UNIT - 4

Transport Layer

Outline

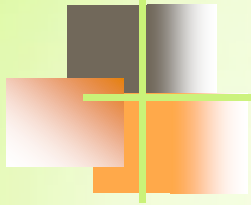
3.1 INTRODUCTION

3.2 TRANSPORT-LAYER PROTOCOLS

- Simplex Protocol
- Stop-and-Wait Protocol
- Go-Back-N Protocol (GBN)
- Selective-Repeat Protocol
- Bidirectional Protocols: Piggybacking

3.3 INTERNET TRANSPORT-LAYER PROTOCOLS

- User Datagram Protocol (UDP)
- Transmission Control Protocol (TCP)

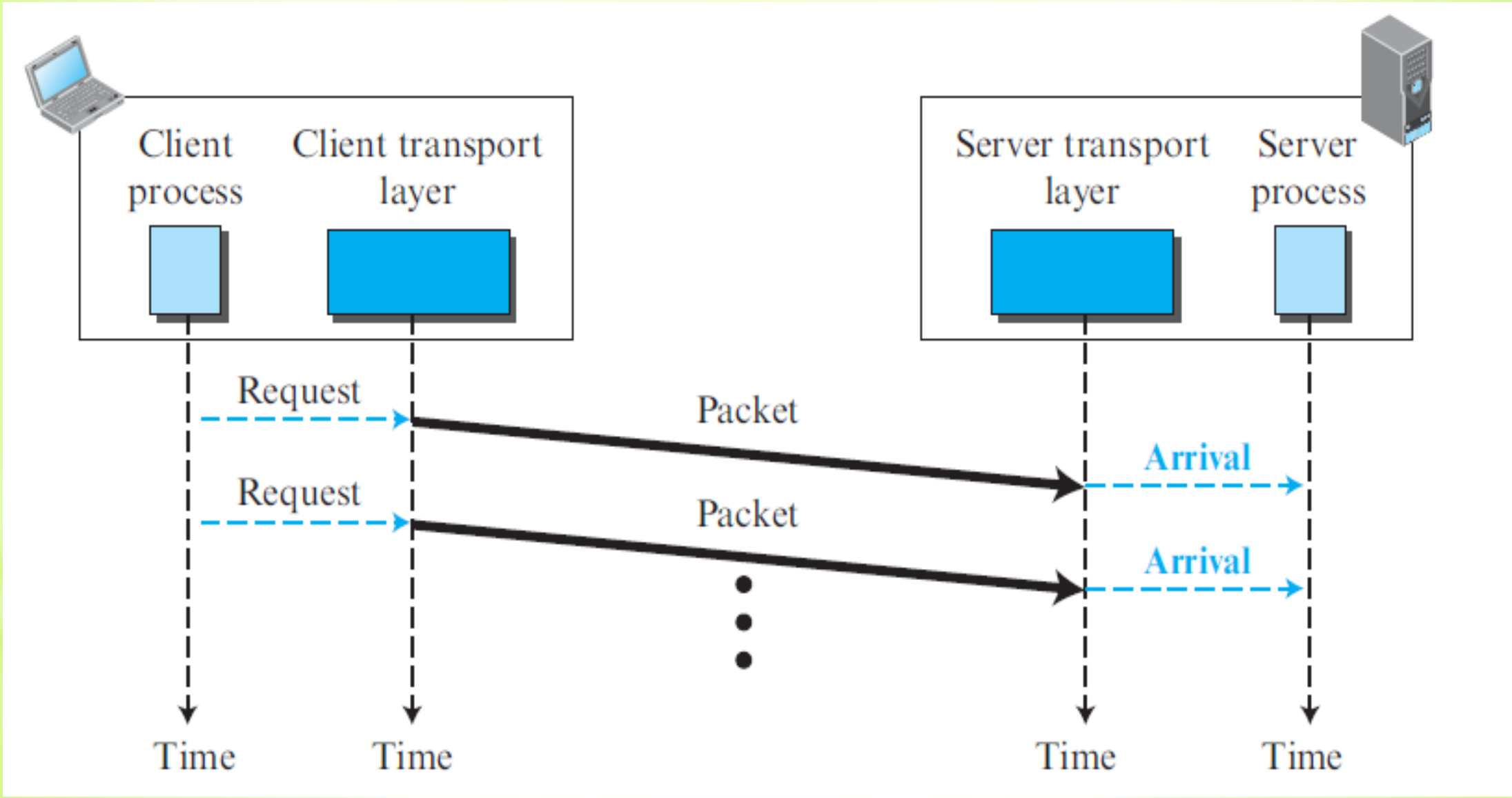


- Sequence Numbers
- Acknowledgment Numbers
- Send Window
- Receive Window
- Timers
- Resending packets
- FSMs (Sender & Reciver)

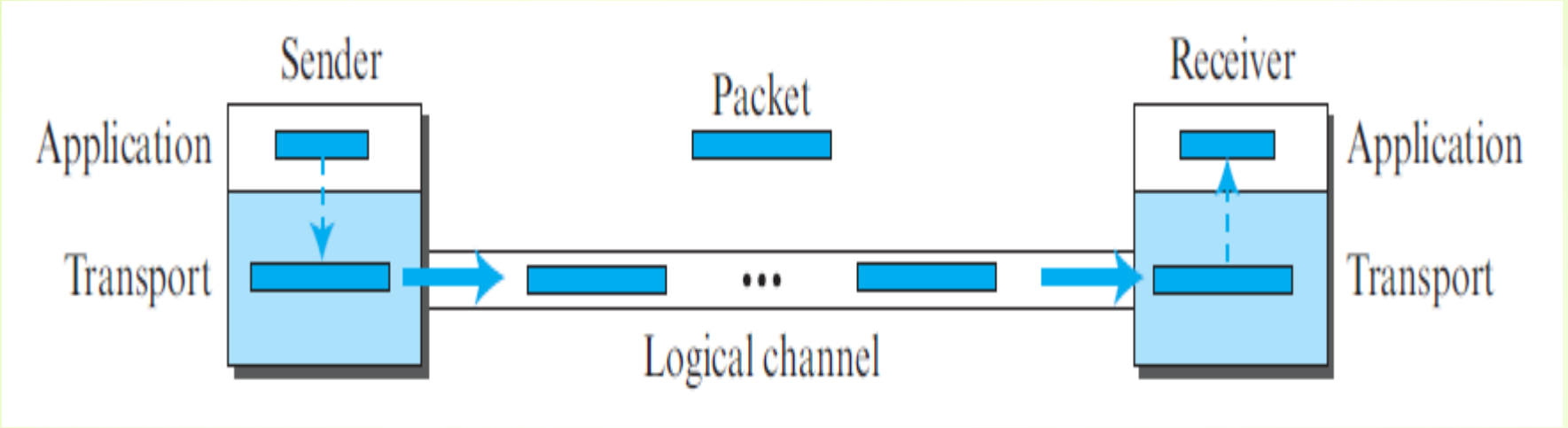
Simplex Protocol

- Used in Noiseless channels
- Simple connectionless protocol with neither flow nor error control
- The transport layer at 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.
- No acknowledgement and no sequence number.
- Each FSM has only one state, the ready state.

