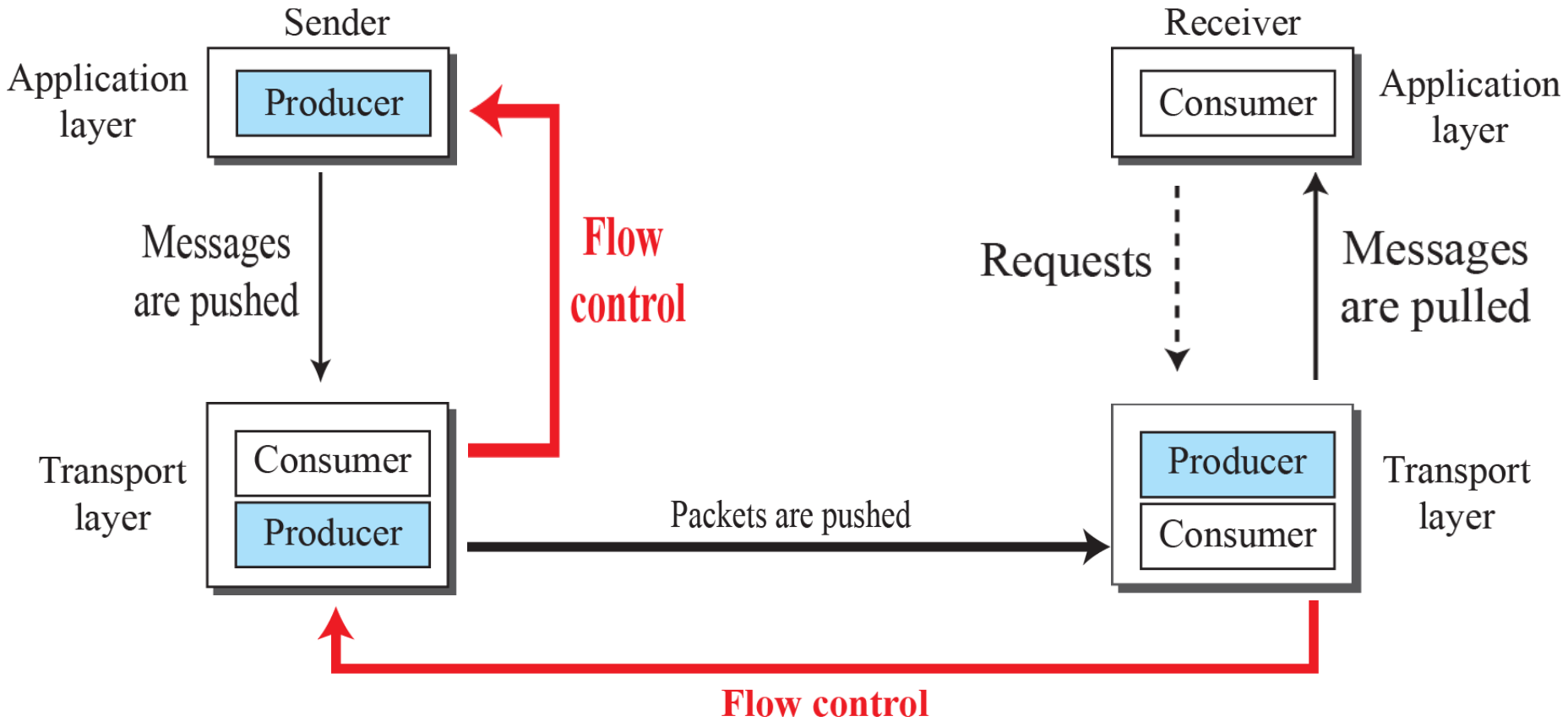


a. Pushing



b. Pulling

Figure 3.10: Flow control at the transport layer



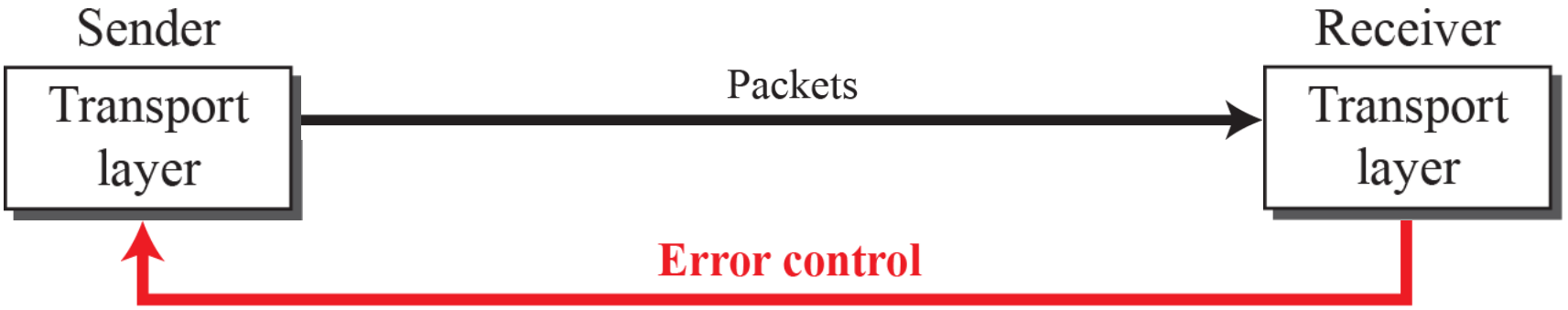
Example 3.2

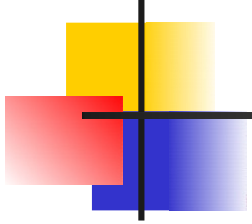
The above discussion requires that the consumers communicate with the producers on two occasions: when the buffer is full and when there are vacancies. If the two parties use a buffer with only one slot, the communication can be easier. Assume that each transport layer uses one single memory location 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.

Error Control

1. Detecting and discarding corrupted packets.
2. Keeping track of lost and discarded packets and resending them.
3. Recognizing duplicate packets and discarding them.
4. Buffering out of order packets until missing packet arrives.

Figure 3.11: Error control at the transport layer

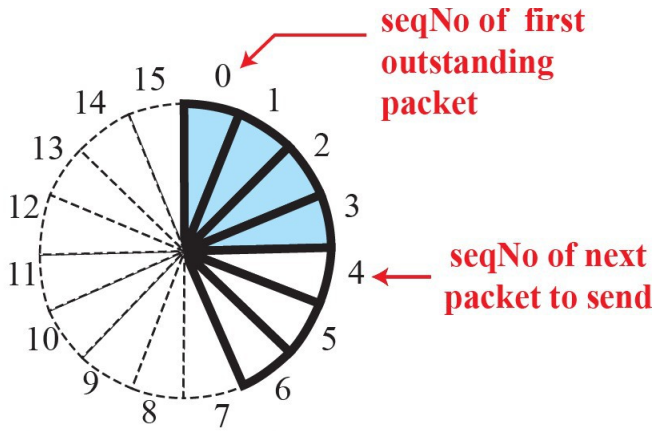




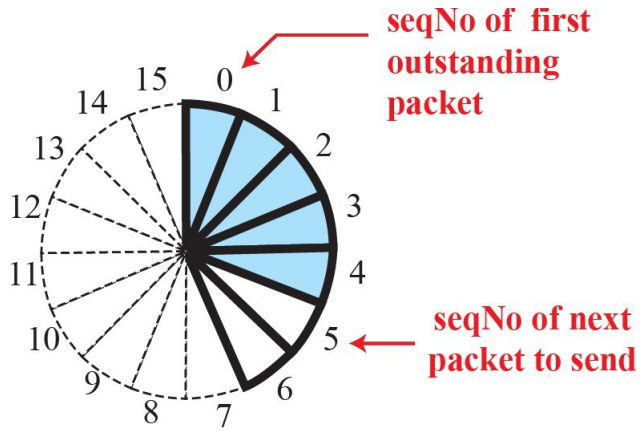
Note

For error control, the sequence numbers are modulo 2^m, where m is the size of the sequence number field in bits.

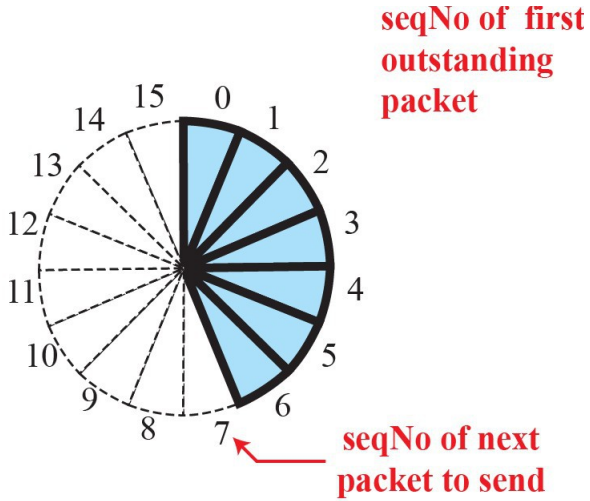
Figure 3.12: Sliding window in circular format



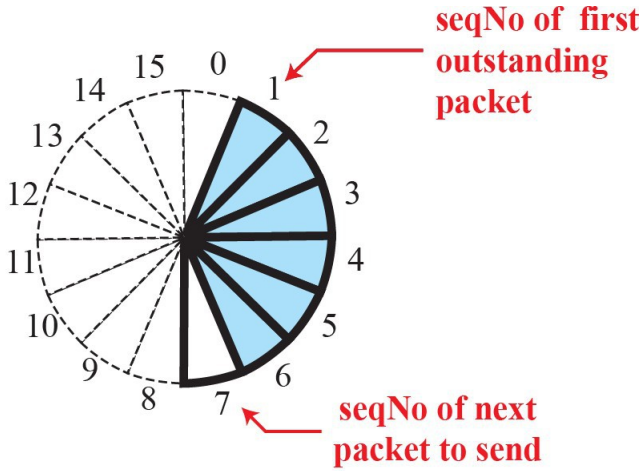
a. Four packets have been sent.



b. Five packets have been sent.

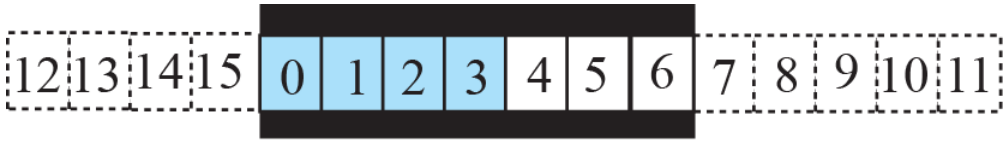


c. Seven packets have been sent; window is full.

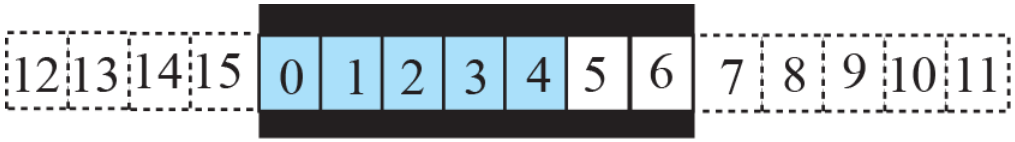


d. Packet 0 has been acknowledged; window slides.

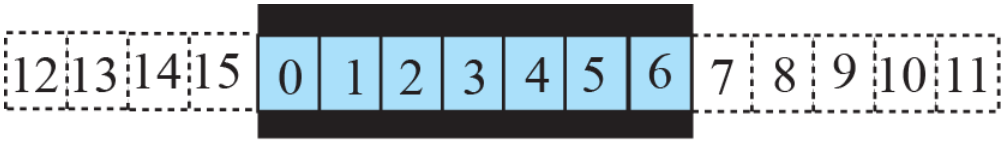
Figure 3.13: Sliding window in linear format



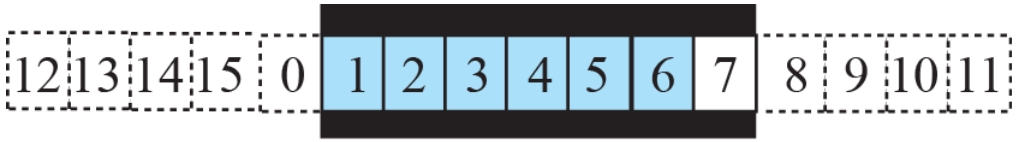
a. Four packets have been sent.



b. Five packets have been sent.



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



d. Packet 0 has been acknowledged;
window slides.

Congestion Control

- Congestion in a network occur if load on the network greater than the capacity of the network.
- Congestion control refers to the mechanisms and techniques that control the congestion and keep the load below the capacity.

Transport Layer

- Connection less- Interdependency between packets.
- Connection -oriented- Dependency.

