

Flow and Error Control

Flow Control

- 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

- A technique for assuring that a transmitting station does not overwhelm a receiving station with data

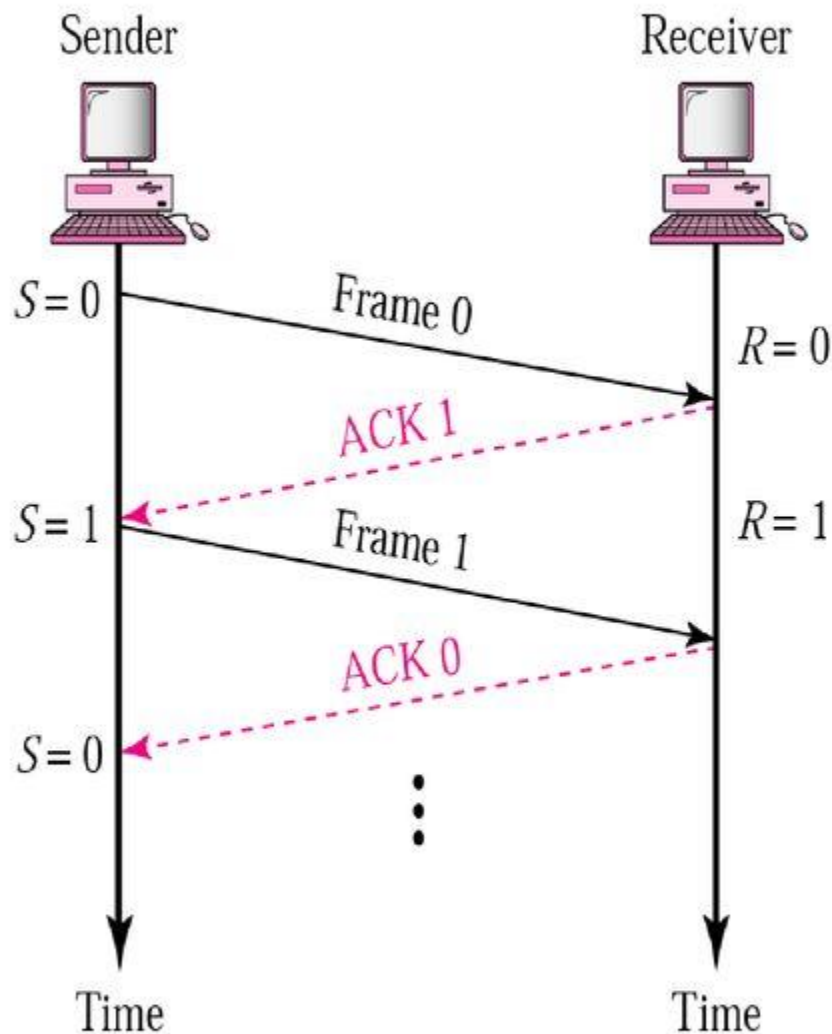
Why flow control needed

- Receiver has a limited speed at which it can process incoming data
- 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.