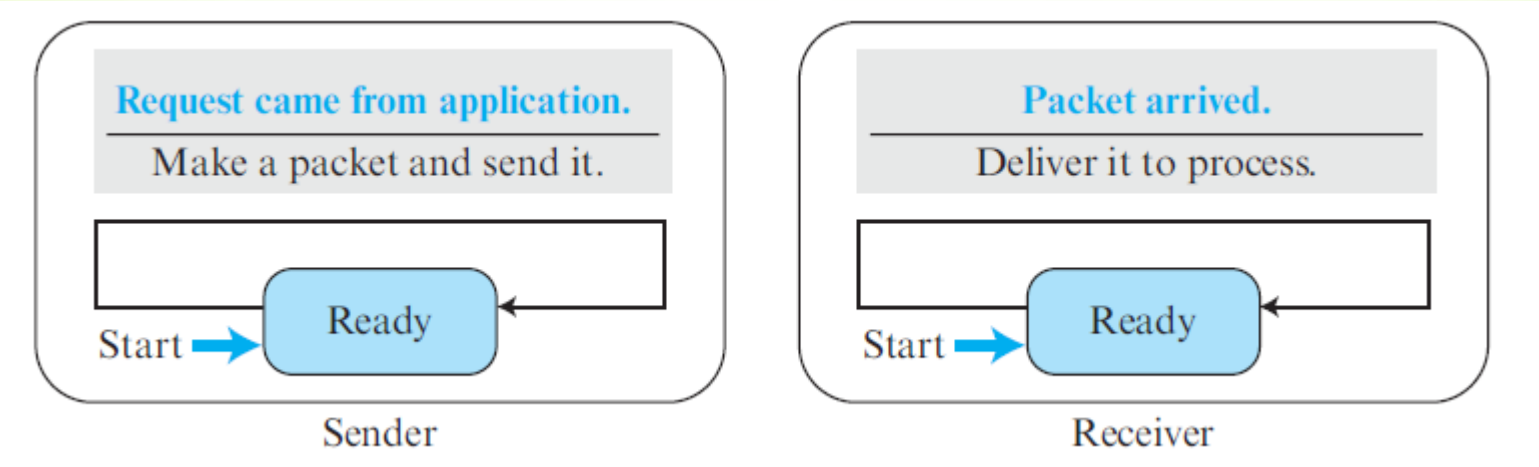
Flow diagram



Simplex protocol



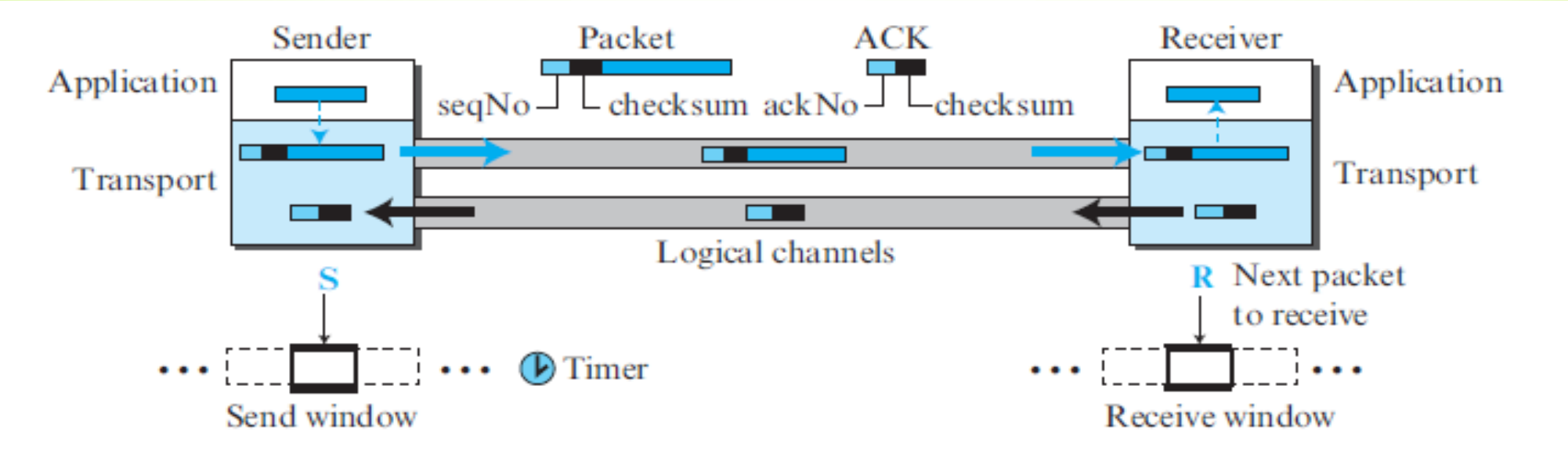
FSMs for the simplex protocol



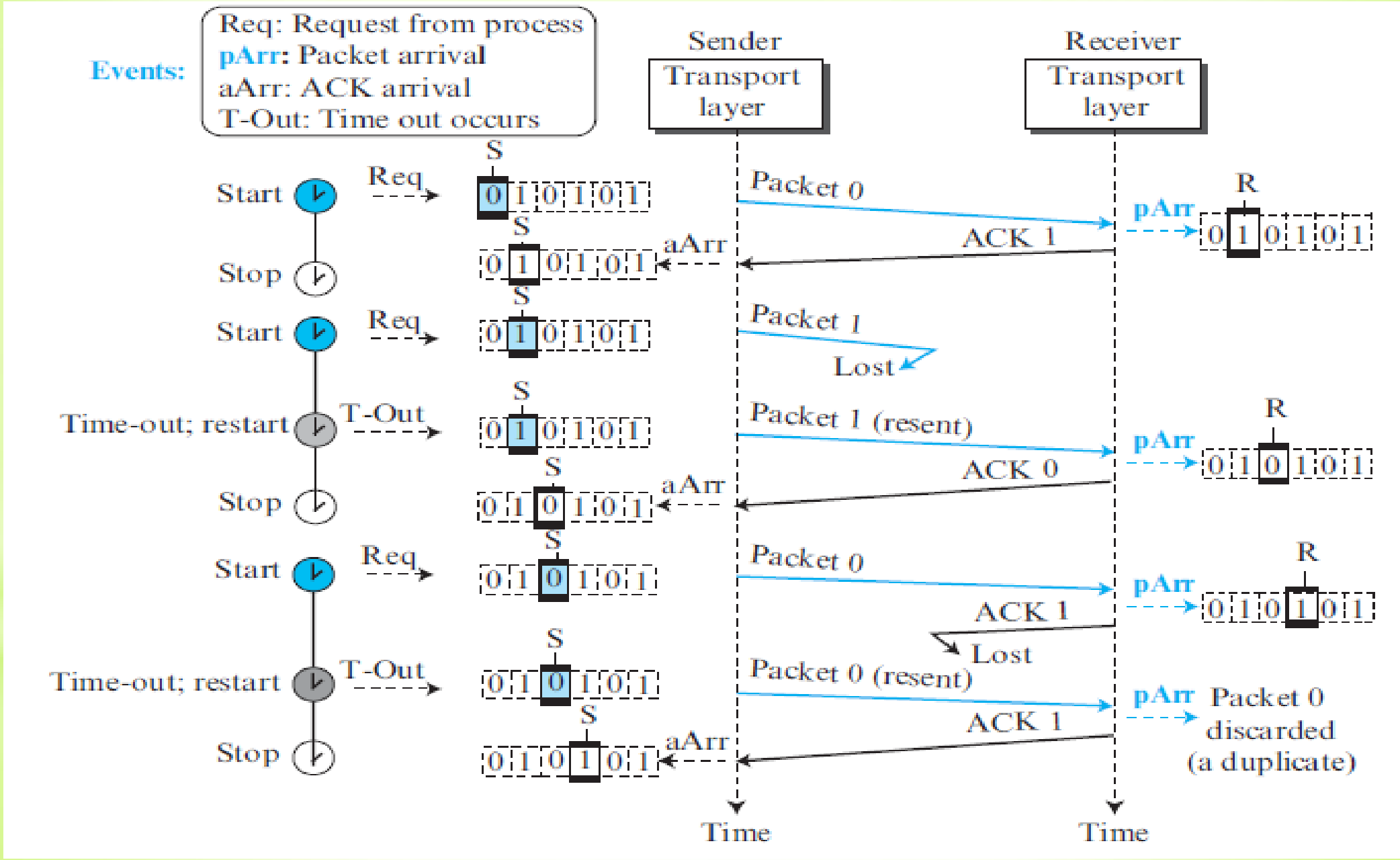
Stop and Wait protocol

- It is a connection-oriented protocol
- 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, if checksum not correct the packet is discarded.
- 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 (if it has one to send).
- If the timer expires, the sender resends the previous packet, assuming that the packet was either lost or corrupted. This means that the sender needs to keep a copy of the packet until its acknowledgment arrives.

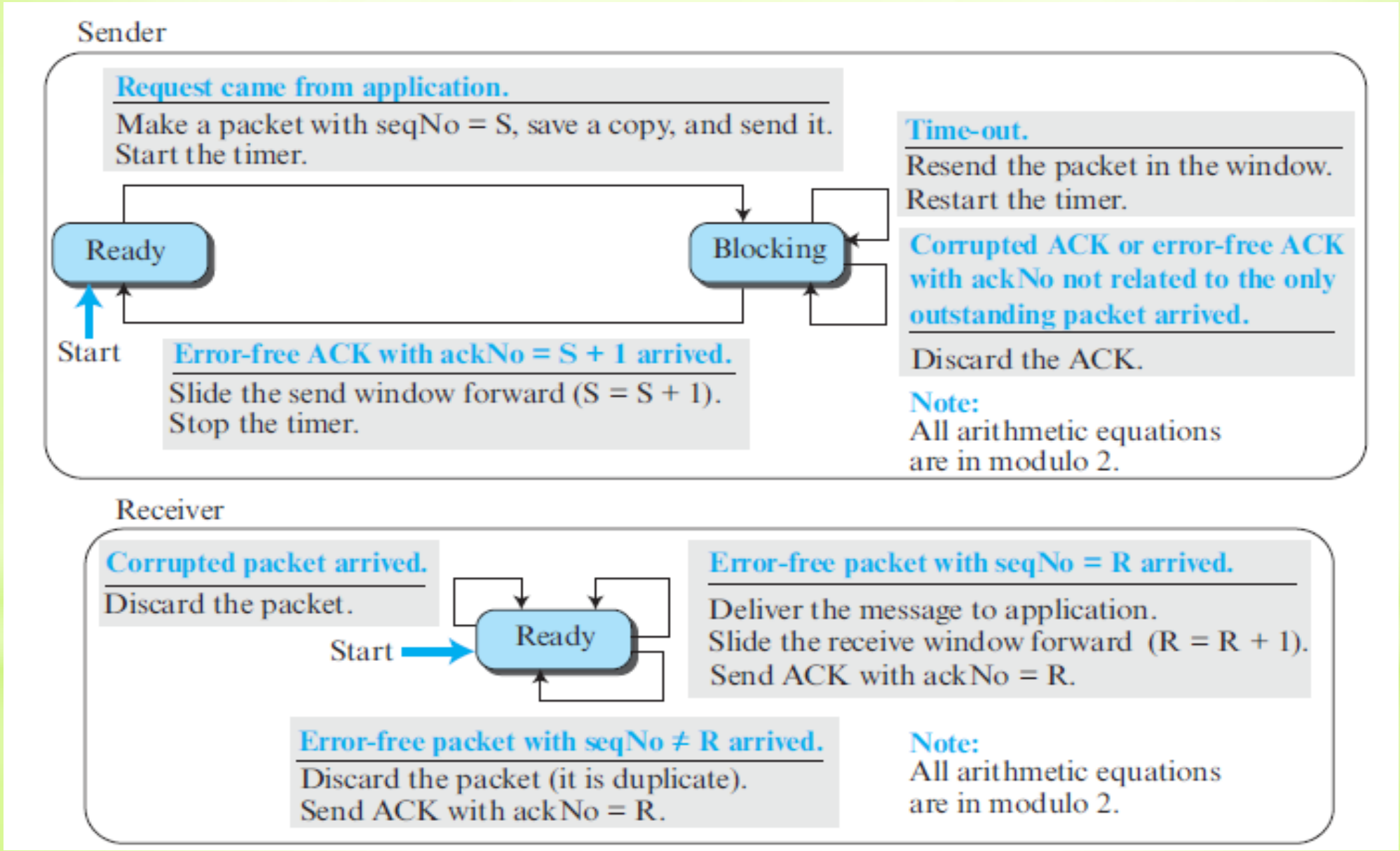
- To prevent duplicate packets, the protocol uses sequence numbers and acknowledgment Numbers
- The sender is initially in the ready state, but it can move between the ready and blocking state.
- The receiver is always in the ready state



Flow diagram of stop and wait



FSM for the Stop-and-Wait protocol



Shortfalls of Stop and Wait Protocol:

- The Stop-and-Wait protocol is very inefficient if our channel is thick and long.
- By thick, we mean that our channel has a large bandwidth (high data rate); by long, we mean the round-trip delay is long.
- The product of these two is called the **bandwidth-delay product**.

