FLOW CONTROL

Flow Control

- The most important responsibilities of the data link layer are flow control and error control
- Collectively, these functions are known as data link control
- It is one of the most important functions of data link layer

Flow Control

- Flow control refers to a set of procedures used to restrict the amount of data that the sender can send before waiting for acknowledgment
- Flow control coordinates the amount of data that can be sent before receiving acknowledgement
- Flow control is a set of procedures that tells the sender how much data it can transmit before it must wait for an acknowledgement from the receiver

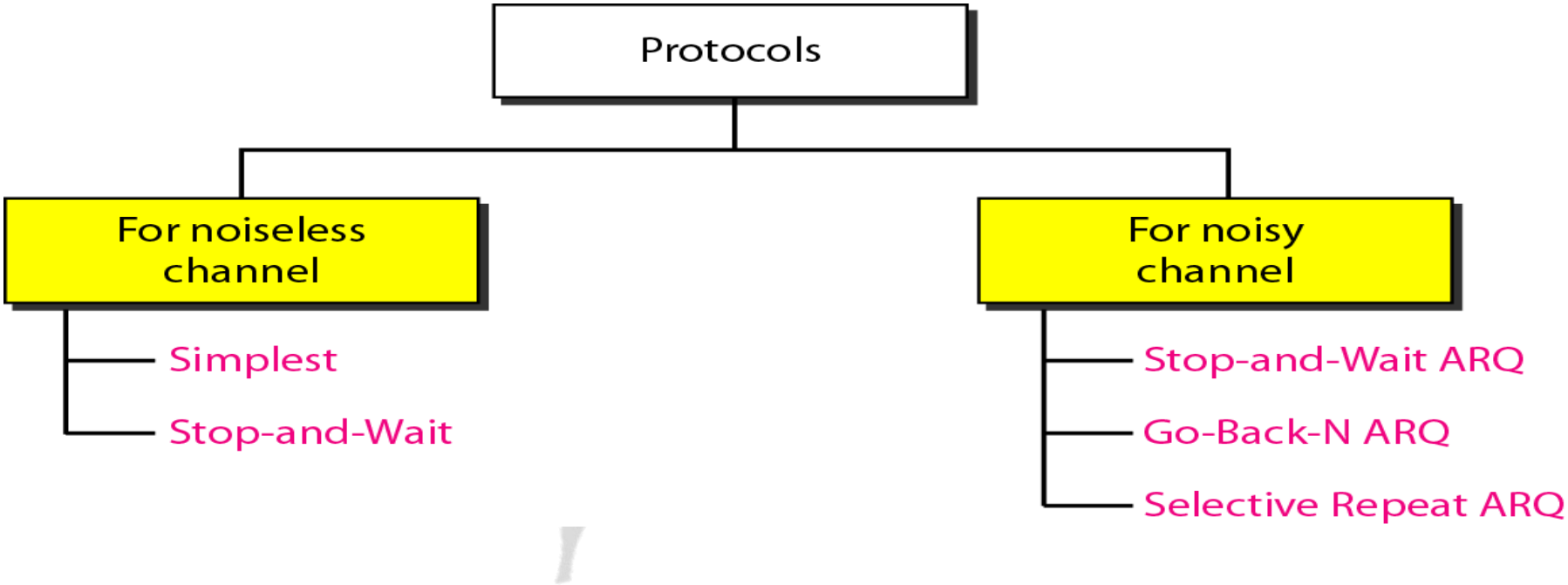
Flow Control

- Receiver has a limited speed at which it can process incoming data and a limited amount of memory in which to store incoming data
- Receiver must inform the sender before the limits are reached and request that the transmitter to send fewer frames or stop temporarily
- Since the rate of processing is often slower than the rate of transmission, receiver has a block of memory (buffer) for storing incoming data until they are processed

Flow Control

- Two Approaches:
  - Feedback based flow control
    - The receiver sends back information to the sender giving it permission to send more data or at least telling the sender how the receiver is doing
  - Rate based flow control
    - The protocol has a built in mechanism that limits the rate at which senders may transmit data, without using feedback from the receiver
- Various flow control schemes uses a common protocol that contains well-defined rules about when a sender may transmit the next frame
- These rules often prohibit frames from being sent until the receiver has granted permission, either implicitly or explicitly

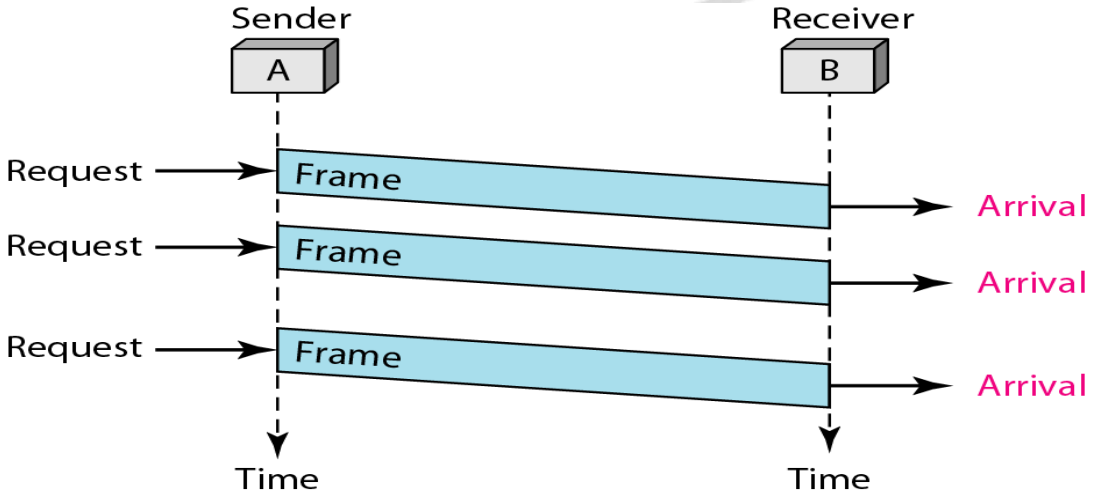
Flow Control



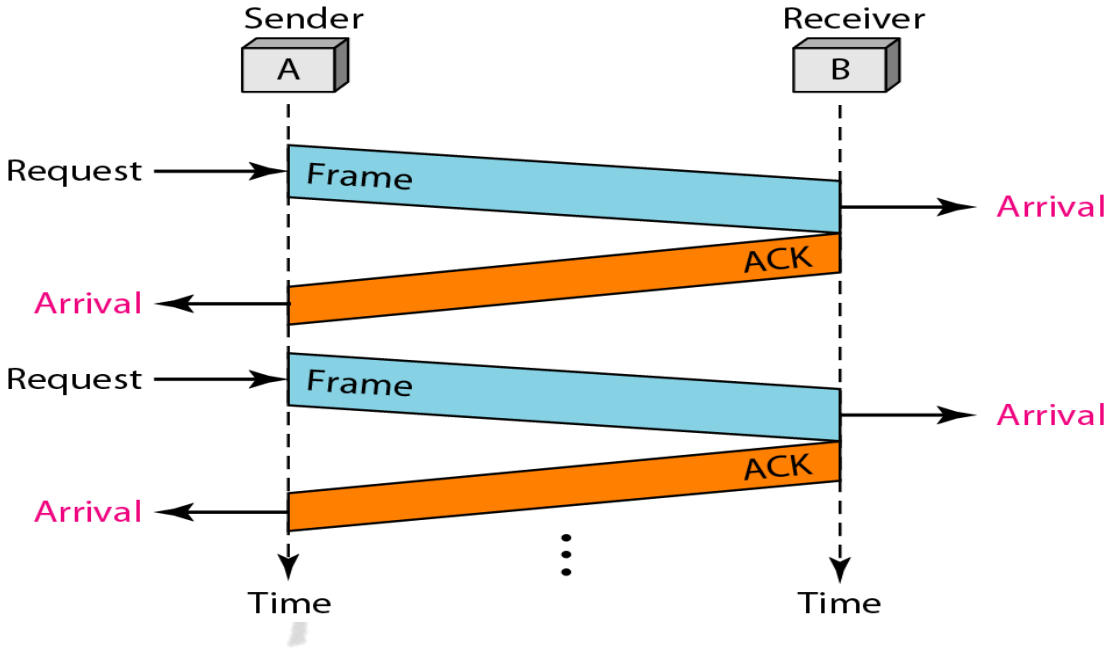
Noiseless channel

- Let us first assume we have an ideal channel in which no frames are lost, duplicated, or corrupted
- Two protocols for this type of channel
  - Simplest Protocol
  - Stop-and-Wait Protocol

