Chapter 3: Transport Layer

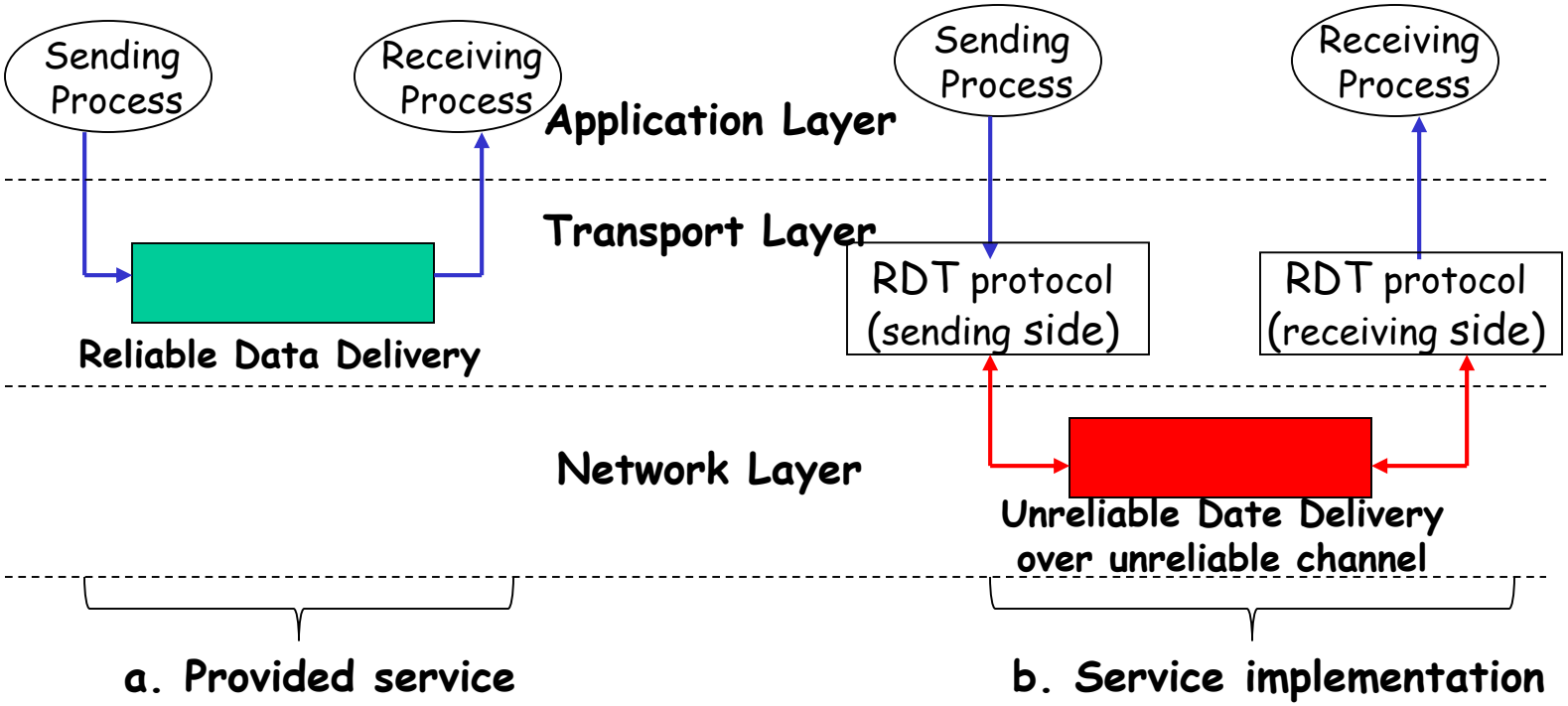
Principle of Reliable Data Transfer

Instructor: HOU, Fen

2025

Principles of Reliable Data Transfer (rdt)

- It is important in some applications



- How to provide the reliable data transfer on the top of an unreliable end-to-end network layer
- characteristics of unreliable channel will determine the complexity of reliable data transfer (RDT) protocol

Three Types of Channels (信道)

- Perfect channel: underlying channel is perfectly reliable
  - no bit errors
  - no loss of packets
- Channel with bit errors
  - All packets are received
  - Packets may be corrupted (i.e., bits may be flipped)
- "Lossy" channel: underlying channel not only corrupts bits in packets but also loses packets
  - Packets may be lost
  - Packets may be corrupted

Data Transfer over a Perfect Channel

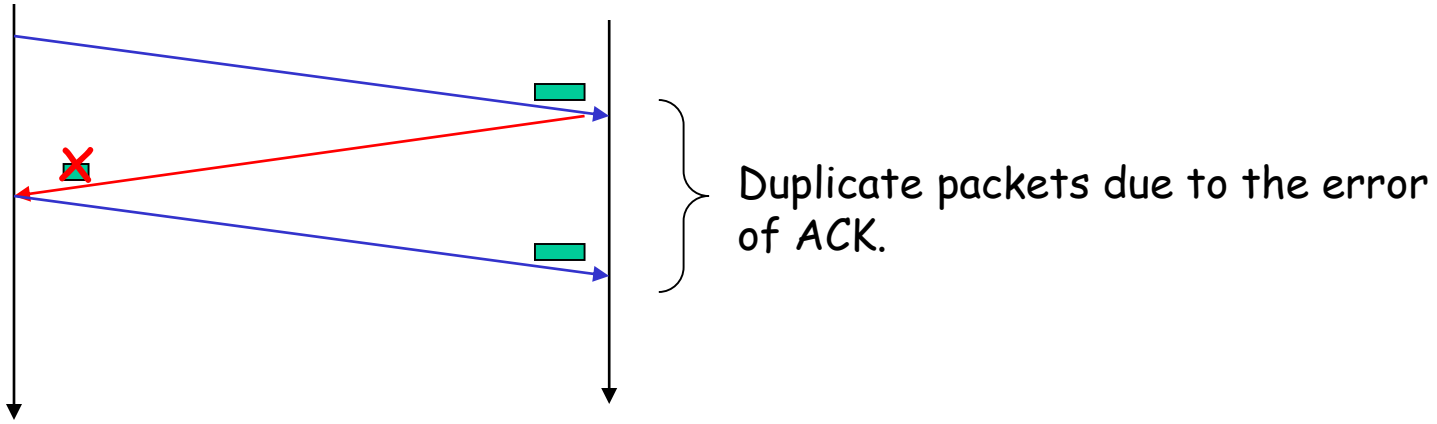
- underlying channel perfectly reliable
  - no bit errors
  - no loss of packets
- sender sends data into underlying channel
- receiver receives data from underlying channel

Data Transfer over Channel with Bit Errors

- Assumptions
  - All packets are received
  - Packets may be corrupted (i.e., bits may be flipped)
- How to recover from bit errors?
- Use ARQ (automatic repeat request自动重传请求) mechanism
  - *acknowledgements (ACKs):* receiver explicitly tells sender that packet is received correctly
  - *negative acknowledgements (NAKs):* receiver explicitly tells sender that packet is received with errors
  - sender retransmits packet on receipt of NAK
  - Reliable data transfer based on such mechanism (error detection, receiver feedback, retransmission or transmission) is known as ARQ mechanism.

Data Transfer over Channel with Bit Errors

- What happens when ACK or NAK has bit error?
  - Resend the current data packets → Duplicate packets

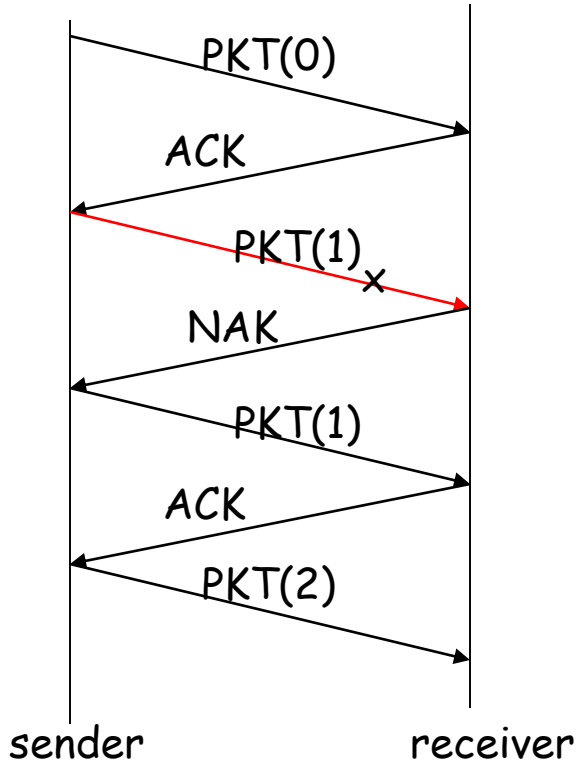


- Handling duplicate packets
  - Sender adds sequence number to each packet
  - Sender retransmits current packet if ACK/NAK is corrupted
  - Receiver deletes (doesn't deliver up) duplicate packet

Handling Duplicate Packets

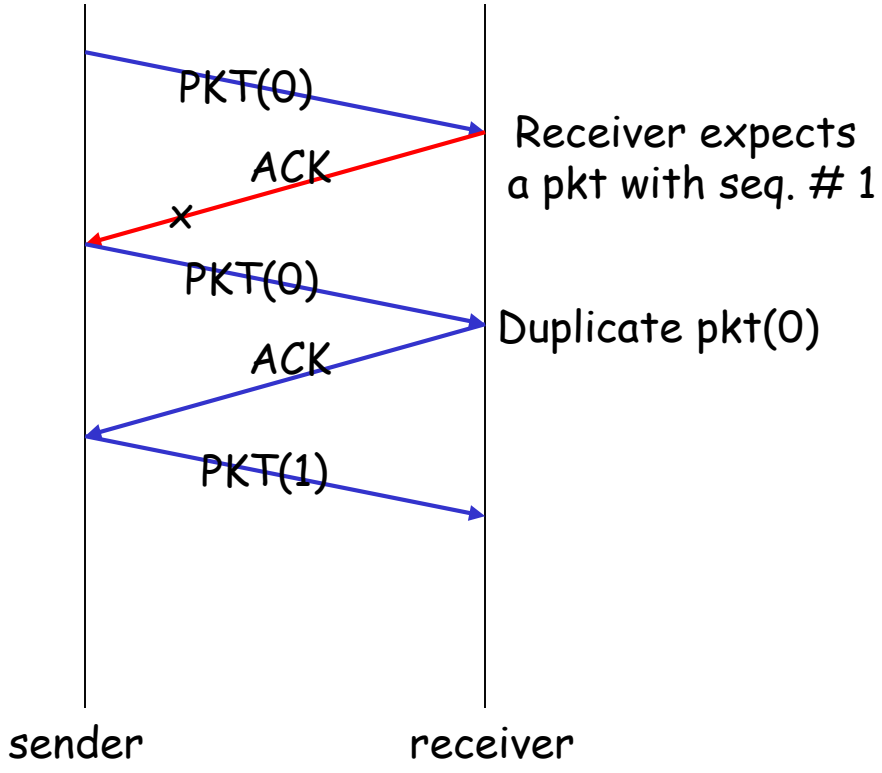
Sender

Add sequence # to each packet.



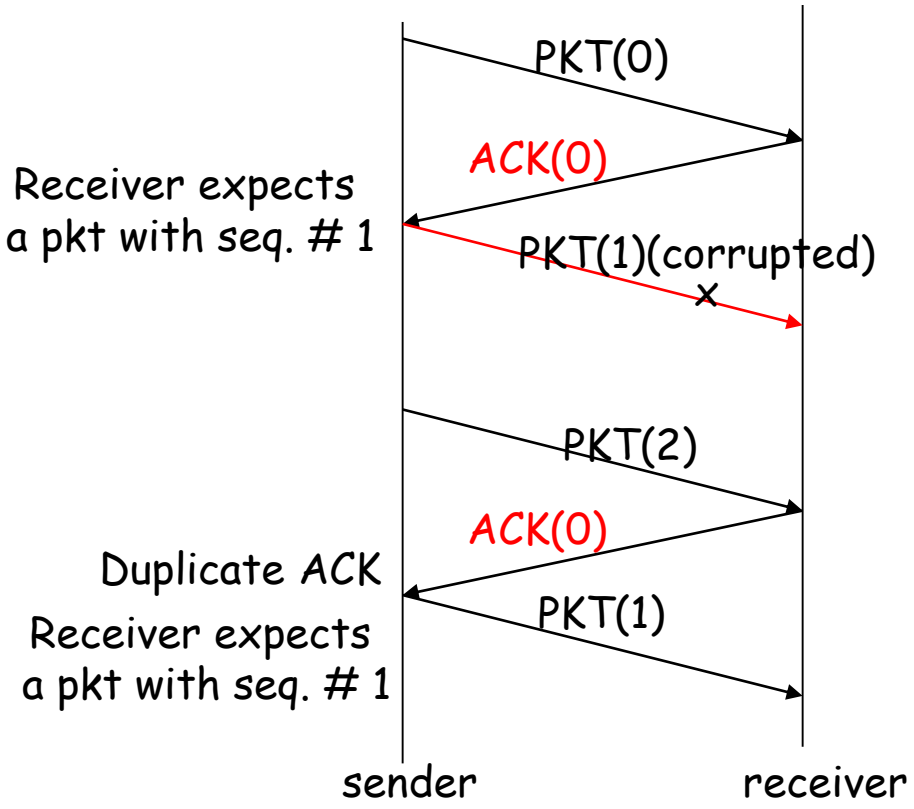
Receiver

Check if the received packet is duplicate. It knows the expected packet sequence number.