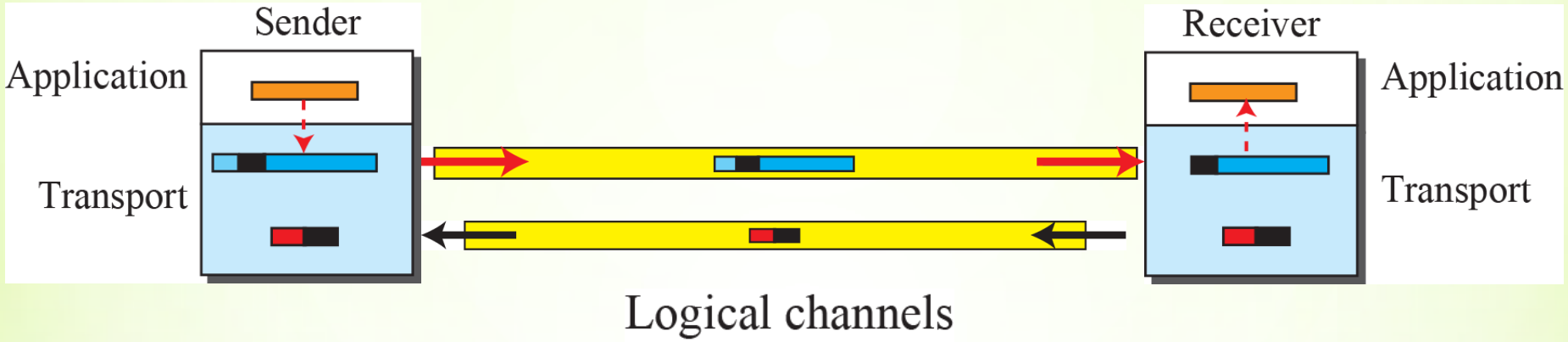
Pipelining:

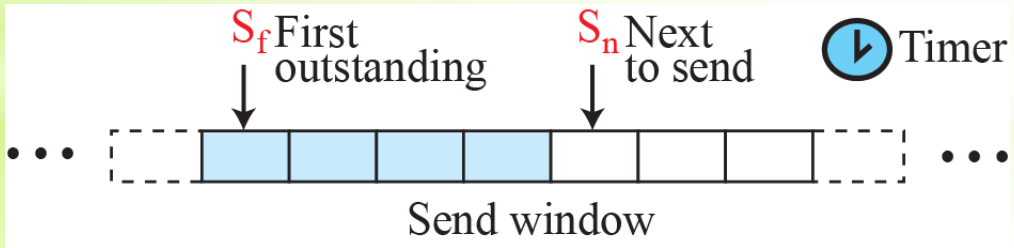
- In networking and in other areas, a task is often begun before the previous task has ended. This is known as pipelining.
- There is no pipelining in the Stop-and-Wait protocol because a sender must wait for a packet to reach the destination and be acknowledged before the next packet can be sent.

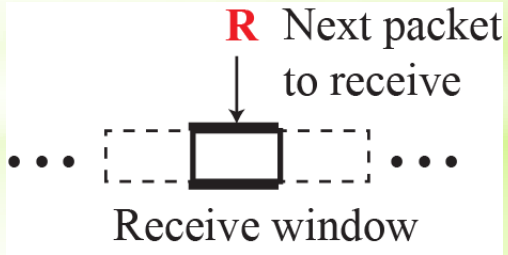
Go-Back-N protocol

- To improve the efficiency of transmission, multiple packets must be in transition while the sender is waiting for acknowledgment. i.e. using the concept of pipelining.
- A simple protocol that can achieve this goal is called Go-Back-N (GBN)
- The key to Go-back-N is that we can send several packets before receiving acknowledgments, but the receiver can only buffer one packet.
- We keep a copy of the sent packets until the acknowledgments arrive.
- The sequence numbers are modulo 2^m, where ‘m’ is the size of the sequence number field in bits.
- An acknowledgment number in this protocol is cumulative and defines the sequence number of the next packet expected

Go-Back-N protocol

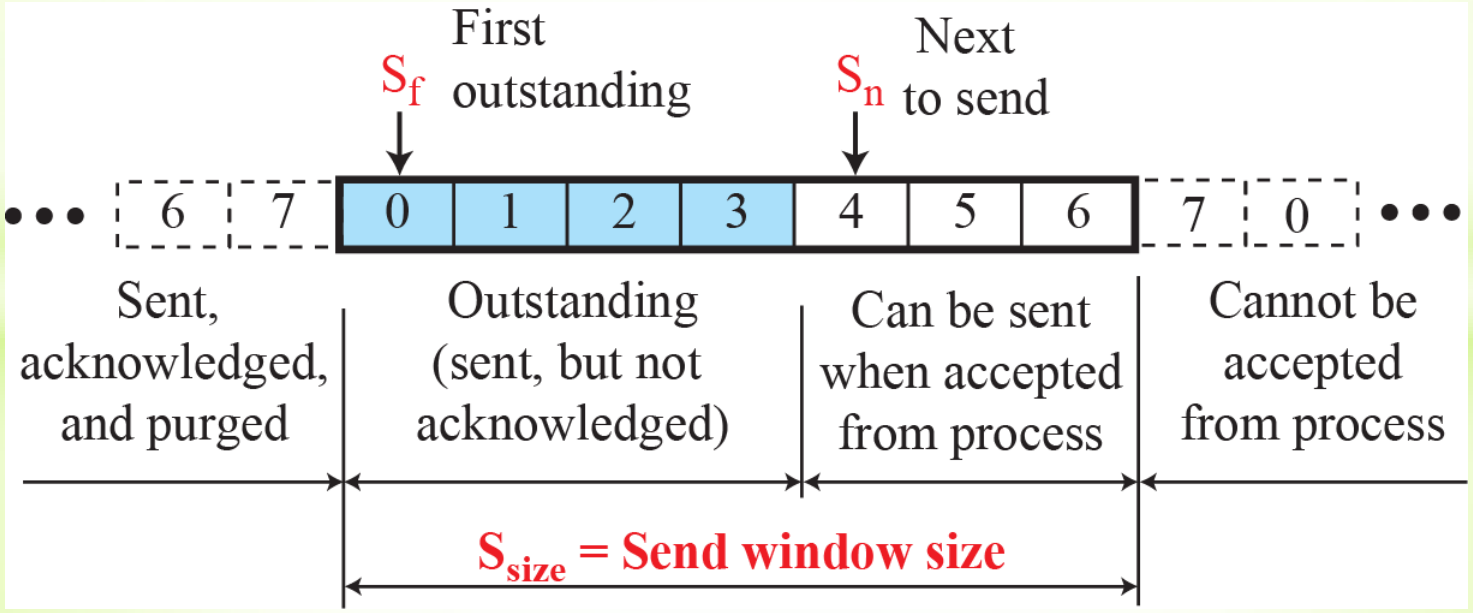




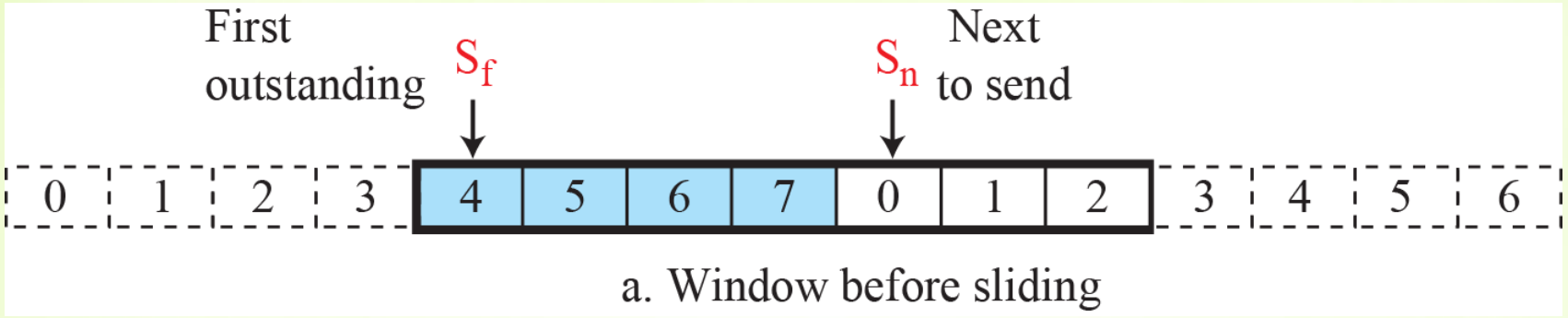


Send window for Go-Back-N

- The maximum **size of the senders window is $2^m - 1$**.
- The sender needs to wait to find out if the packets sent have been received or were lost. We call these **outstanding packets**.
- Three variables are defined:
  - S_f (send window, the first outstanding packet),
  - S_n (send window, the next packet to be sent), and
  - S_{size} (send window, size)= $2^m - 1$.