Simplest Protocol



Simplest Protocol



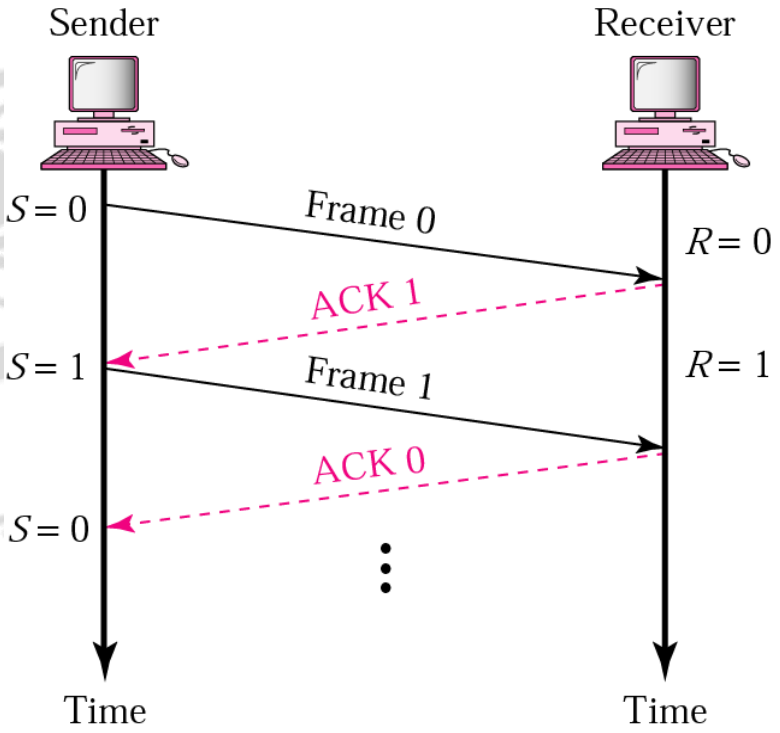
Noisy channel

- Three protocols
  - Stop-and-Wait Automatic Repeat Request
  - Go-Back-N Automatic Repeat Request
  - Selective Repeat Automatic Repeat Request

- NOTE:
- Round Trip Time (RTT)
  - Time taken by frame to travel from sender to receiver + Time taken by corresponding frame ACK to travel from receiver to sender
- Thus,
  - RTT = 2* Tp (Twice Propagation Time)

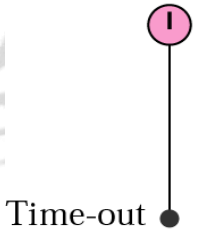
Stop and Wait ARQ (General Scenario)

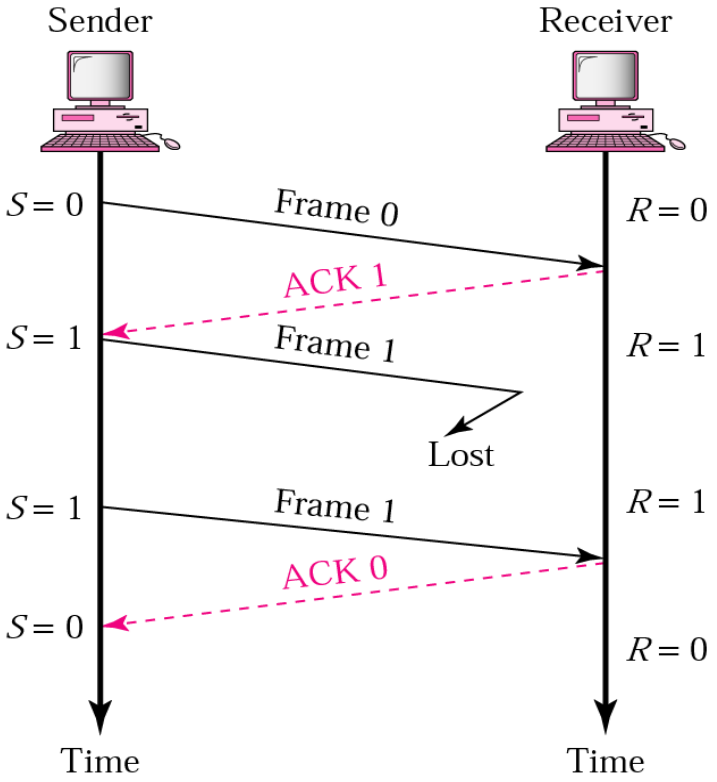
- Sender keeps a copy of the last frame until it receives an acknowledgement
- For identification, data frame and acknowledgement (ACK) frames are numbered alternatively 0 and 1
- Sender has a control variable “S” that holds the number of the recently sent frame. (0 or 1)
- Receiver has a control variable “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 received
- ACK number always defines the number of the next expected frame



Stop and Wait ARQ (Damaged frame)

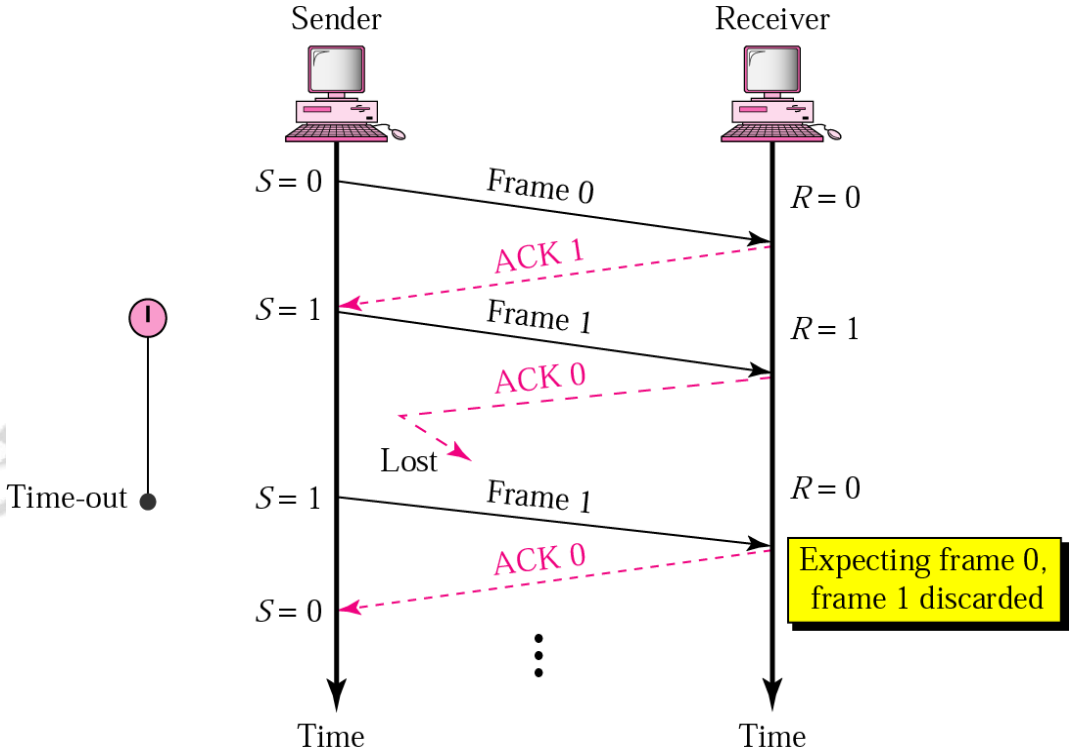
- When a receiver receives a damaged frame, it discards it and keeps its value of R
- After the timer at the sender expires, another copy of frame 1 is sent





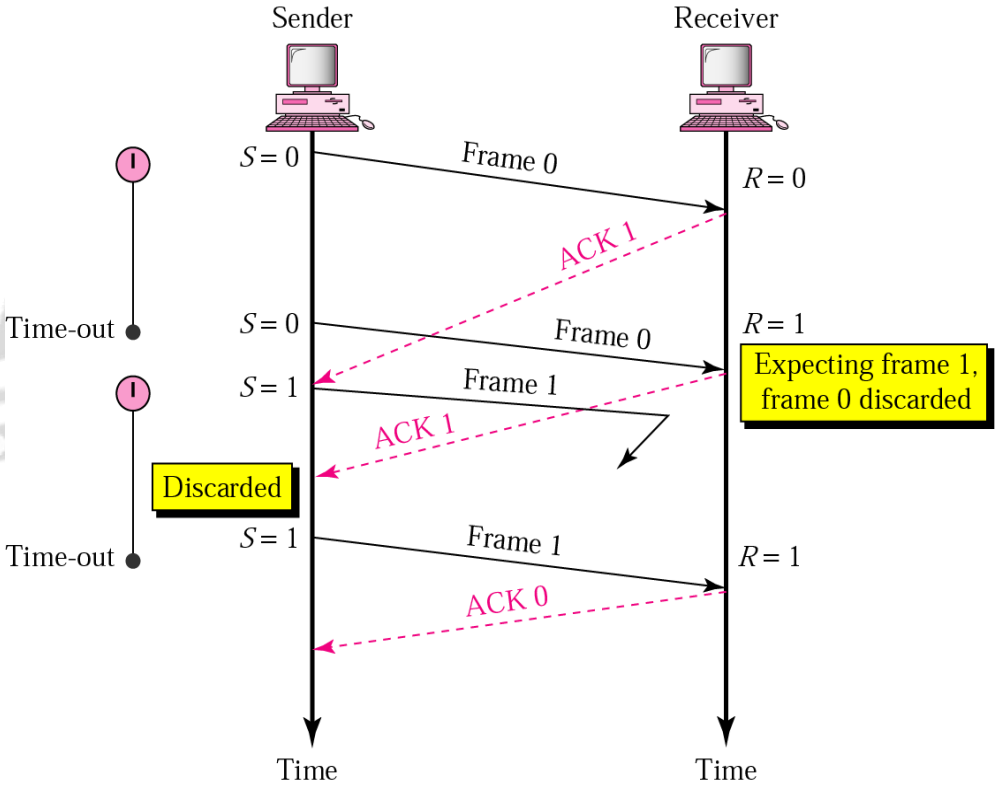
Stop and Wait ARQ (Damaged ACK)