Figure 3.14: Connectionless service

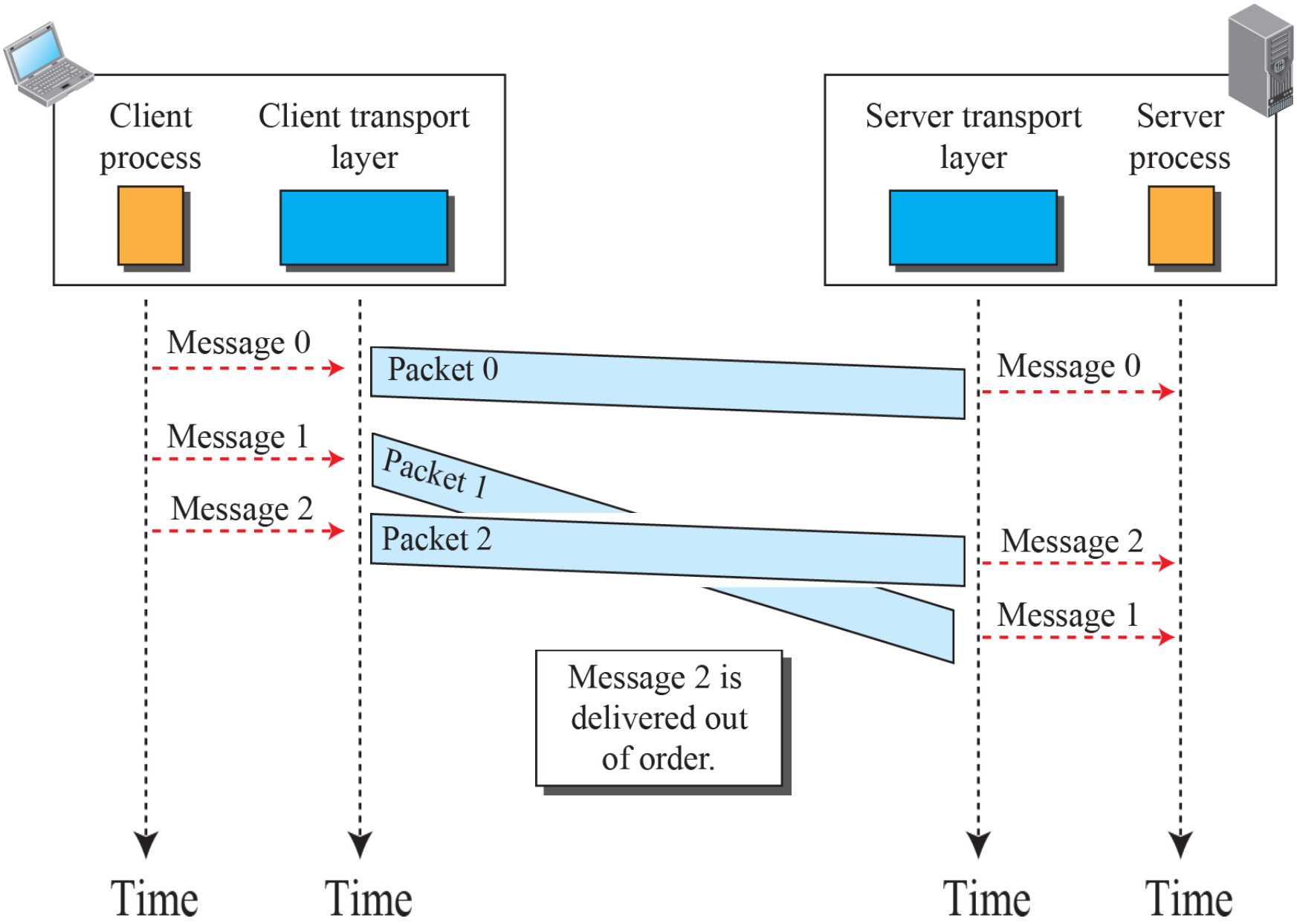


Figure 3.15: Connection-oriented service

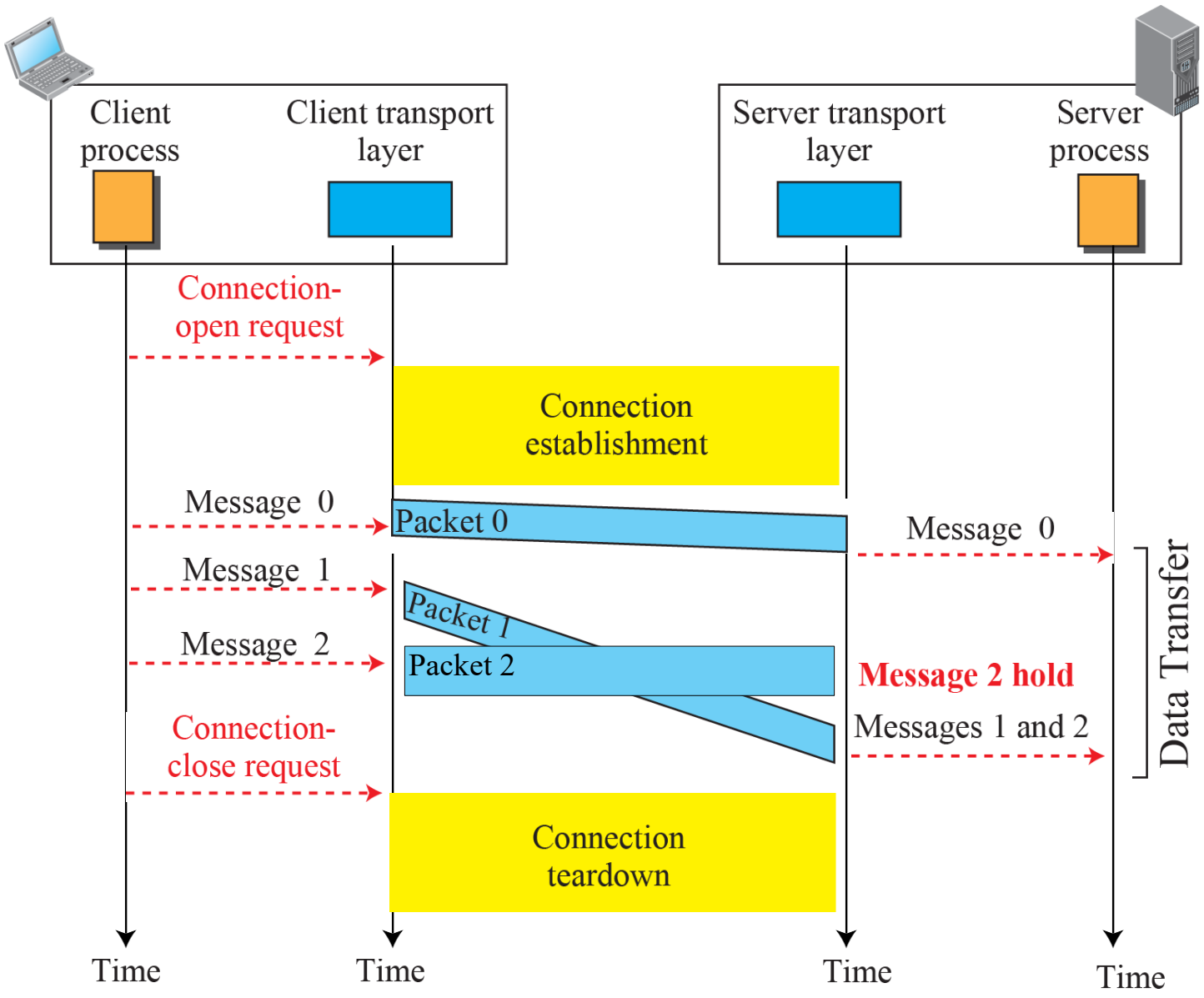
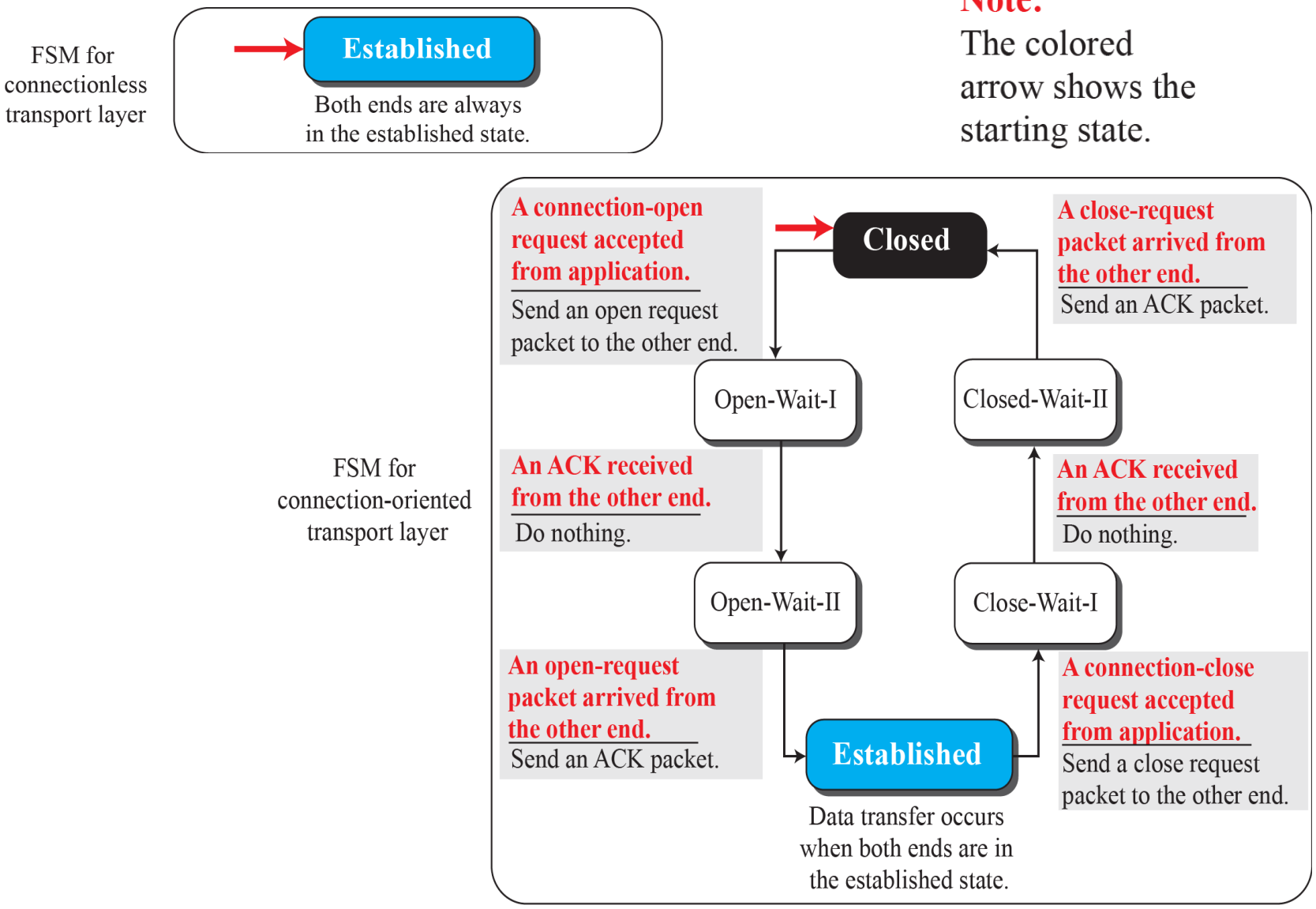


Figure 3.16: Connectionless and connection-oriented service represented as FSMs

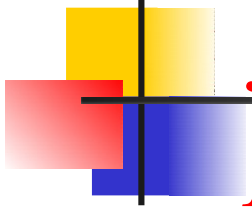


3-2 TRANSPORT-LAYER PROTOCOLS

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.

Topics Discussed in the Section

- Simple Protocol
- Stop-and-Wait Protocol
- Go-Back-*N* Protocol
- Selective-Repeat Protocol
- Bidirectional Protocols: Piggybacking



3.2.1 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 3.17: Simple protocol

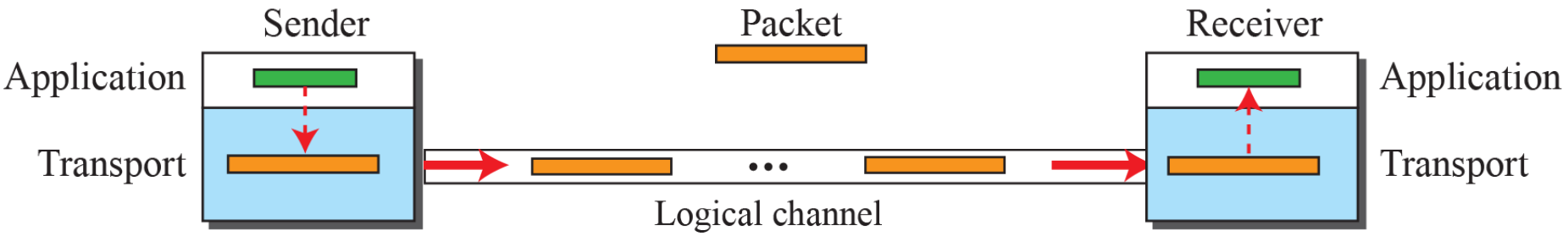
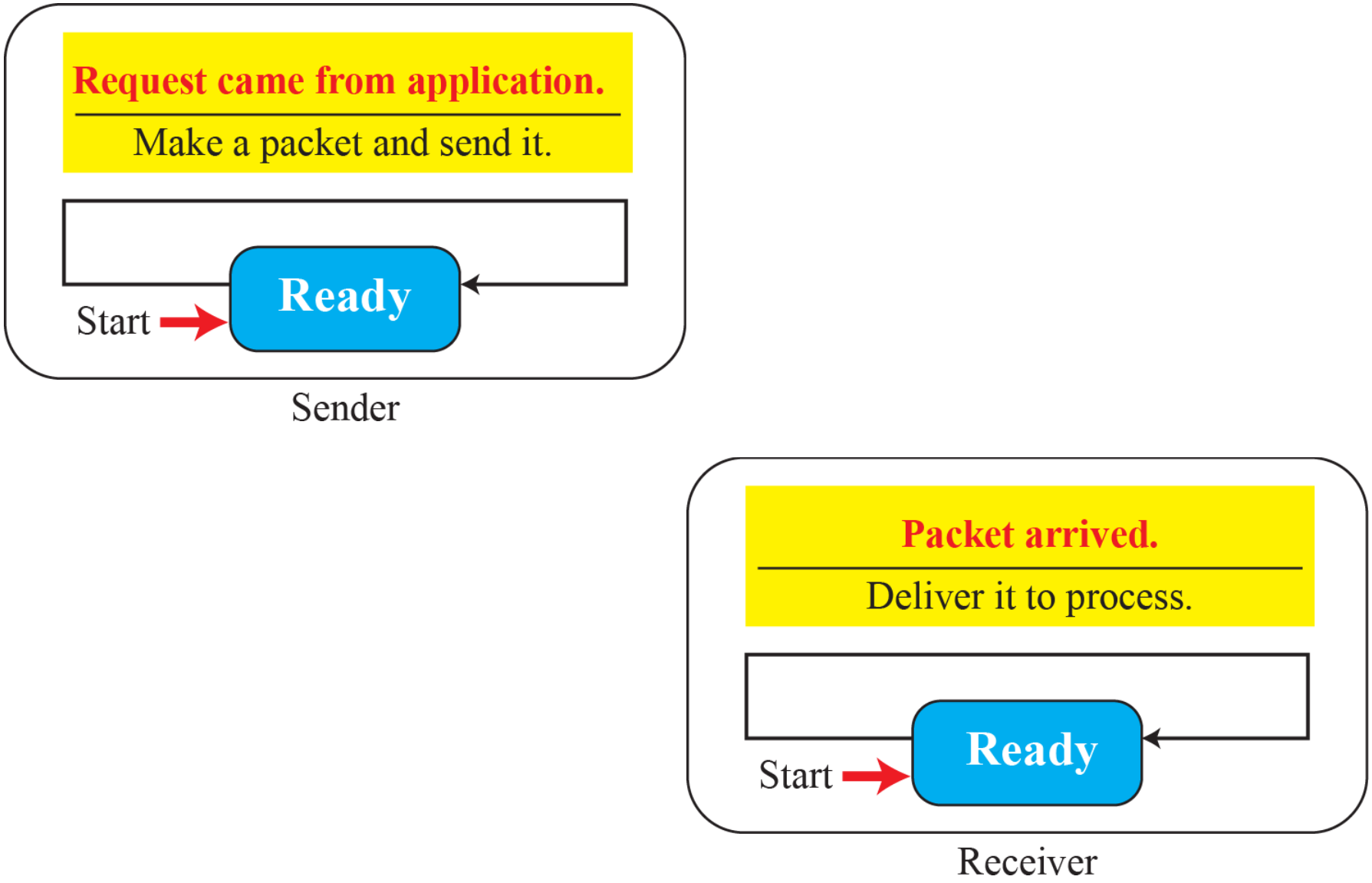


Figure 3.18: FSMs for the simple protocol



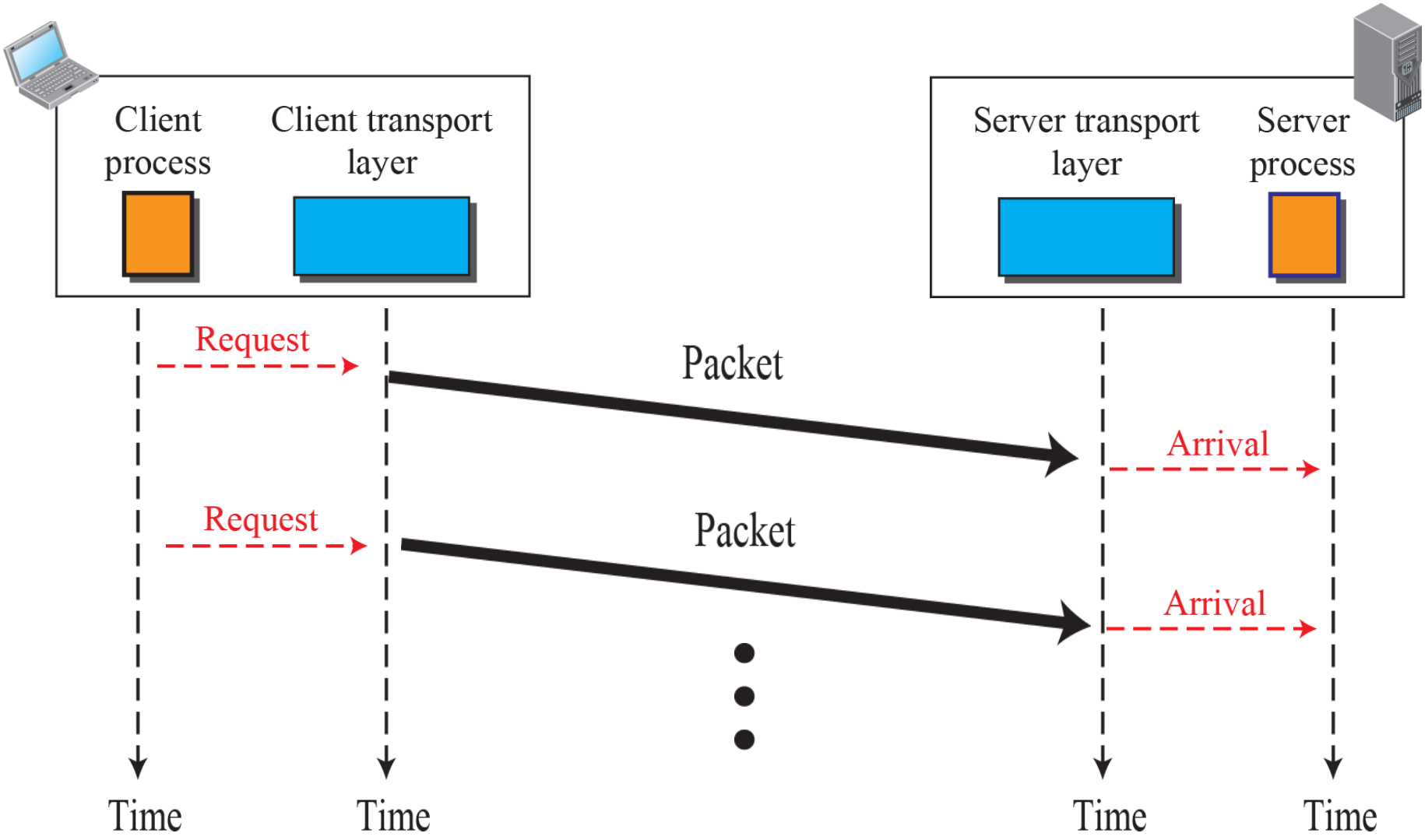
Note

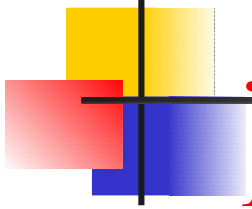
The simple protocol is a connectionless protocol that provides neither flow nor error control.

Example 3.3

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 3.19: Flow diagram for Example 3.3

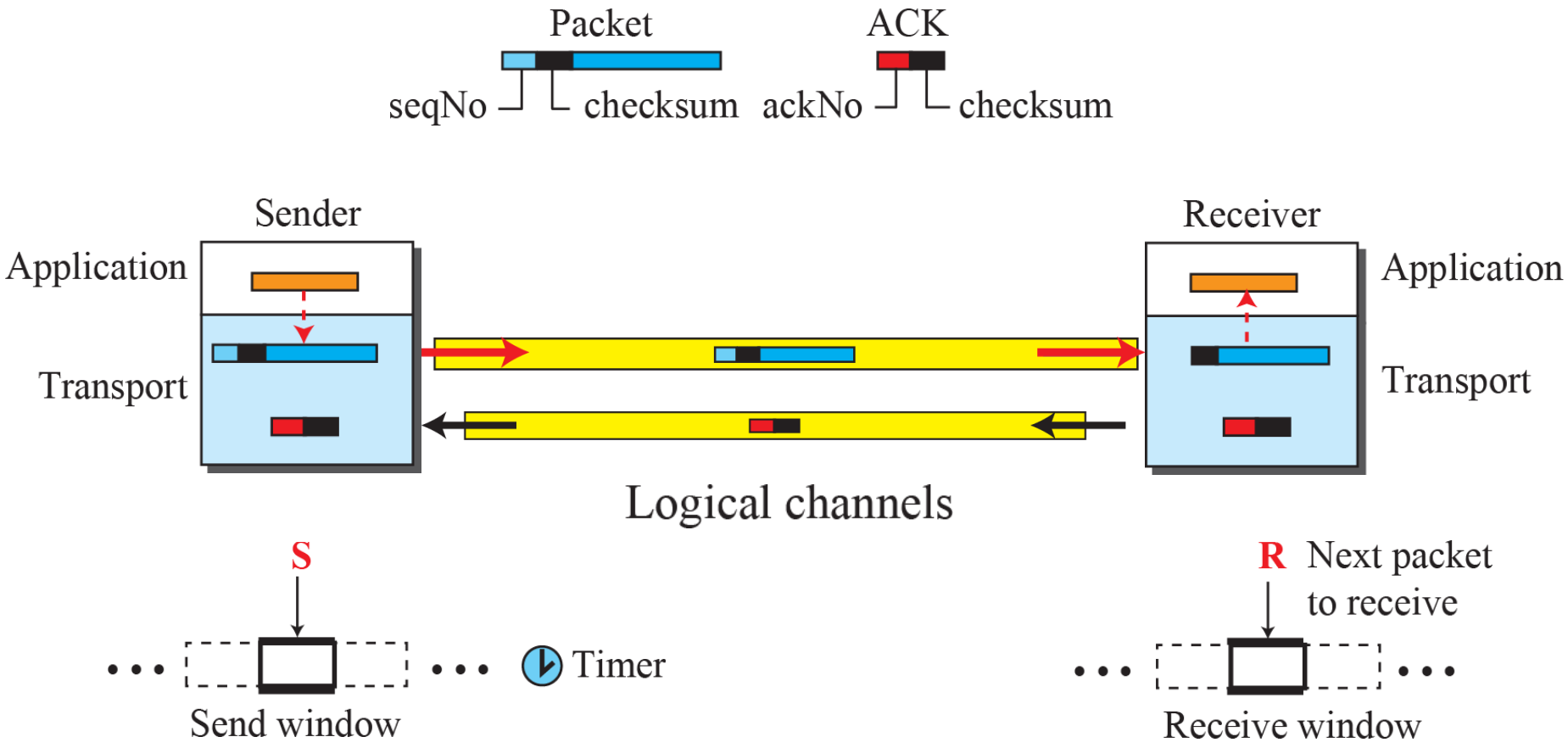




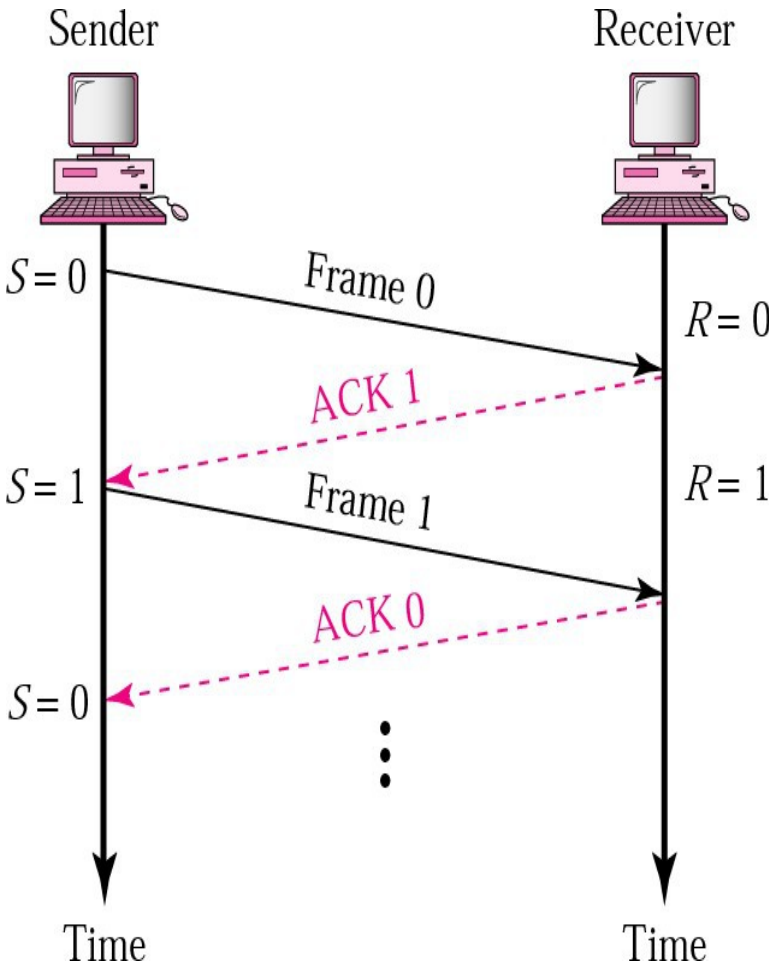
3.2.2 Stop-and-Wait Protocol

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 1. 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.