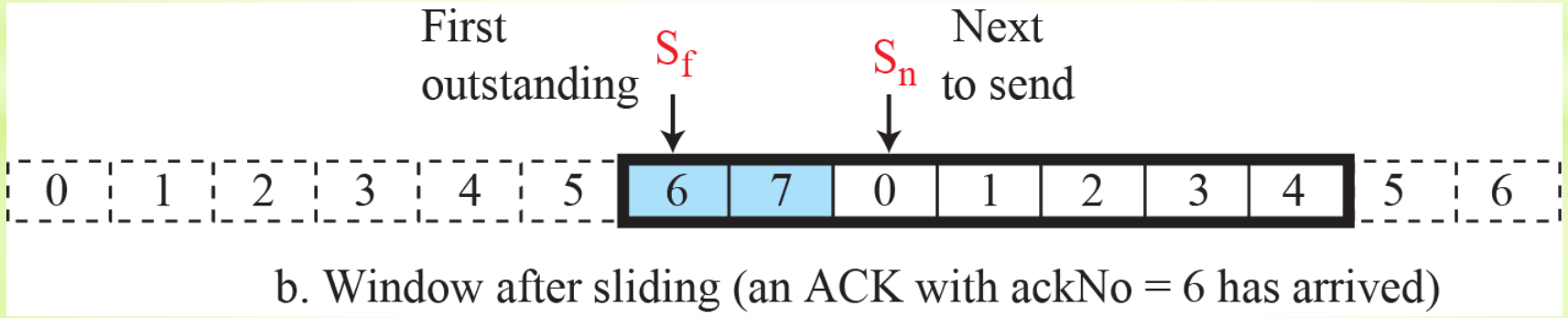
Sliding the send window



a. Window before sliding

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

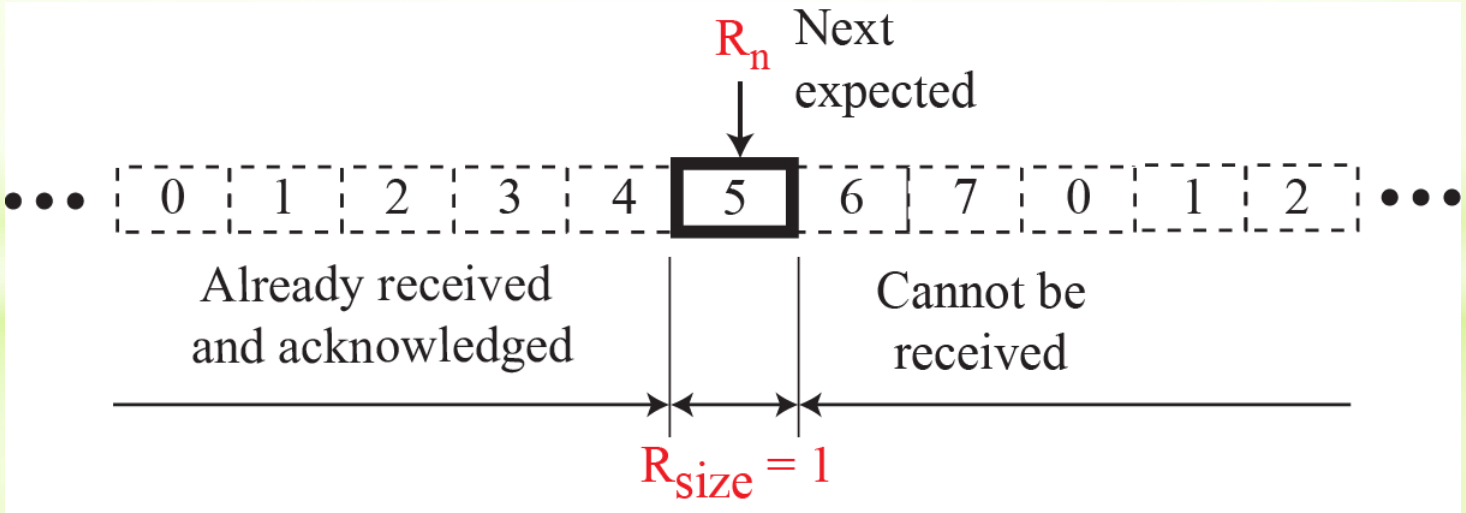
Sliding direction



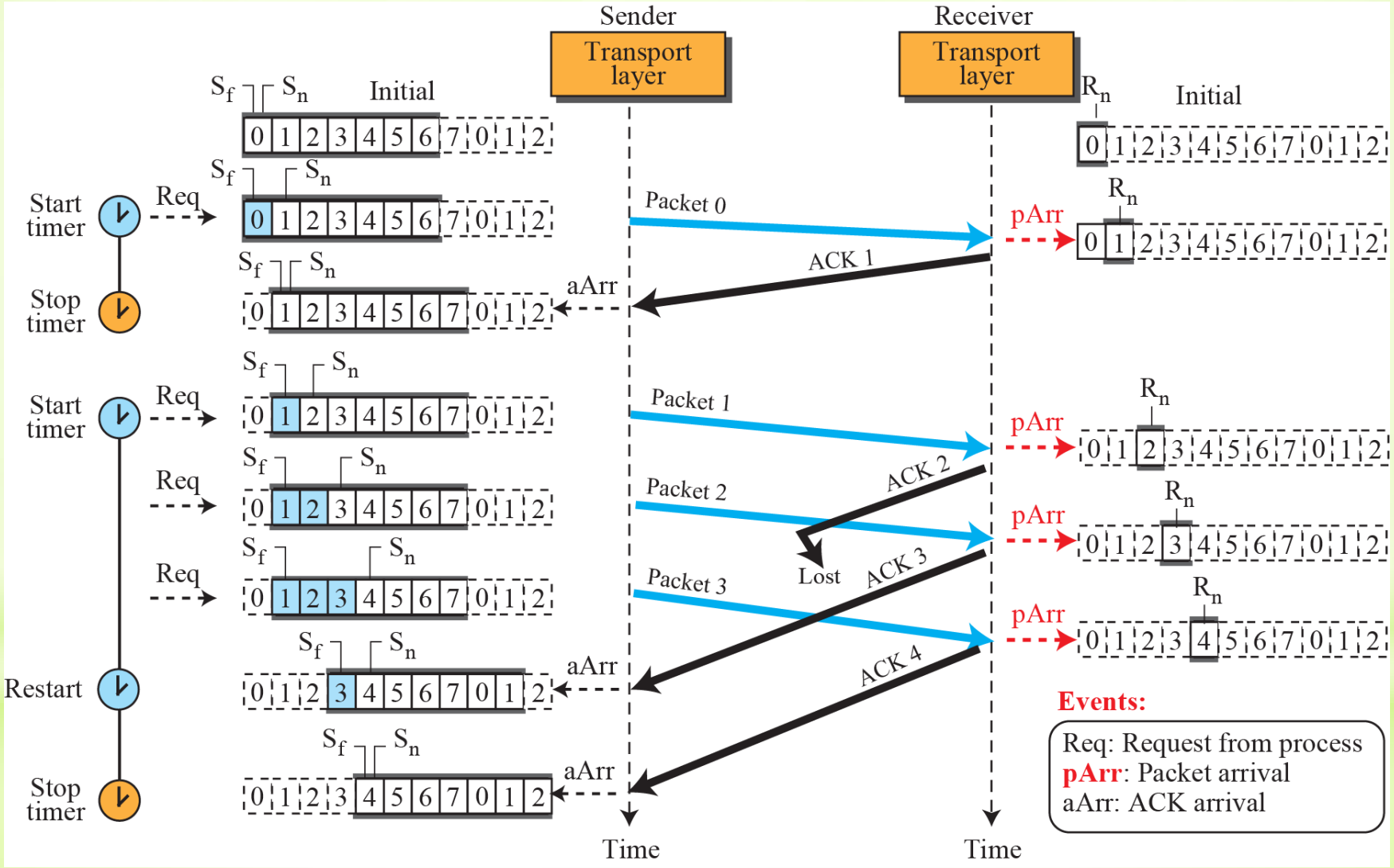
b. Window after sliding (an ACK with ackNo = 6 has arrived)