- 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 ARQ (ACK delayed)

- 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

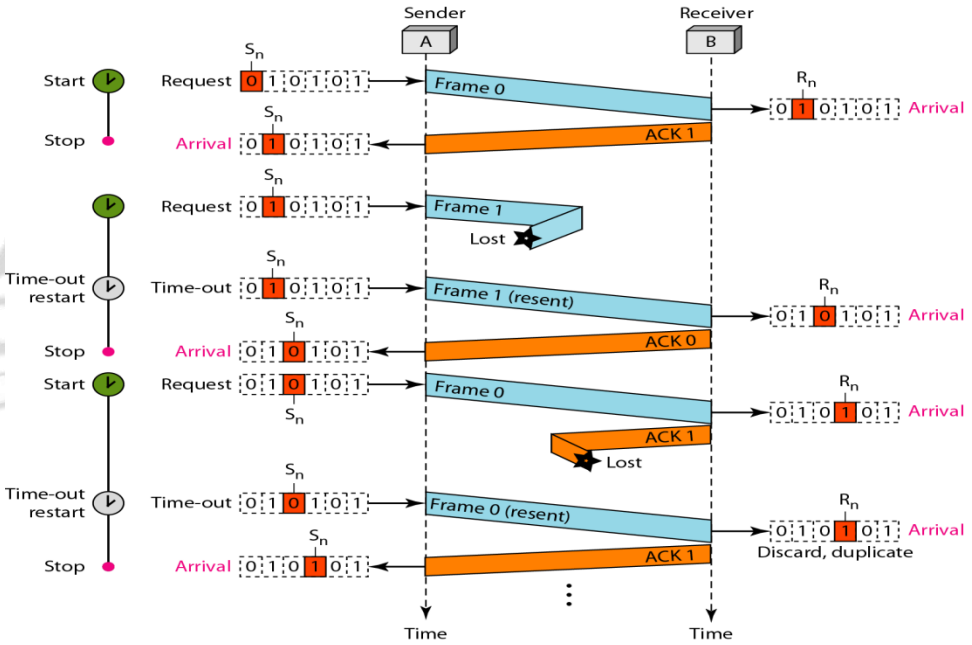


Stop and Wait ARQ

- Error correction in Stop-and-Wait ARQ is done by keeping a copy of the sent frame and retransmitting of the frame when the timer expires
- In Stop-and-Wait ARQ, we use sequence numbers to number the frames
- The sequence numbers are based on modulo-2 arithmetic
- In Stop-and-Wait ARQ, the acknowledgment number always announces in modulo-2 arithmetic the sequence number of the next frame expected

Stop and Wait ARQ

- Eg:
  - As seen from the figure, Frame 0 is sent and acknowledged
  - Frame 1 is lost and resent after the time-out
  - The resent frame 1 is acknowledged and the timer stops
  - Frame 0 is sent and acknowledged, but the acknowledgment is lost
  - The sender has no idea if the frame or the acknowledgment is lost, so after the time-out it resends frame 0, which is acknowledged



Stop and Wait ARQ

- Eg:
  - Assume that, in a Stop-and-Wait ARQ system, the bandwidth of the line is 1 Mbps, and 1 bit takes 20 ms to make a round trip. What is the bandwidth-delay product?

Solution

- The bandwidth-delay product is

$$(1 \times 10^6) \times (20 \times 10^{-3}) = 20{,}000 \text{ bits}$$

Stop and Wait ARQ

- Eg: (contd)
  - If the system data frames are 1000 bits in length, what is the utilization percentage of the link?

Solution

- The system can send 20,000 bits during the time it takes for the data to go from the sender to the receiver and then back again
- However, the system sends only 1000 bits
- We can say that the link utilization is only 1000/20,000, or 5 percent
- For this reason, a link with a high bandwidth or long delay, the use of Stop-and-Wait ARQ wastes the capacity of the link

Stop and Wait ARQ

- Eg: (contd)
  - What is the utilization percentage of the link if we have a protocol that can send up to 15 frames before stopping and worrying about the acknowledgments?

Solution

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

Stop and Wait ARQ

- Disadvantages
  - 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
  - Efficiency is very low
  - To improve efficiency, multiple frames should be in transition while waiting for ACK

- Two protocol use the above concept
  - Go back N ARQ
  - Selective repeat ARQ

Stop and Wait ARQ

Total time taken to send one packet,

```
= Tt(data) + Tp(data) + Tq + Tpro + Tt(ack) + Tp(ack)

Since,
Tp(ack) = Tp(data)

And,
Tt(ack) << Tt(data).

So we can neglect Tt(ack)
Tq = 0 and Tpro = 0

Hence,
Total time = Tt(data) + 2 * Tp
```

Stop and Wait ARQ

- Efficiency (η) = Useful time / Total cycle time
  = Tt / (Tt + 2*Tp)
  = 1 / (1+2*(Tp/Tt))
  = 1 / (1+2*a)

where, a = Tp / Tt

- Throughput
  - Number of bits send per second, which is also known as Effective Bandwidth or Bandwidth utilization
- Throughput = η * BW

where, BW is Bandwidth

Stop and Wait ARQ

- Eg:
  - If the bandwidth of the line is 1.5 Mbps, RTT is 45 msec and packet size is 1 KB, then find the link utilization in stop and wait

Solution

$$\text{Transmission delay } (T_t) = \text{Packet size / Bandwidth}$$
$$= 1 \text{ KB} / 1.5 \text{ Mbps}$$
$$= (2^{10} \text{ x } 8 \text{ bits}) / (1.5 \text{ x } 10^6 \text{ bits/sec})$$
$$= 5.461 \text{ msec}$$

Stop and Wait ARQ

Propagation delay (Tp) = Round Trip Time / 2
= 45 msec/2
= 22.5 msec

Calculating Value Of ‘a’

a = Tp / Tt
a = 22.5/5.461
a = 4.12 msec

Link Utilization or Efficiency (η) = 1 / 1+2a
= 1 / (1 + 2 x 4.12)
= 1 / 9.24
= 0.108
= 10.8 %

Stop and Wait ARQ

- Eg:
  - A channel has a bit rate of 4 Kbps and one way propagation delay of 20 msec. The channel uses stop and wait protocol. The transmission time of the acknowledgement frame is negligible. To get a channel efficiency of at least 50%, the minimum frame size should be
  - 80 bytes
  - 80 bits
  - 160 bytes
  - 160 bits

Solution

Bandwidth = 4 Kbps, Propagation delay (Tp) = 20 msec, Efficiency >= 50%

Let the required frame size = L bits

Transmission delay (Tt) = Packet size / Bandwidth

= L bits / 4 Kbps

Stop and Wait ARQ

Condition for efficiency to be at least 50 %

For efficiency to be at least 50%, we must have

$$1 / 1+2a >= 1/2$$

$$a <= 1/2$$

Substituting the value of ‘a’, we get

$$(20 \text{ msec} \times 4 \text{ Kbps}) / L \text{ bits} <= 1/2$$

$$L \text{ bits} >= (20 \text{ msec} \times 4 \text{ Kbps}) \times 2$$

$$L \text{ bits} >= (20 \times 10\text{-}3 \text{ sec} \times 4 \times 103 \text{ bits per sec}) \times 2$$

$$L \text{ bits} >= 20 \times 4 \text{ bits} \times 2$$

$$L >= 160$$

From here, frame size must be at least 160 bits

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 ARQ
  - Selective Repeat ARQ