Figure 3.20: Stop-and-Wait protocol



Stop-and-Wait



- Sender keeps a copy of the last frame until it receives an acknowledgement.
- For identification, both data frames and acknowledgements (ACK) frames are numbered alternatively 0 and 1.
- Sender has a control variable (S) that holds the number of the recently sent frame. (0 or 1)
- Receiver has a control variable(R) that holds the number of the next frame expected (0 or 1).
- Sender starts a timer when it sends a frame. If an ACK is not received within a allocated time period, the sender assumes that the frame was lost or damaged and resends it
- Receiver send only positive ACK if the frame is intact.
- ACK number always defines the number of the next expected frame

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.

Note

In the Stop-and-Wait protocol, we can use a 1-bit field to number the packets. The sequence numbers are based on modulo-2 arithmetic.

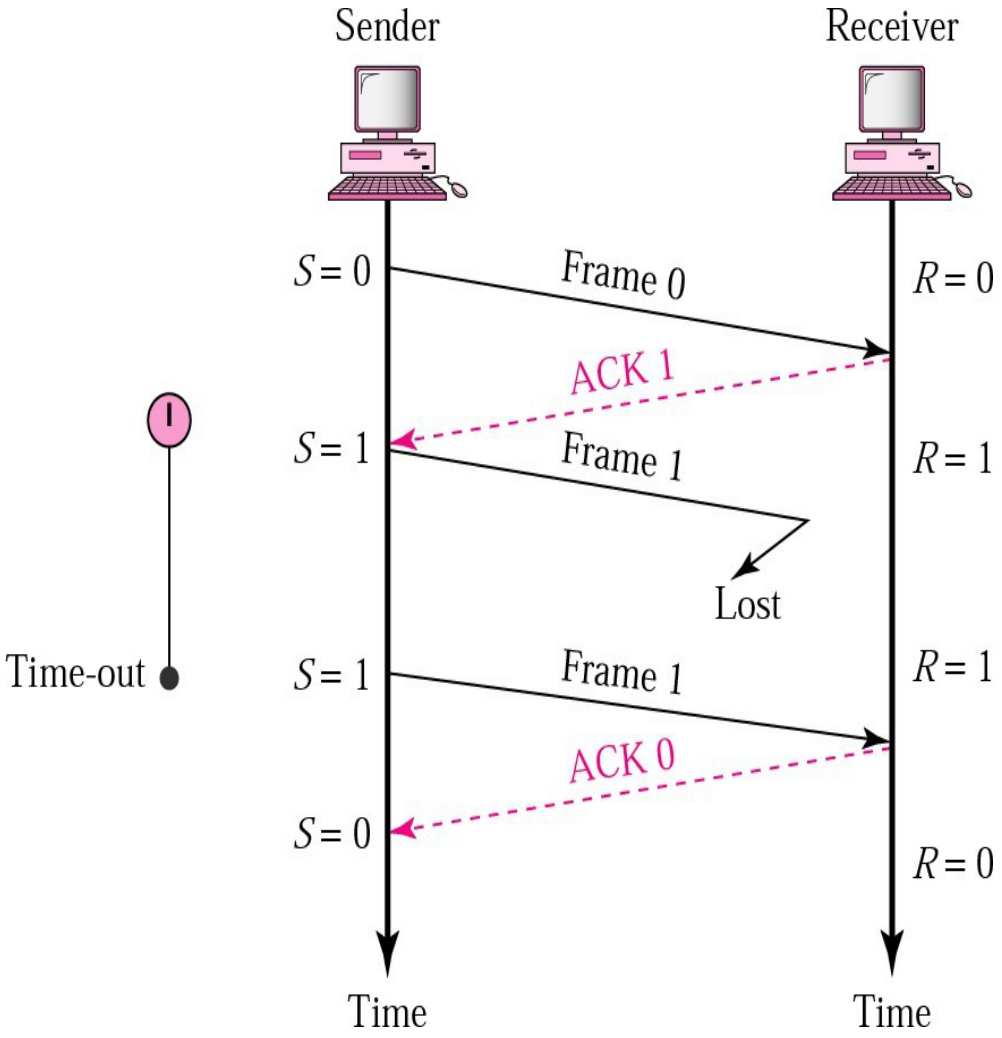
Note

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

Note

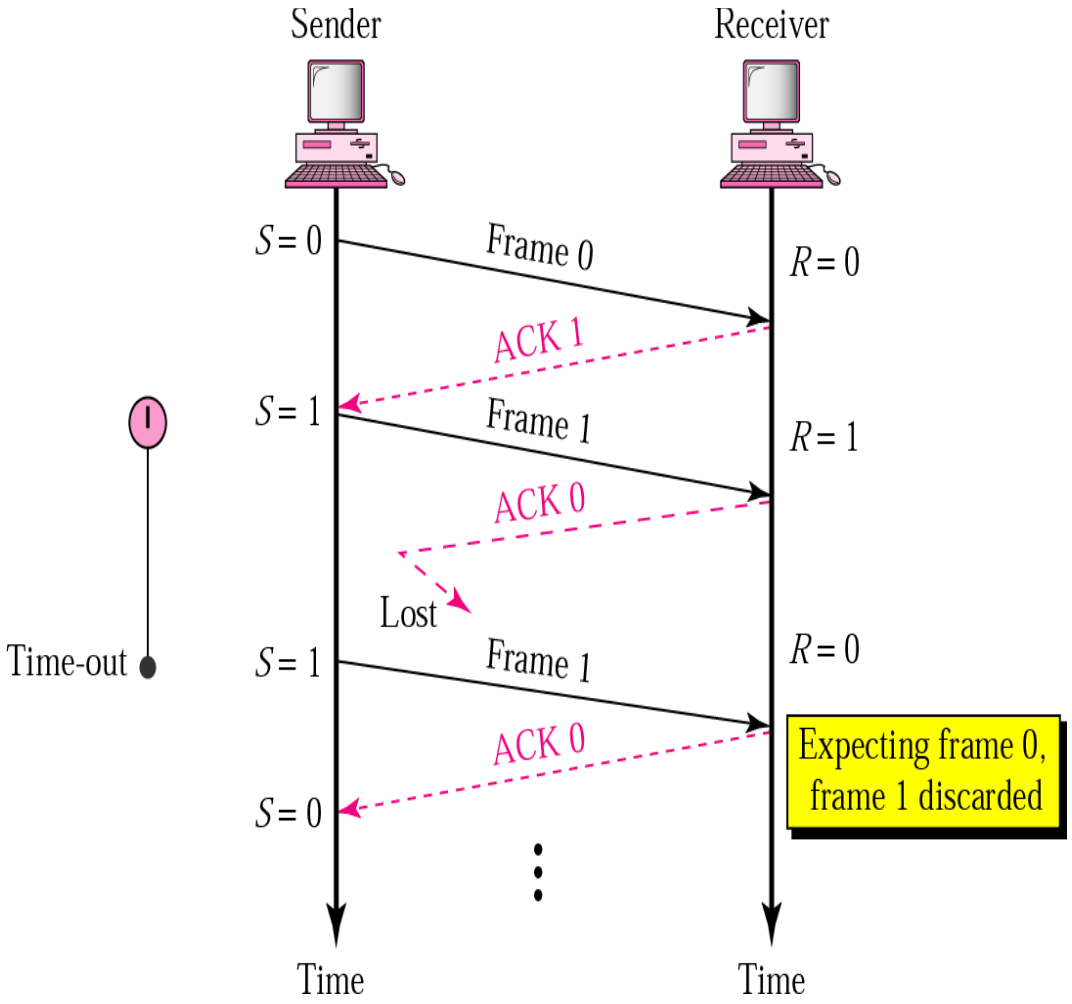
All calculation in the Stop-and-Wait protocol is in modulo 2.

Stop-and-Wait , lost frame



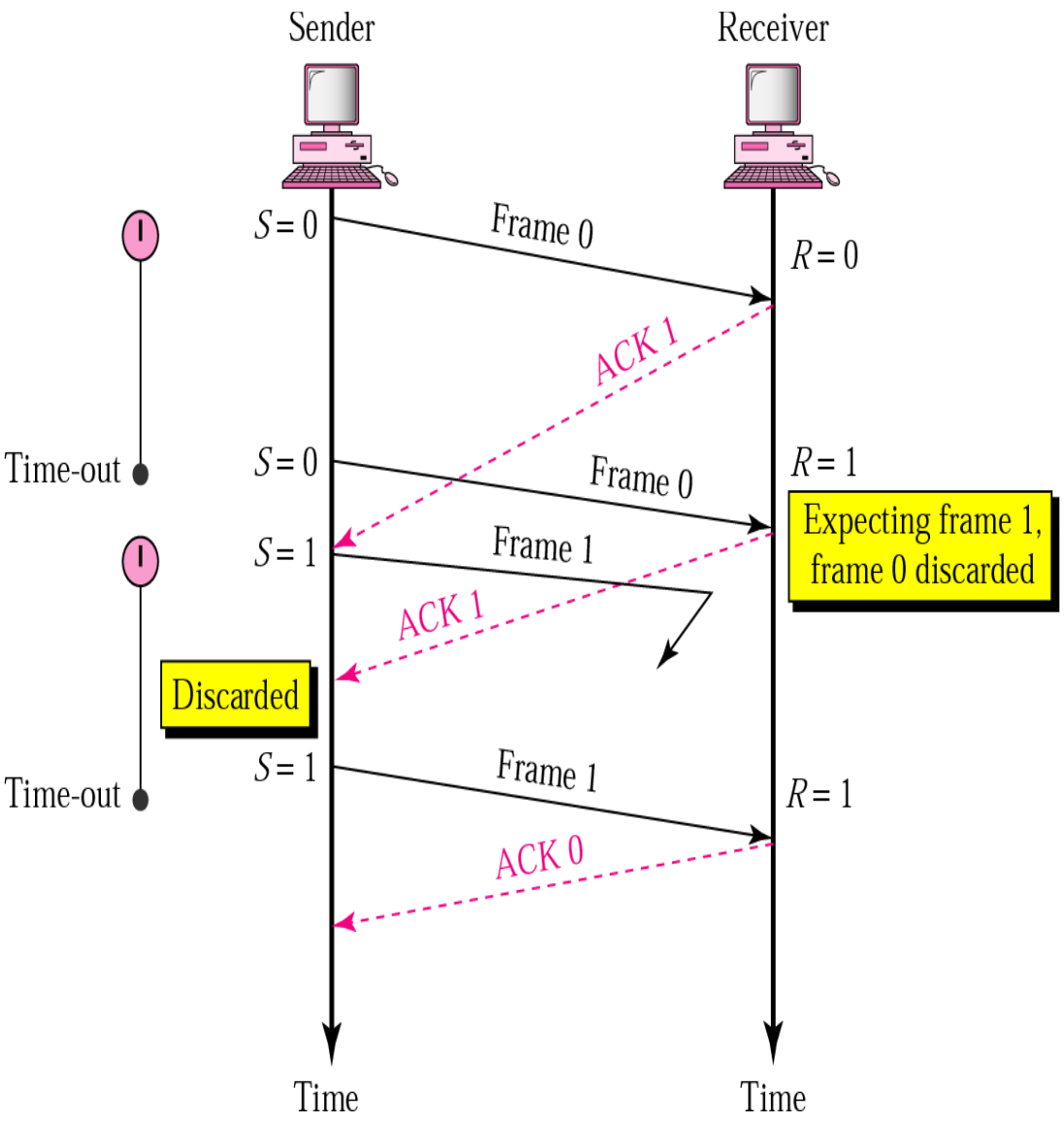
When a receiver receives a damaged frame, it discards it and keeps its value of R. After the timer at the sender expires, another copy of frame 1 is sent.

Stop-and-Wait, lost ACK frame



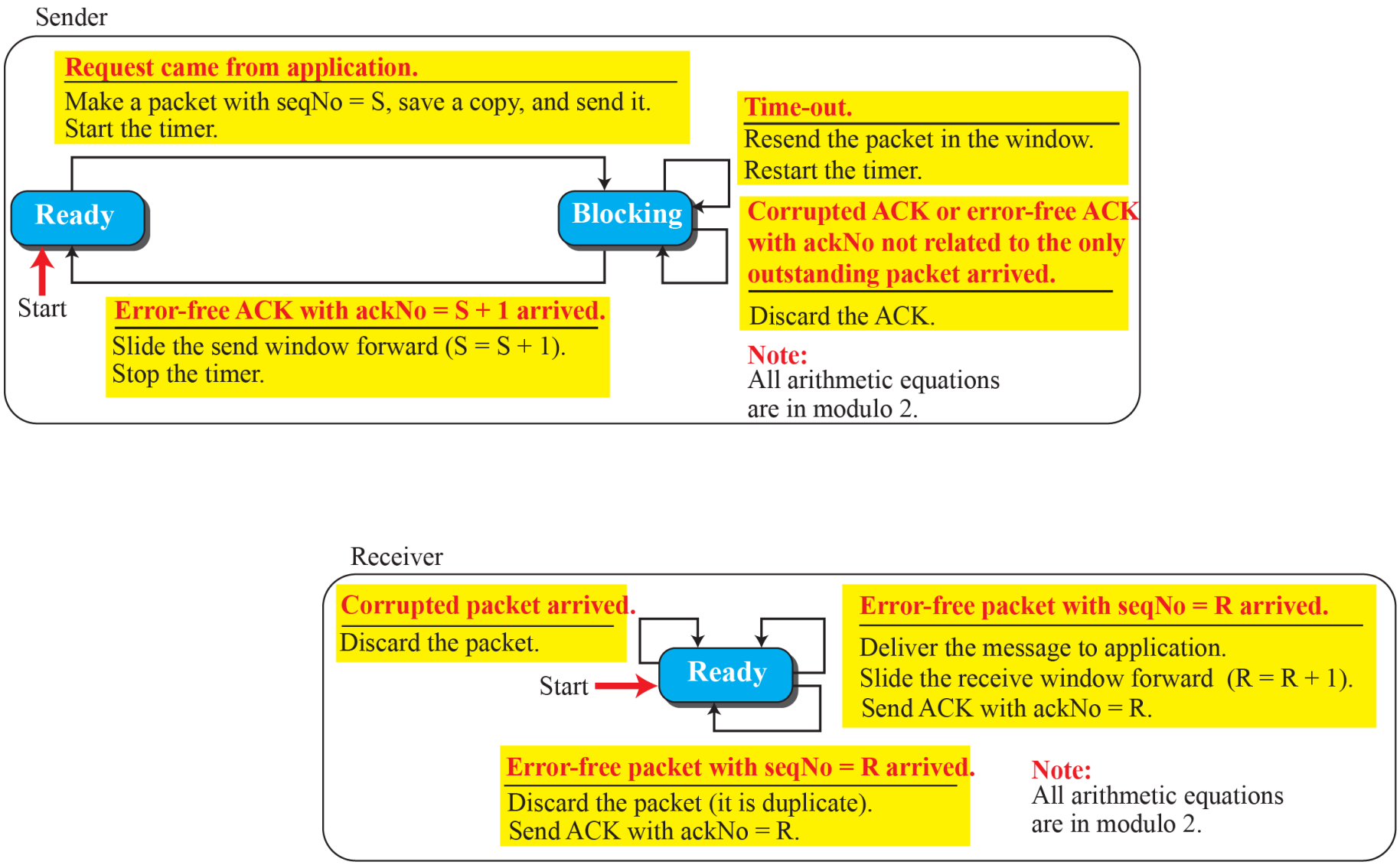
- If the sender receives a damaged ACK, it discards it.
- When the timer of the sender expires, the sender retransmits frame 1.
- Receiver has already received frame 1 and expecting to receive frame 0 (R=0). Therefore it discards the second copy of frame 1.

Stop-and-Wait, delayed ACK frame



- The ACK can be delayed at the receiver or due to some problem
- It is received after the timer for frame 0 has expired.
- Sender retransmitted a copy of frame 0. However, R =1 means receiver expects to see frame 1. Receiver discards the duplicate frame 0.
- Sender receives 2 ACKs, it discards the second ACK.