Flow Control Mechanisms

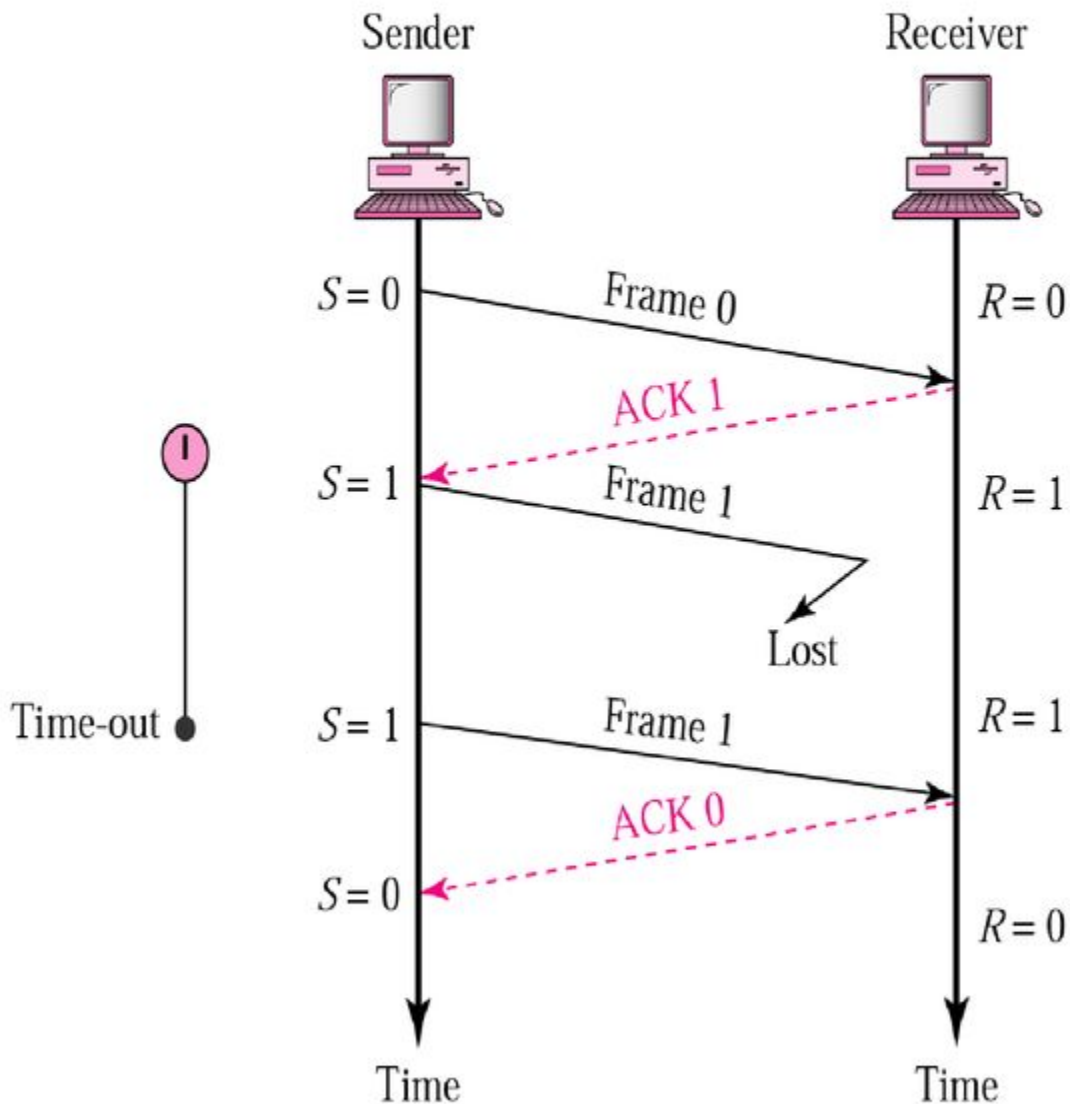
- Stop-and-Wait
- Go-Back-N ARQ
- Selective-Repeat ARQ