Go Back N ARQ

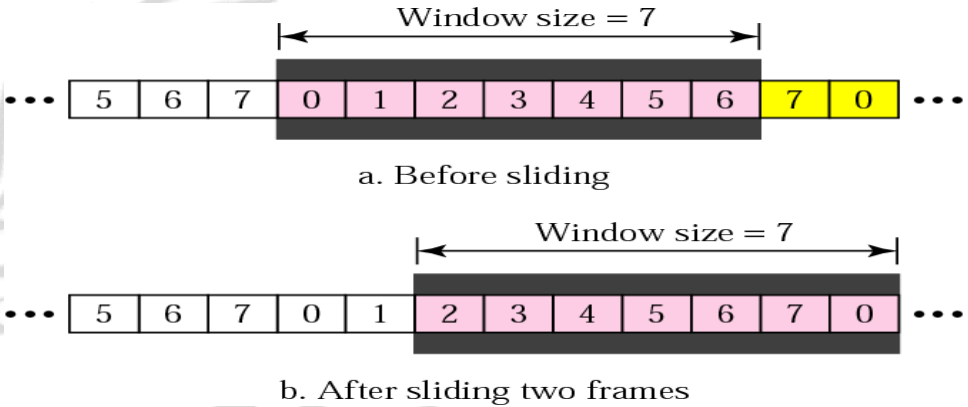
- We can send up to W frames before worrying about ACKs
- We keep a copy of these frames until the ACKs arrive
- This procedure requires additional features to be added to Stop and Wait ARQ

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, …

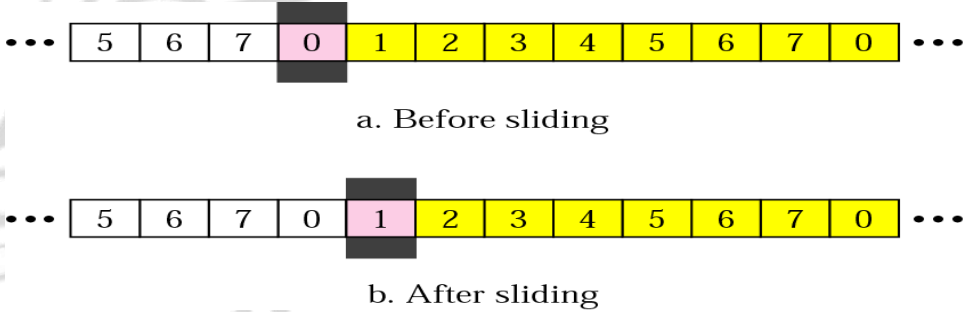
Sender Sliding Window

- 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



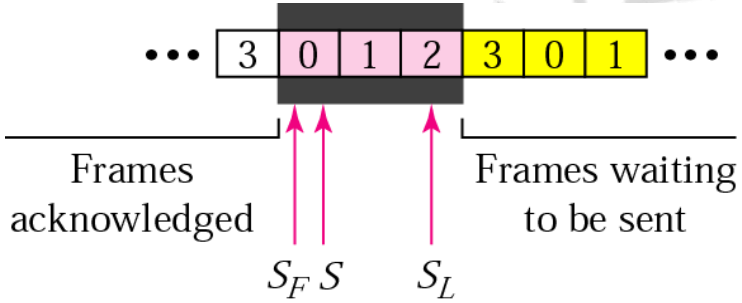
Receiver Sliding Window

- 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

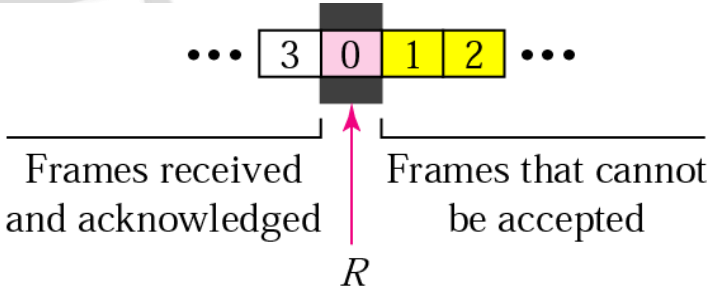


Control Variables

- Sender has 3 variables: S, SF, and SL
- S holds the sequence number of recently sent frame
- SF holds the sequence number of the first frame
- SL holds the sequence number of the last frame
- Receiver only has the one variable, R, that holds the sequence number of the frame it expects to receive. If the seq. no. is the same as the value of R, the frame is accepted, otherwise rejected



a. Sender window



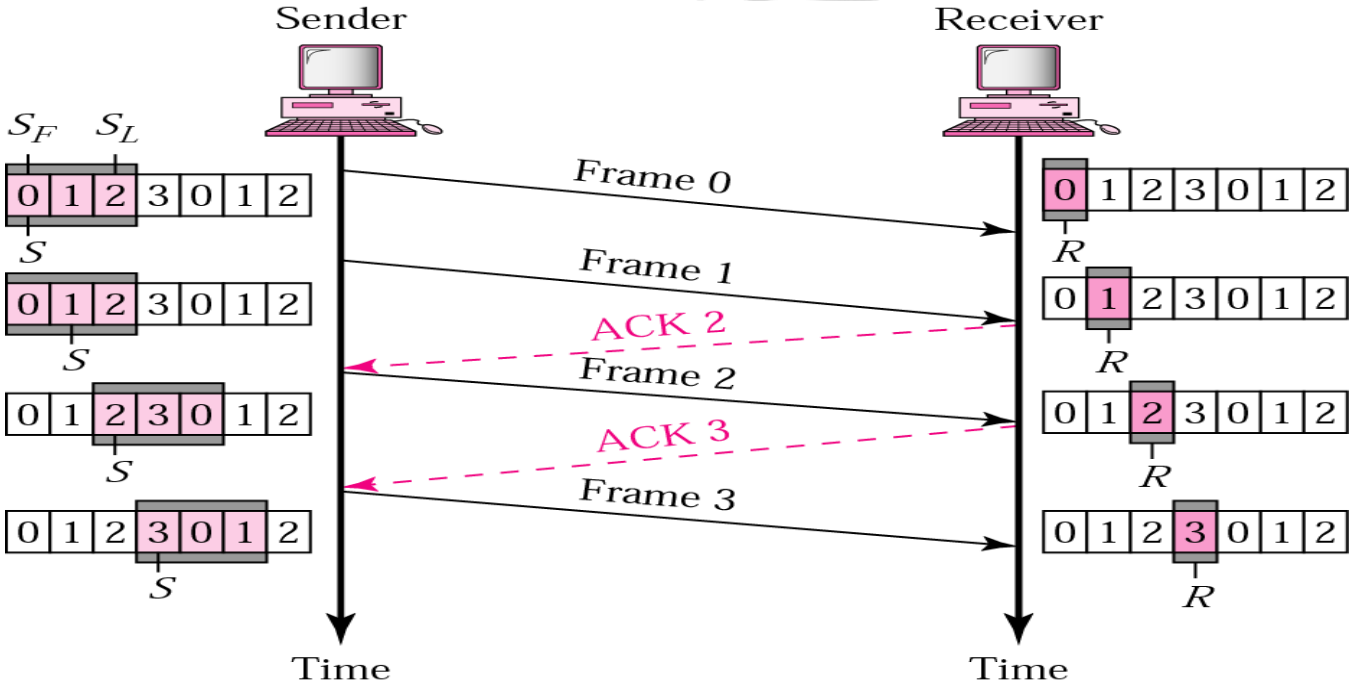
b. Receiver window

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-ARQ
- The receiver does not have to acknowledge each frame received, it can send one cumulative ACK for several frames

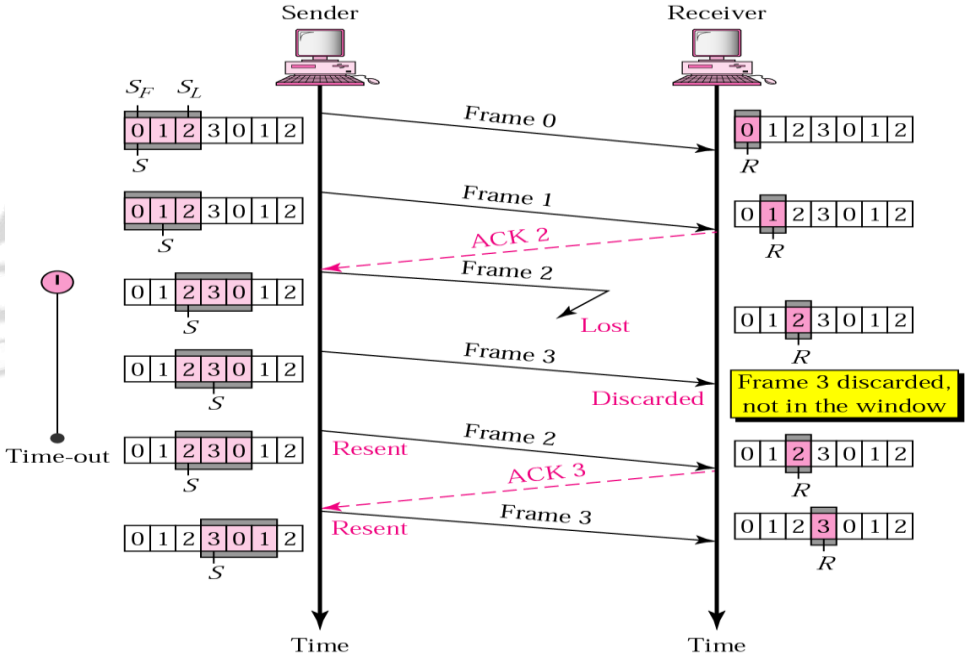
Go Back N ARQ, normal operation

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



Go Back N ARQ, 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)