Figure 3.21: FSM for the Stop-and-Wait protocol



Disadvantage of Stop-and-Wait

- In stop-and-wait, at any point in time, there is only one frame that is sent and waiting to be acknowledged.
- This is not a good use of transmission medium.
- To improve efficiency, multiple frames should be in transition while waiting for ACK.
- Two protocol use the above concept,

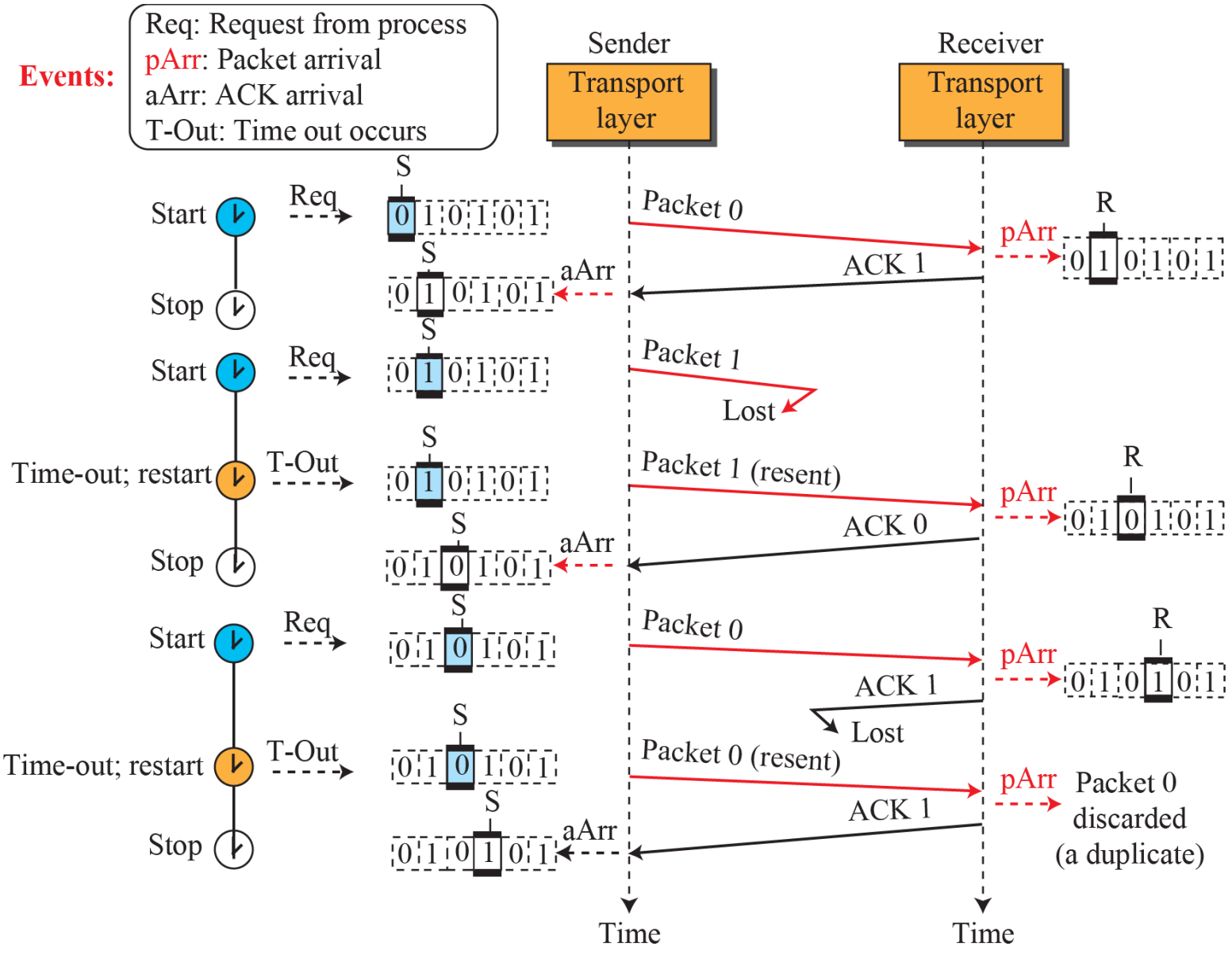
—Go-Back-N

—Selective Repeat

Example 3.4

Figure 3.22 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 3.22: Flow diagram for Example 3.4



Example 3.5

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

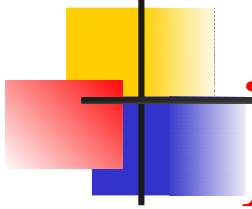
The bandwidth-delay product is $(1 \times 106) \times (20 \times 10{-}3) = 20{,}000$ bits. The system can send 20,000 bits during the time it takes for the data to go from the sender to the receiver and the acknowledgment to come back. However, the system sends only 1,000 bits. The link utilization is only 1,000/20,000, or 5 percent.

Example 3.6

What is the utilization percentage of the link in Example 3.5 if we have a protocol that can send up to 15 packets before stopping and worrying about the acknowledgments?

Solution

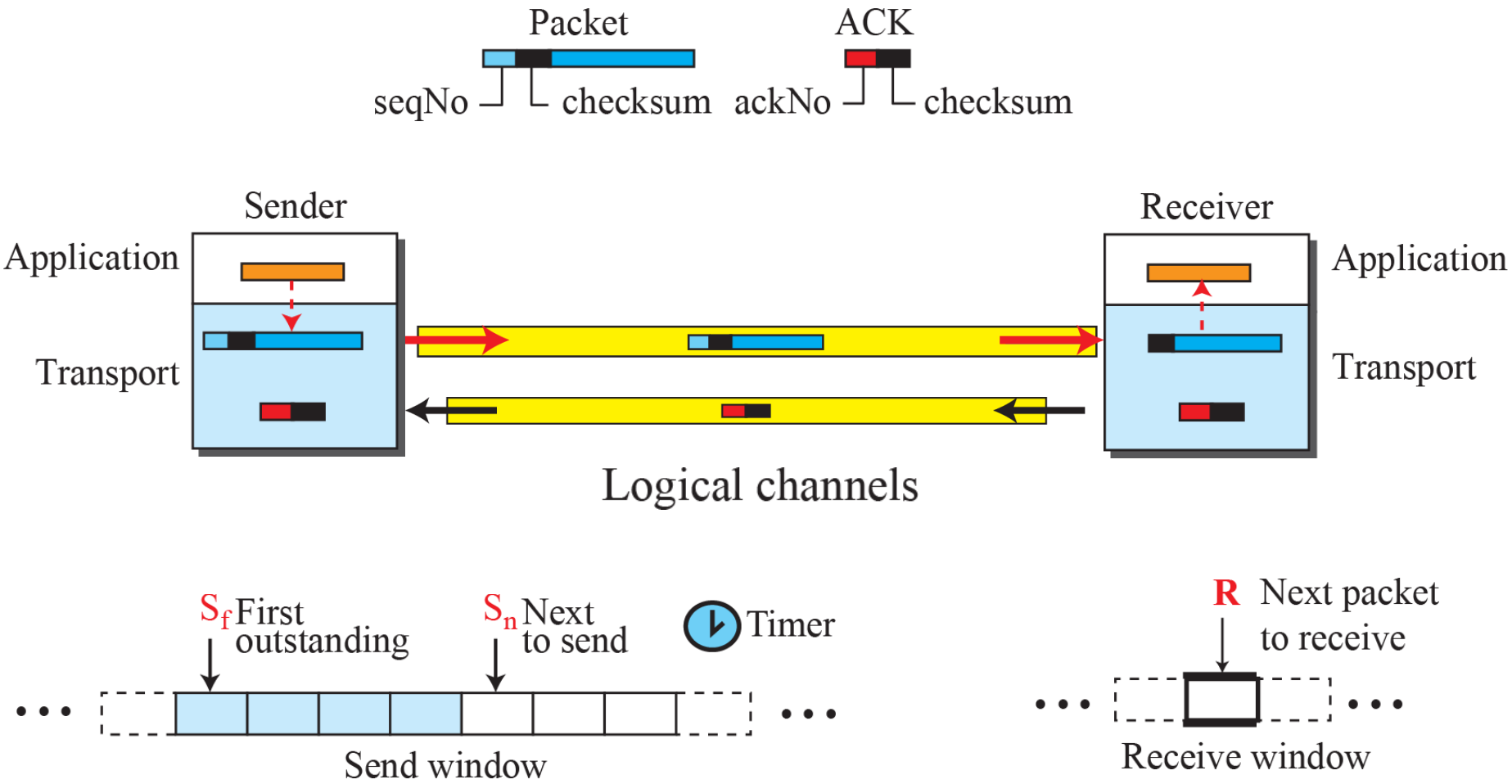
The bandwidth-delay product is still 20,000 bits. The system can send up to 15 packets or 15,000 bits during a round trip. This means the utilization is 15,000/20,000, or 75 percent. Of course, if there are damaged packets, the utilization percentage is much less because packets have to be resent.



3.2.3 Go-Back-N Protocol

To improve the efficiency of transmission multiple packets must be in transition 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 3.23: Go-Back-N protocol



Go-Back-N

- We can send up to W frames before worrying about ACKs.
- We keep a copy of these frames until the ACKs arrive.
- This procedure requires additional features to be added to Stop-and-Wait .

Note

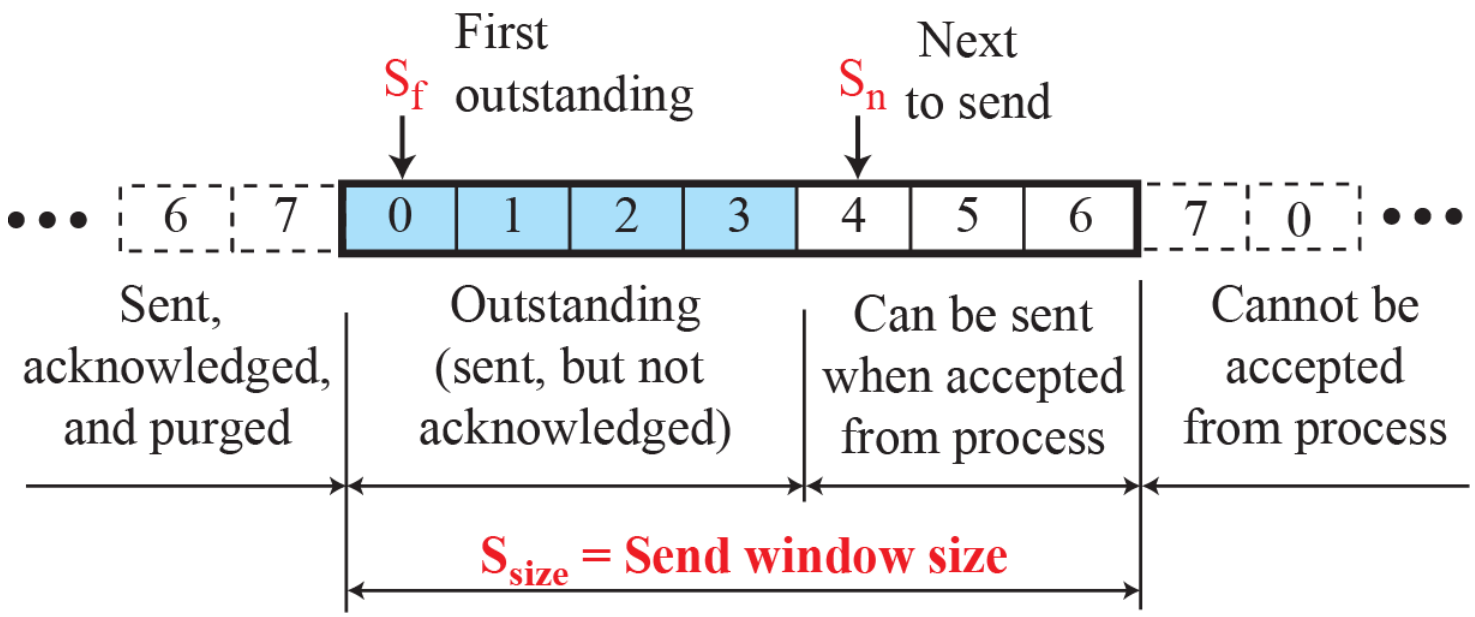
In the Go-Back-N Protocol, the sequence numbers are modulo 2^m, where m is the size of the sequence number field in bits.

Sequence Numbers

- Frames from a sender are numbered sequentially.
- We need to set a limit since we need to include the sequence number of each frame in the header.
- If the header of the frame allows m bits for sequence number, the sequence numbers range from 0 to $2^m - 1$. for m = 3, sequence numbers are: 1, 2, 3, 4, 5, 6, 7.
- We can repeat the sequence number.
- Sequence numbers are:

0, 1, 2, 3, 4, 5, 6, 7, 0, 1, 2, 3, 4, 5, 6, 7, 0, 1, …

Figure 3.24: Send window for Go-Back-N



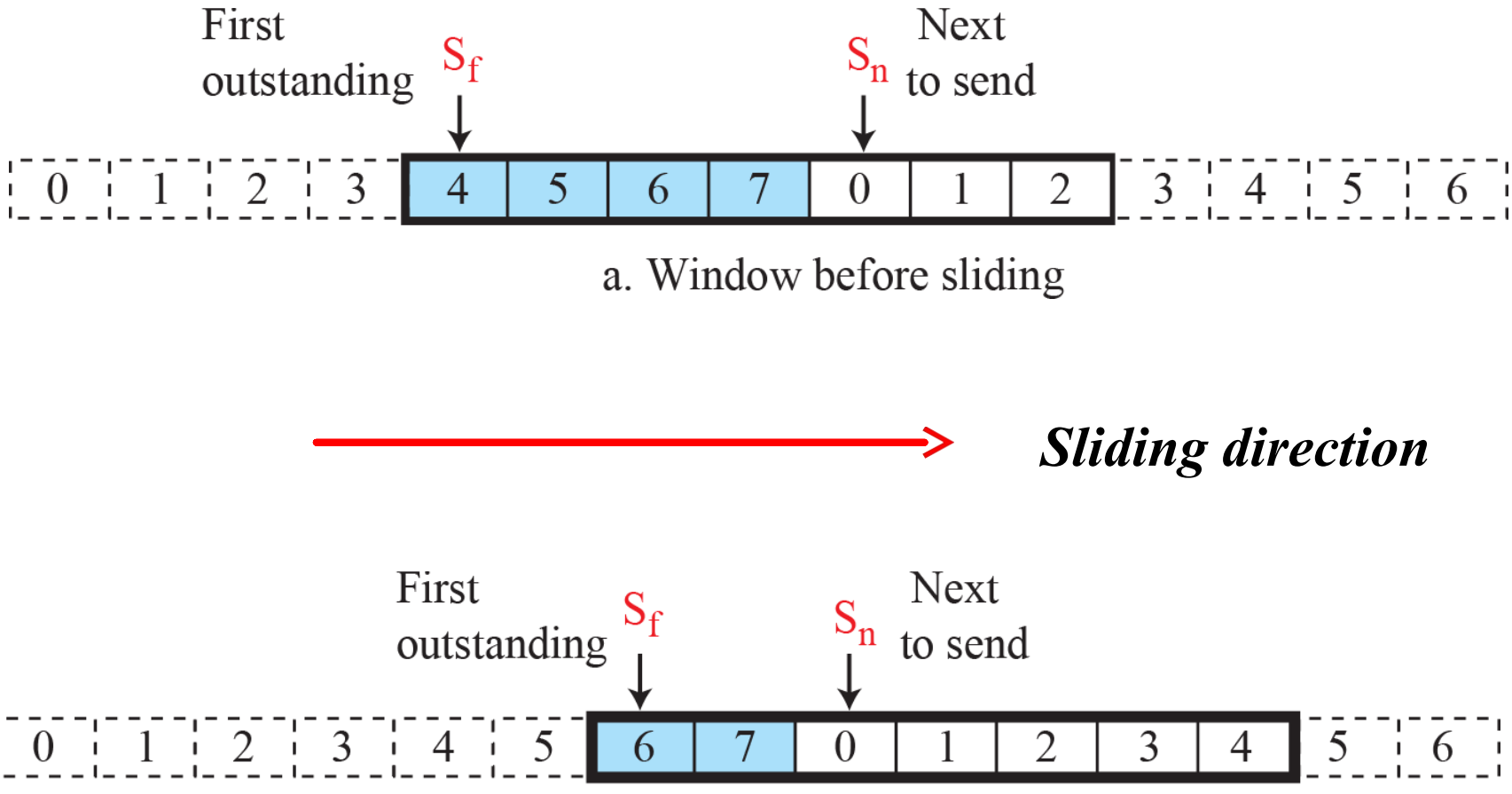
Note

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

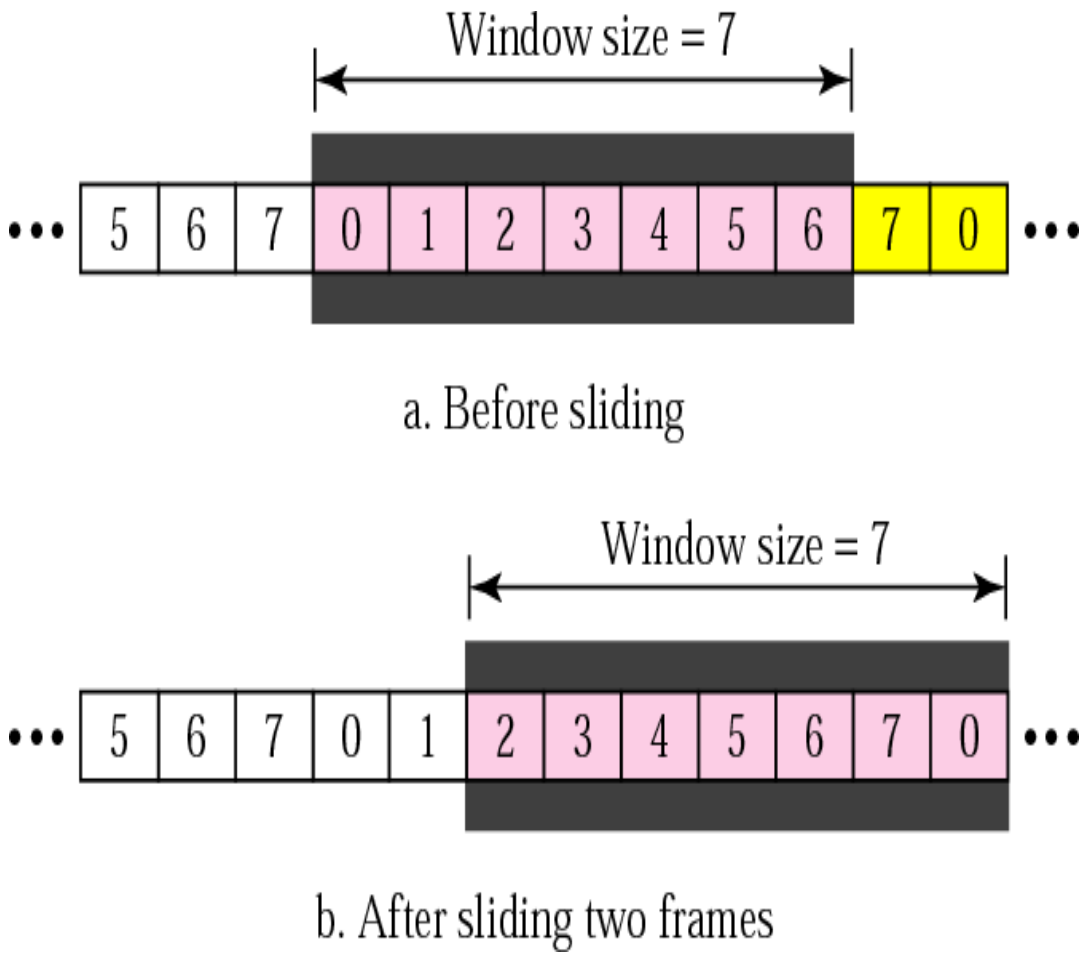
The send window can slide one or more slots when an error-free ACK with ackNo between S_f and S_n (in modular arithmetic) arrives.