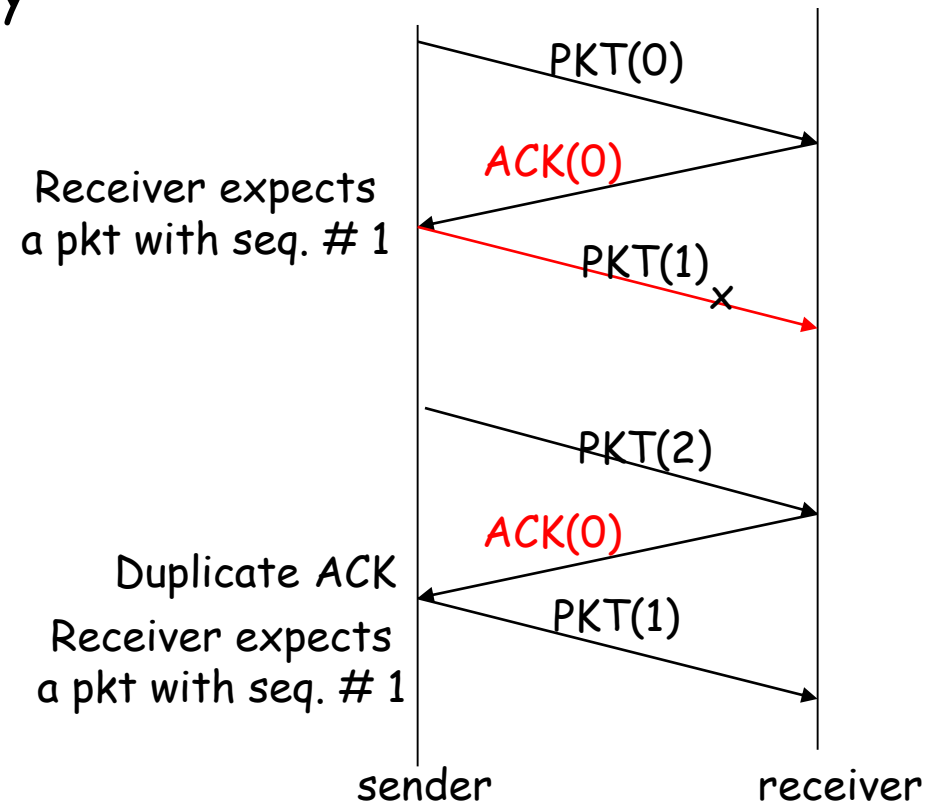
A ACK-only(NAK-free) Protocol

- using ACKs only
- instead of NAK, receiver sends ACK for the correctly received packet with the highest in-order sequence number
  - receiver must *explicitly* include seq # of pkt being ACKed into the ACK message
- The sender that receives two ACKs for the same packet (that is, the sender receives duplicate ACKs) knows that the receiver did not correctly receive the packet following the packet that is being ACKed twice, then retransmit this packet.



A ACK-only Protocol

- With infrequent data transmission, ACK-only protocol can have a long error recovery time.

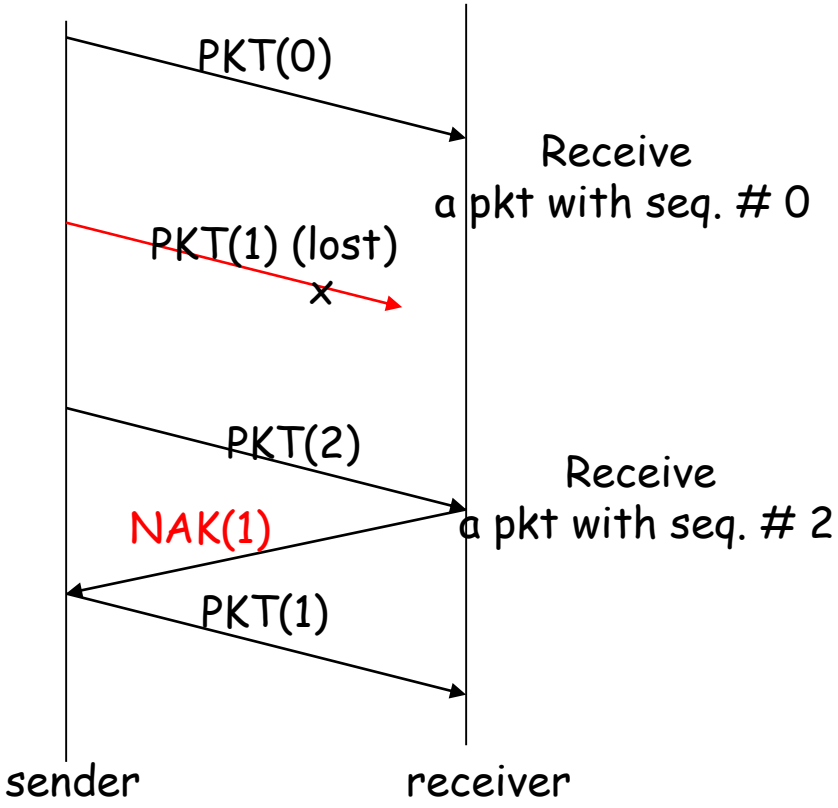


The Case of "Lossy" Channels

- Assumption: underlying channel not only corrupts bits in packets but also loses packets (data or ACKs)
  - Packets may be corrupted
  - Packets may be lost
- How to identify and handle the packet loss

A NAK-only Protocol

- using NAKs only
- instead of ACK, receiver sends NAK for the incorrectly received packet with the lowest sequence number
  - receiver must *explicitly* include seq # of pkt being NAKed into the NAK message
- Detection of a packet loss: The receiver that receives two packets with a gap in their sequence numbers knows the lose of packets.
- With infrequent data transmission, NAK-only protocol can have a long error recovery time.



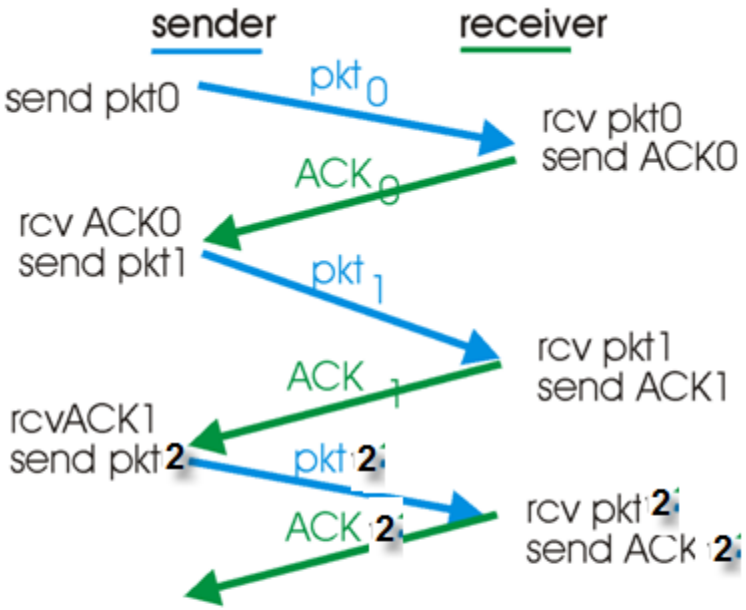
The case of "Lossy" Channels

- Use timer to identify and handle the loss of a packet/ACK/NAK?
  - Set a timer: sender waits "reasonable" amount of time for ACK/NAK (a Time-Out).
  - If the ACK for a transmitted packet is not received within the duration of the timer for this packet, the packet (or its ACK or NAK) is assumed to be lost.

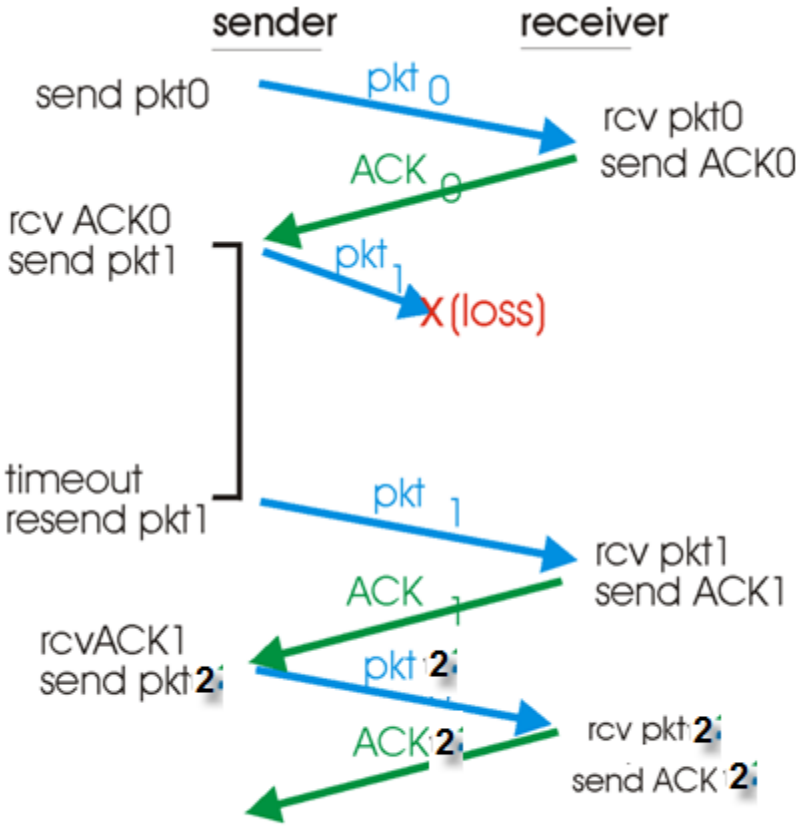
The case of "Lossy" Channels

- Approach used in the case of "lossy" channel
  - Set a timer: sender waits "reasonable" amount of time for ACK (a Time-Out).
  - Sender adds sequence number to each packet.
  - Receiver must specify sequence # of packet being ACKed/NAKed.
  - Sender retransmits current packet if it does receive the ACK after timeout.
  - If the receiver receives duplicated packets, the receiver discards it.