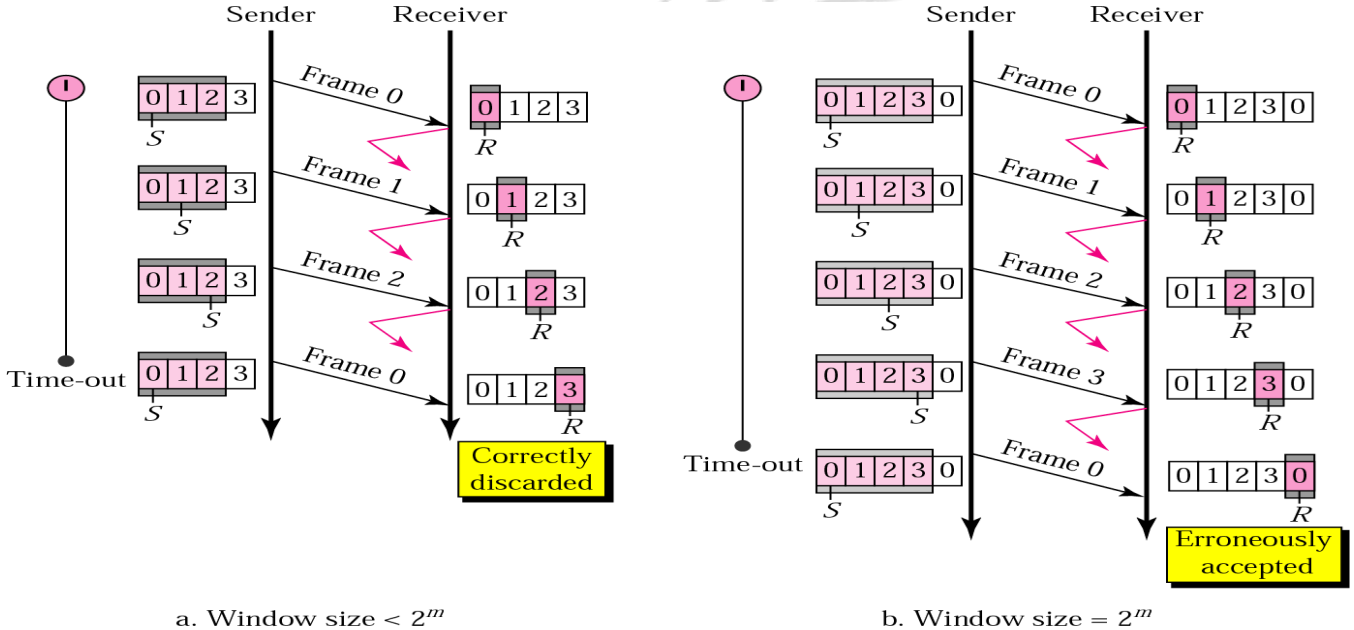


Go Back N ARQ, damaged/lost/delayed ACK

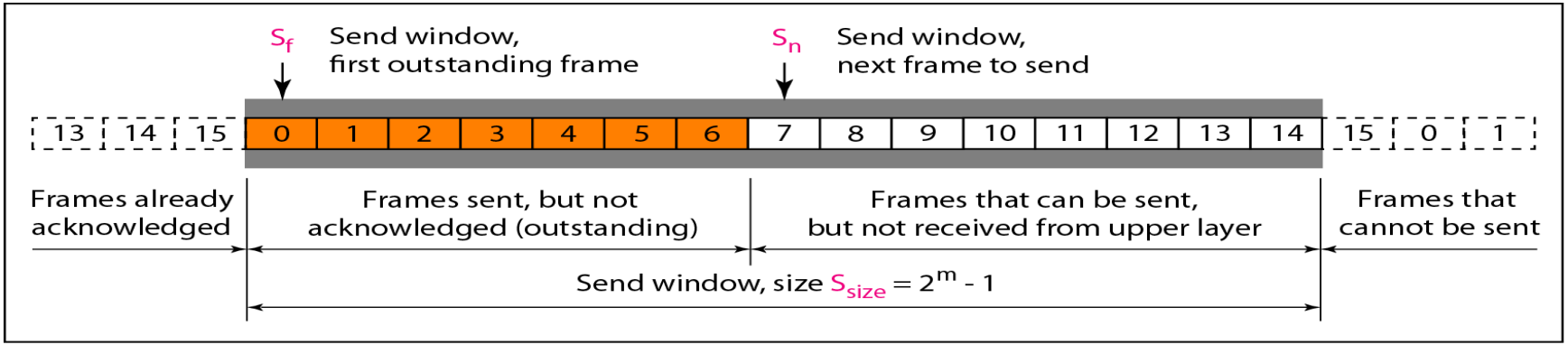
- If an ACK is damaged/lost, we can have two situations
- If the next ACK arrives before the expiration of any timer, there is no need for retransmission of frames because ACKs are cumulative in this protocol
- If ACK1, ACK2, and ACk3 are lost, ACK4 covers them if it arrives before the timer expires
- 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

Go Back N ARQ, sender window size

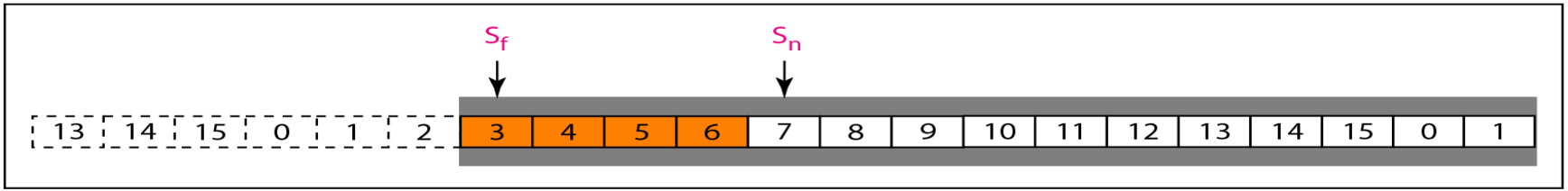
- Size of the sender window must be less than 2^m
- Size of the receiver is always 1



Go Back N ARQ (sending window)

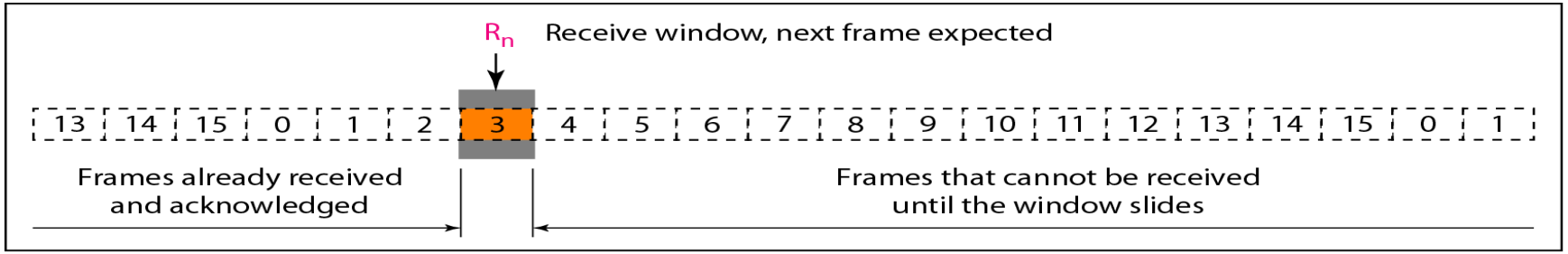


a. Send window before sliding

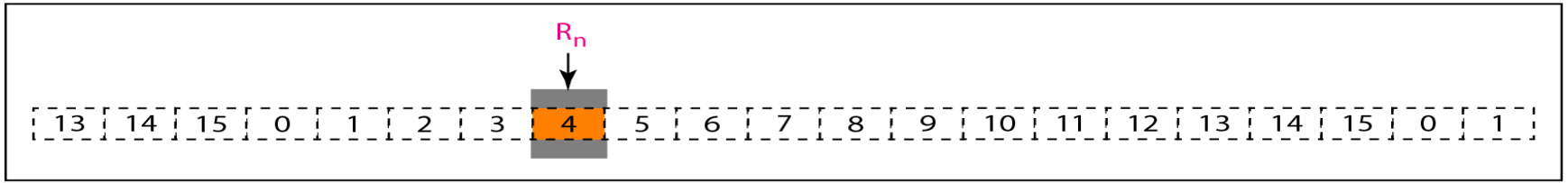


b. Send window after sliding

Go Back N ARQ (receiving window)