Figure 3.25: Sliding the send window



Sender Sliding Window

- At the sending site, to hold the outstanding frames until they are acknowledged, we use the concept of a window.
- The size of the window is at most $2^m - 1$ where m is the number of bits for the sequence number.
- Size of the window can be variable, e.g. TCP.
- The window slides to include new unsent frames when the correct ACKs are received

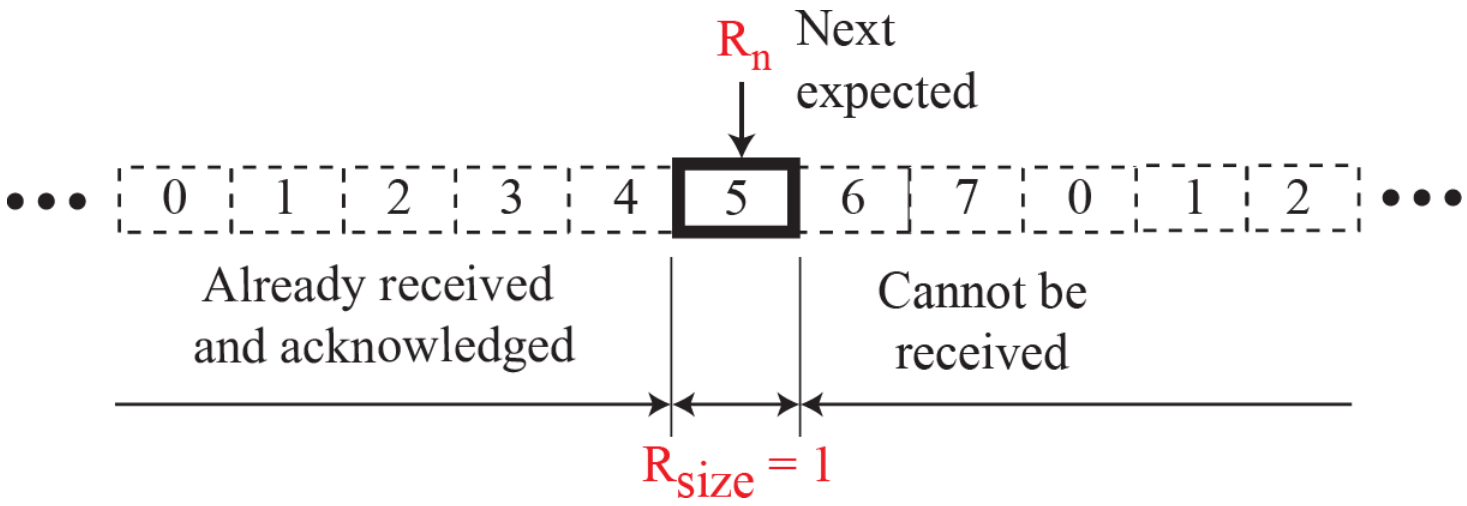


Note

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

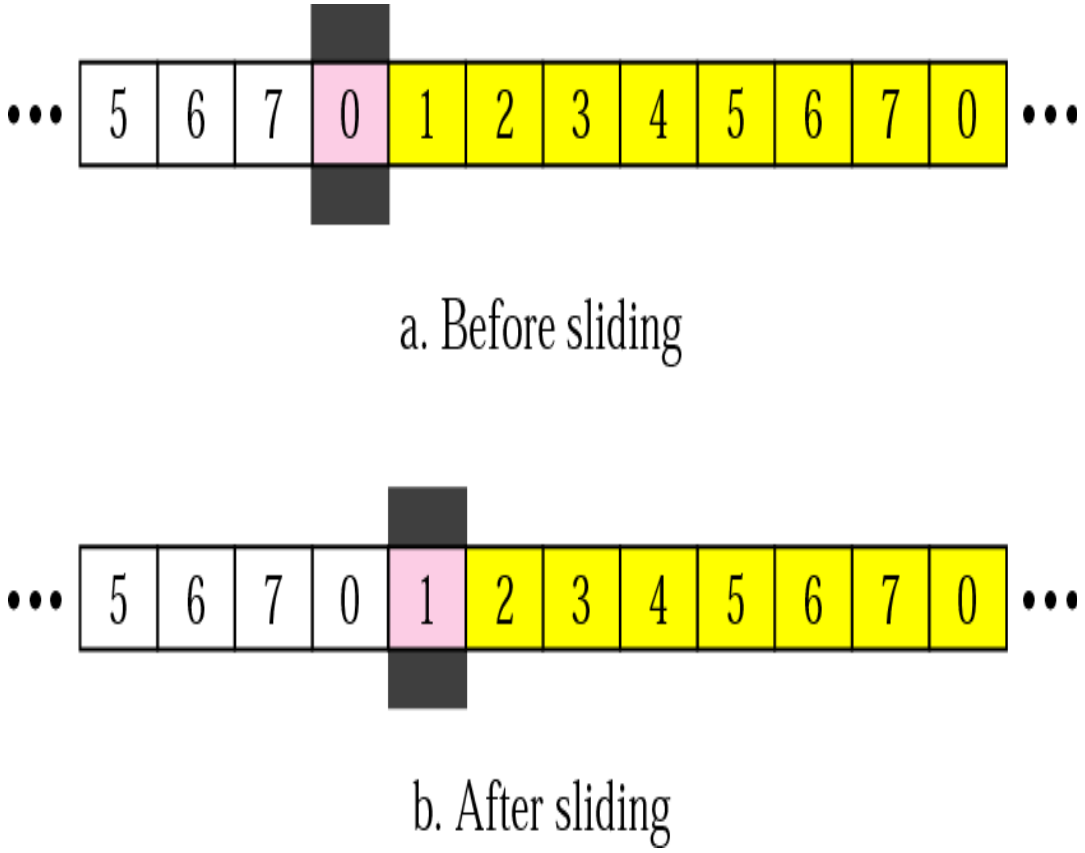
The window slides when a correct packet has arrived; sliding occurs one slot at a time.

Figure 3.26: Receive window for Go-Back-N



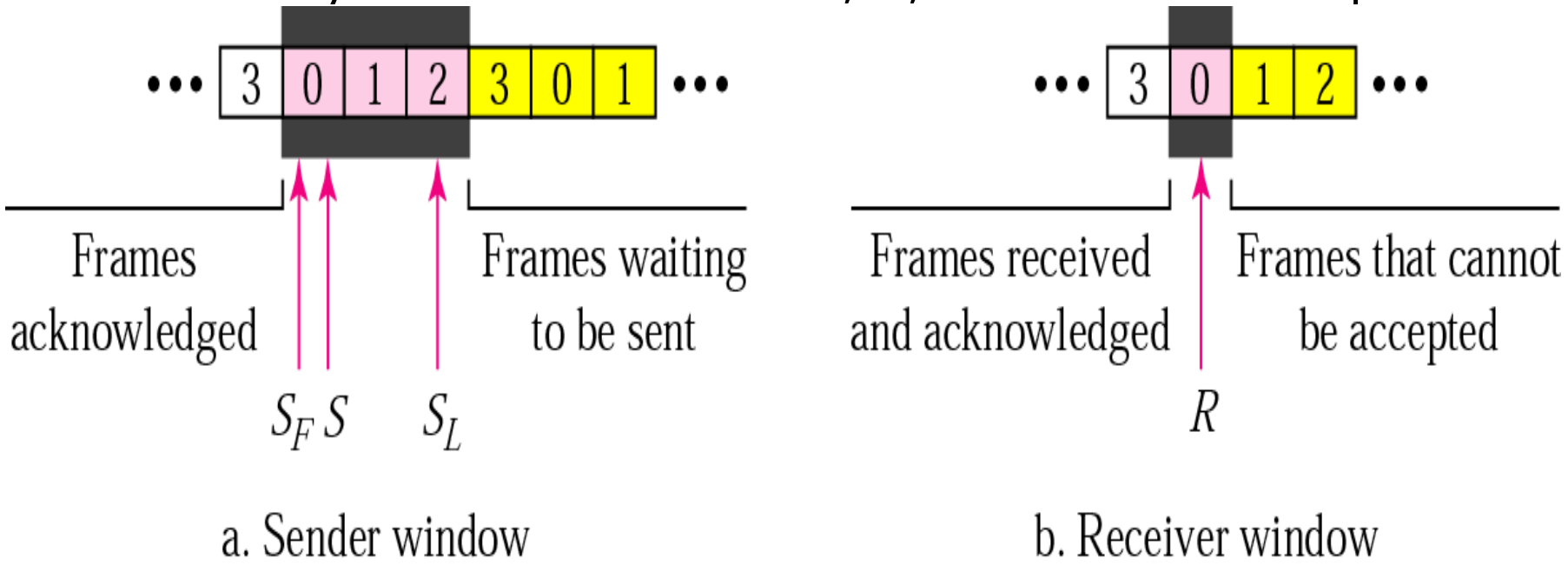
Receiver Sliding Window

- Size of the window at the receiving site is always 1 in this protocol.
- Receiver is always looking for a specific frame to arrive in a specific order.
- Any frame arriving out of order is discarded and needs to be resent.
- Receiver window slides as shown in fig. Receiver is waiting for frame 0 in part a.



Control Variables

- Sender has 3 variables: S, S_F, and S_L
- S holds the sequence number of recently sent frame
- S_F holds the sequence number of the first frame
- S_L holds the sequence number of the last frame
- Receiver only has the one variable, R, that holds the sequence



Note

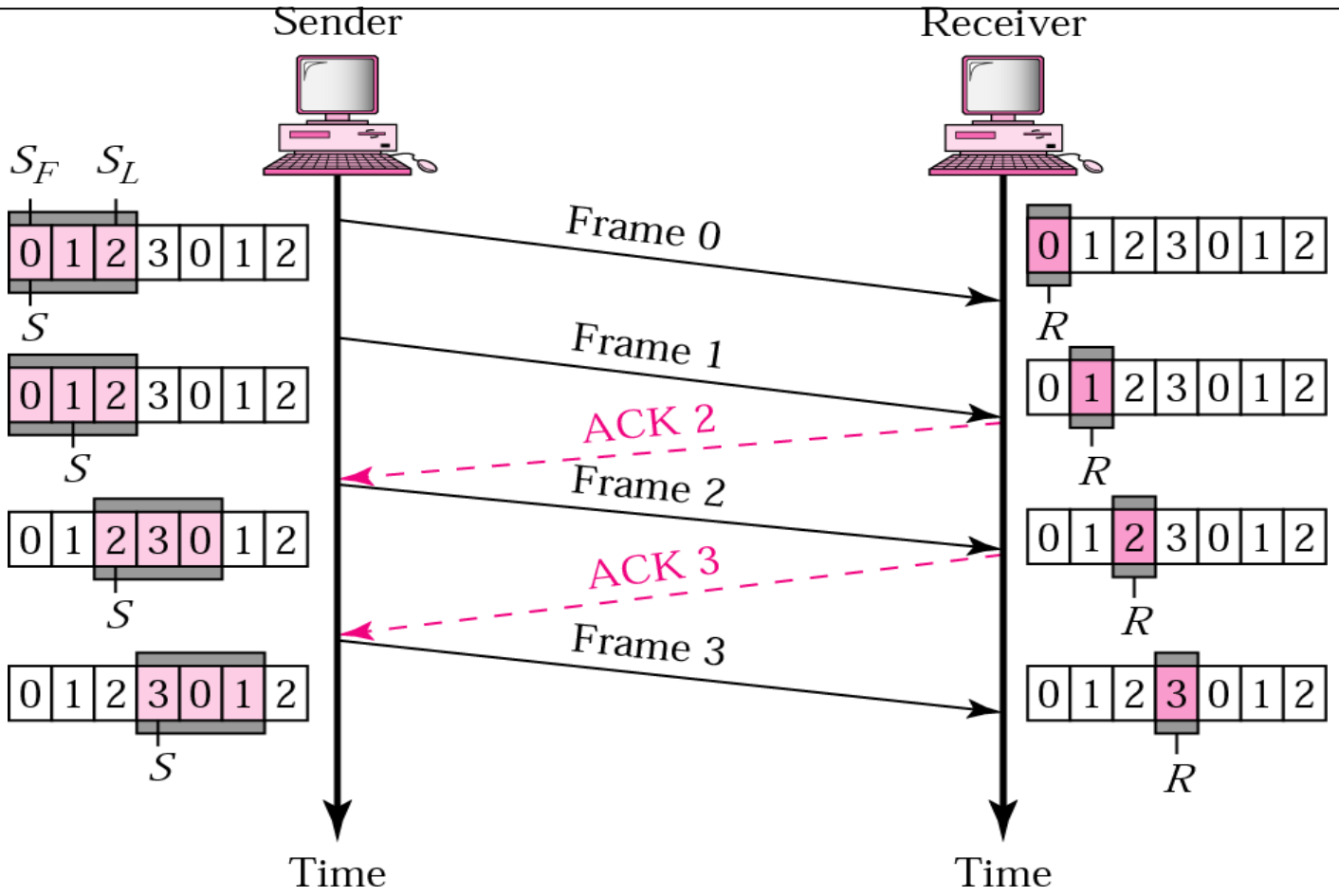
In the Go-Back-N protocol, the acknowledgment number is cumulative and defines the sequence number of the next packet expected to arrive.

Acknowledgement

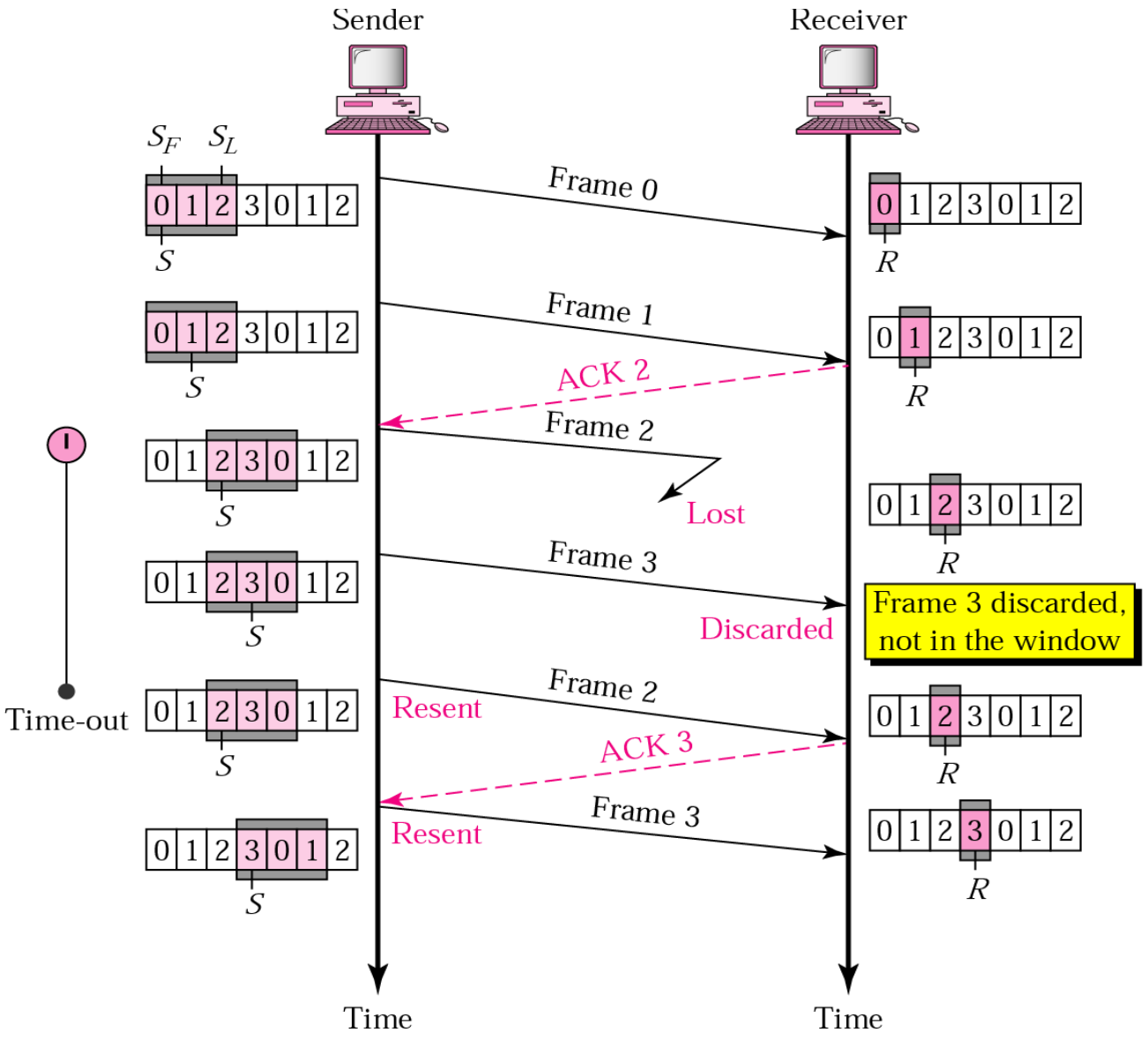
- Receiver sends positive ACK if a frame arrived safe and in order.
- If the frames are damaged/out of order, receiver is silent and discard all subsequent frames until it receives the one it is expecting.
- The silence of the receiver causes the timer of the unacknowledged frame to expire.
- Then the sender resends all frames, beginning with the one with the expired timer.
- For example, suppose the sender has sent frame 6, but the timer for frame 3 expires (i.e. frame 3 has not been acknowledged), then the sender goes back and sends frames 3, 4, 5, 6 again. Thus it is called Go-Back-N.
- The receiver does not have to acknowledge each frame received, it can send one cumulative ACK for several frames.

Go-Back-N , normal operation

- The sender keeps track of the outstanding frames and updates the variables and windows as the ACKs arrive.