Examples

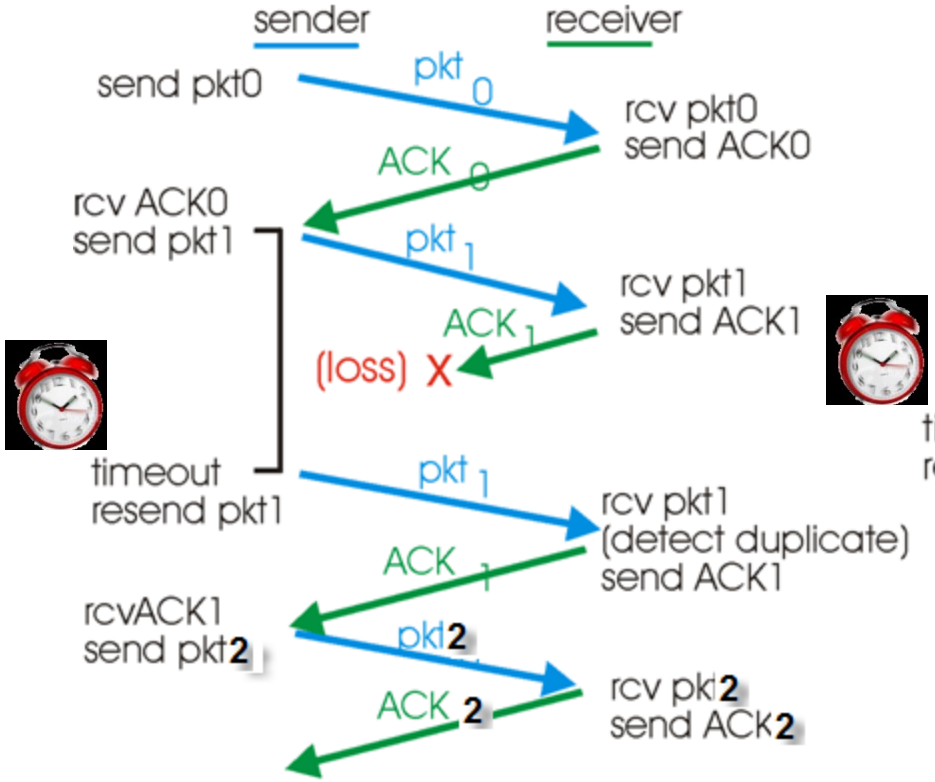


(a) operation with no loss

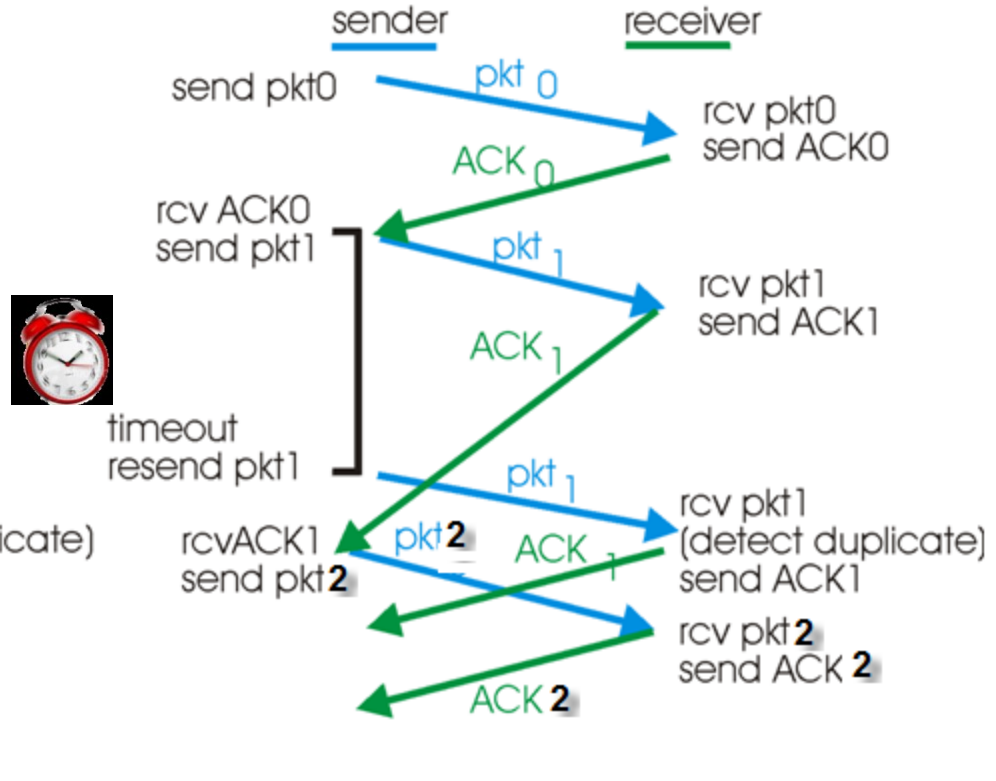


(b) lost packet

Examples



(c) lost ACK



(d) premature timeout

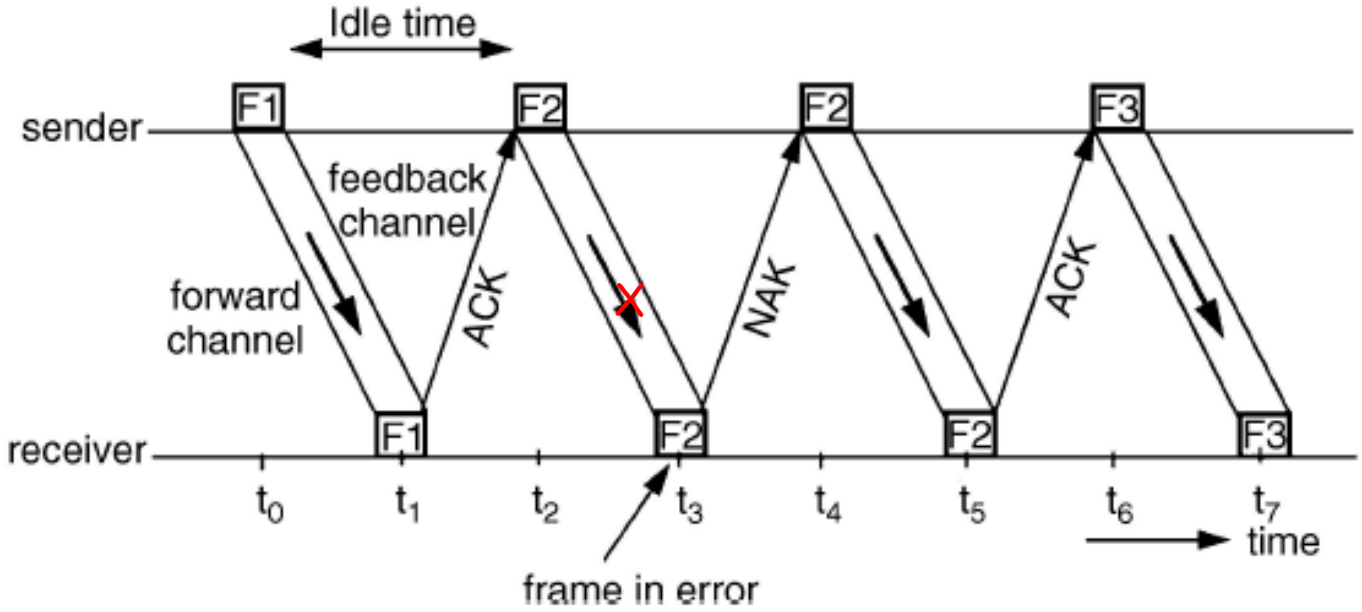
How to set the value of time out is a key issue!

Some Reliable Data Transfer (rdt) Protocols

- Stop-and-wait protocol
- Pipelined protocol
  - Go-back-N
  - Selective repeat

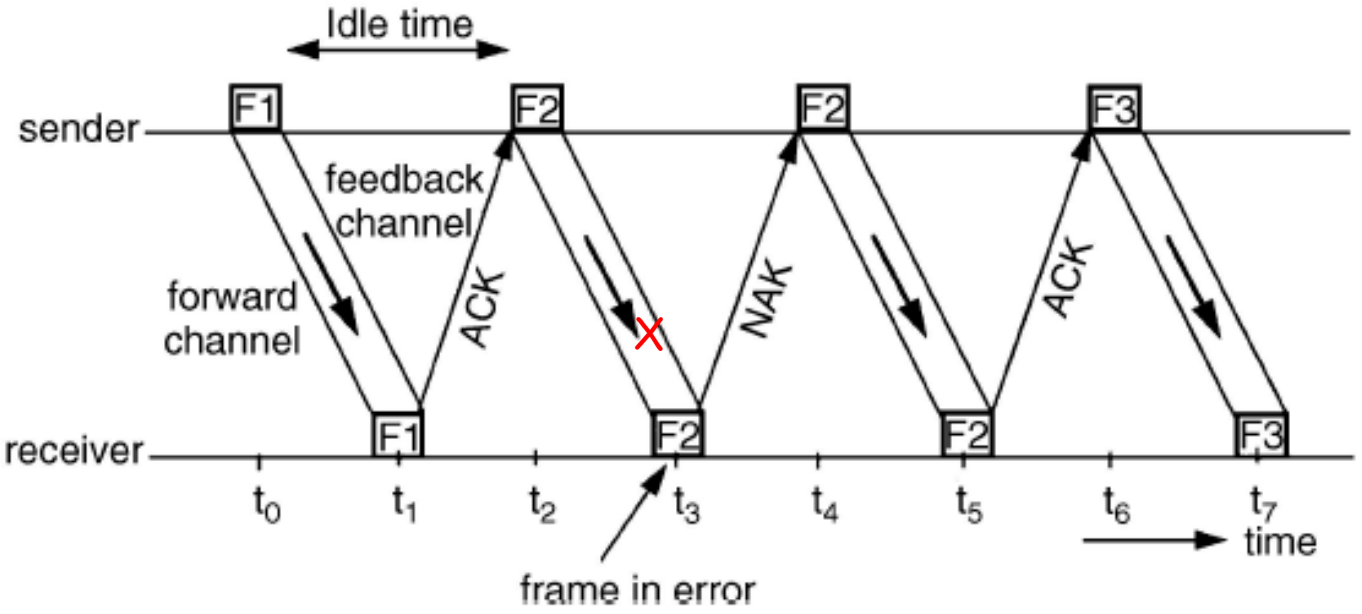
Stop-and-Wait Protocol: Operation

- Sender transmits a single frame
- Wait for ACK
- If received frame is damaged, discard it
  - Send back a NAK
- If the sender receives a NAK or no ACK before timeout, retransmit the frame

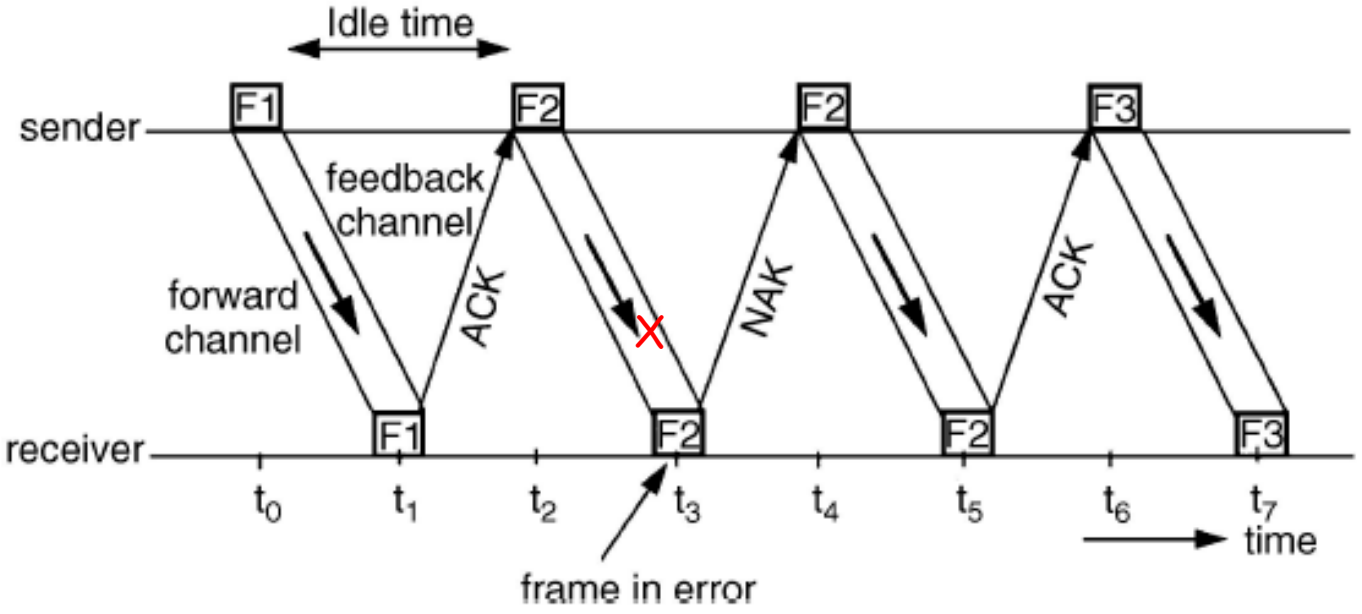


Stop-and-Wait Protocol: Operation

- If ACK is damaged, sender will not recognize it
  - Sender will do the retransmission
  - Receiver gets two copies of frame, delete the duplicated one.

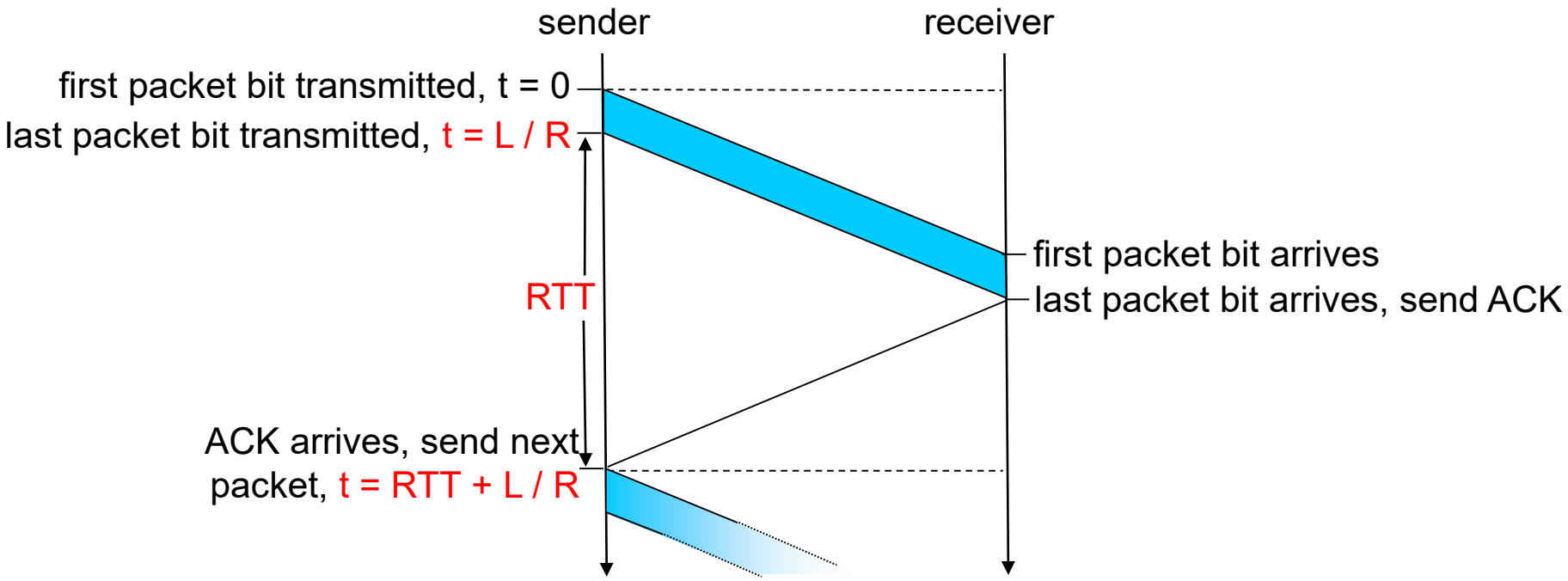


Stop-and-Wait Protocol: Operation



- Stop-and-wait allows the sender to only have a single unACKed packet at any time.
- This method requires the sender to maintain a copy of a transmitted packet until an ACK is received.

stop-and-wait operation



Link/Channel Utilization/Efficiency = (L/R)/(RTT+L/R)

where R is the transmission rate, L is the packet length, and RTT (Round-trip Time) is also called the round-trip propagation delay.

Example

- Example: Transmission Rate （R）of 1 Mbps, the round trip propagation delay of 92ms, 1Kbyte(字节)/packet (L): channel utilization is

$$T_{transmit} = \frac{L \text{ (packet length in bits)}}{R \text{ (transmission rate, bps)}} = \frac{8\text{kbit}}{1\ 0^3\text{kb/sec}} = 8 \text{ ms}$$

$$DTT + T_{transmit} = 92\text{ms} + 8\text{ms} = 100\text{ms}$$

$$U = \frac{L / R}{RTT + L / R} = \frac{8\text{ms}}{100\text{ms}} = 0.08$$

  - Transmitting 1Kbyte every 100 ms -> 80Kbps throughput (the link bandwidth is 1 Mbps) -> only 8 % of channel capacity is used, waste 92% of resource ->Low efficiency, low utilization

Example

- Example: Transmission Rate is 1 Gbps; the propagation delay is 15 ms; the packet size is 8000bit/packet. Calculate the channel utilization

$$T_{transmit} = \frac{L \text{ (packet length in bits)}}{R \text{ (transmission rate, bps)}} = \frac{8\text{kbit}}{1\ 0^9\text{bits/sec}} = 8 \text{ microsecs}$$

$$RTT + T_{transmit} = 30\text{ms} + 0.008\text{ms} = 30.008\text{ms}$$

$$U = \frac{L / R}{RTT + L / R} = \frac{0.008}{30.008} = 0.00027$$

- That is, Throughput on 1 Gbps link = 0.00027 of 1 Gbps = 270 kbpt. Low efficiency, low utilization
- Network protocol limits use of physical link resources.