Stop-and-Wait



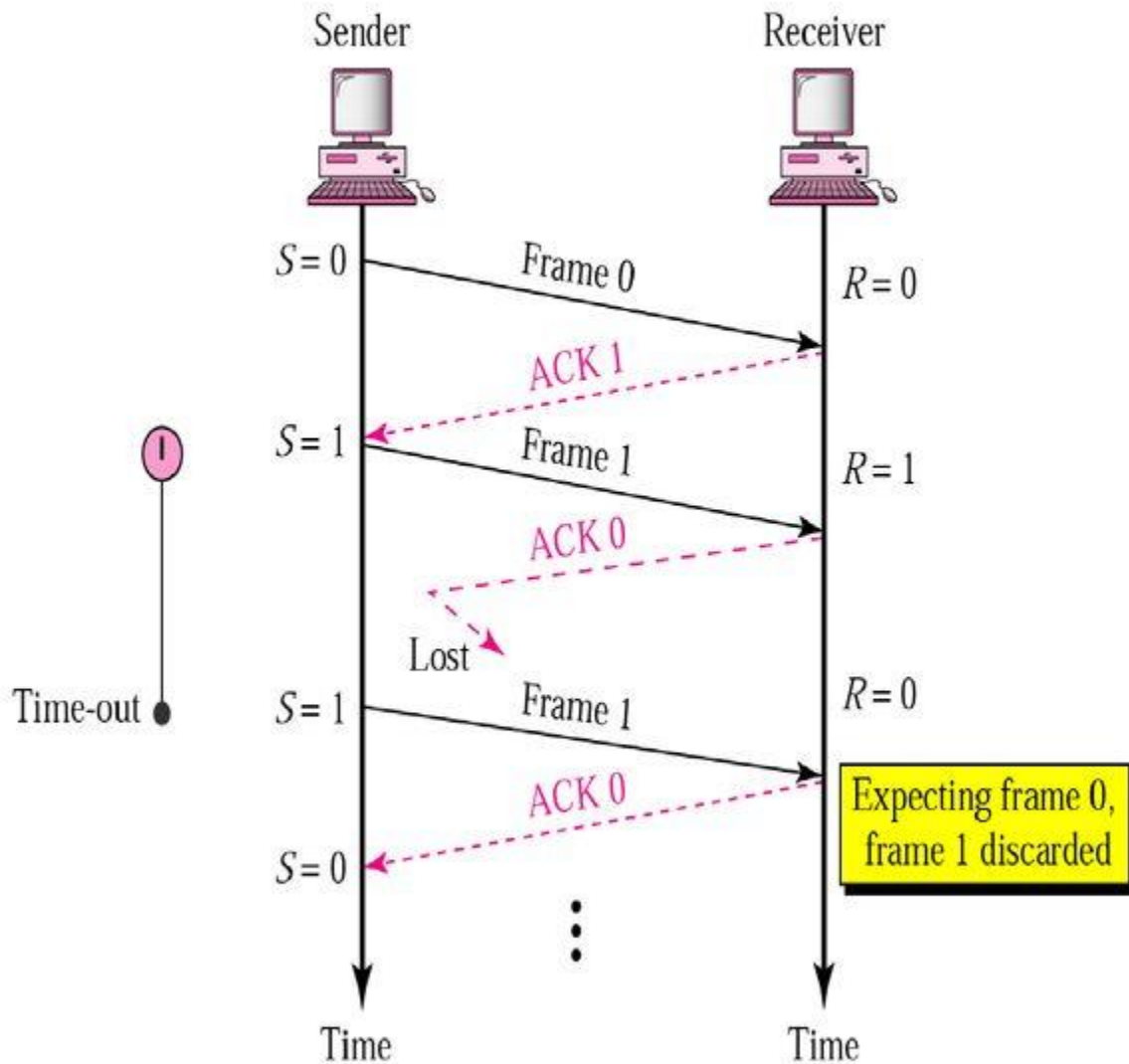
- Sender keeps a copy of the last frame until it receives an acknowledgement.
- For identification, both data frames and acknowledgements (ACK) frames are numbered alternatively 0 and 1.
- **Sender has a control variable (S)** that holds the number of the recently sent frame. (0 or 1)
- **Receiver has a control variable [®]** 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