Go-Back-N , lost frame



- Frame 2 is lost
- When the receiver receives frame 3, it discards frame 3 as it is expecting frame 2 (according to window).
- After the timer for frame 2 expires at the sender site, the sender sends frame 2 and 3. (go back to 2)

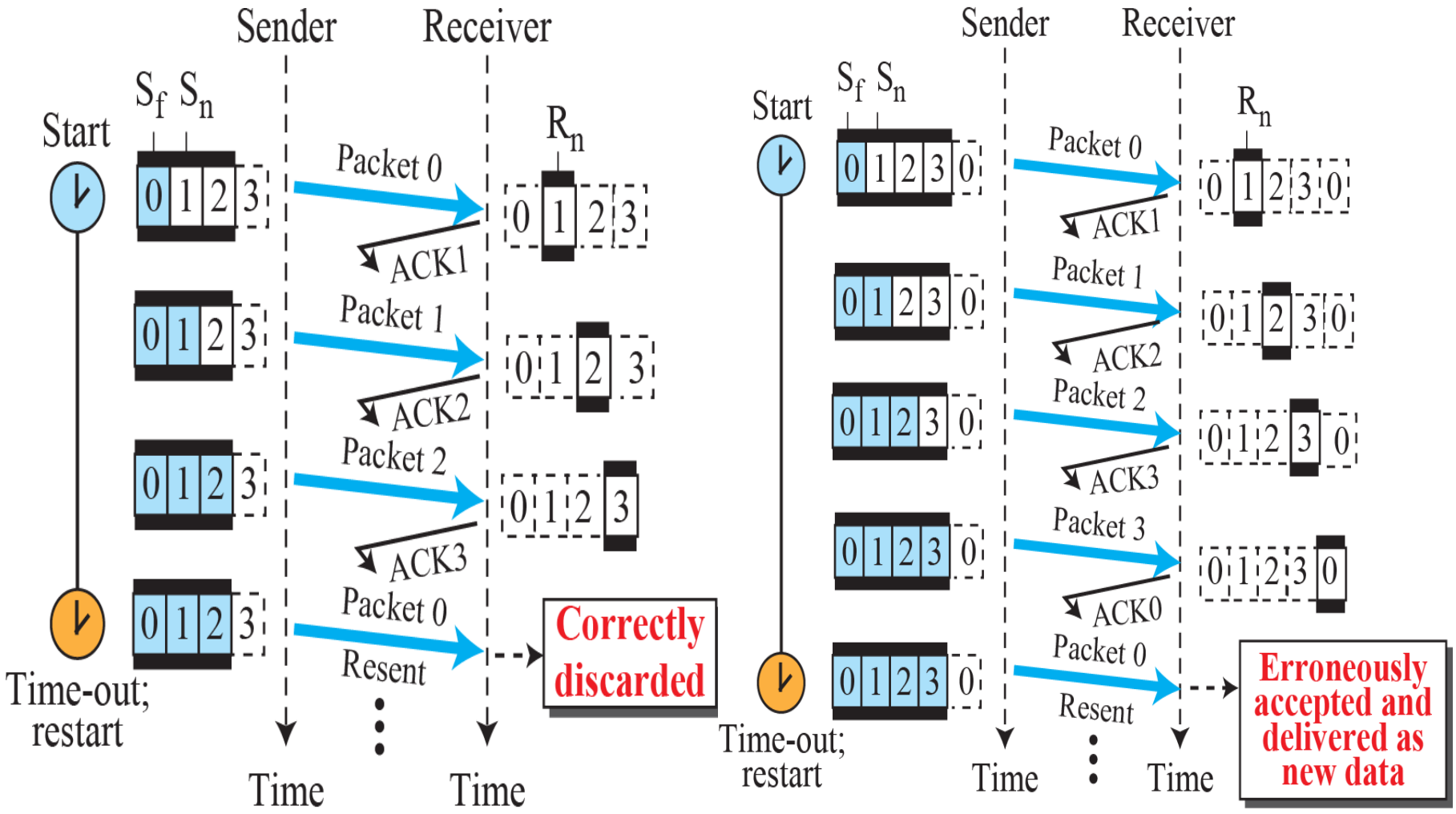
Go-Back-N , damaged/lost/delayed ACK

- If an ACK is damaged/lost, we can have two situations:
- If the next ACK arrives before the expiration of any timer, there is no need for retransmission of frames because ACKs are cumulative in this protocol.
- If ACK1, ACK2, and ACk3 are lost, ACK4 covers them if it arrives before the timer expires.
- If ACK4 arrives after time-out, the last frame and all the frames after that are resent.
- Receiver never resends an ACK.
- A delayed ACK also triggers the resending of frames

Note

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

Figure 3.28: Send window size for Go-Back-N

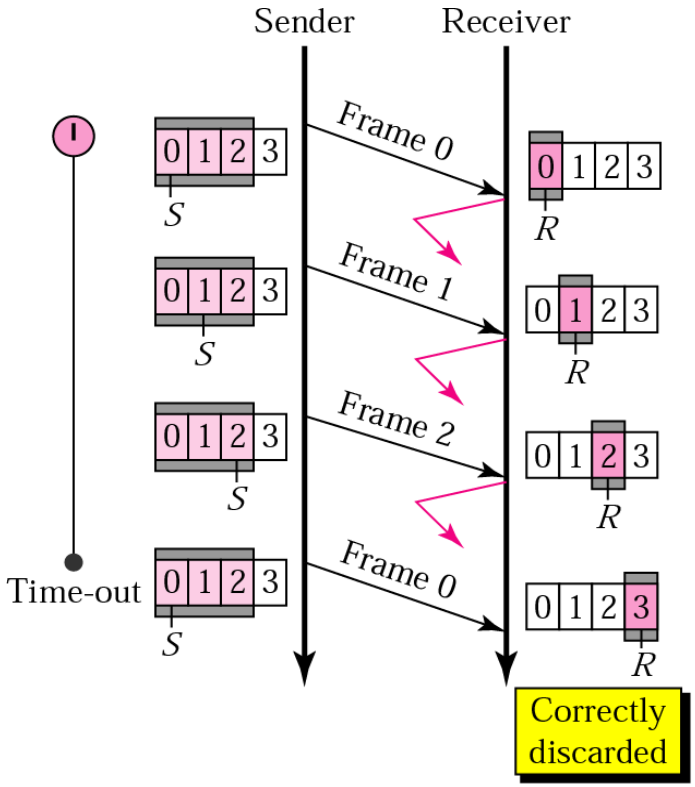


a. Send window of size $< 2^m$

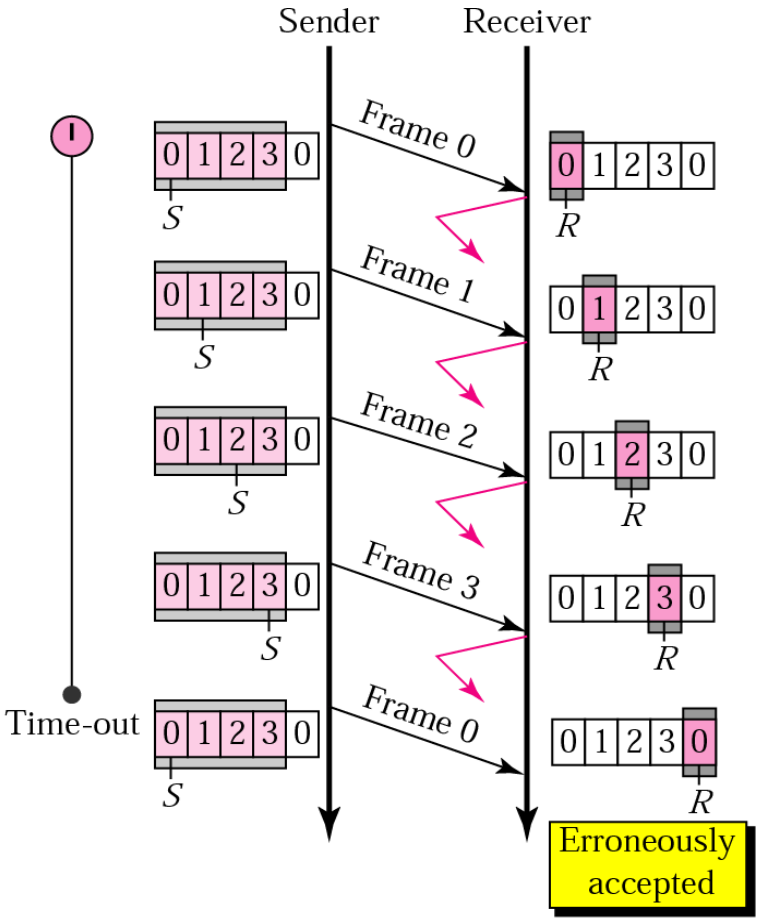
b. Send window of size $= 2^m$

Go-Back-N , sender window size

- Size of the sender window must be less than 2 m. Size of the receiver is always 1. If m = 2, window size = 2 m – 1 = 3. Fig compares a window size of 3 and 4.



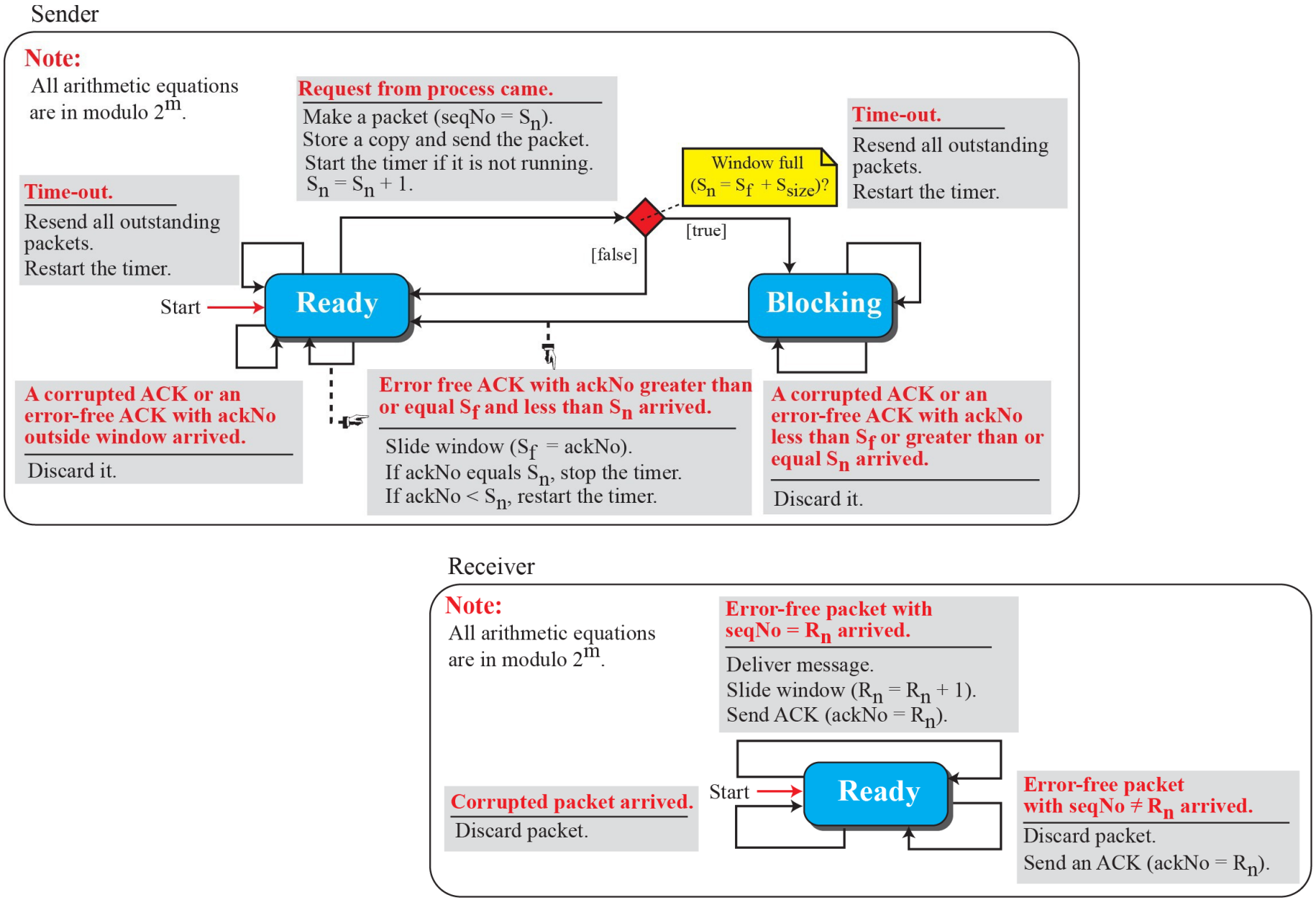
a. Window size < 2^m



b. Window size = 2^m

Accepts as the 1st frame in the next cycle-an **error**

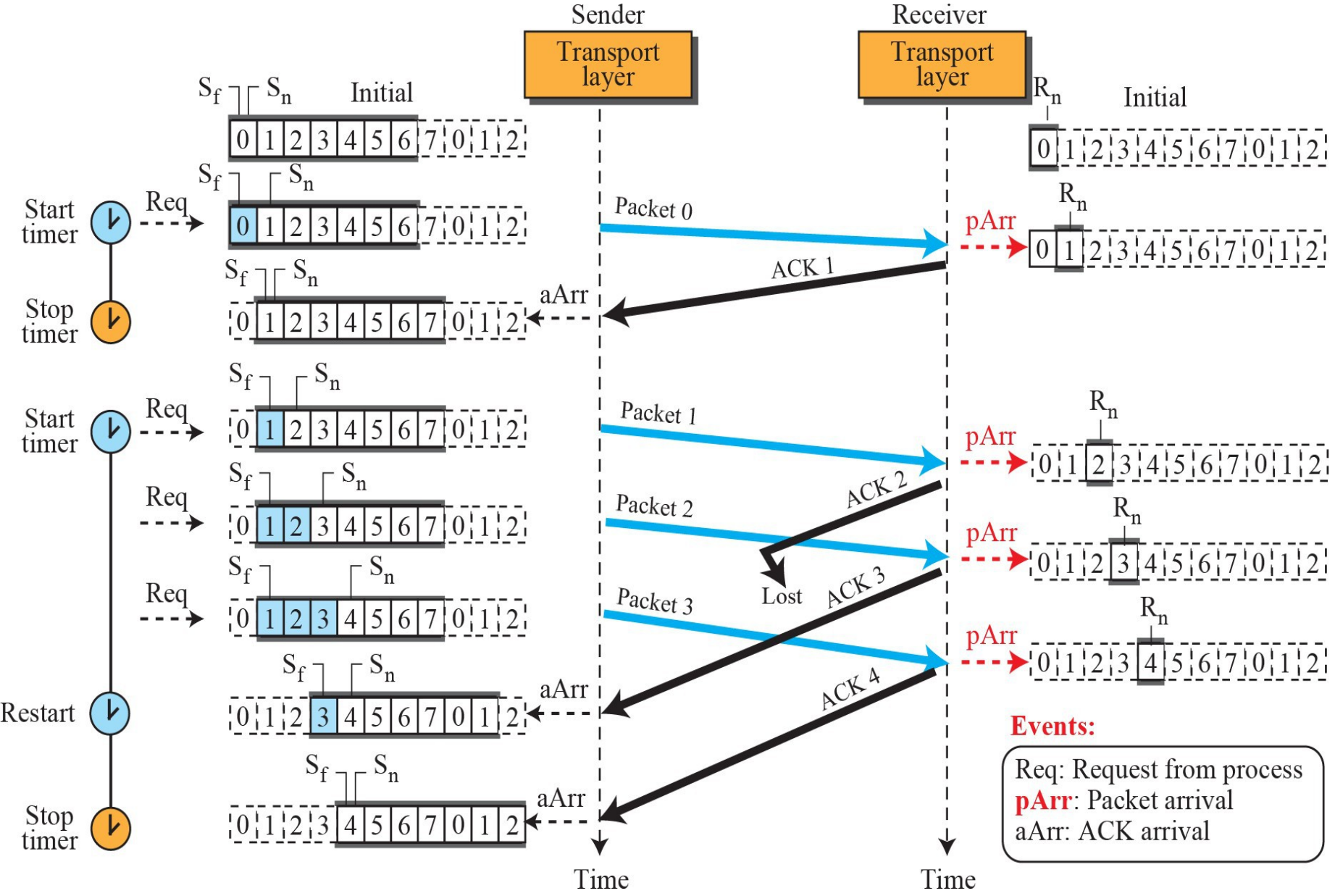
Figure 3.27: FSMs for the Go-Back-N protocol



Example 3.7

Figure 3.29 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.

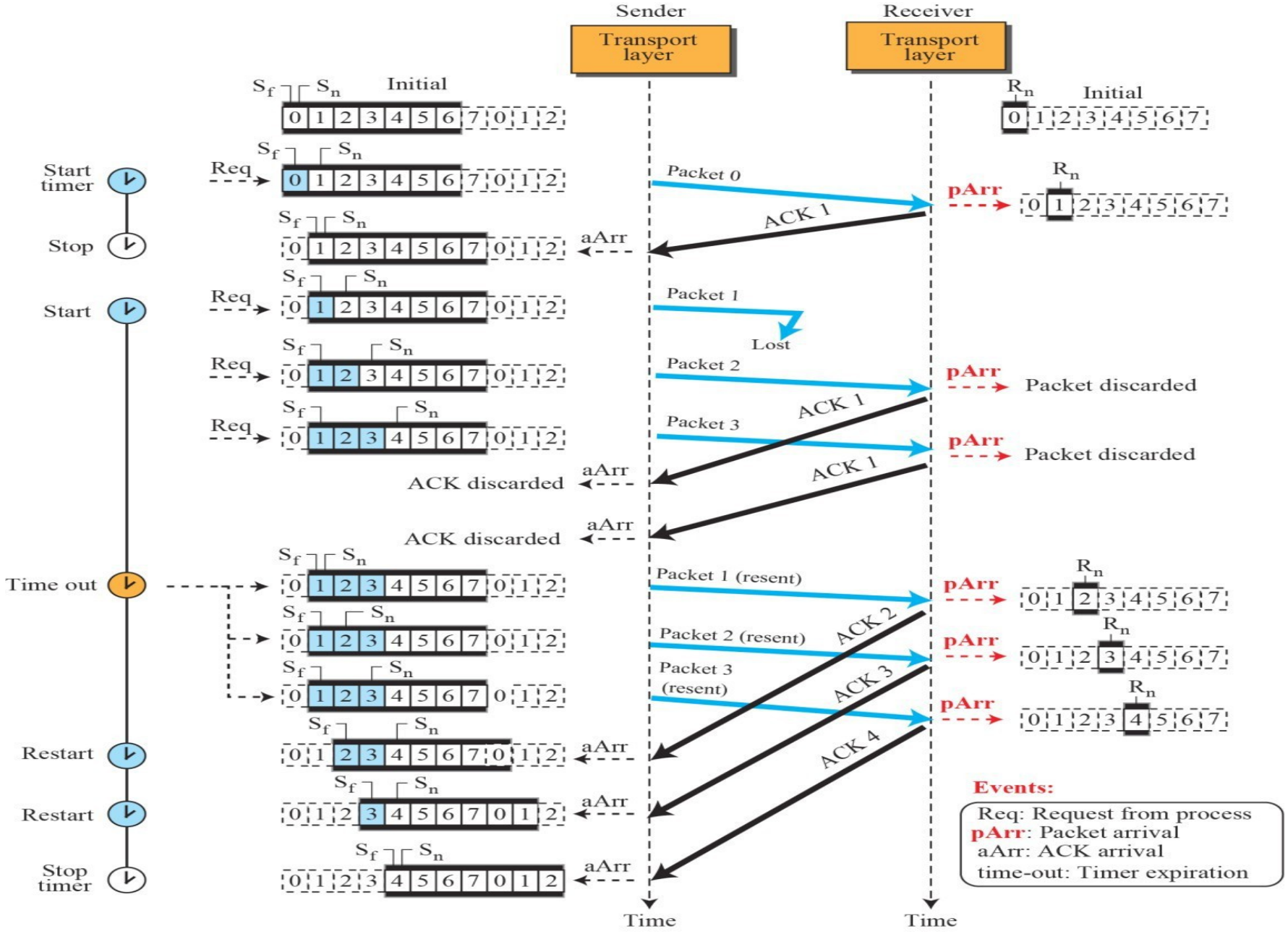
Figure 3.29: Flow diagram for Example 3.7

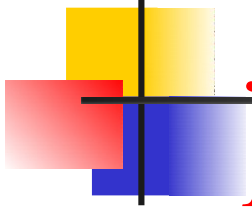


Example 3.8

Figure 3.30 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 1. However, these ACKs are not useful for the sender because the ackNo is equal to *Sf*, not greater that *Sf* . So the sender discards them. When the time-out occurs, the sender resends packets 1, 2, and 3, which are acknowledged.