Stop-and-Wait Protocol

- Features
  - It has a timer in the implementation
  - Timer should be set for each individual packet
  - Only 1 packet is sent at a time
  - No pipelining
  - Sender window size is 1
  - Receiver window size is 1
  - Minimum number of the sequence numbers is 2
- Advantage
  - Simple
- Disadvantage
  - Inefficient: efficiency and utilization are very low (e.g.,8%)

Exercise

- If the RTT is 45ms, packet size is 1 KByte, the transmission rate of the link is 1.5Mbps.
  - Calculate the link/channel utilization with Stop and Wait protocol?

- Solution
  - Transmission Time: $(1 \times 10^3 \times 8) / (1.5 \times 10^6)$ =5.33 ms
  - Link Utilization: 5.33/(5.33 + 45) =10.5%

Exercise

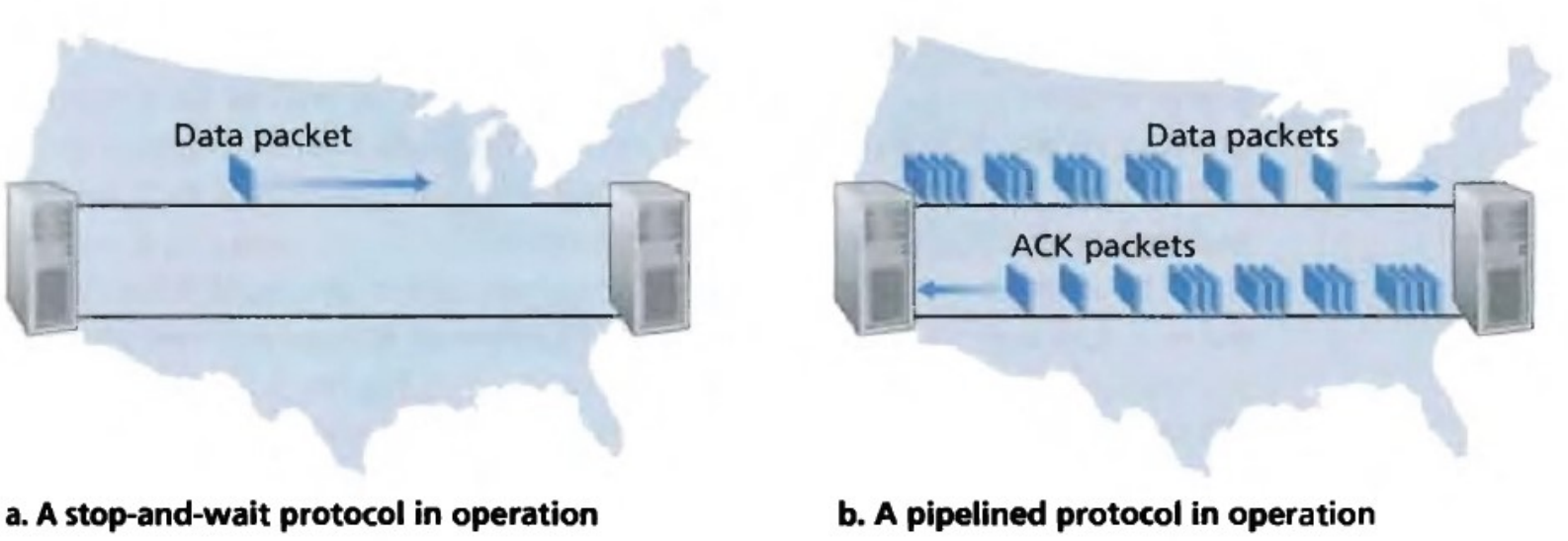
- A channel has transmission rate of 4 Kbps and one way propagation delay of 20 msec. The channel uses stop and wait protocol. To get a channel efficiency of at least 50%, the minimum packet size should be:
  - A: 80 bytes
  - B: 80 bits
  - C: 160 bytes
  - D: 160 bits
- Solution
  - D

How to Address the Low Efficiency of Stop-and-Wait Protocol

- Solution: the sender is allowed to send multiple packets when waiting for acknowledgements
- Window size: the maximum allowable number of unACKed packets.
- Pipelining: sender allows multiple, "in-flight", yet-to-be-acknowledged packets.
- Two features of pipelined protocol
  - The range of sequence numbers must be increased.
  - The sender and the receiver may have to buffer more than one packet.

Pipelined protocols

- Two generic forms of pipelined protocols
  - *go-Back-N*
  - *selective repeat*



a. A stop-and-wait protocol in operation

b. A pipelined protocol in operation

Figure 3.17 ♦ Stop-and-wait versus pipelined protocol

Go-Back-N

- It introduces a window of size N: N is called as window size. It specifies the maximum allowable number of unACKed packets.
- It allows up to N unACKed packets in the network
  - Sender can transmit N packets into the network before receiving an ACK.
  - Base: Seq. # of the oldest unACKed packet.
  - Nextseqnum: the smallest unused seq. #.
  - Window slides forward over the seq. # space.

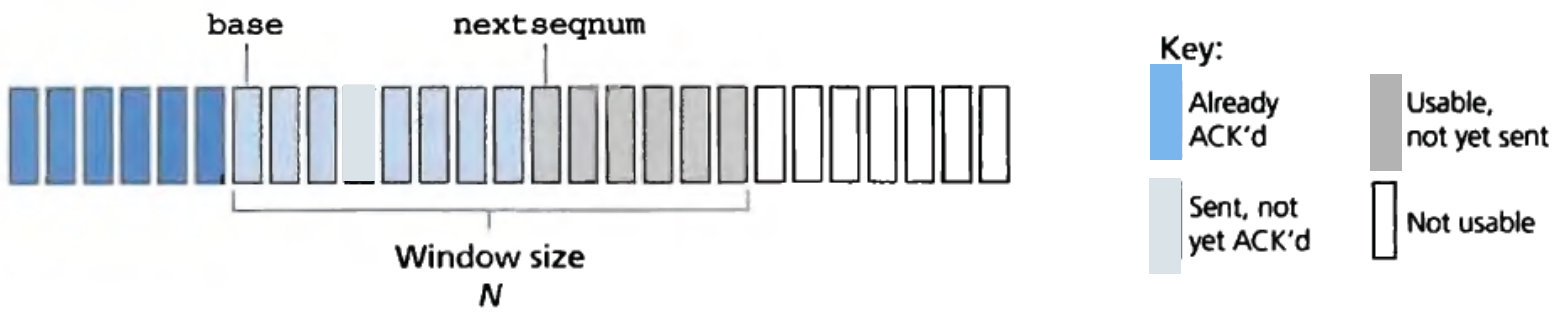


Figure 3.19 ♦ Sender's view of sequence numbers in Go-Back-N

Go-Back-N

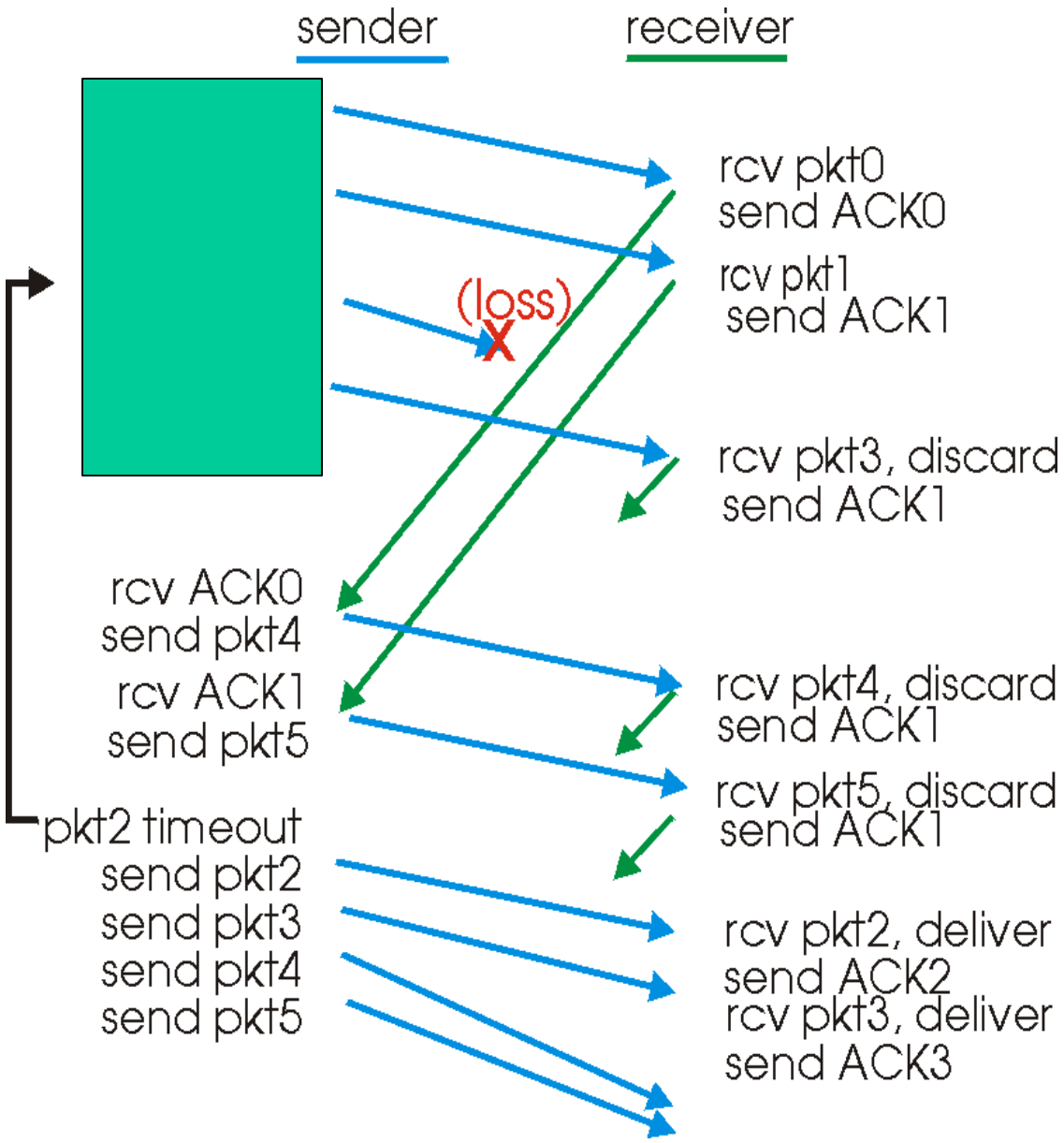
- Concepts
  - Window size
    - specify the maximum allowable number of unACKed packets.
  - Sequence number:
    - Sequence number is required for a receiver to find out whether an arriving packet contains new data or it is a retransmission.
    - In practice, a packet's seq. number is carried in a fixed-length field in the packet header.
  - Cumulative acknowledgement
  - How to deal with out-of-order packets
  - Timer and timeout
    - To handle packet loss.

Go-Back-N

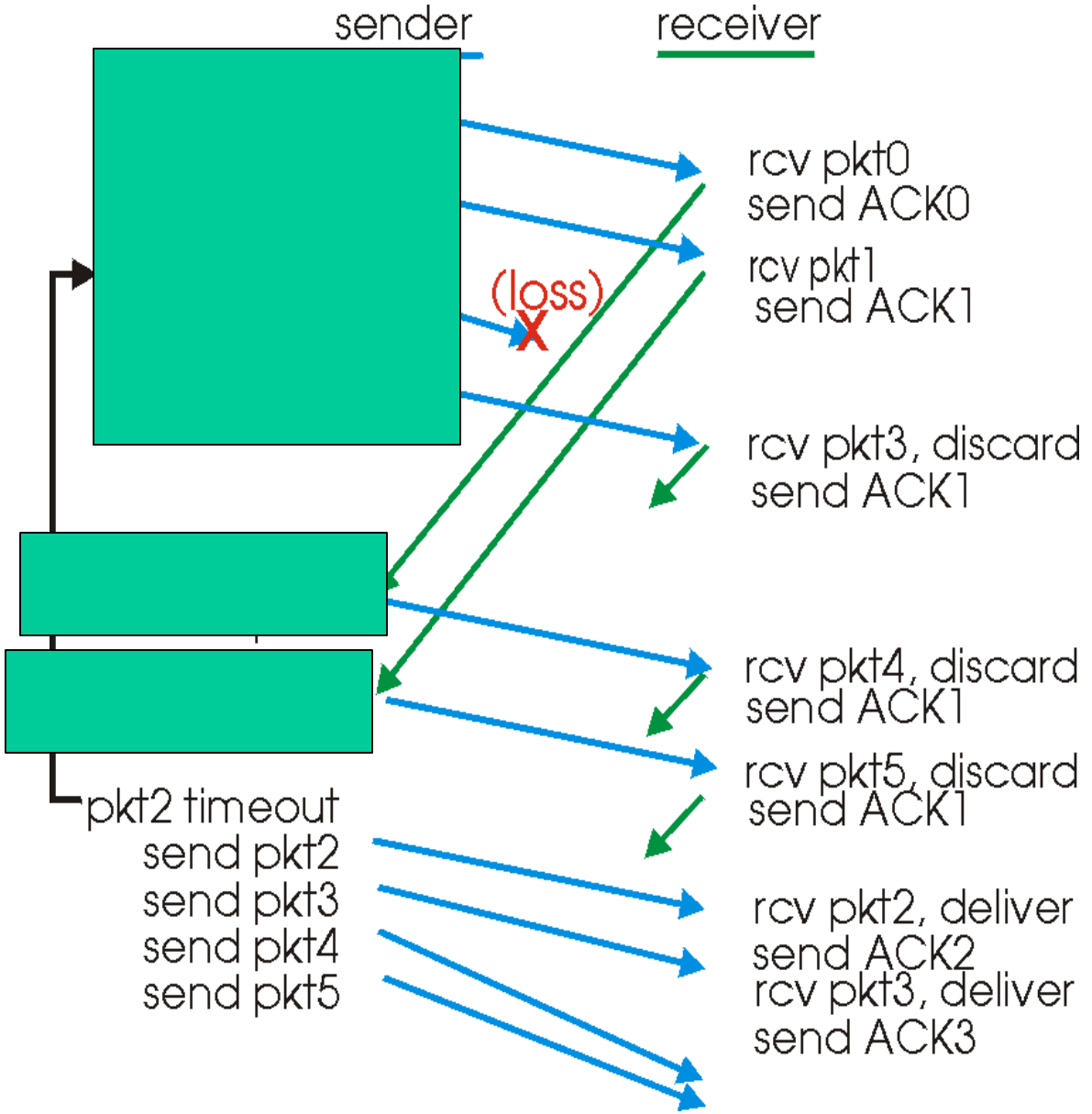
- Sender Operation:
  - Label each packet with a sequence number
  - A window is a collection of adjacent sequence numbers; The size of the collection is the sender's window size
  - If window is not full, sender transmits packets
  - ACKs are cumulative:
    - ACK all pkts up to, including seq. # n, called as "cumulative ACK".
    - Doesn't ACK packet if there is a sequence gap.
  - Delete the out-of-order packets: deal with the out-of-order packets by simply discarding these packets.
  - Timer and timeout
    - Uses a single timer – Sender has the timer for the oldest unacked packet.
    - On timeout, sender retransmits all unacked packets.