Stop-and-Wait ARQ, lost ACK frame



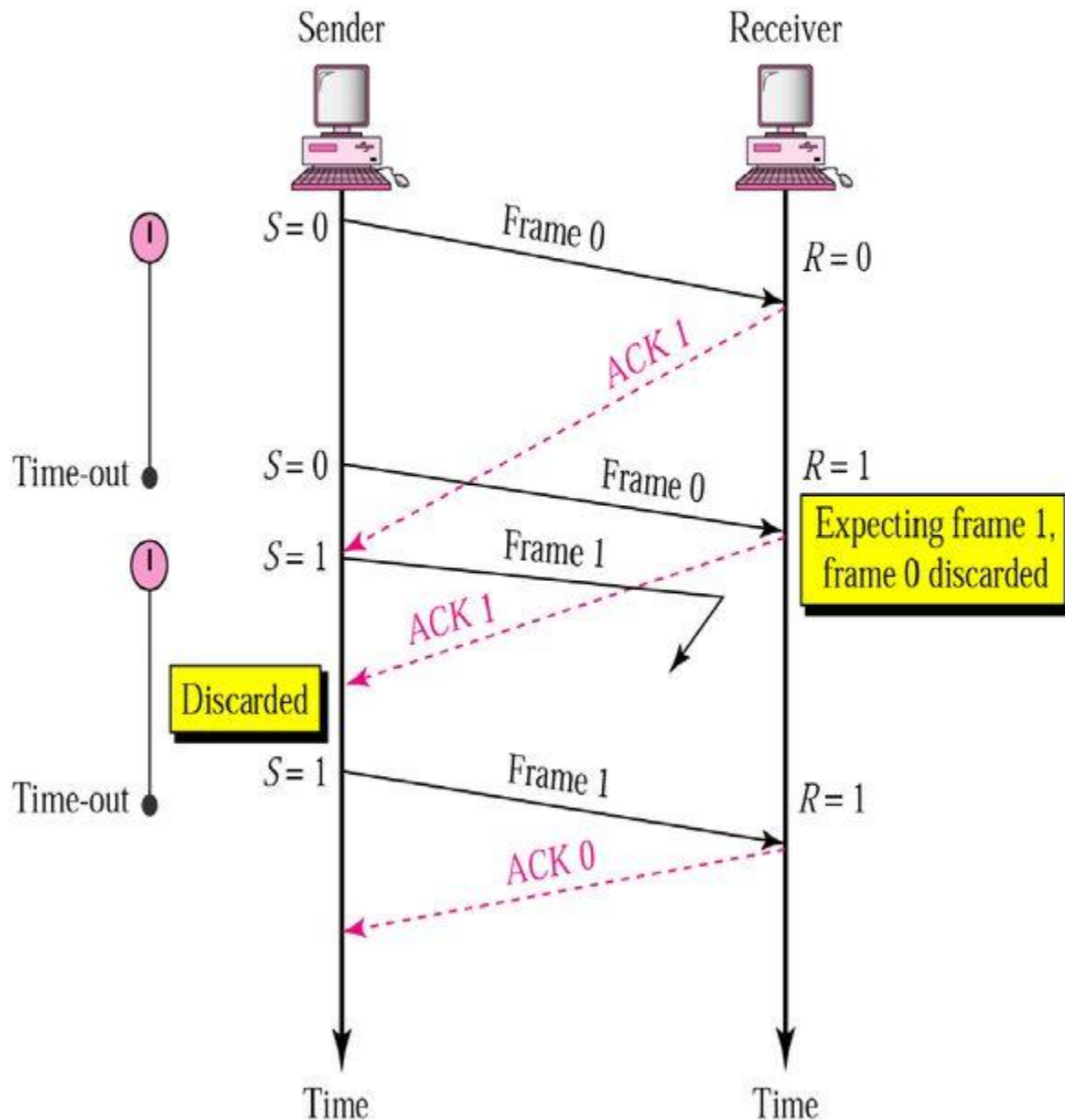
- When a receiver receives a damaged frame, it discards it and keeps its value of R .
- After the timer at the sender expires, another copy of frame 1 is sent.