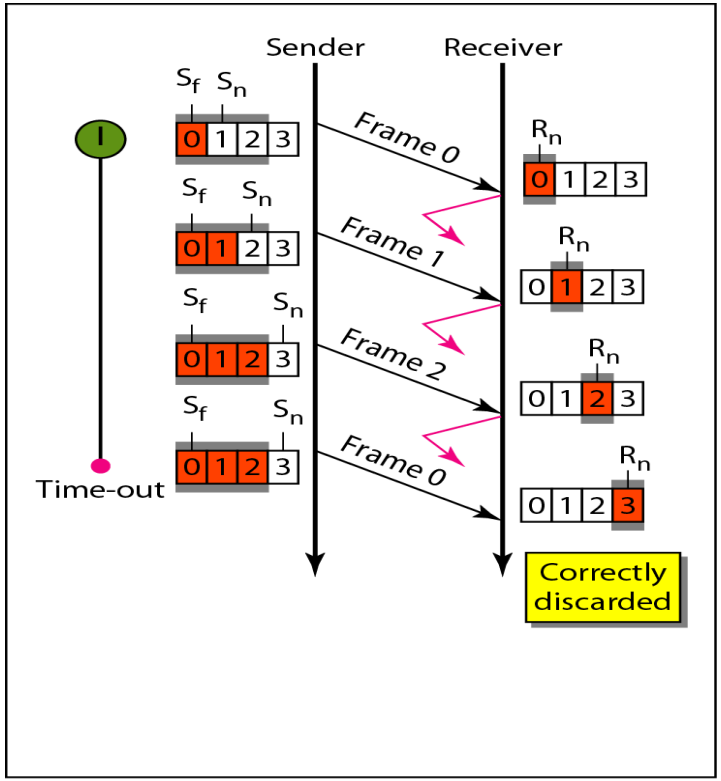


a. Receive window

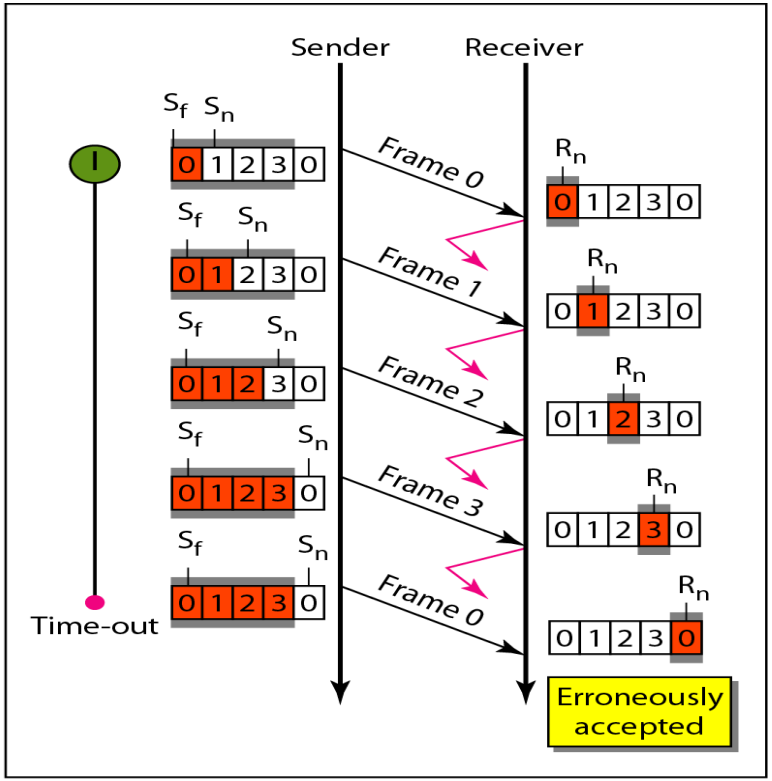


b. Window after sliding

Go Back N ARQ (sending and receiving window)



a. Window size $< 2^m$

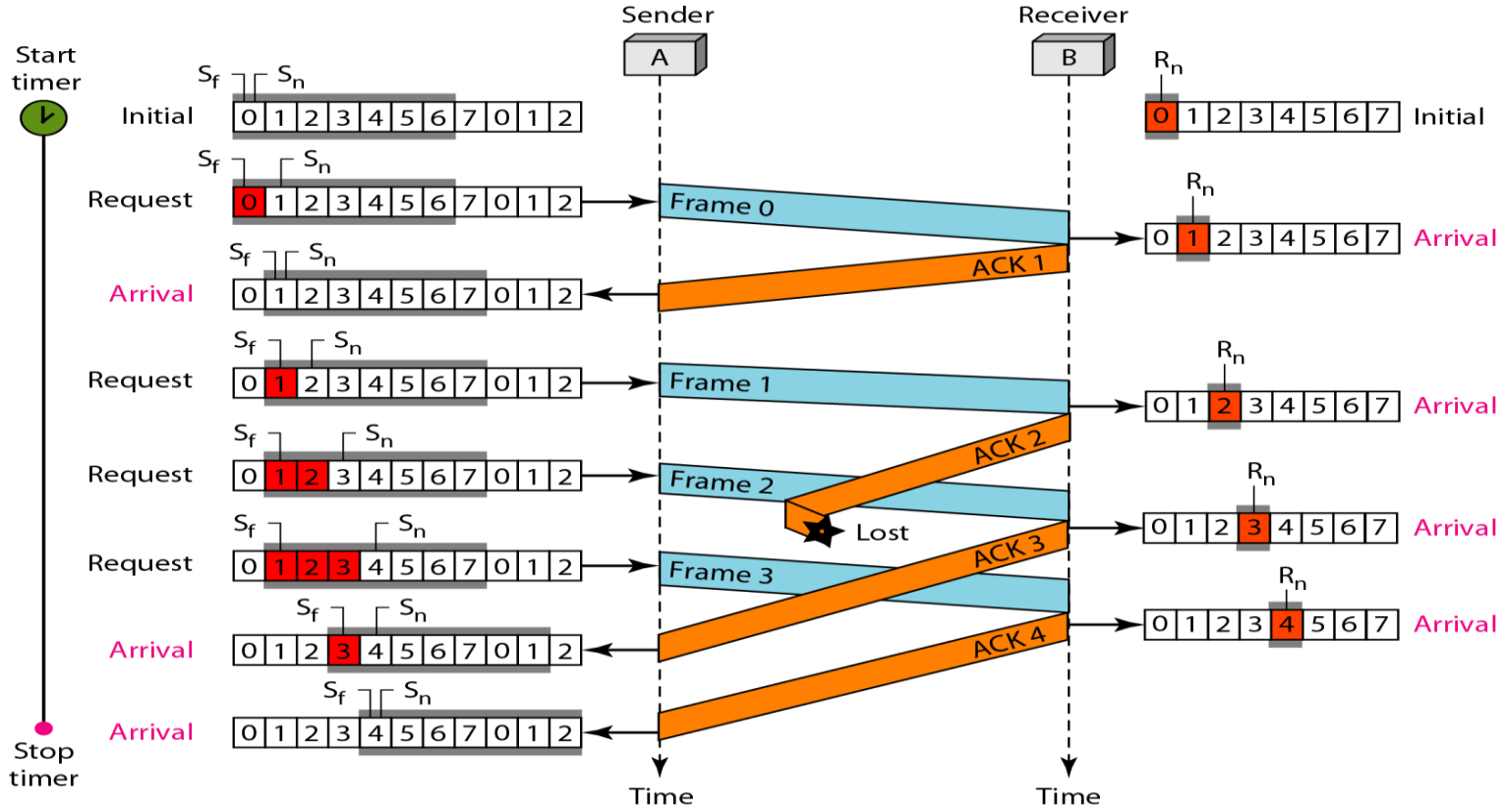


b. Window size $= 2^m$

Go Back N ARQ

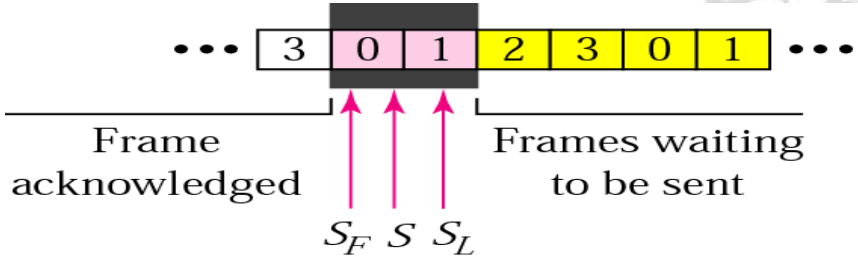
- Eg:
  - Figure shows an example of Go Back N. This is an example of a case where the forward channel is reliable, but the reverse is not. No data frames are lost, but some ACKs are delayed and one is lost. The example also shows how cumulative acknowledgments can help if acknowledgments are delayed or lost. After initialization, there are seven sender events. Request events are triggered by data from the network layer; arrival events are triggered by acknowledgments from the physical layer. There is no time-out event here because all outstanding frames are acknowledged before the timer expires. Note that although ACK 2 is lost, ACK 3 serves as both ACK 2 and ACK 3.