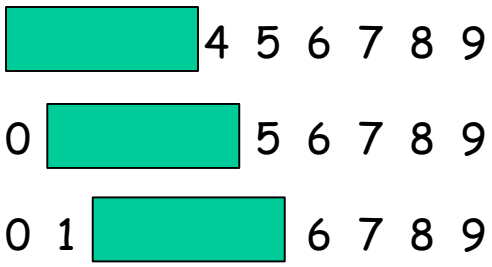
GBN in action

- Window size N (here N=4):
  - Label each packet with a sequence number.
  - A window is a collection of adjacent sequence numbers.
  - The size of the collection is the sender's window size.
  - If window is not full, transmit.



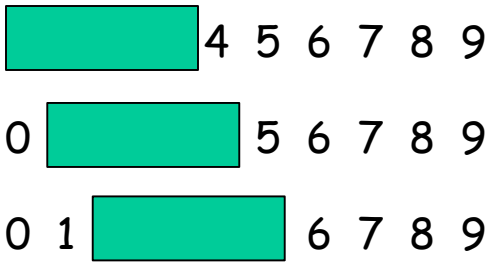
GBN in action

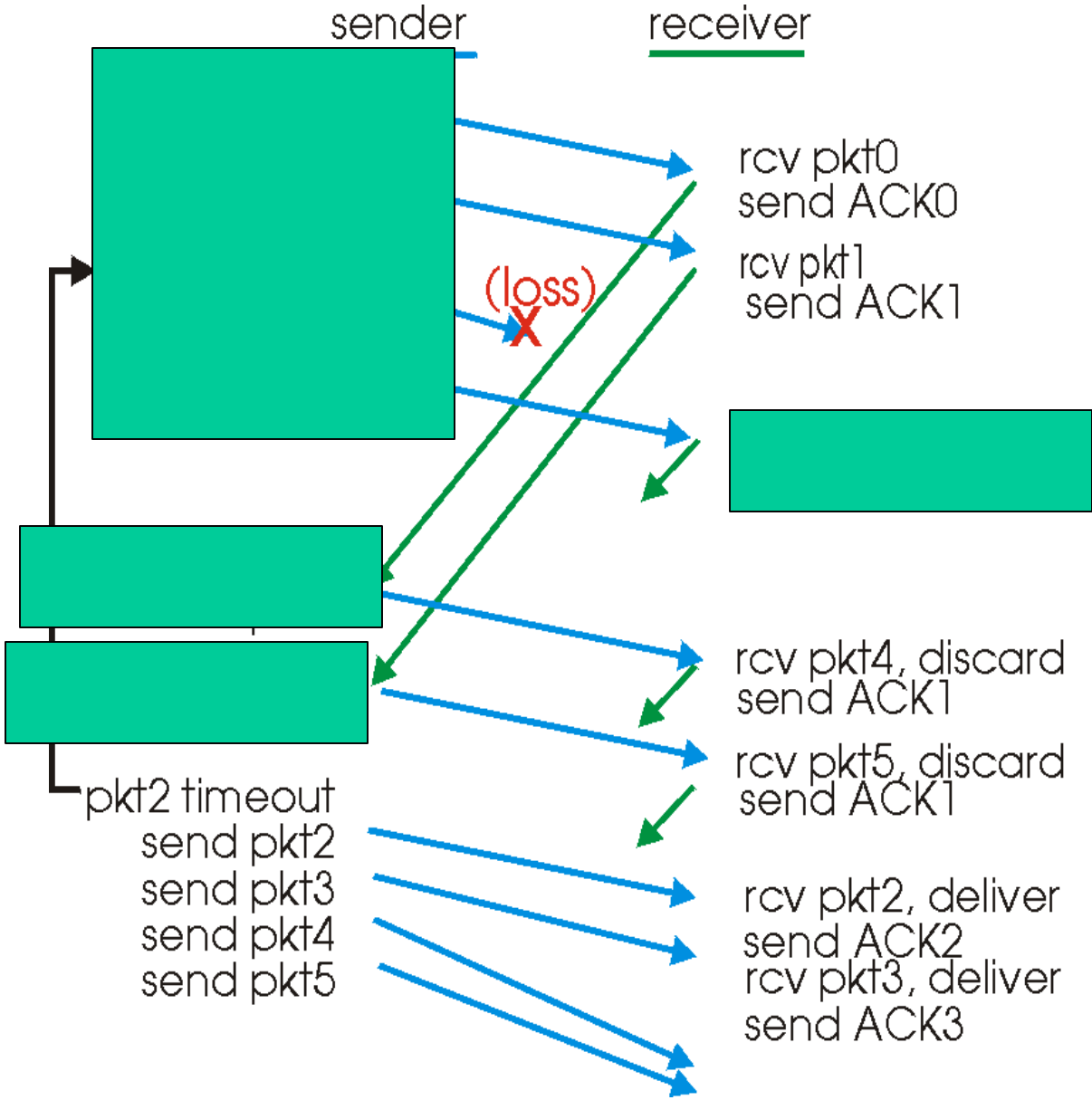




Out-of-Order PKT

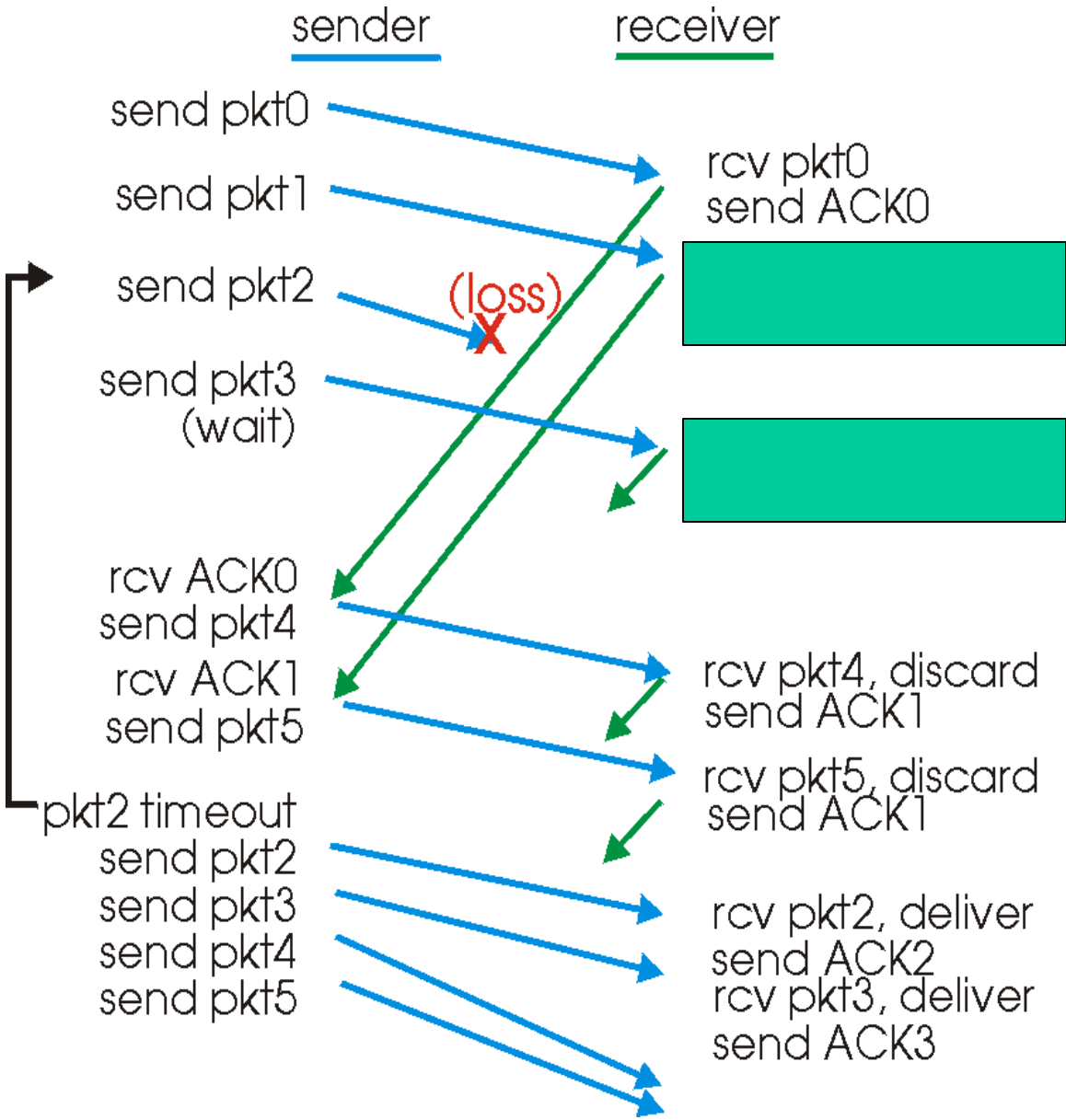
- out-of-order packet:
  - discard (don't buffer) -> no receiver buffering!
  - Re-ACK the pkt with the highest in-order seq #





Cumulative ACK

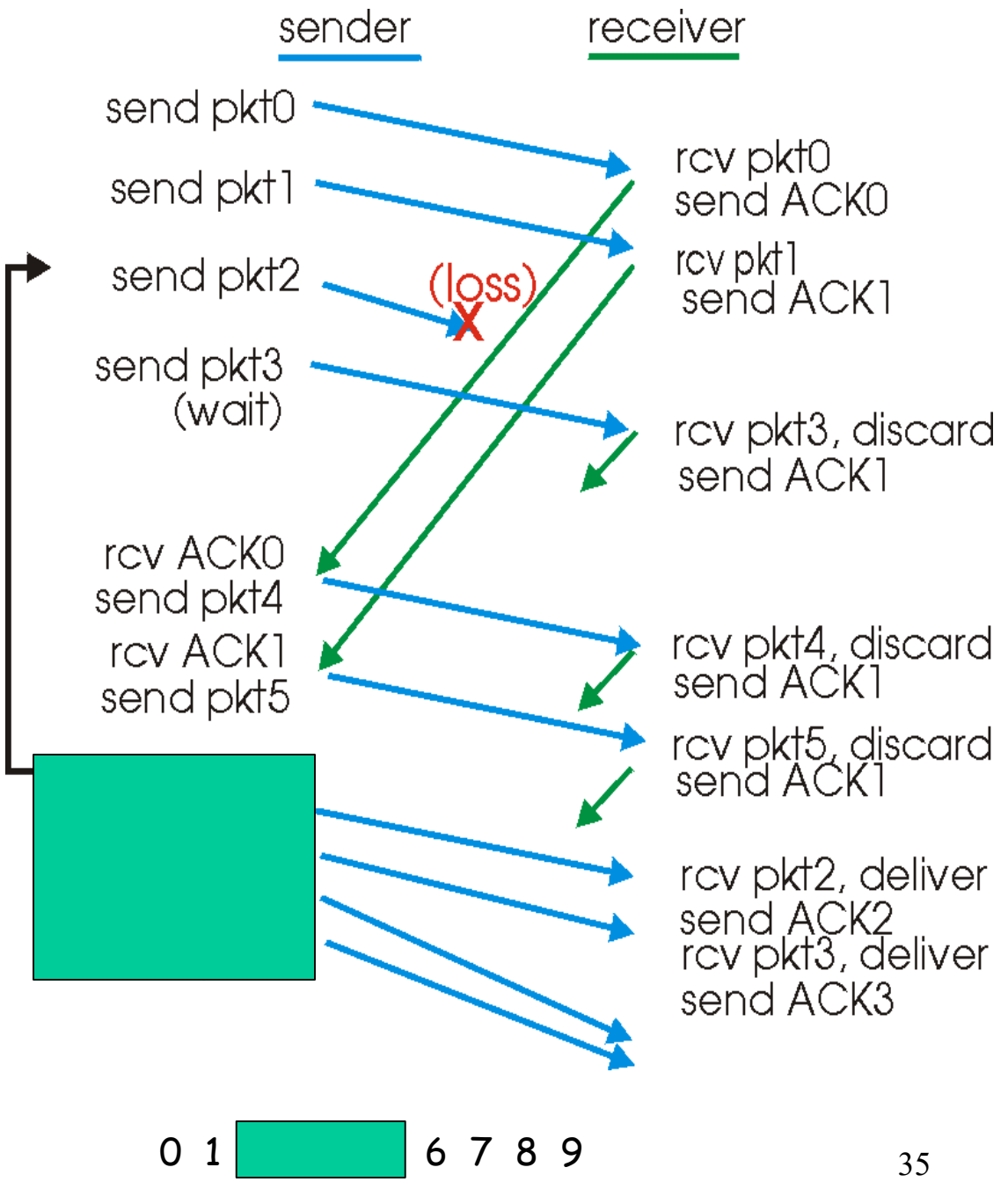
- Cumulative ack: always send ACK for correctly-received packet with highest *in-order seq #*
  - need only remember **`expected sequence number`**
  - may generate duplicate ACKs



With cumulative ack, an ACK indicates that all packets with a sequence number up to and including the number specified in the ACK have been correctly received.