Figure 3.30: Flow diagram for Example 3.8

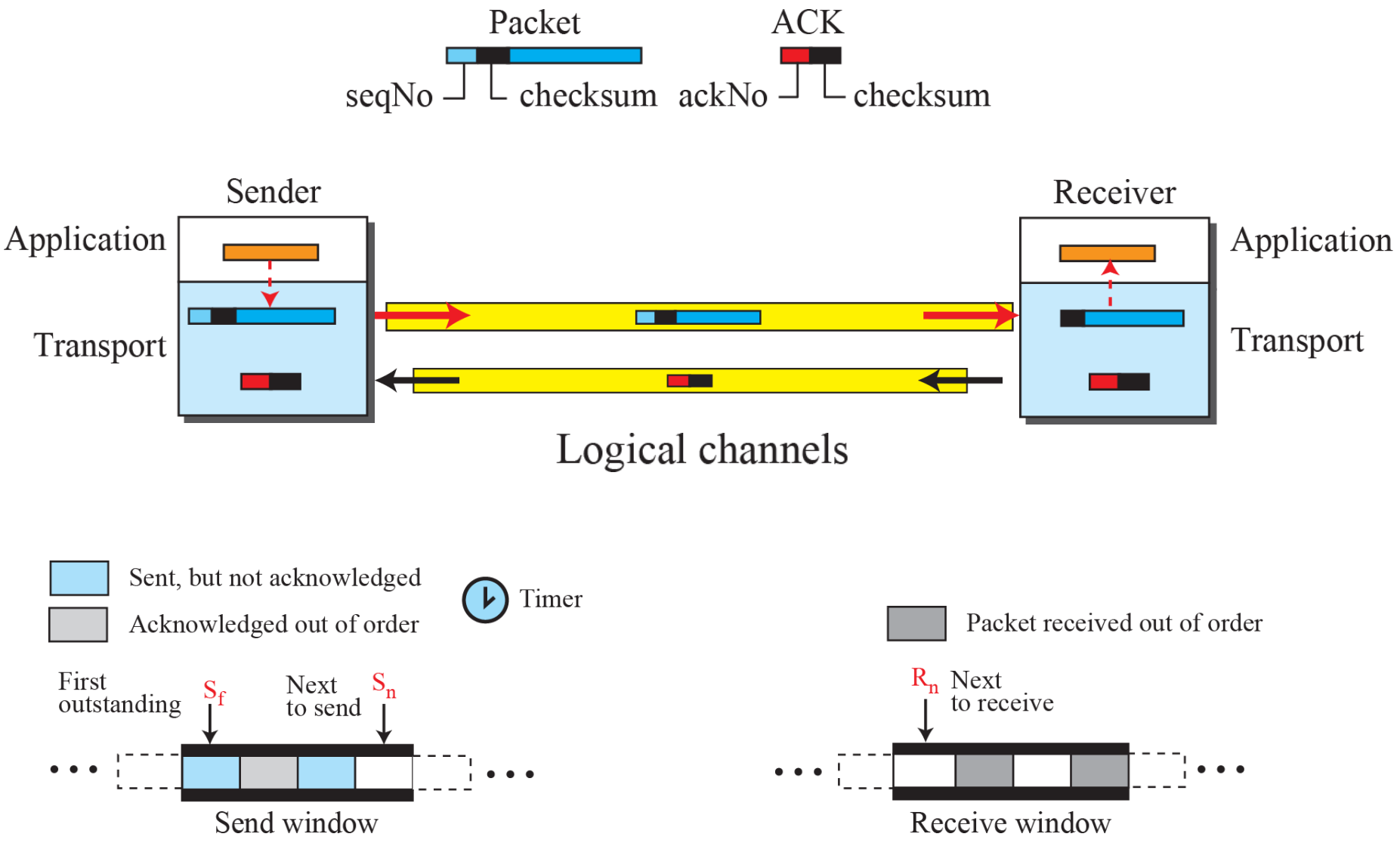




3.2.4 Selective-Repeat Protocol

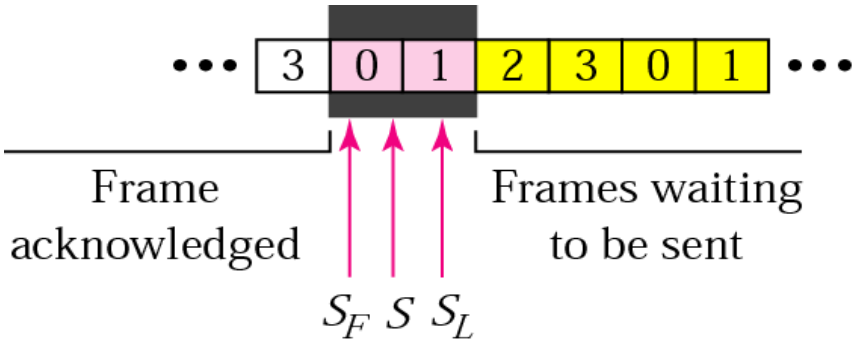
The Go-Back-N protocol simplifies the process at the receiver. The receiver keeps track of only one variable, and there is no need to buffer out-of-order packets; they are simply discarded. Another protocol, called the Selective-Repeat (SR) protocol, has been devised, which, as the name implies, resends only selective packets, those that are actually lost.

Figure 3.31: Outline of Selective-Repeat

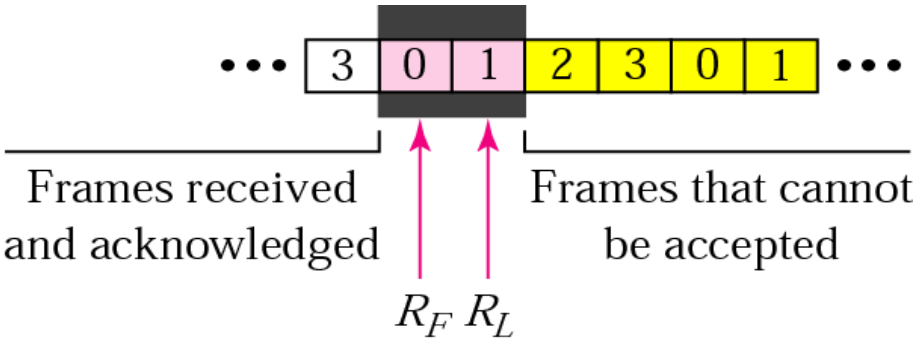


Selective Repeat , sender and receiver windows

- Go-Back-N simplifies the process at the receiver site. Receiver only keeps track of only one variable, and there is no need to buffer out-of- order frames, they are simply discarded.
- However, Go-Back-N protocol is inefficient for noisy link. It bandwidth inefficient and slows down the transmission.
- In Selective Repeat , only the damaged frame is resent. More bandwidth efficient but more complex processing at receiver.
- It defines a negative ACK (NAK) to report the sequence number of a damaged frame before the timer expires.



a. Sender window



b. Receiver window

Figure 3.32: Send window for Selective-Repeat protocol

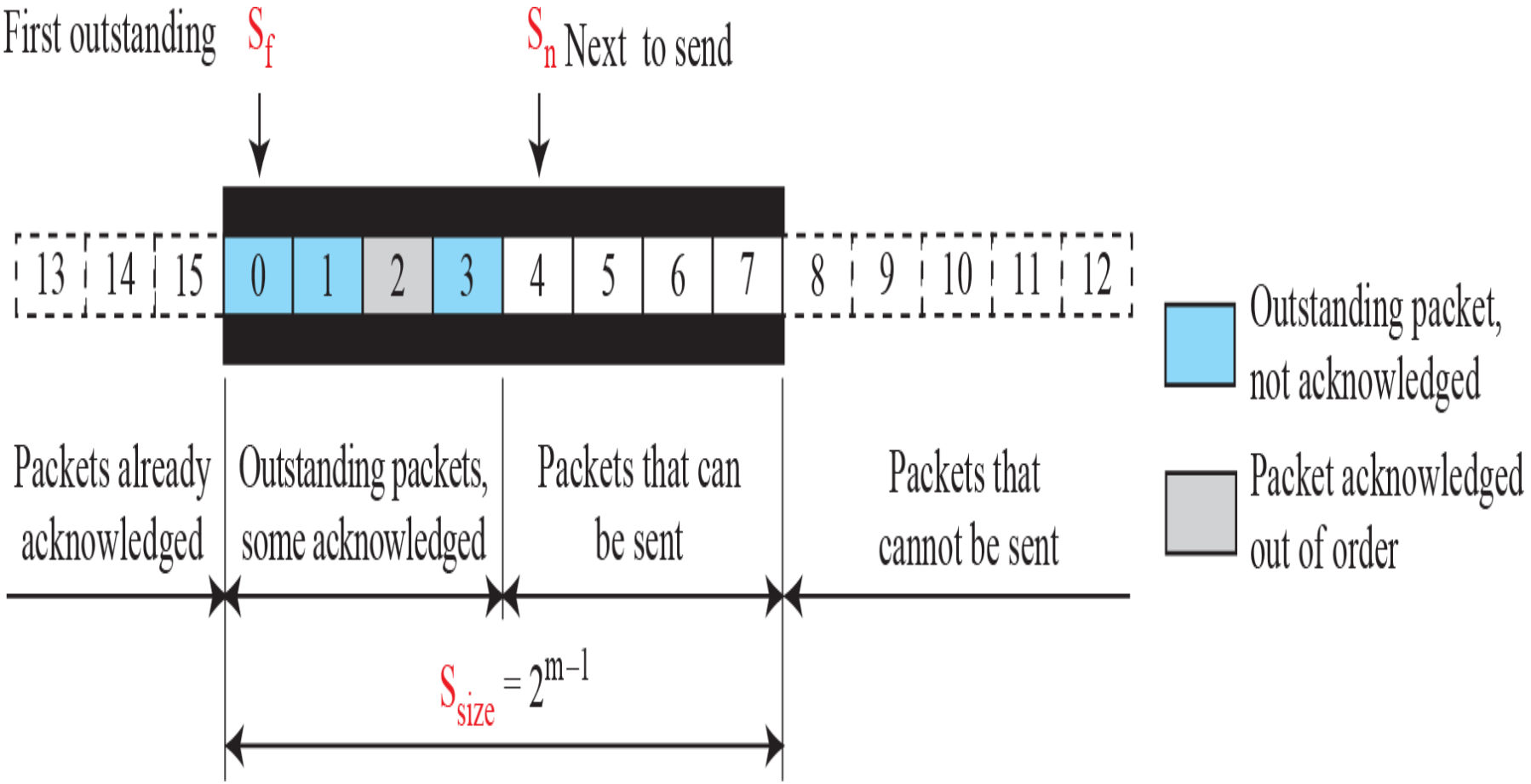
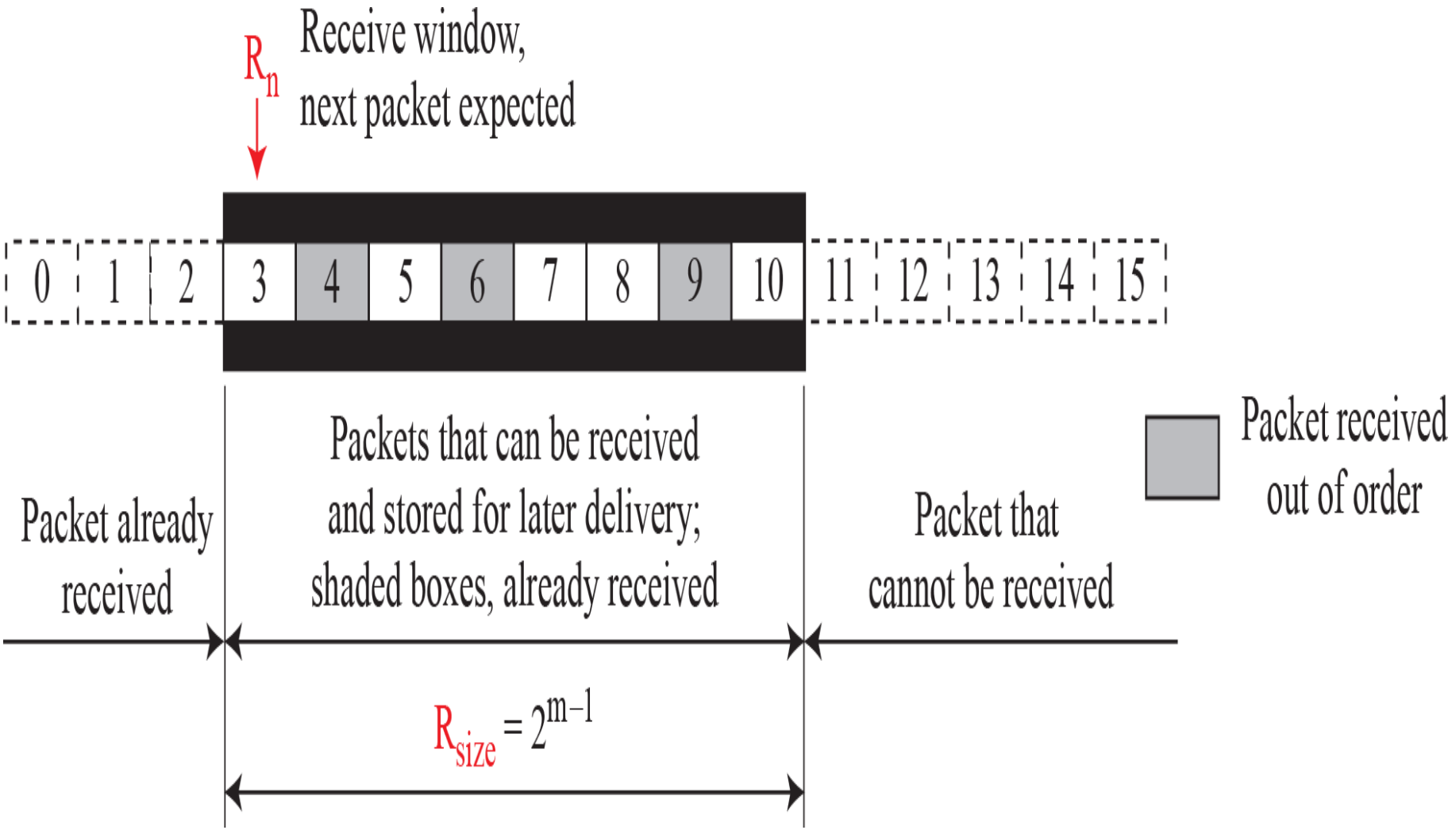


Figure 3.33: Receive window for Selective-Repeat protocol



Note

In the Selective-Repeat protocol, an acknowledgment number defines the sequence number of the error-free packet received.

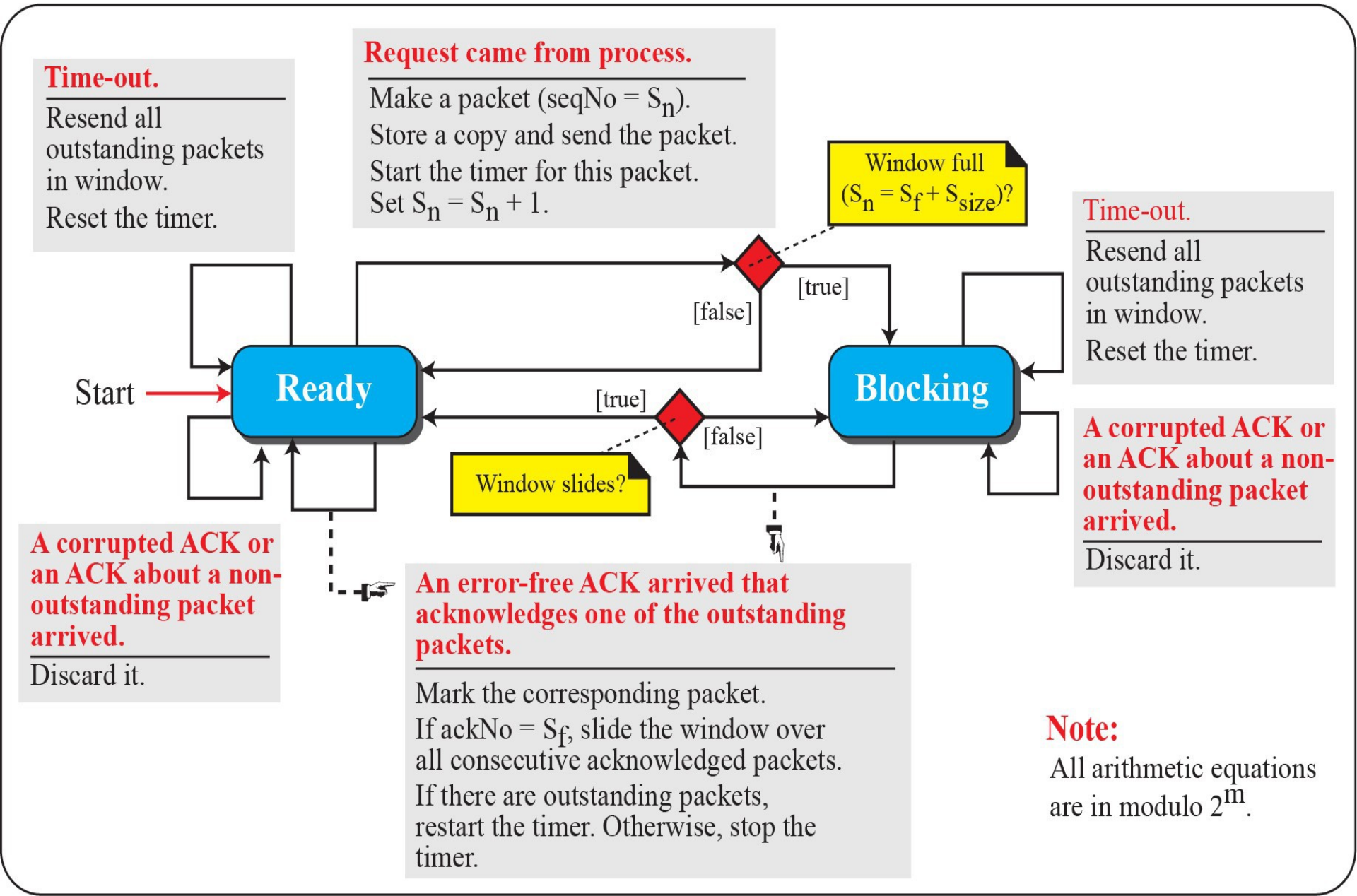
Example 3.9

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 3.34: FSMs for SR protocol



Receiver

Error-free packet with seqNo inside window arrived.