Go Back N ARQ

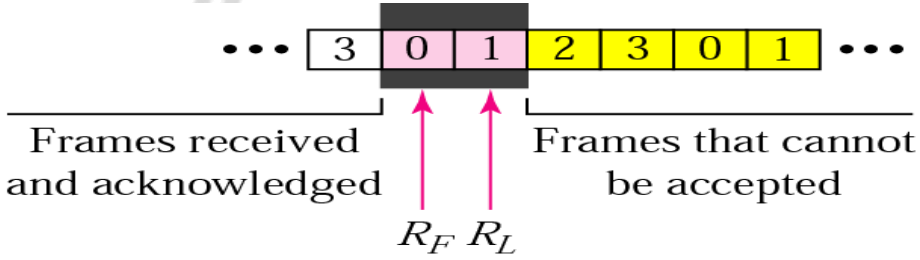

Sender
A
Receiver
B
Start timer
Stop timer
Initial
Request
Arrival
Frame 0
ACK 1
Frame 1
ACK 2
Frame 2
Lost
Frame 3
ACK 3
ACK 4
Time
Time

Selective Repeat ARQ, sender and receiver windows

- Go Back N ARQ 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 ARQ protocol is inefficient for noisy link
- In Selective Repeat ARQ, 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



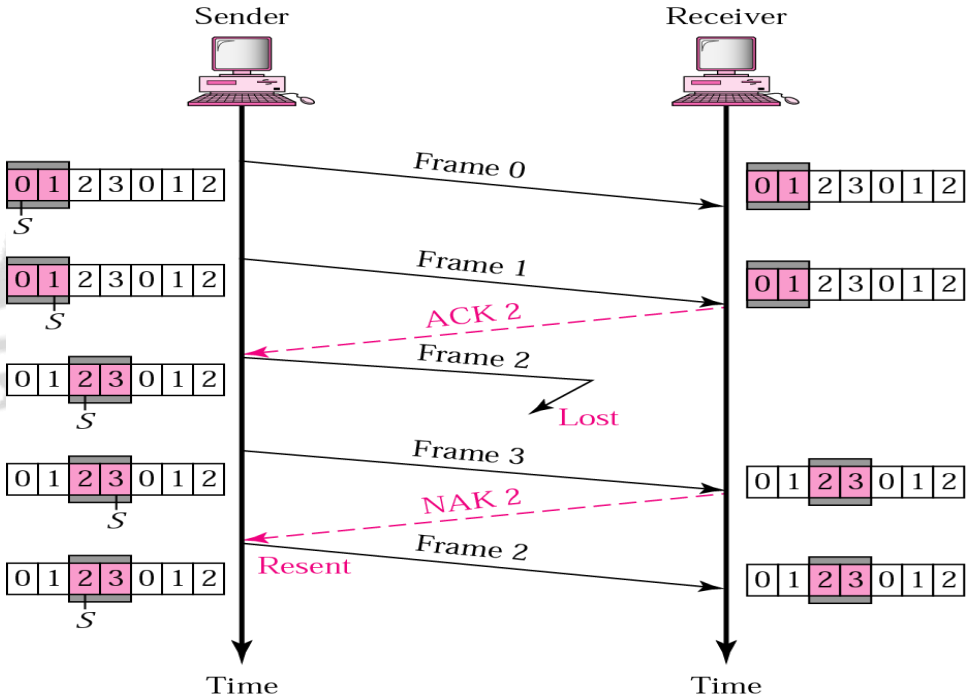
a. Sender window



b. Receiver window

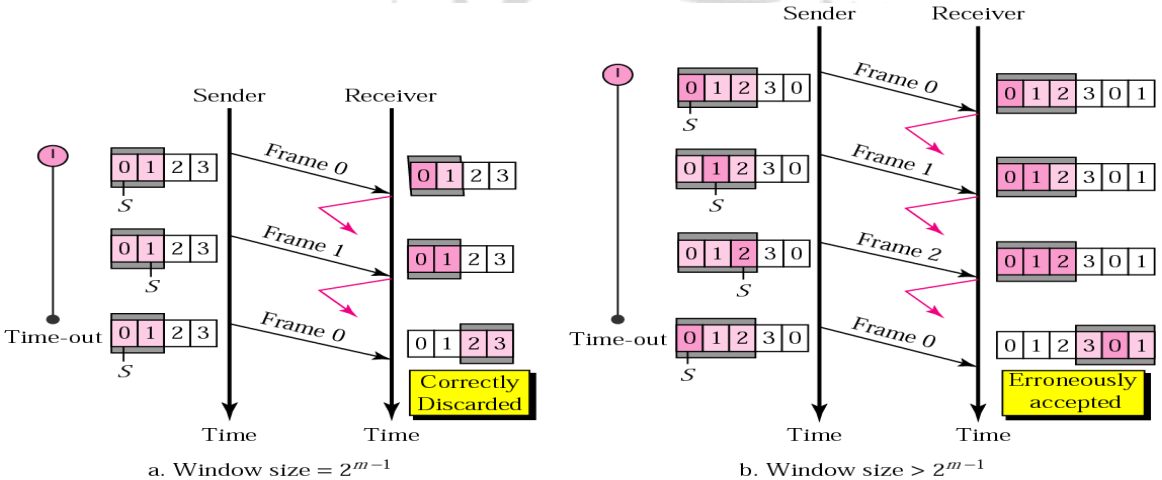
Selective Repeat ARQ, lost frame

- Frames 0 and 1 are accepted when received because they are in the range specified by the receiver window. Same for frame 3
- Receiver sends a NAK2 to show that frame 2 has not been received and then sender resends only frame 2 and it is accepted as it is in the range of the window