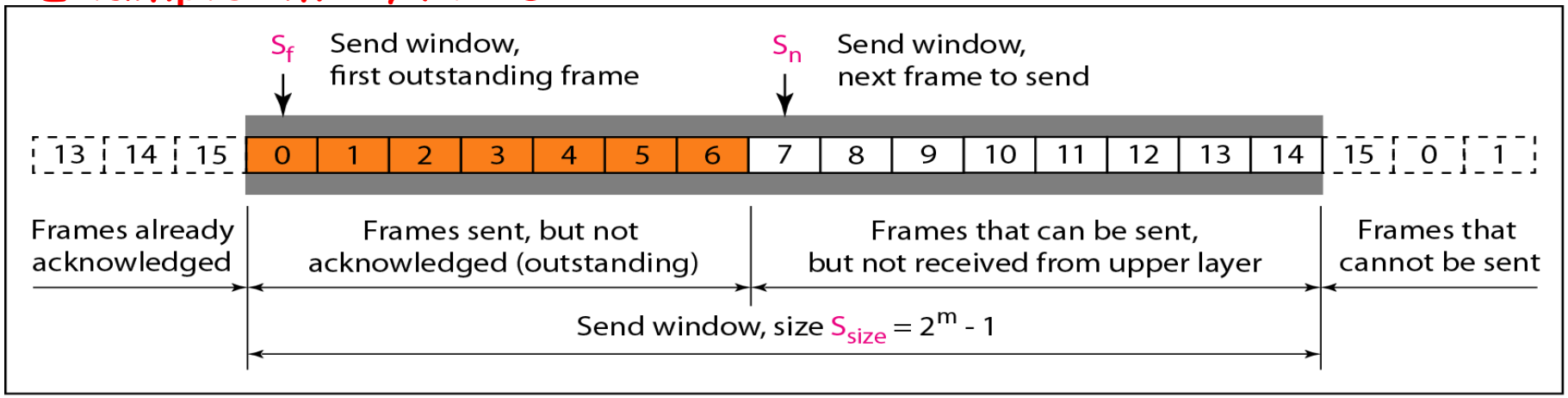
Timer and Timeout

- Uses a single timer – for the oldest unACKed pkt
- If an ACK is received but there are still other unACKed packets, the timer is restarted.
- On timeout, retransmit all unACKed packets.

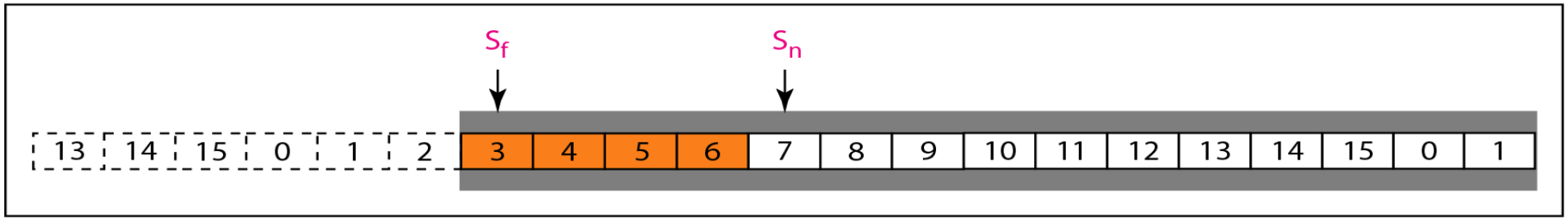


Sender/Receiver Window Size

- In go-back-N ARQ, The size of receiver window is 1; the size of the sender window N must be no more than (2^m) - 1, where m is the number of bits used to identify the sequence number of packets;
- Example: m=4; N=15



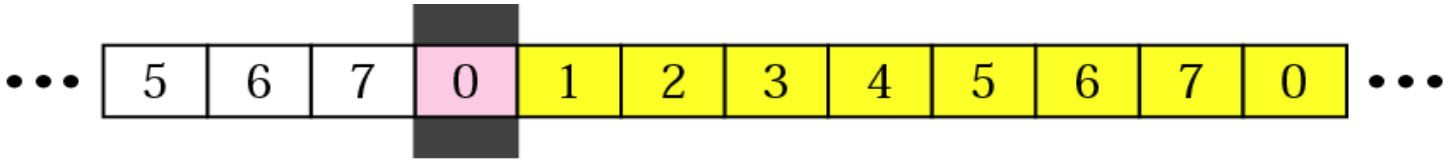
a. Send window before sliding



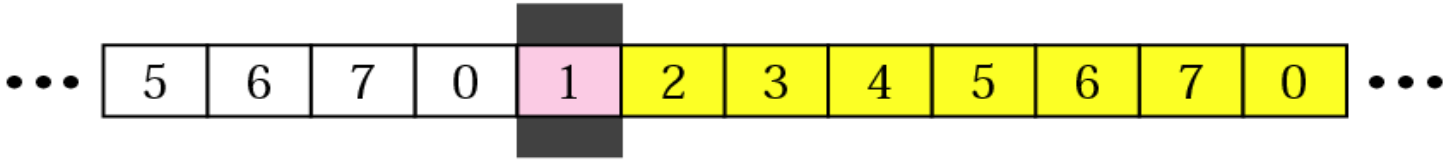
b. Send window after sliding

Receiver Window Size

- Receiver Operation:
  - The size of receiver window is 1;
  - The receive window indicates the packet's sequence number that is expected to receive.
  - The window slides when a pkt/frame has successfully received, and sliding occurs one window size at a time.



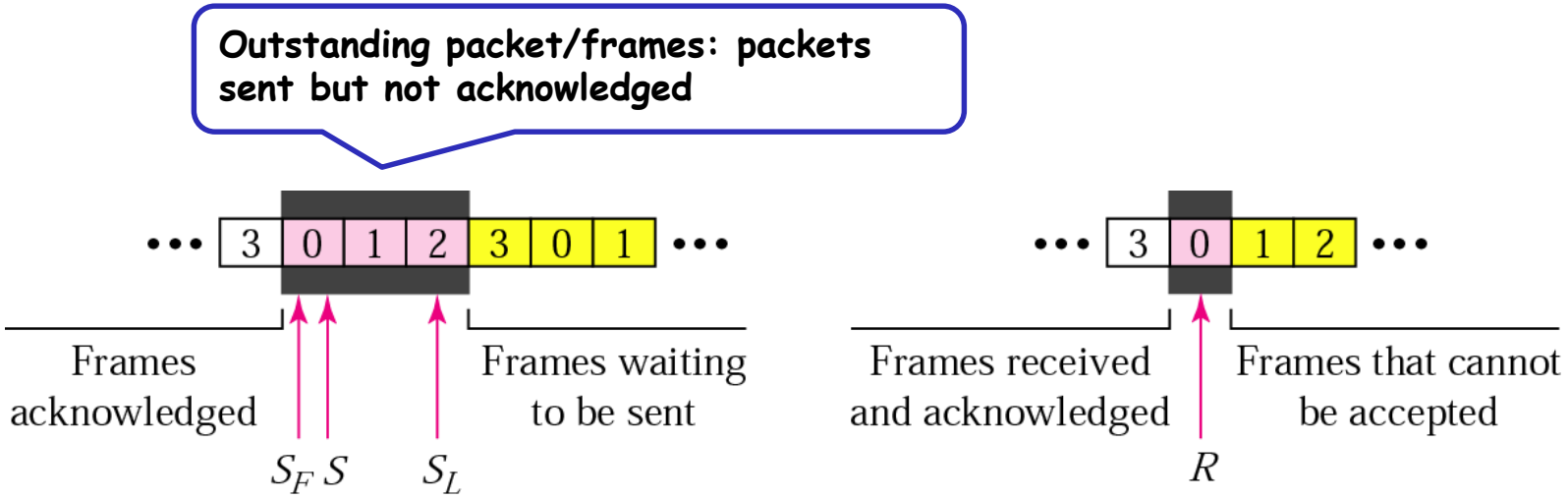
a. Before sliding



b. After sliding

Go-Back-N

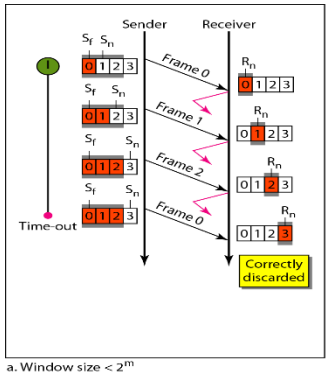
- Sender and Receiver Operation control variables (变量):
  - N: sender window size. （for example，N=3）
  - The size of receiver window is 1。
  - The number of bits to identify the sequence number is m=2.
  - The total packet sequence numbers (2^m)= (2^2) = 4: 0, 1, 2,3;
  - The maximum sender window size (2^m)-1= (2^2)-1 = 3: (N=3)
  - S_F: holds the sequence number of the first packet in the window;
  - S_L: holds the sequence number of the last packet.



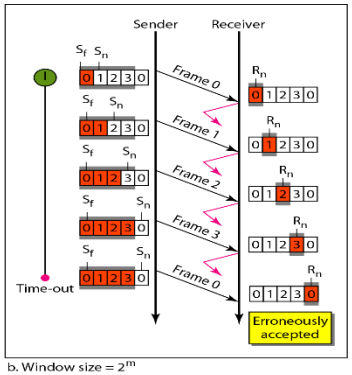
Sender window size

- In go-back-N ARQ, The size of receiver window is 1; the size of the sender window N must be no more than (2^m) -1, where m is the number of bits used to identify the sequence number of packets;

- Why?
  - This is to avoid packets being recognized incorrectly.
  - If the window size is greater than (2^m) -1, then if the ACK loses, the sender retransmits packets. However, the the receiver may recognize the retransmitted packet as a new packet.



a. Window size < 2^m

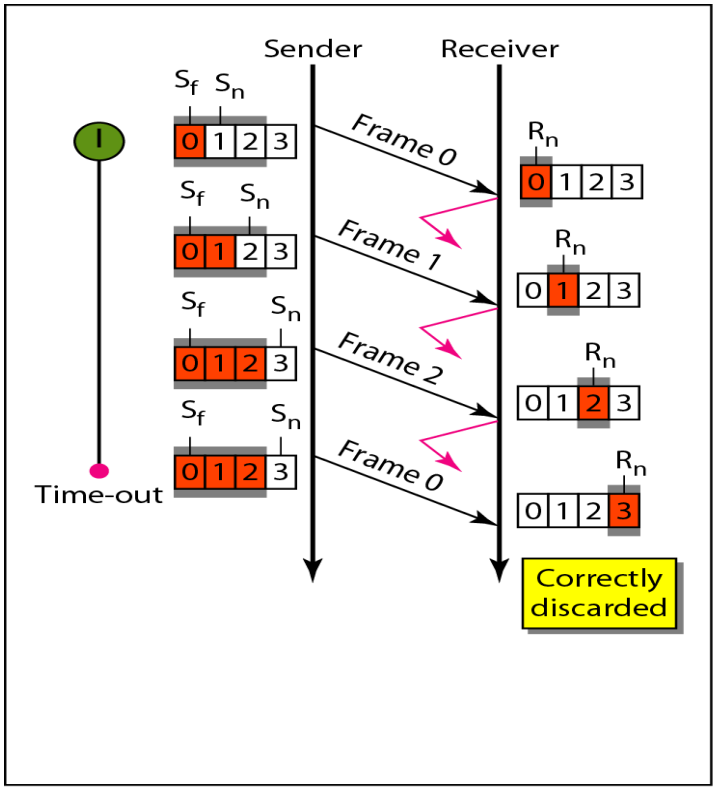


b. Window size = 2^m

Sender window size

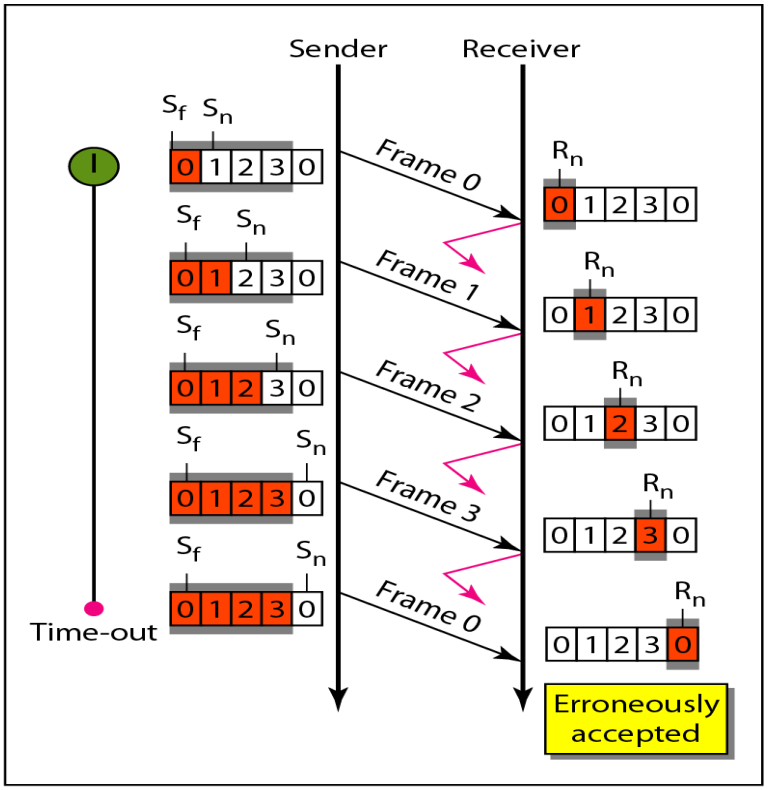
- m=2, Packet sequence numbers: 0, 1, 2, 3
- (2^m) -1 =3; Window size N ≤ 3;