Stop-and-Wait, lost ACK frame



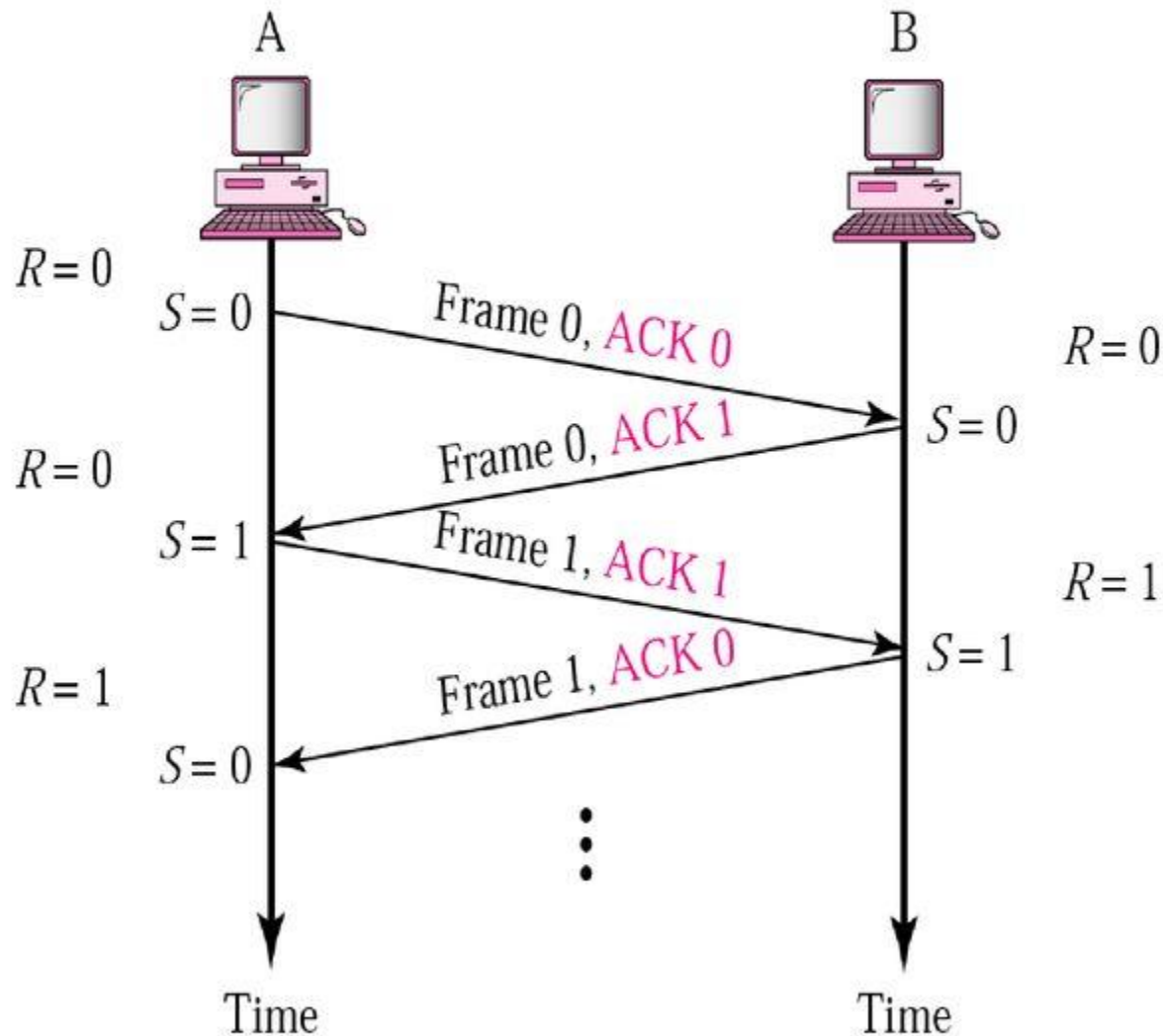
- If the sender receives a damaged ACK, it discards it.
- When the timer of the sender expires, the sender retransmits frame 1.
- Receiver has already received frame 1 and expecting to receive frame 0 ($R=0$). Therefore it discards the second copy of frame 1.

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.

Piggybacking



- A method to combine a data frame with ACK.
- Station A and B both have data to send.
- Instead of sending separately, station A sends a data frame that includes an ACK.
- Station B does the same thing.
- Piggybacking saves bandwidth.