Receive window for Go-Back-N

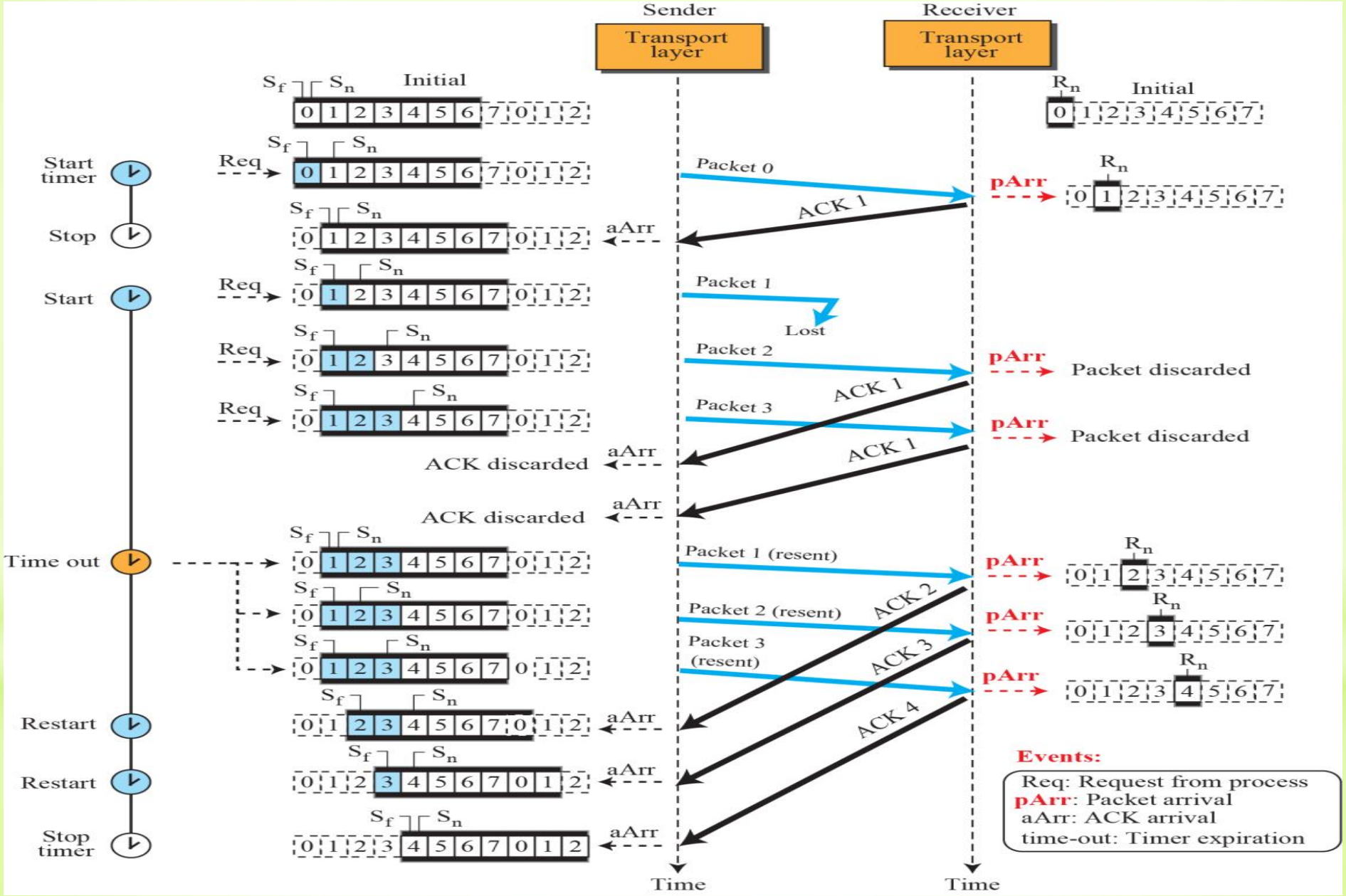
- The receive window makes sure that the correct data packets are received and that the correct acknowledgments are sent. In Go-back-N, the size of the **receive window is always 1**.
- Only one variable: R_n (receive window, next packet expected).
- Only a packet with a sequence number matching the value of R_n is accepted and acknowledged
- When a correct packet is received, the window slides, $R_n = (R_n + 1)$ modulo 2^m



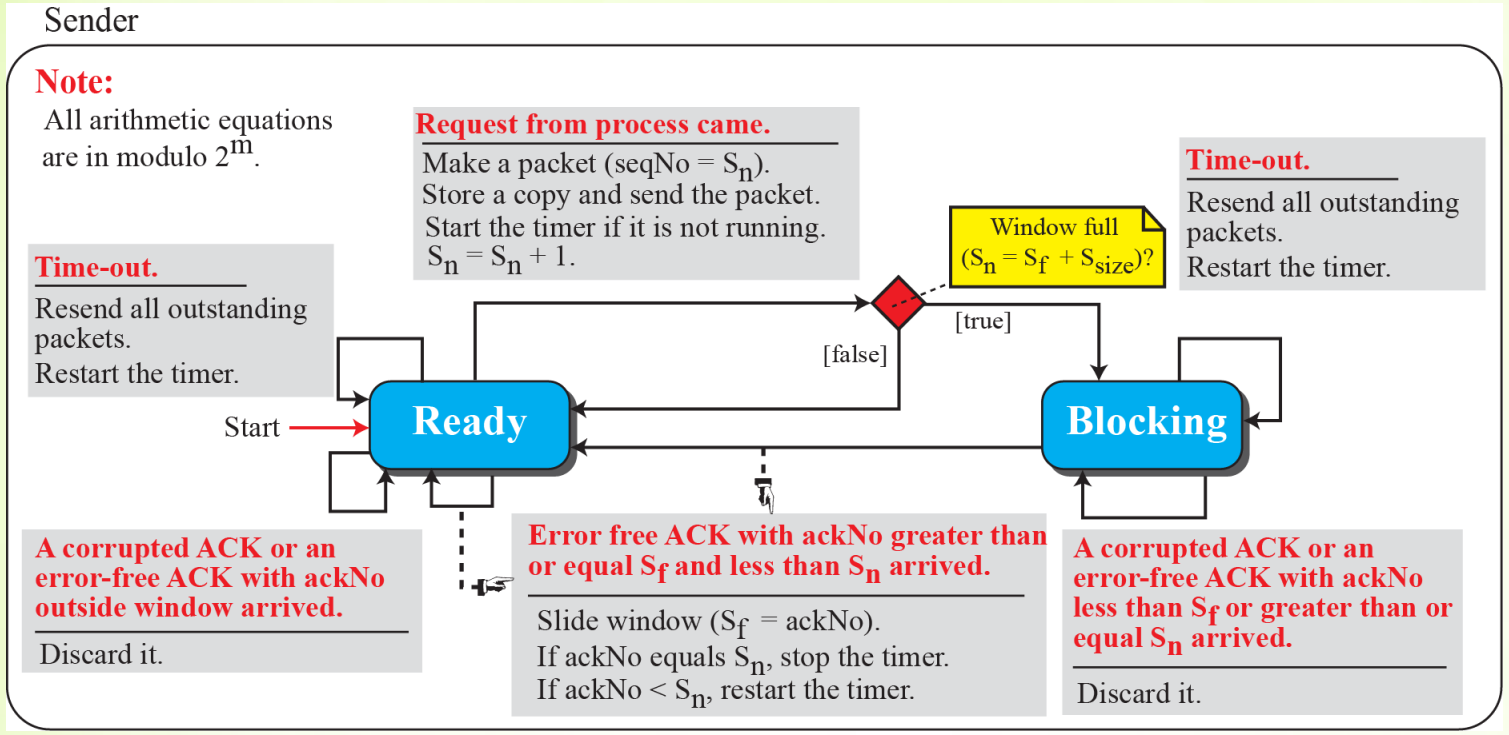
Flow diagram for reliability on senders side