Case 1:
Window size N=3



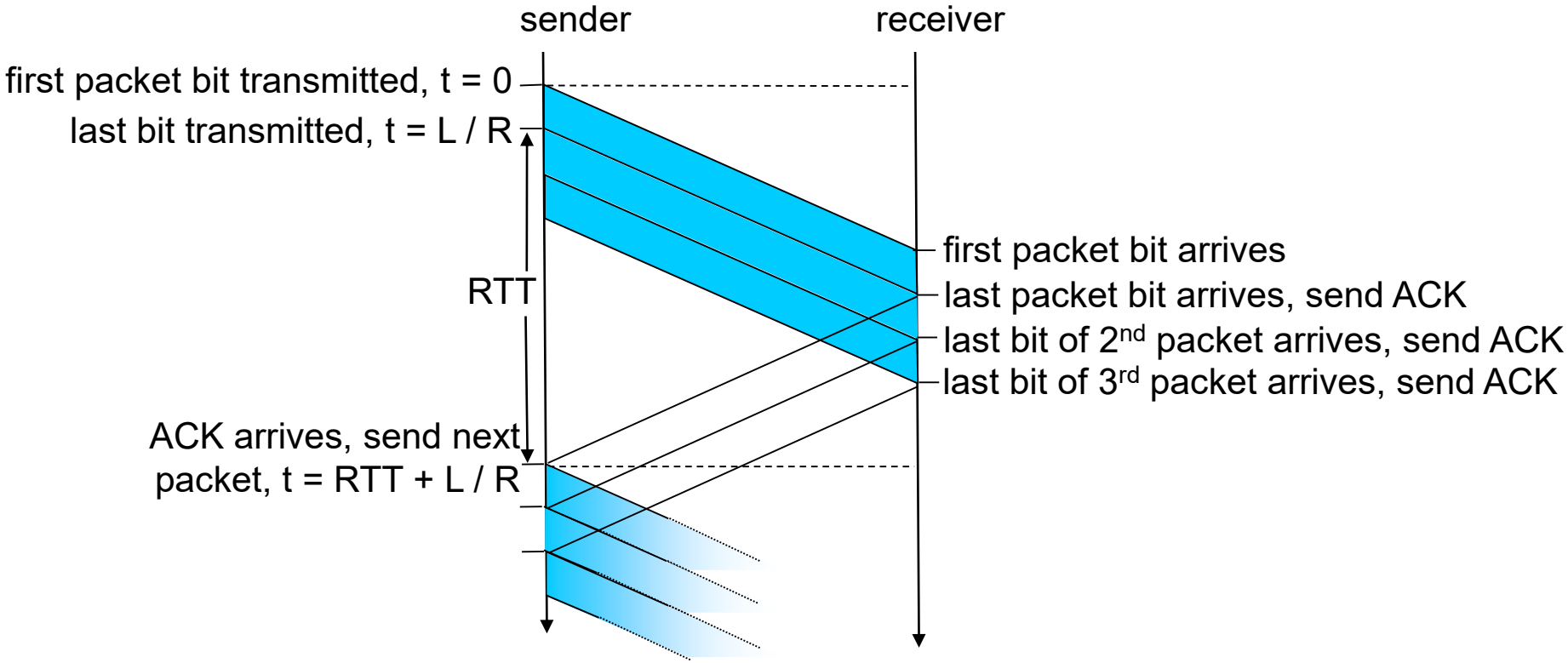
a. Window size < 2^m

Case 2:
Window size N=4



b. Window size = 2^m

The utilization



Link/Channel Utilization/Efficiency = N*(L/R)/(RTT+L/R)

Example

- example: Transmission Rate （R）of 1 Mbps, the round trip propagation delay (D) of 92ms, 1Kbyte(字节)/packet (L). Using the Go-Back-3 scheme, channel utilization is

$$T_{transmit} = \frac{L \text{ (packet length in bits)}}{R \text{ (transmission rate, bps)}} = \frac{8kbit}{1\ 0^3 kb/sec} = 8 \text{ ms}$$

$$RTT + T_{transmit} = 92ms + 8ms = 100ms$$

$$\text{Channel Efficiency} = \frac{3*L / R}{RTT + L / R} = \frac{24ms}{100ms} = 0.24$$

  - Channel utilization becomes 24%

Increase utilization by a factor of 3

Exercise

- A satellite link has transmission rate of 20 Kbps and RTT of 800 msec. The transmitter employs the "Go-Back-N "ARQ scheme with N=10. The transmission time of the acknowledgement frame is negligible. The size of each packet is 100 bytes. Please calculate the channel utilization.

- Answer:

Transmission Delay = (100 * 8)/ (20 * 10 ^3) = 0.04 sec = 40 msec.

Efficiency = (10 * 40)/(40 + 800) = 0.476 = 47.6%

Go-Back-N

- Advantages
  - Sender can send multiple packets at a time (a pipelined protocol)
  - Efficiency is high compared with stop-and-wait protocol
  - We can configure the window size
  - No buffer is needed for received out-of-order packets
- Disadvantage
  - Sender needs to store the last unAcked N packets
  - Retransmission of many error-free packets following an lost/corrupted packet.
  - It is inefficient when round-trip delay is large

Selective Repeat

- It is proposed to overcome the unnecessarily retransmitting the error-free packets.
- Selective Repeat will retransmit only those lost or corrupted packets.
- Receiver accepts and buffers the packets following a damaged or lost one. That is, receiver is able to accept out-of-order packets.
- SR ARQ is more efficient for noisy links, but the processing at the receiver is more complex.

Selective Repeat

- Sender window operation
  - Data is received from above layer.
  - Label each packet with a sequence number
  - If there is the next available seq # in the window, send pkt
  - Window size limits the number of sent but unACKed packets
  - The range of sender window is decided based on window size N and the unACKed packet with the smallest sequence #. That is, when the packet at the left side of the window is ACKed, the window will move forward and the range of sender window will update accordingly.

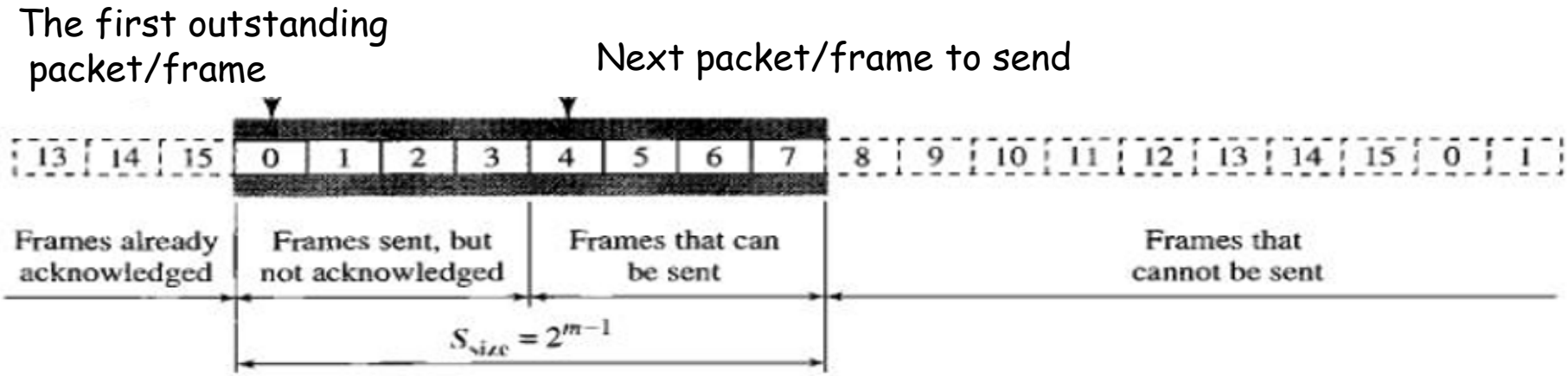


Fig. 1. Sender window for SR ARQ, where the window size is 8

Selective Repeat

- Receiver window operation
  - The receiver window indicates packets/frames that have arrived out of order or are expected to be received.
  - When the packet at the left side of the window is received successfully, the window will move forward and the range of receiver window will update accordingly.

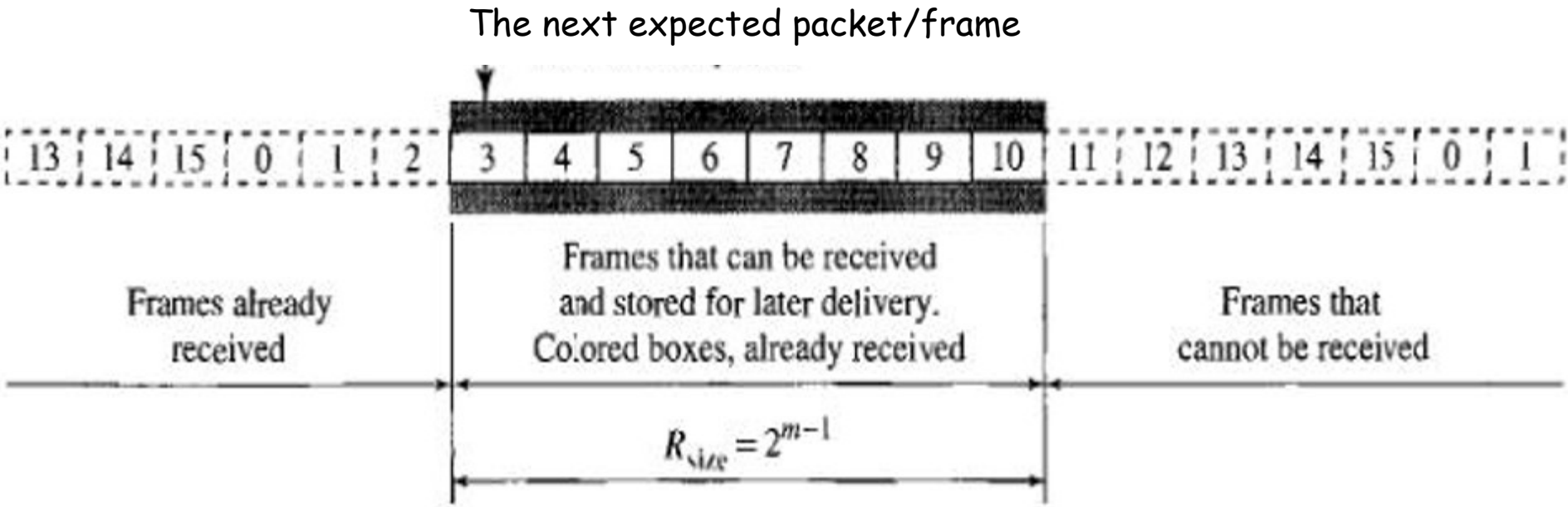
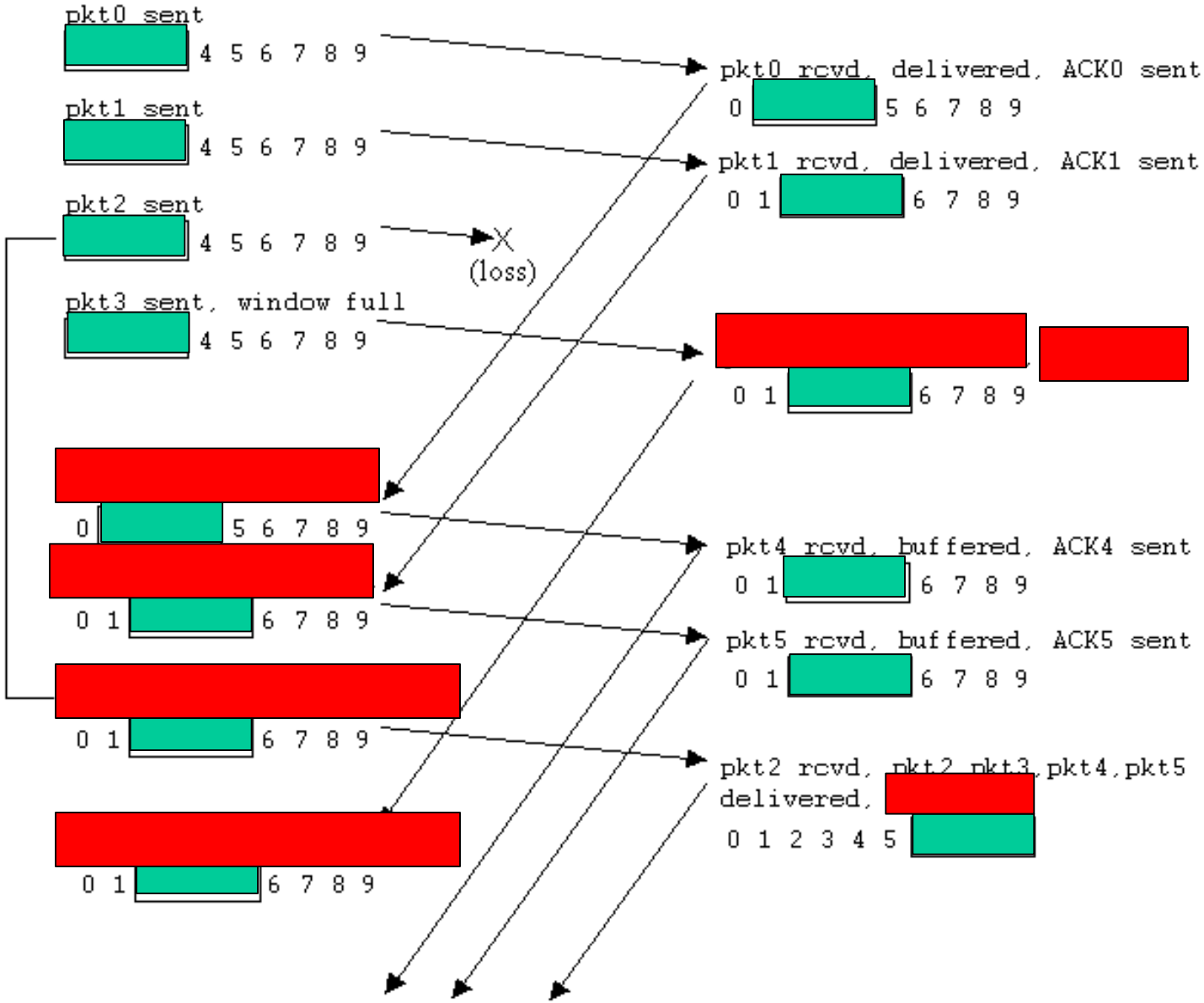


Fig. 1. Receiver window for SR ARQ, where the window size is 8

Selective Repeat

- How to slide window
  - The range of sender window is decided based on window size N and the unACKed packet with the smallest sequence #.
  - Window moves when the packet with the smallest sequence # in the window is ACKed (for the sender window) or received (for the receiver window)
  - After moving window, if there are untransmitted packets with sequence numbers within the window, these packets are transmitted.