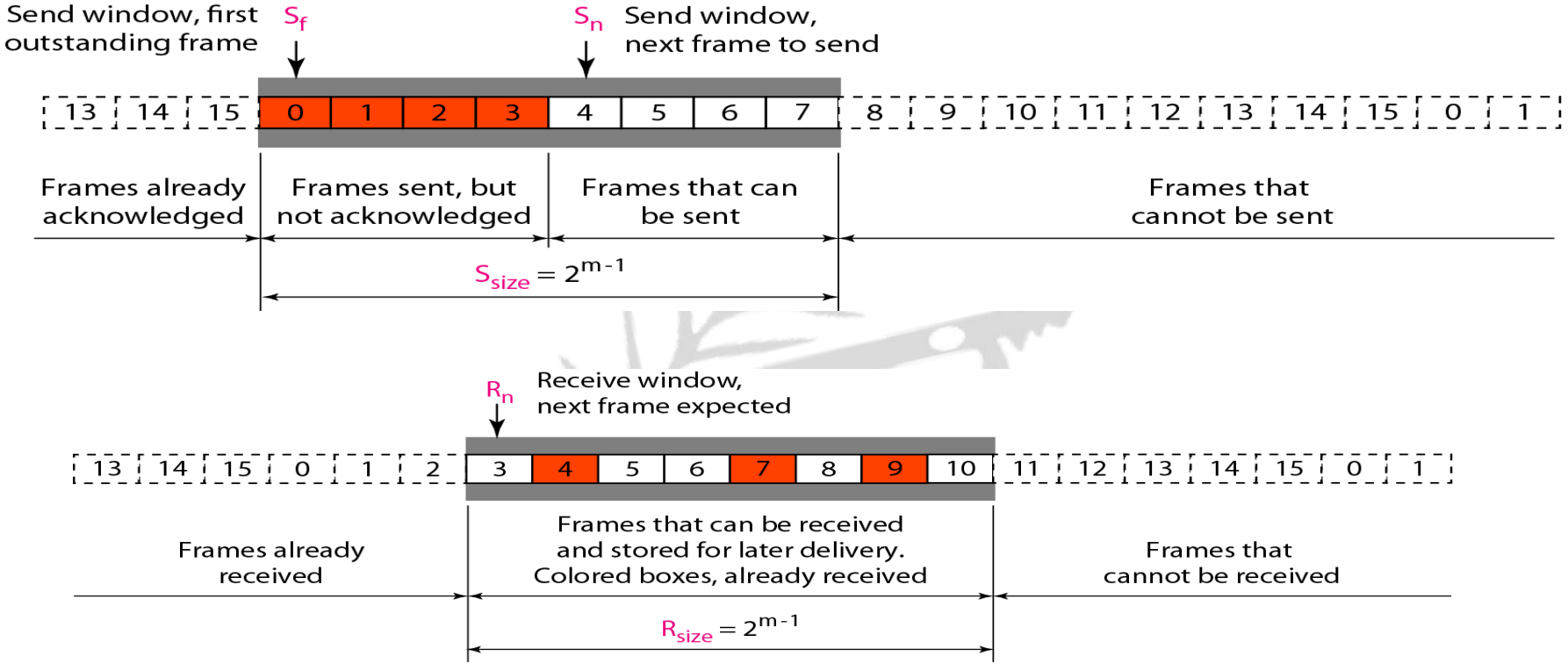


Selective Repeat ARQ, sender window size

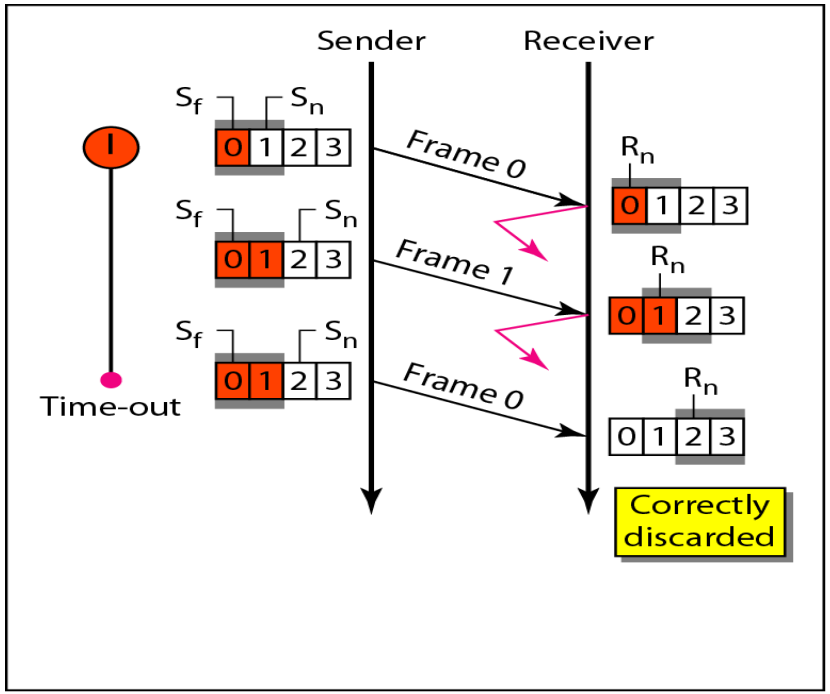
- Size of the sender and receiver windows must be at most one-half of 2^m. If m = 2, window size should be $2^m/2 = 2$
- Fig compares a window size of 2 with a window size of 3. Window size is 3 and all ACKs are lost, sender sends duplicate of frame 0, window of the receiver expect to receive frame 0, so accepts frame 0, as the 1st frame of the next cycle – an error



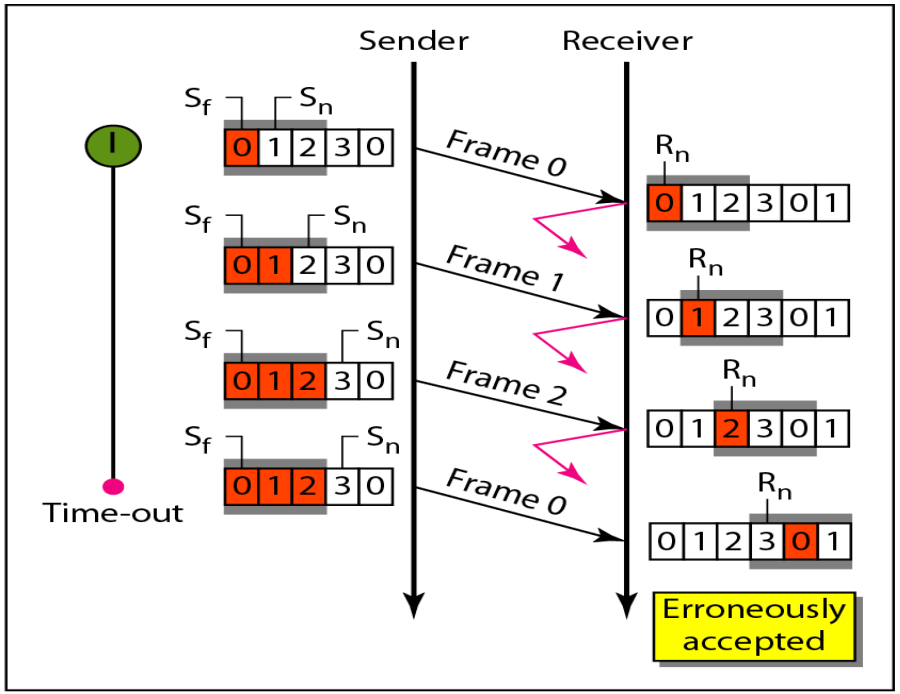
Window for Selective Repeat ARQ



Window for Selective Repeat ARQ

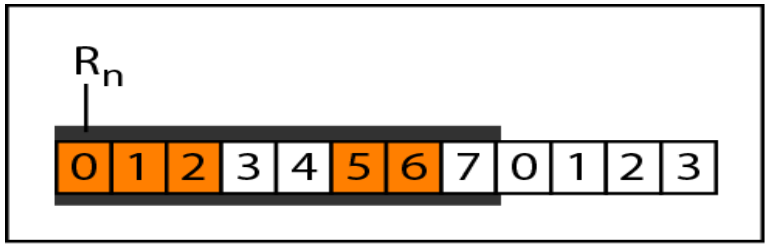


a. Window size = 2^{m-1}

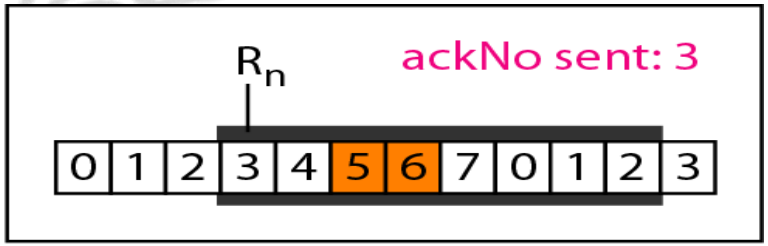


b. Window size > 2^{m-1}

Delivery of data in Selective Repeat ARQ



a. Before delivery



b. After delivery

Selective Repeat ARQ

- Eg:
  - This example is similar to previous example in which frame 1 is lost. We show how Selective Repeat behaves in this case. Figure 11.23 shows the situation. One main difference is the number of timers. Here, each frame sent or resent needs a timer, which means that the timers need to be numbered (0, 1, 2, and 3). The timer for frame 0 starts at the first request, but stops when the ACK for this frame arrives. The timer for frame 1 starts at the second request, restarts when a NAK arrives, and finally stops when the last ACK arrives. The other two timers start when the corresponding frames are sent and stop at the last arrival event.
  - At the receiver site we need to distinguish between the acceptance of a frame and its delivery to the network layer. At the second arrival, frame 2 arrives and is stored and marked, but it cannot be delivered because frame 1 is missing. At the next arrival, frame 3 arrives and is marked and stored, but still none of the frames can be delivered. Only at the last arrival, when finally a copy of frame 1 arrives, can frames 1, 2, and 3 be delivered to the network layer. There are two conditions for the delivery of frames to the network layer: First, a set of consecutive frames must have arrived. Second, the set starts from the beginning of the window.

Selective Repeat ARQ

- Eg: (contd…)
  - Another important point is that a NAK is sent after the second arrival, but not after the third, although both situations look the same. The reason is that the protocol does not want to crowd the network with unnecessary NAKs and unnecessary resent frames. The second NAK would still be NAK1 to inform the sender to resend frame 1 again; this has already been done. The first NAK sent is remembered (using the nakSent variable) and is not sent again until the frame slides. A NAK is sent once for each window position and defines the first slot in the window.
  - The next point is about the ACKs. Notice that only two ACKs are sent here. The first one acknowledges only the first frame; the second one acknowledges three frames. In Selective Repeat, ACKs are sent when data are delivered to the network layer. If the data belonging to n frames are delivered in one shot, only one ACK is sent for all of them.

Selective Repeat ARQ