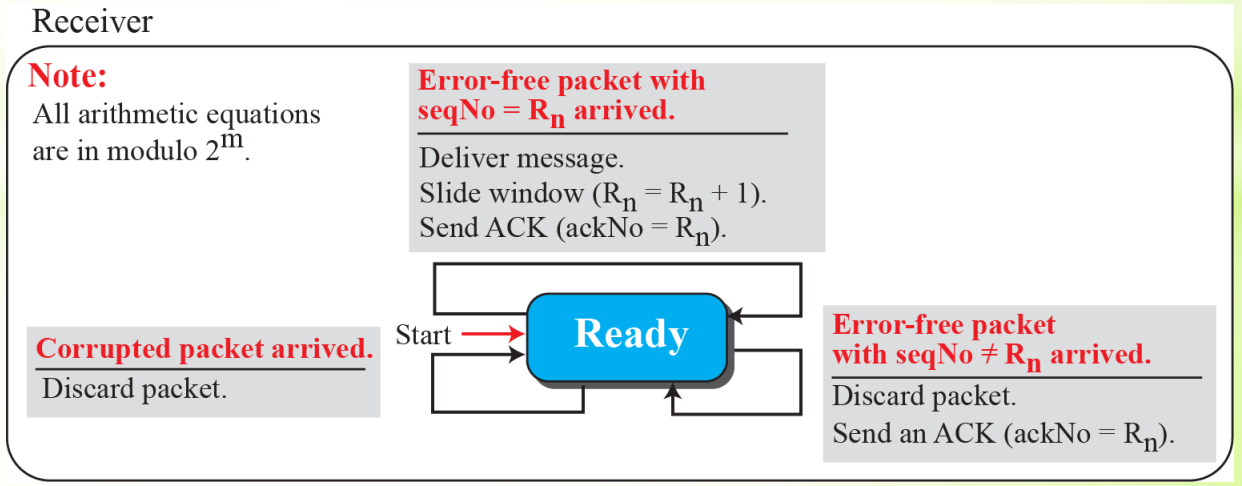


Flow diagram for unreliability in senders site

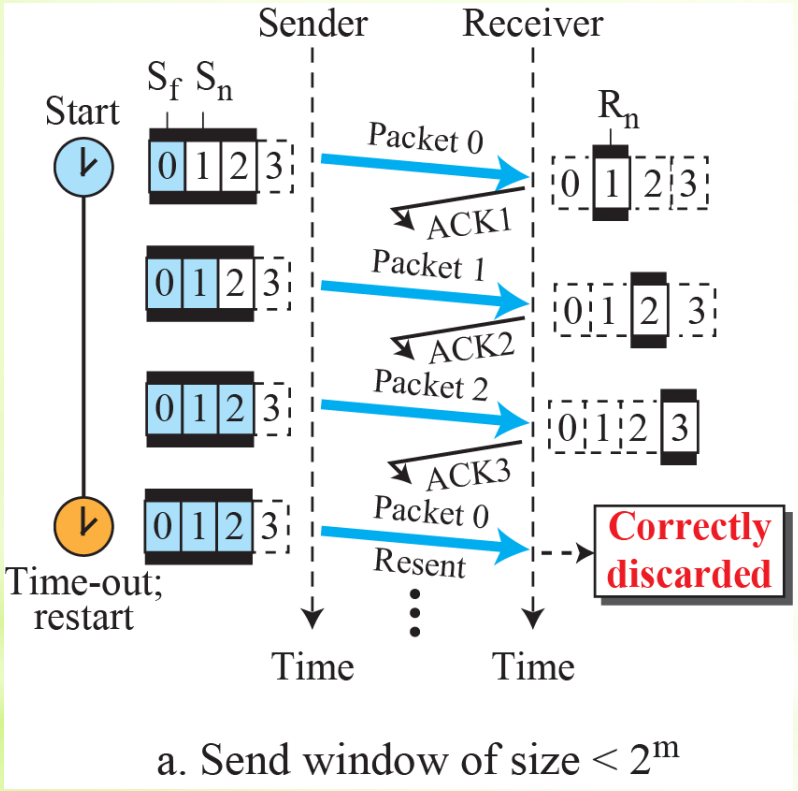


FSMs for the Go-Back-N protocol

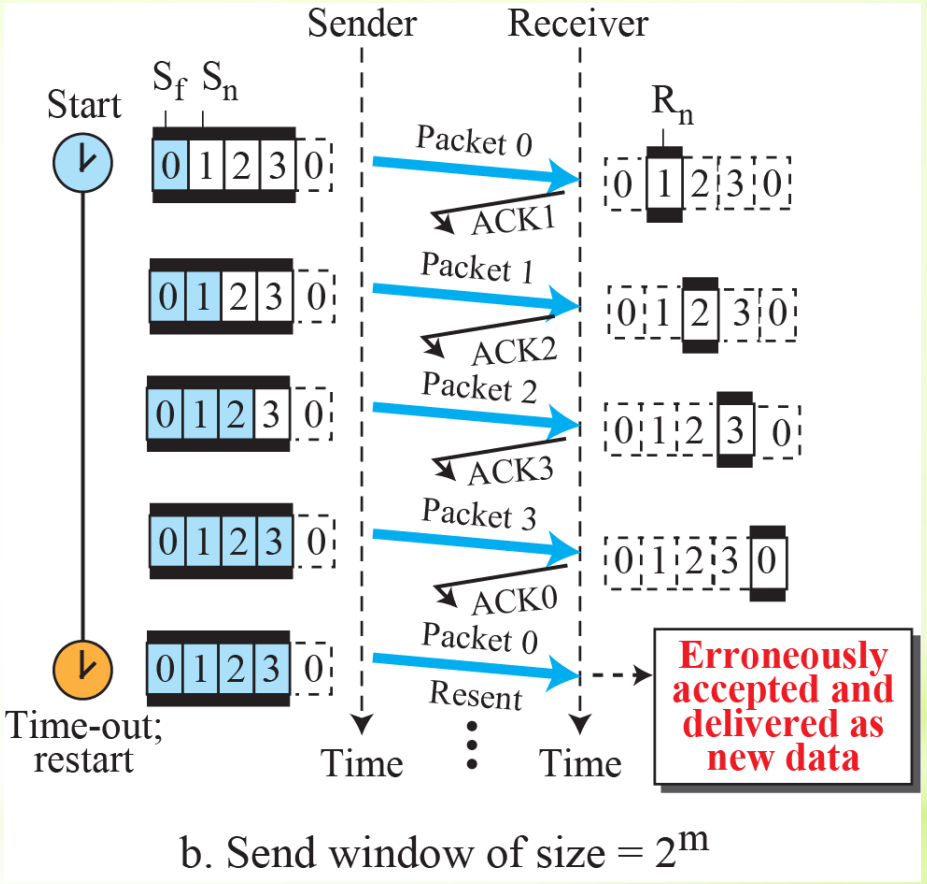




Send window size for Go-Back-N



a. Send window of size < 2^m



b. Send window of size = 2^m

- Cumulative acknowledgments can help if acknowledgments are delayed or lost

Shortfalls of Go-Back-N Protocol:

- 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.

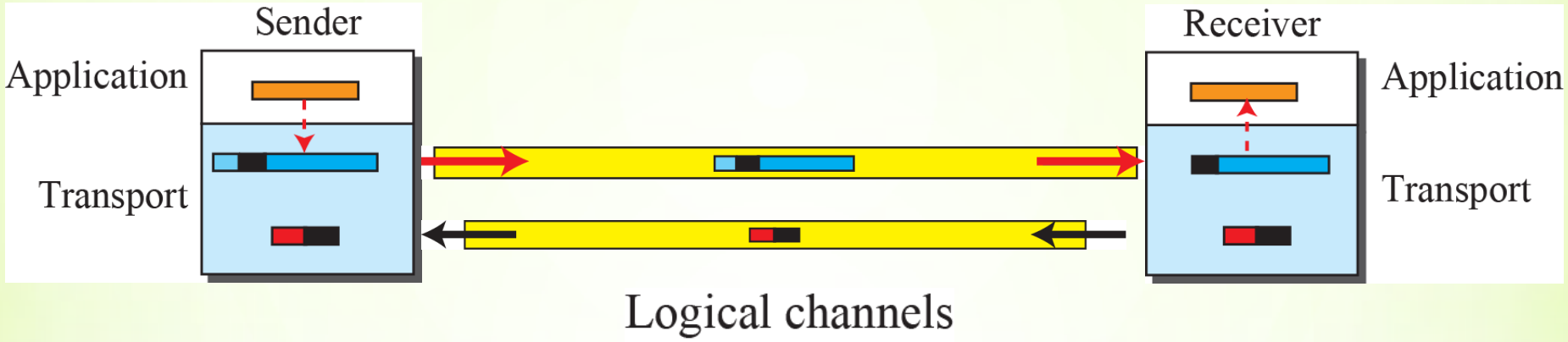
- If the network layer is losing many packets because of congestion in the network, the resending of all of these outstanding packets makes the congestion worse, and eventually more packets are lost. This has an avalanche effect that may result in the total collapse of the network.

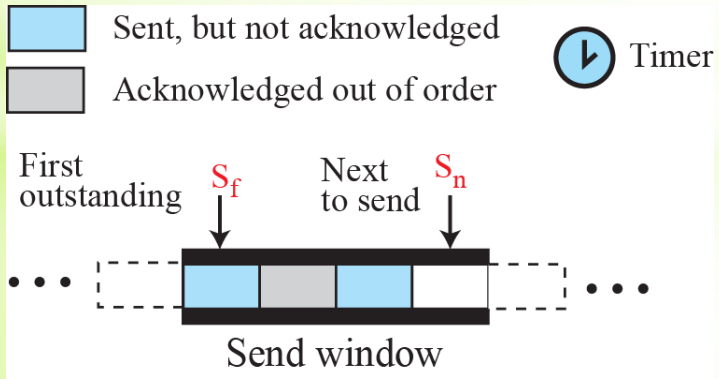
Selective Repeat protocol

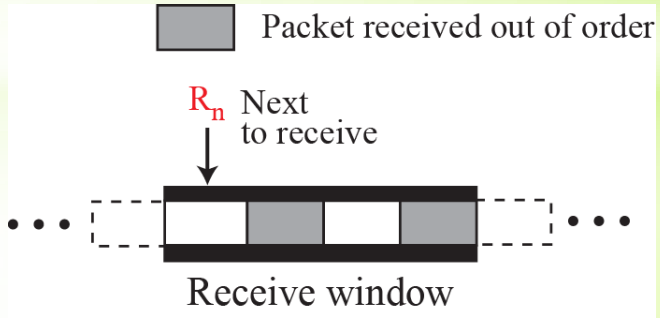
The Go-Back-N protocol simplifies the process at the receiver. The receiver keeps track of only one variable, and there is no need to buffer out-of-order packets; they are simply discarded.

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.