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 protocols use the above concept,
 - **Go-Back-N ARQ**
 - **Selective Repeat ARQ**

Go-Back-N ARQ

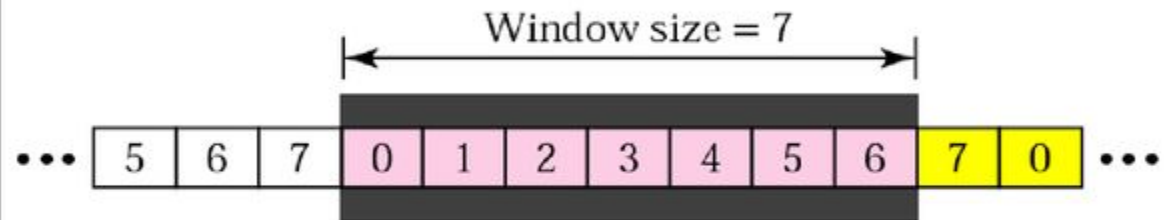
- We can send up to W frames before worrying about ACKs.
- We keep a copy of these frames until the ACKs arrive.
- This procedure requires additional features to be added to Stop-and-Wait ARQ.

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



a. Before sliding



b. After sliding two frames

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.



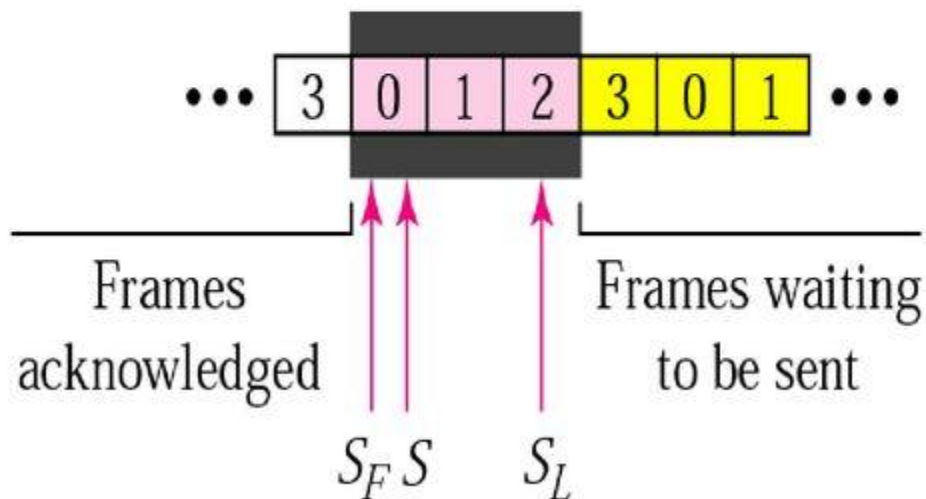
a. Before sliding



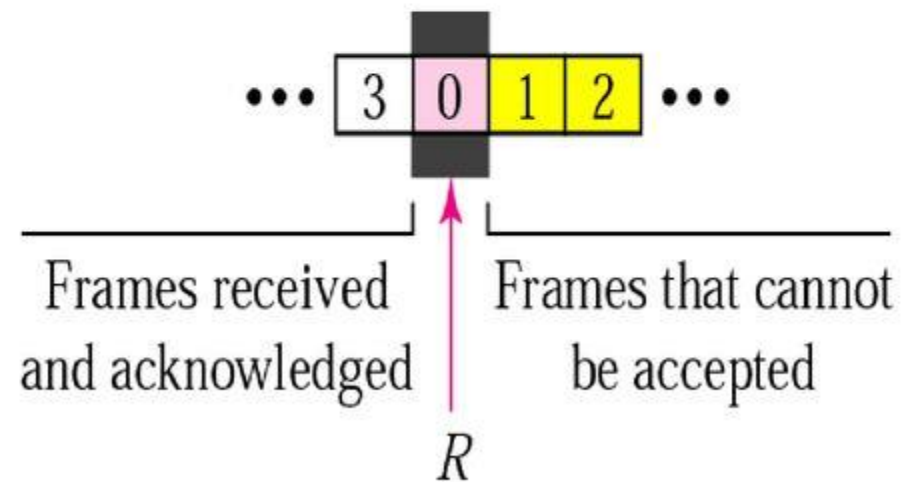
b. After sliding

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 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