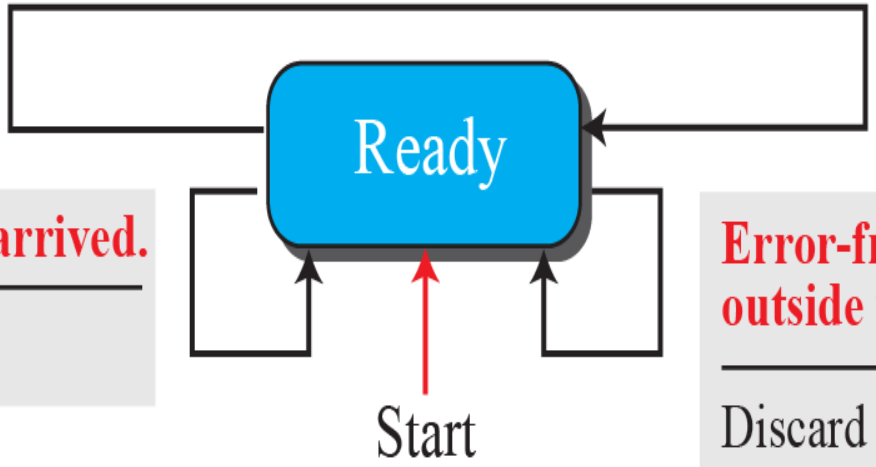
If duplicate, discard; otherwise, store the packet.

Send an ACK with ackNo = seqNo.

If seqNo = R_n, deliver the packet and all consecutive previously arrived and stored packets to application, and slide window.

Note:

All arithmetic equations are in modulo 2^m.



Corrupted packet arrived.

Discard the packet.

Error-free packet with seqNo outside window boundaries arrived.

Discard the packet.

Send an ACK with ackNo = R_n.

Example 3.10

This example is similar to Example 3.8 (Figure 3.30) 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 3.10 (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 3.35: Flow diagram for Example 3.10

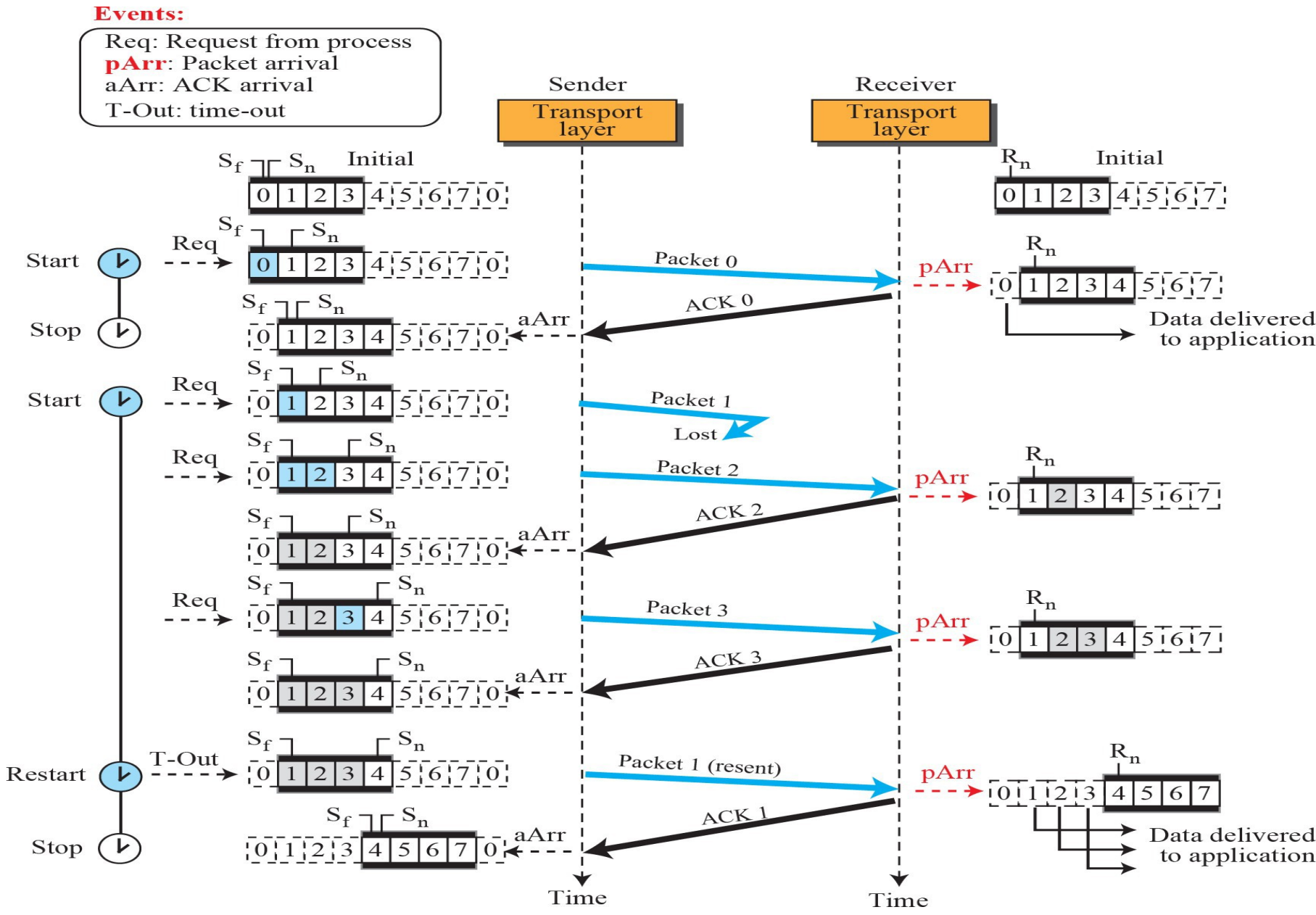
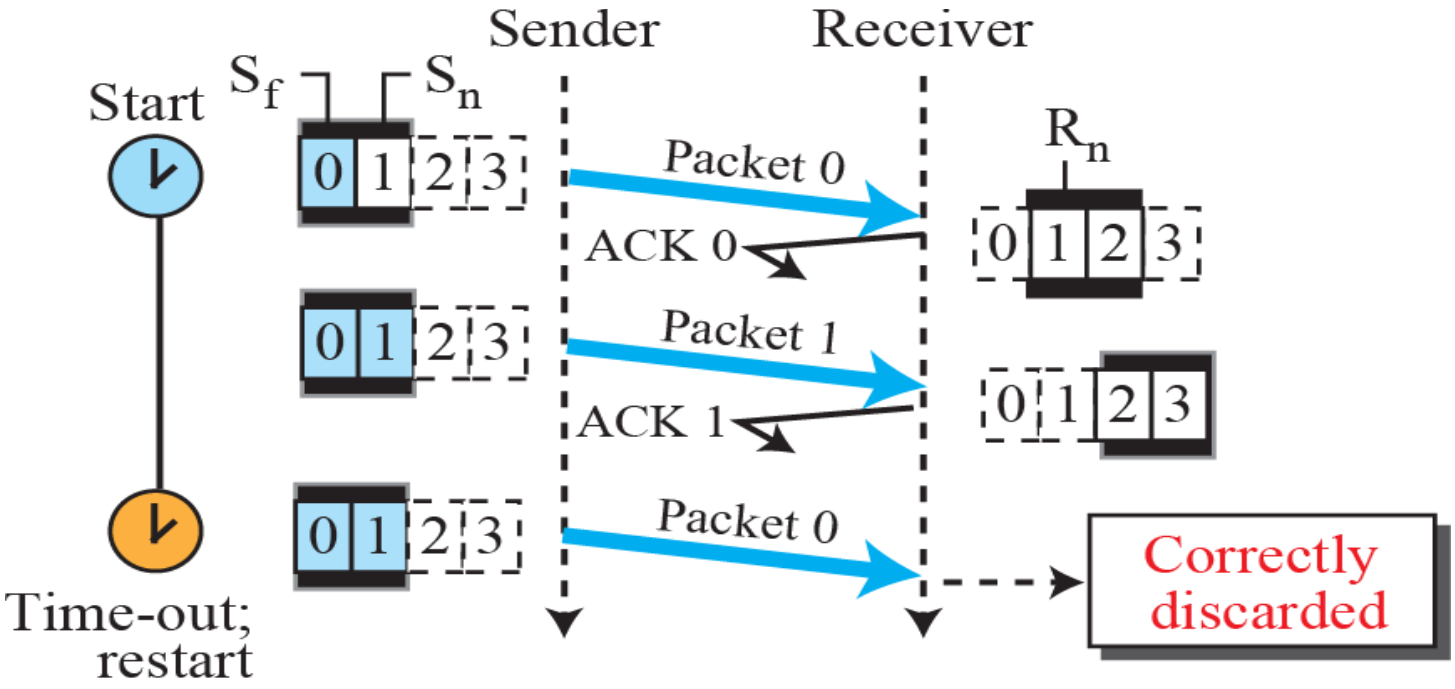
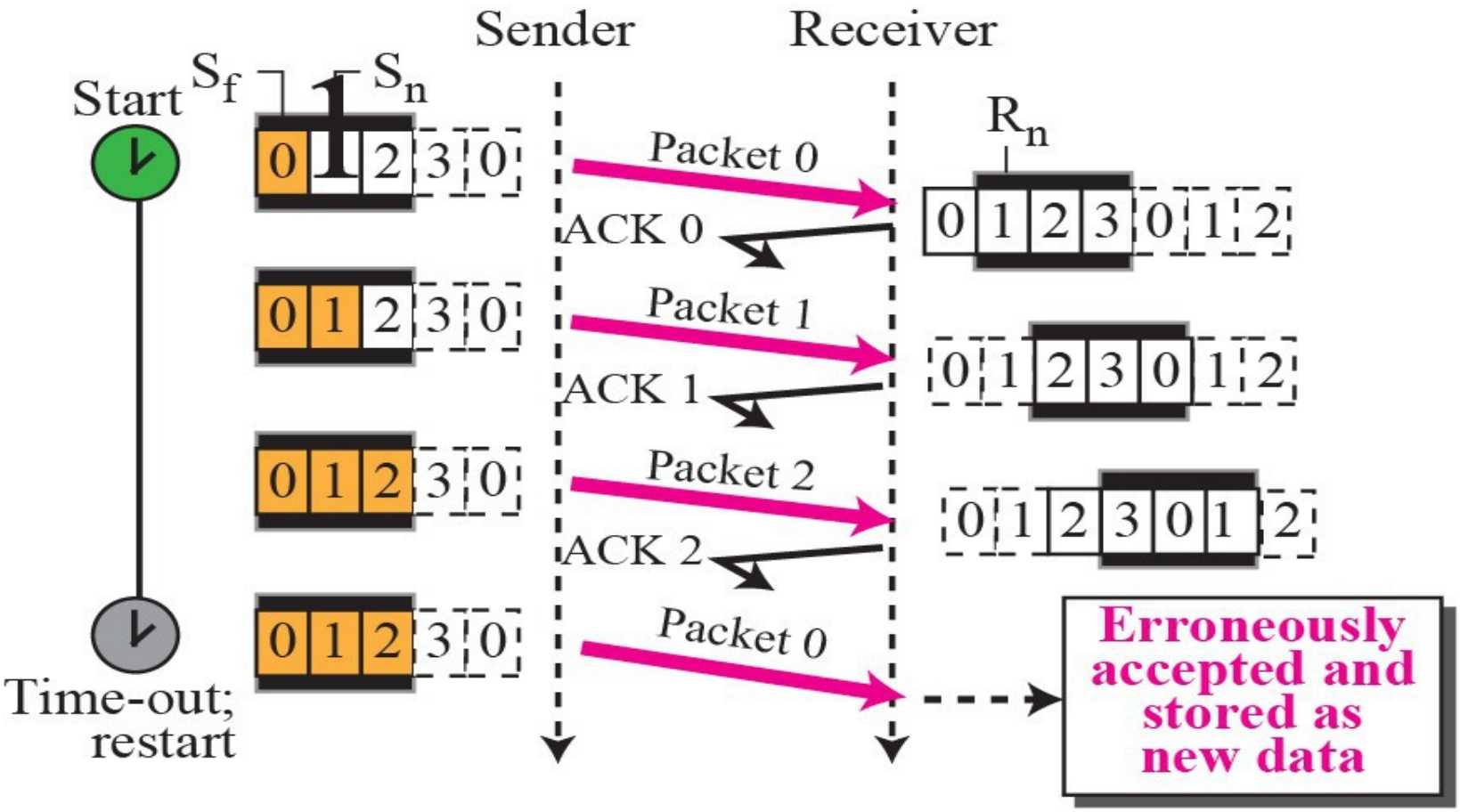


Figure 3.36: Selective-Repeat, window size



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

Figure 13.35 *Selective-Repeat window size*



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

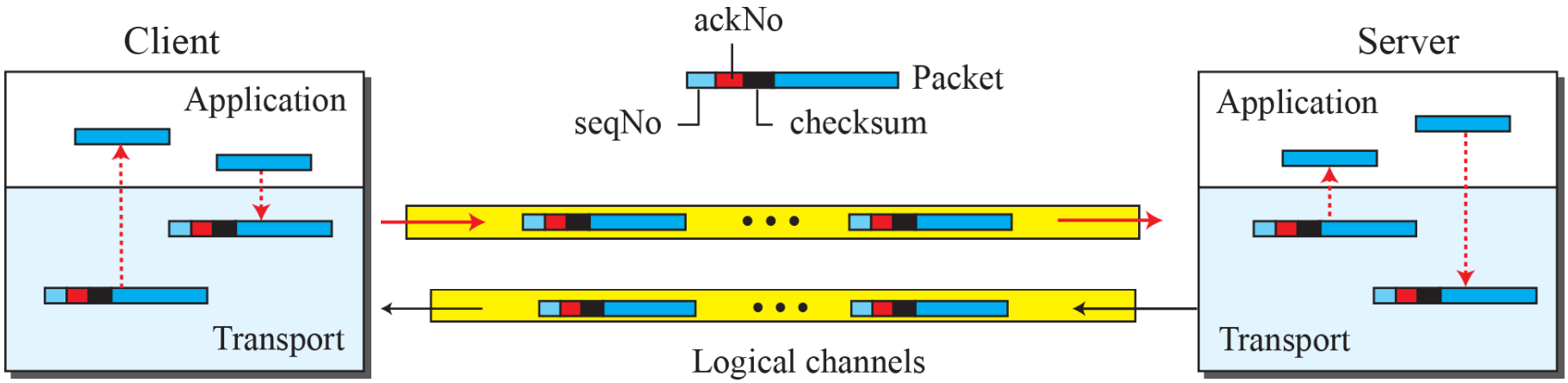
Note

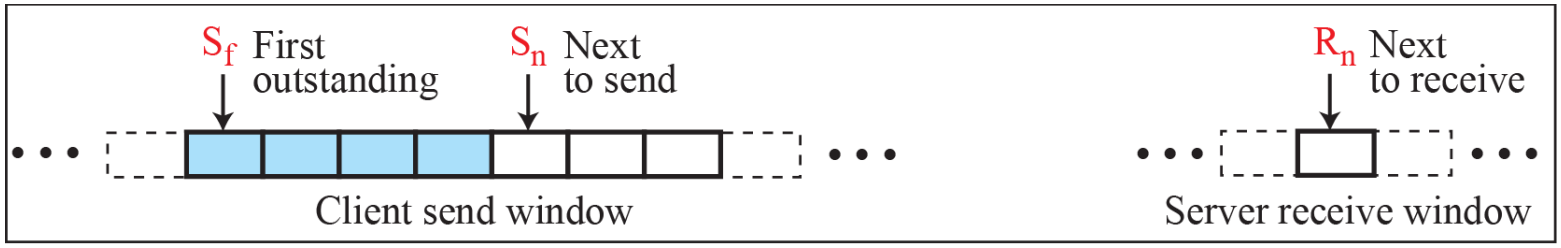
In Selective-Repeat, the size of the sender and receiver window can be at most one-half of 2m.

3.2.5 Bidirectional Protocols

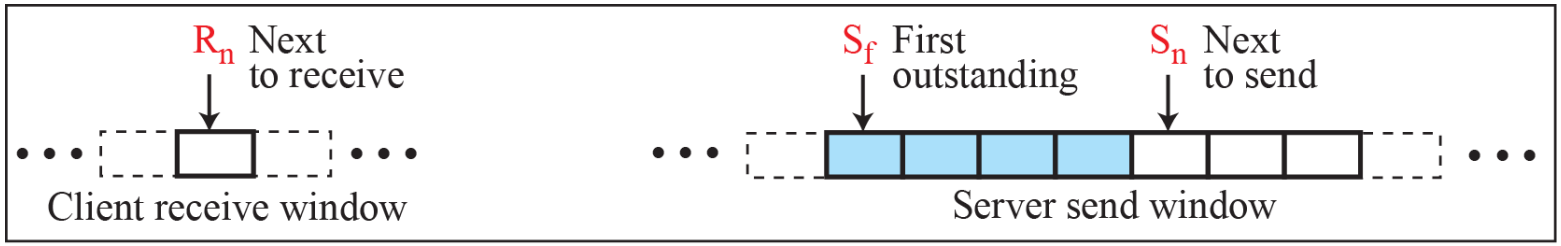
The four protocols we discussed earlier in this section are all unidirectional: data packets flow in only one direction and acknowledgments travel in the other direction. In real life, data packets are normally flowing in both directions: from client to server and from server to client. This means that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

Figure 3.37: Design of piggybacking in Go-Back-N





Windows for communication from client to server



Windows for communication from server to client