Outline of Selective-Repeat

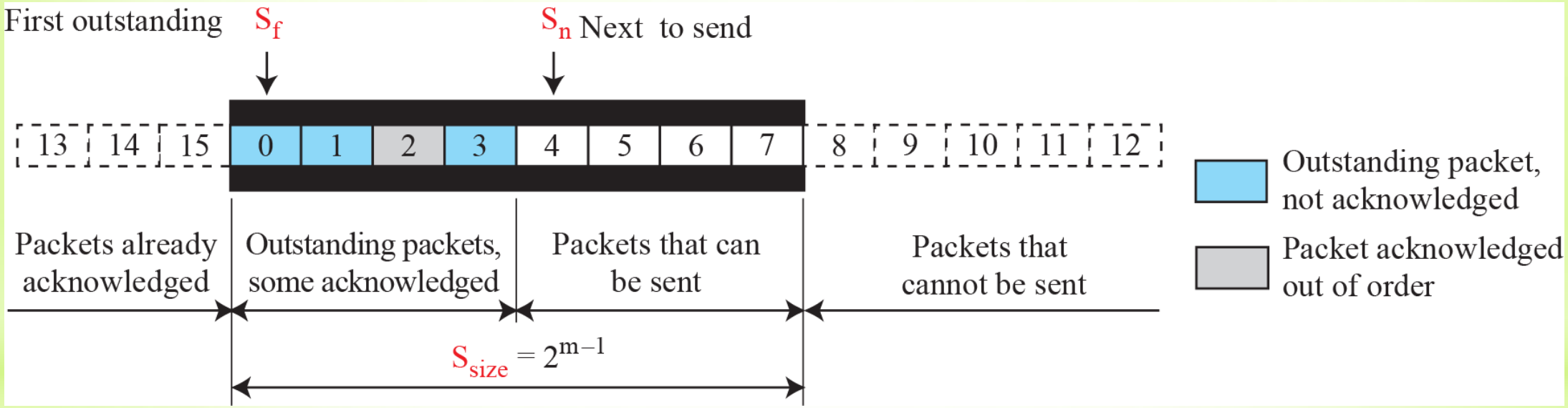






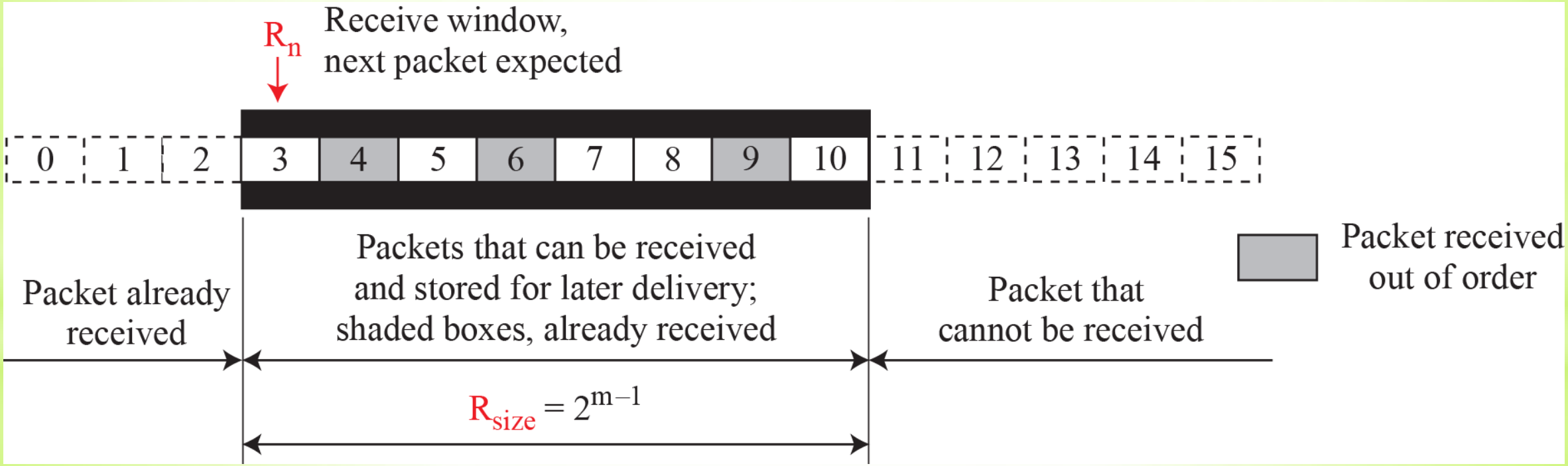
Send window for Selective-Repeat protocol

- The maximum **size of the send & receive window is 2^{m-1}.**



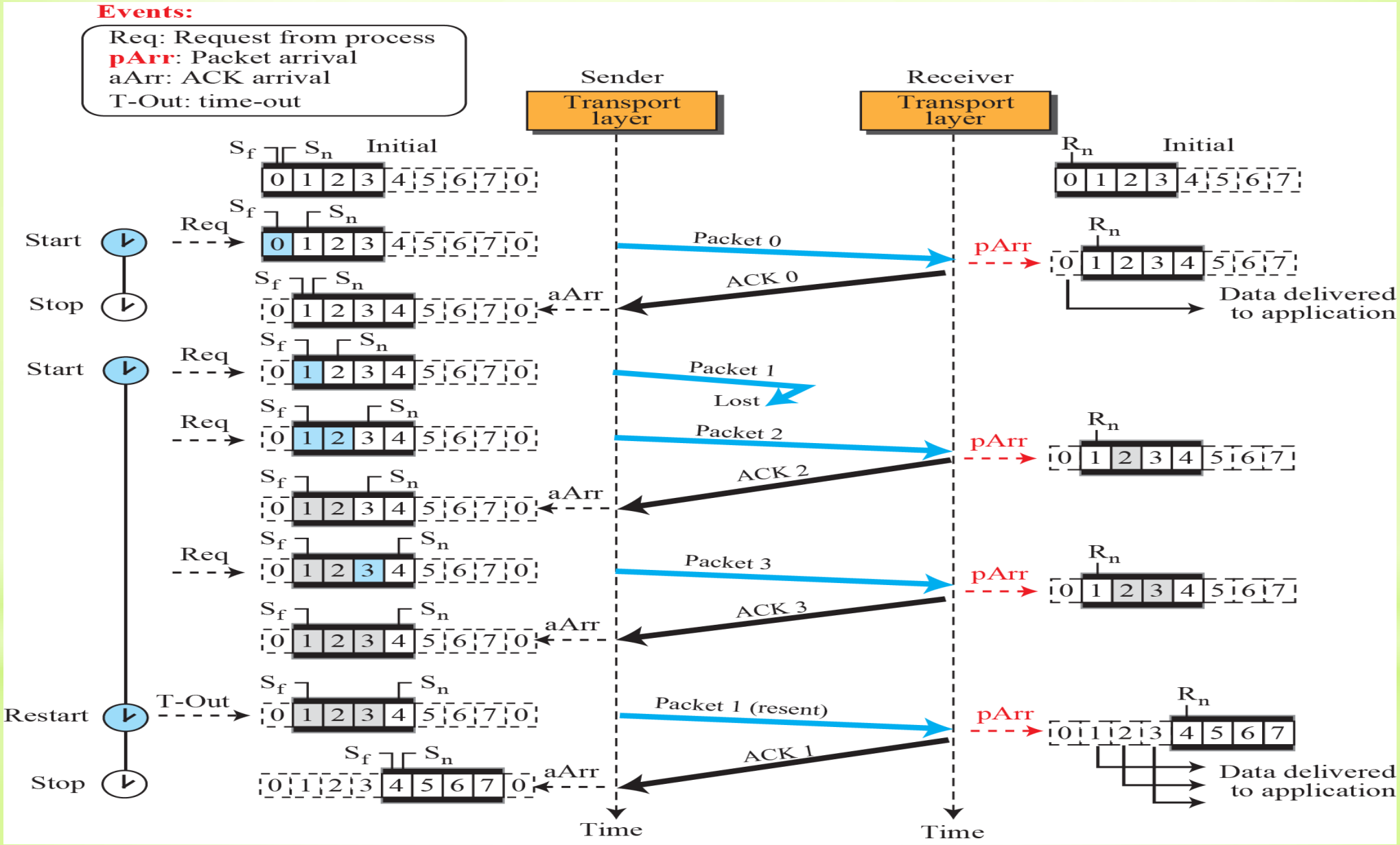
Receive window for Selective-Repeat protocol

- The Selective-Repeat protocol allows as many packets as the size of the receive window to arrive out of order and be kept until there is a set of consecutive packets to be delivered to the application layer.