Selective Repeat Example: Window Size N=4

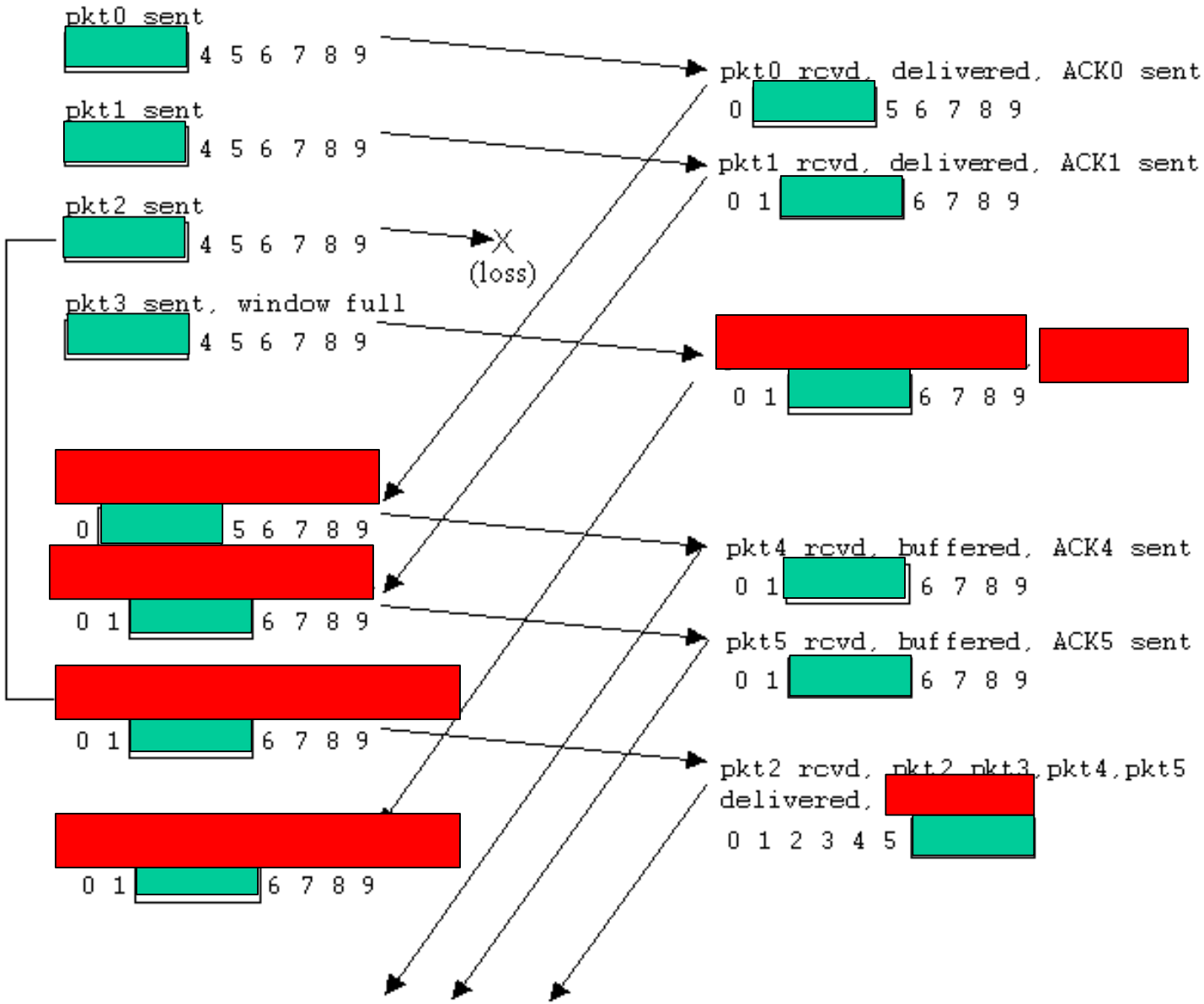


Selective Repeat

- ACK is not cumulative
  - Receiver sends *individual ack for each* correctly received pkts (ACK is not cumulative)
  - Mark packet n as received
  - Buffer is needed at the receiver for eventual in-order delivery packets to upper layer

- Sender resends pkts when timeout (due to the lost of pkt or ACK) or receiving NAK for the corrupted pkts
  - Sender sets timer for each unACKed pkt
  - Sender resends pkts for which ACK is not received when timeout

Selective Repeat Example: Window Size N=4



Selective Repeat

- How to deal with the out-of-order packets:
  - Buffer the out-of-order packets
- Timer and timeout
  - sender maintains timer for each unACKed pkt (multiple timers)
  - If packet n is timeout, retransmit only pkt n and restart timer

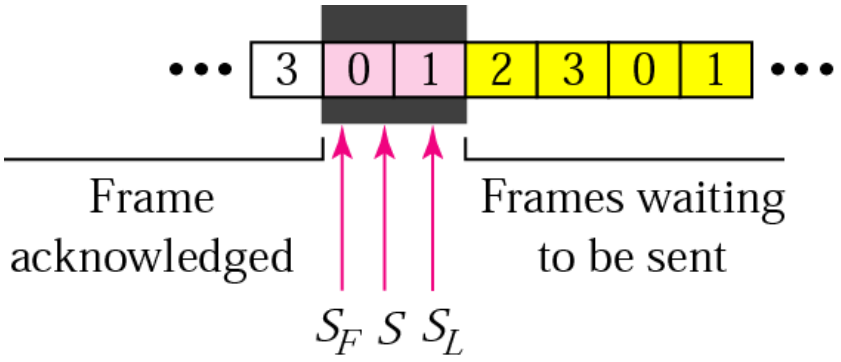
Selective Repeat

- Window size
  - Sender window size = receiver window size
  - The size of the sender and receiver windows must be no more than half of 2^m, where m is the number of bits used to identify the sequence number.
    - If m bits are used to identify the sequence number of packets, the range of the sequence numbers is from 0 to (2^m) – 1.
    - The sequence numbers are modulo 2^m. if m=3, sequence number range from 0 to 7: 0,1,2,3,4,5,6,7,0, 1,2,3, 4, 5,6,7,0,1,…….
  - For example, m=4, which means the size of window is (2^m)/2 = 2^(m-1)=8. It means the sequence numbers go from 0 to 7.

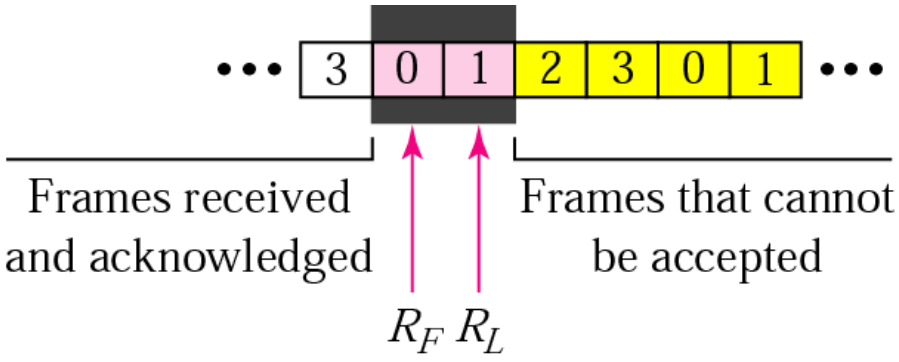
- Why?
  - This is to avoid packets being recognized incorrectly. If the window size is greater than the half of the sequence number space, then if an ACK is lost, the sender may retransmit packets that the receiver will recognize as a new packet.

Window Size and Sequence Number

- Let m=2
  - Sequence numbers 2^m = 2^2 =4 : 0,1,2,3
  - Window size 2^(m-1) =2^1=2: 0, 1



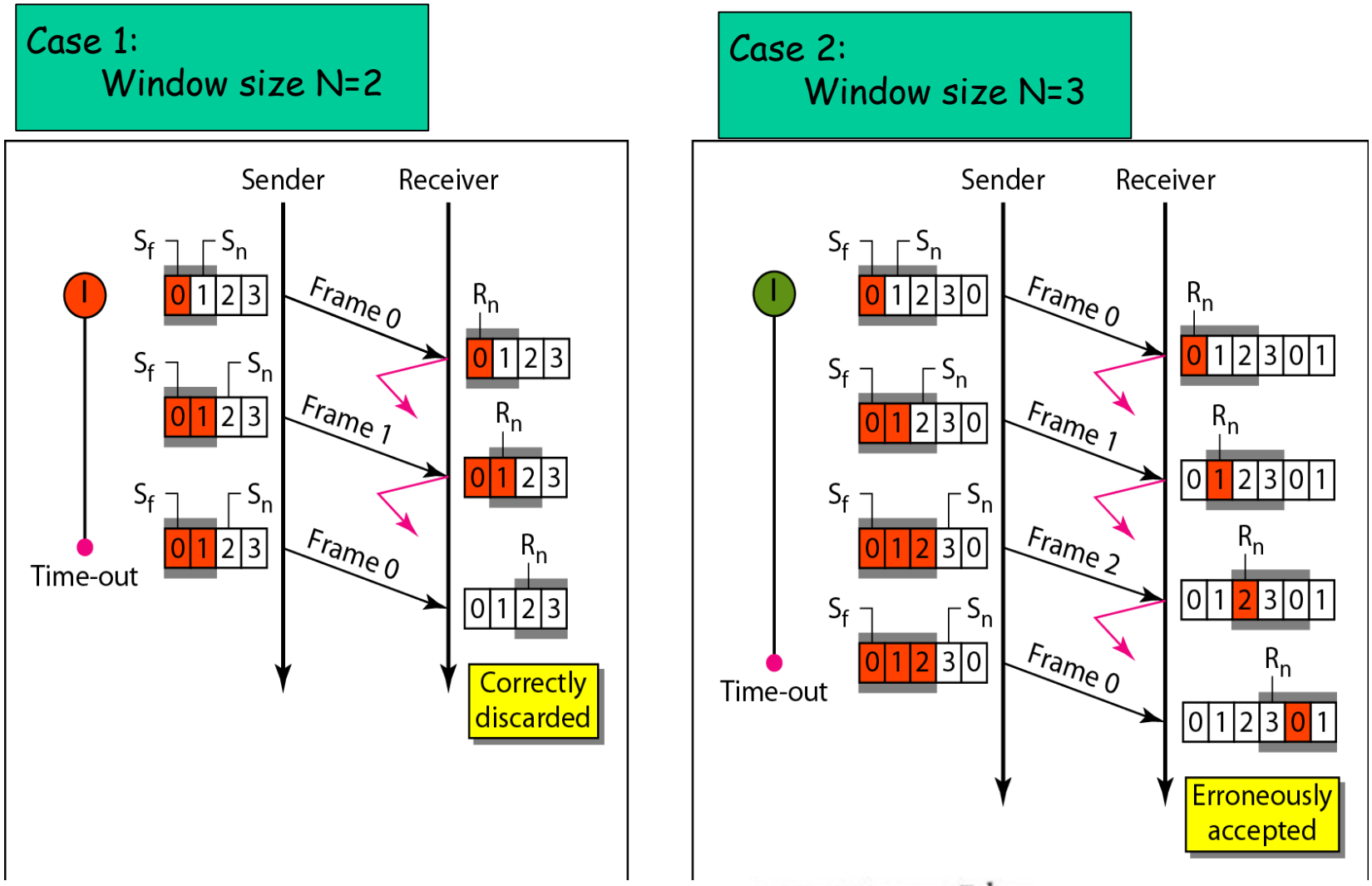
a. Sender window



b. Receiver window

Selective Repeat

- For example, m=2, which limits the maximum window size to be 2.



Comparison

High; Medium; Low
Implement; Not implemented
Buffered; Discarded
No; Yes

Protocol	Stop and Wait	Go-Back-N	Selective Repeat
Channel Utilization			
Pipelining			
Out-of-order packets			
Cumulative ACK	N.A		

Comparison

Protocol	Stop and Wait	Go-Back-N	Selective Repeat
Channel Utilization	Low	Medium	High
Pipelining	Not implemented	Implemented	Implemented
Out-of-order packets	N.A.	Discarded	Buffered
Cumulative ACK	N.A.	Yes	No

Comparison

Protocol	Stop and Wait	Go-Back-N	Selective Repeat
Sender Window Size	1	(2^m)-1	2^(m-1)
Receiver Window Size	1	1	2^(m-1)
Minimum number of the sequence numbers	2	2^m	2^m
Efficiency	(L/R)/ ((L/R) +RTT)	N*(L/R)/ ((L/R) +RTT)	N*(L/R)/ ((L/R) +RTT)