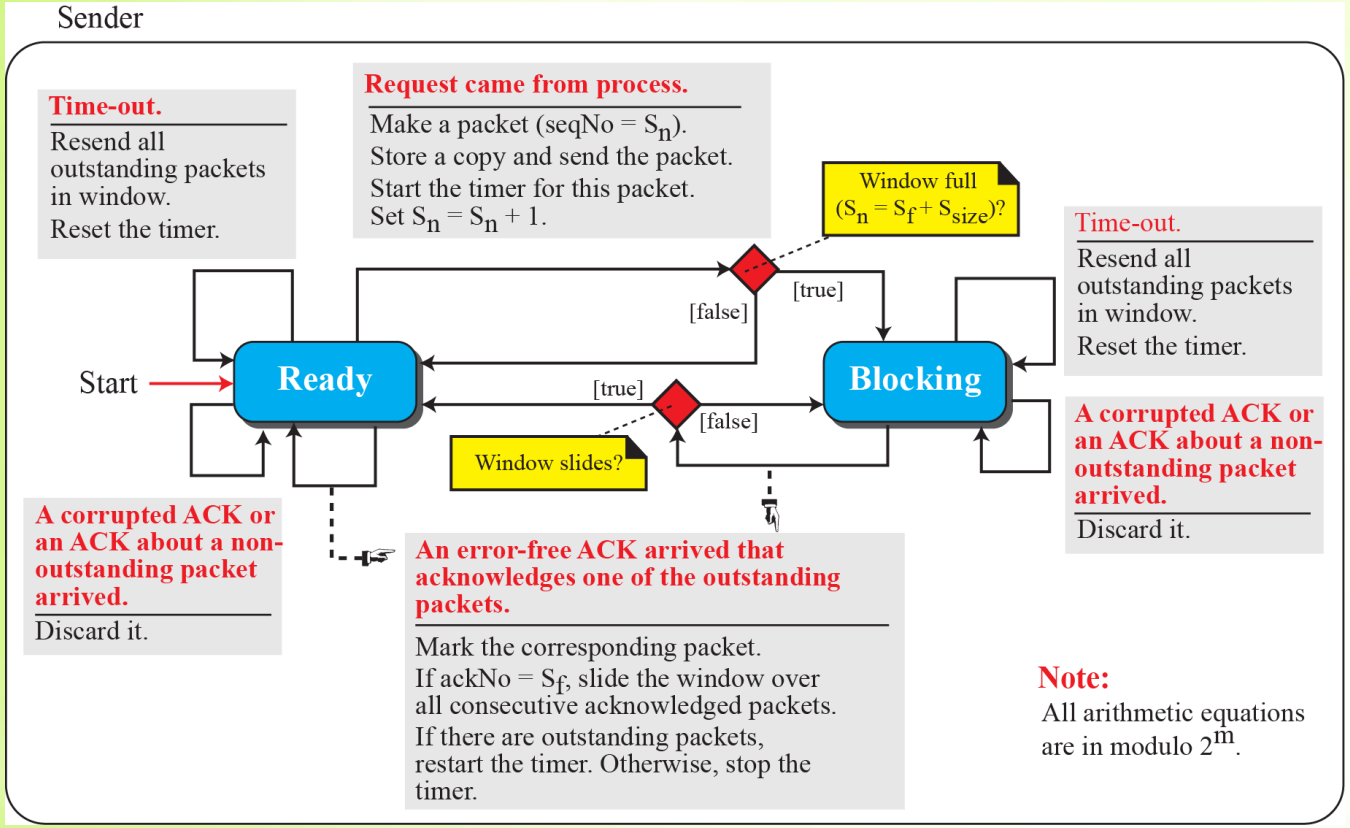


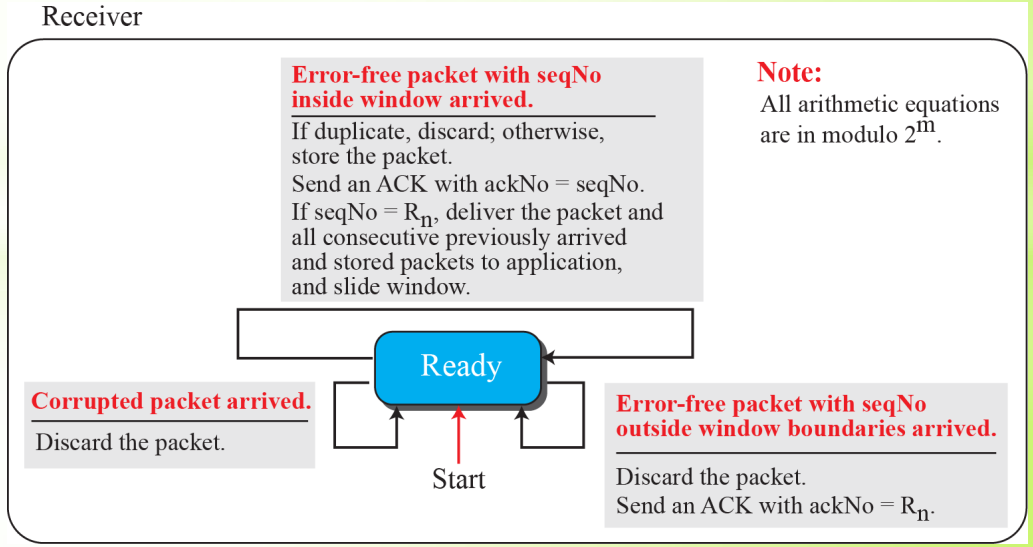
- Selective-Repeat uses one timer for each outstanding packet.
- GBN treats outstanding packets as a group; SR treats them individually.
- In the Selective-Repeat protocol, an acknowledgment number defines the sequence number of the error-free packet received.

Flow diagram for selective repeat

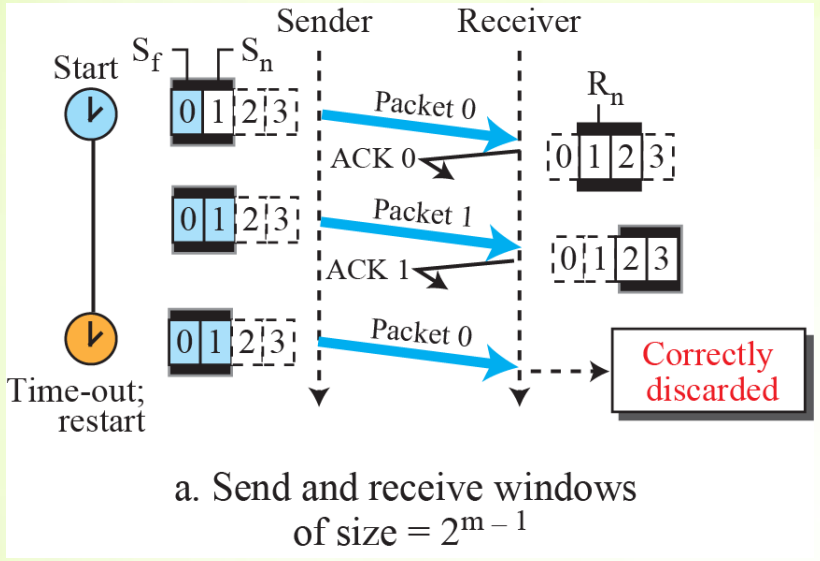


FSMs for SR protocol

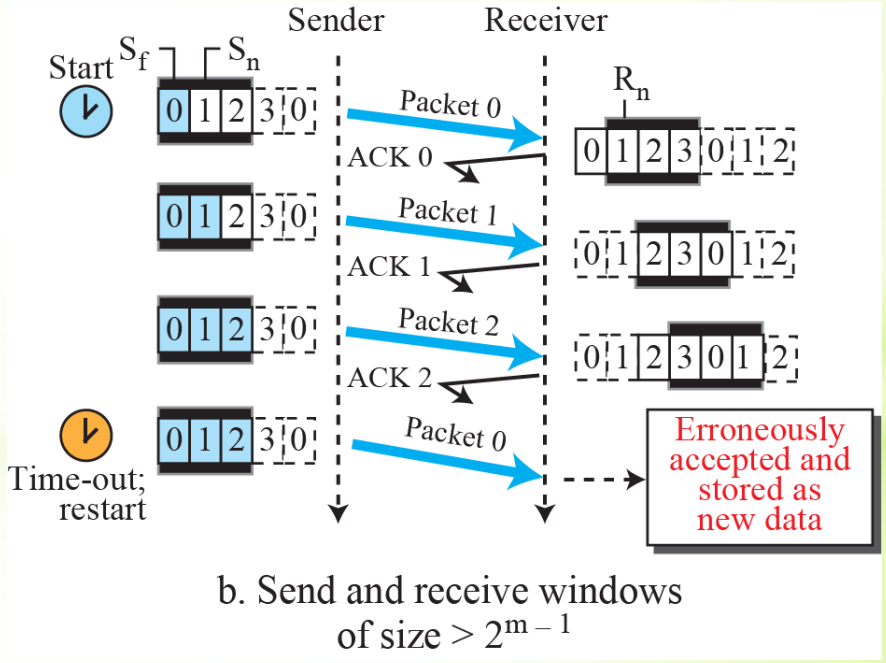




Selective-Repeat, window size



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

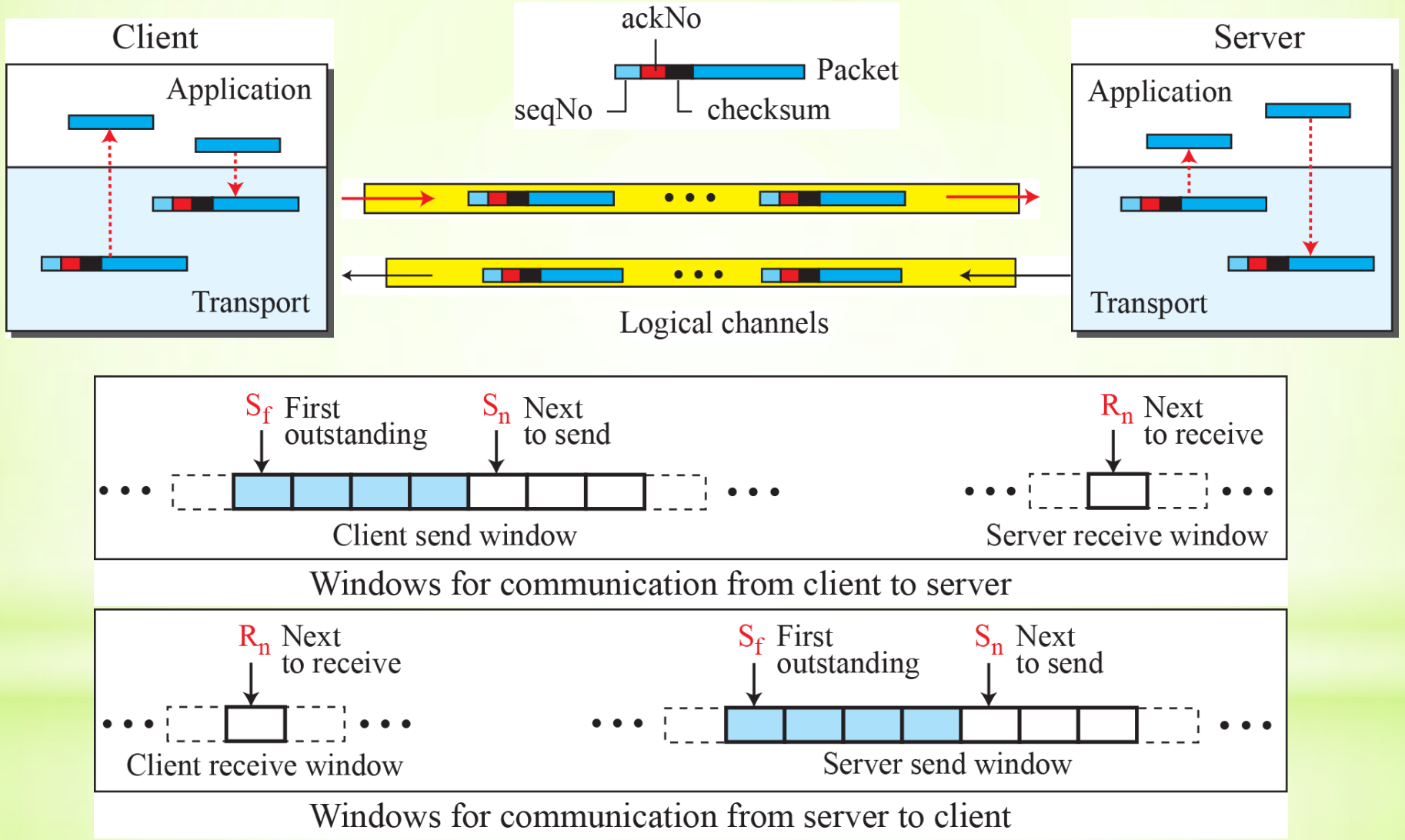


b. Send and receive windows of size > 2^{m-1}

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.

Design of piggybacking in Go-Back-N



Internet Transport-Layer Protocols