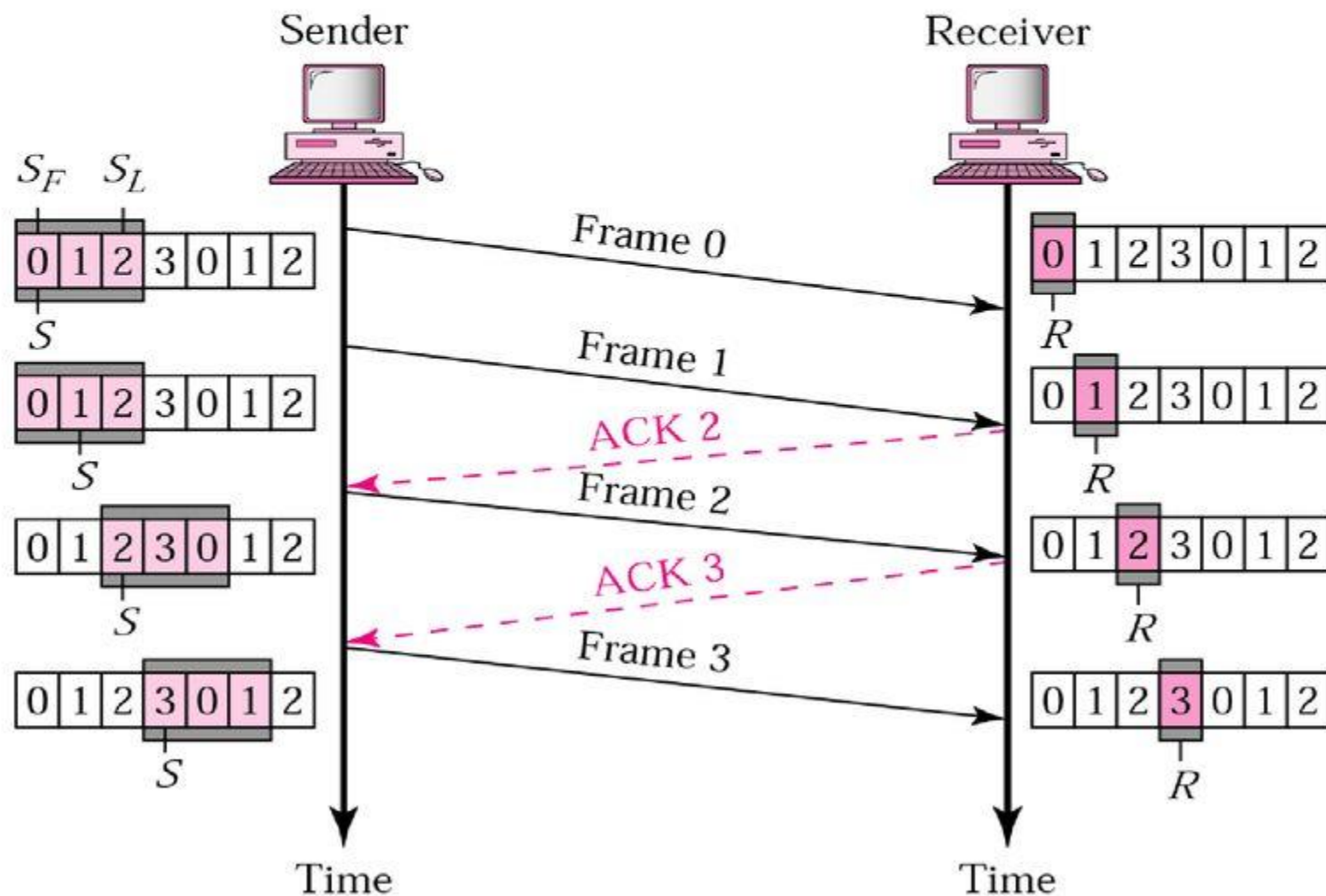
b. Receiver window

Acknowledgement

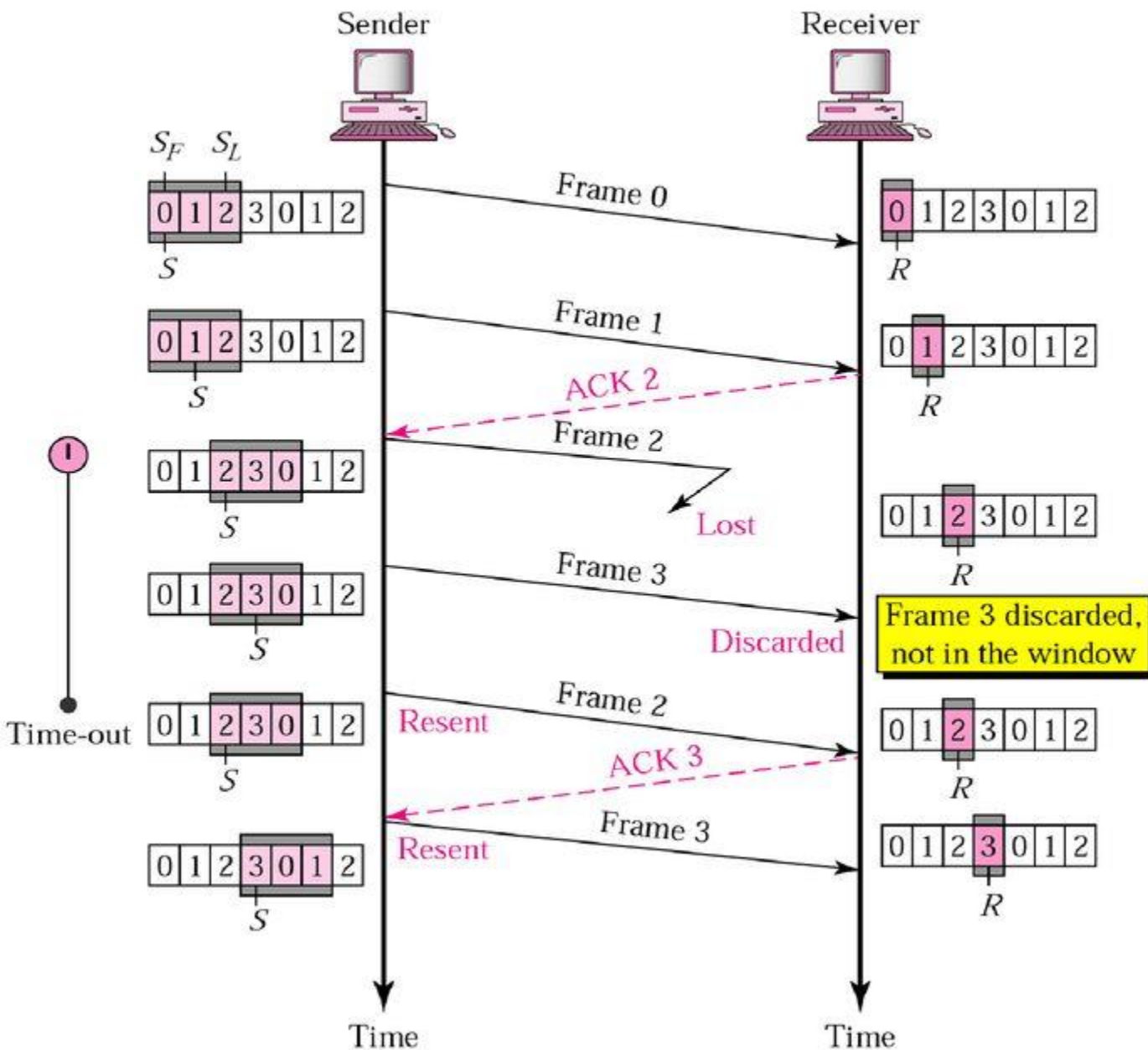
- Receiver sends positive ACK if a frame arrived safe and in order.
- If the frames are damaged/out of order, receiver is silent and discard all subsequent frames until it receives the one it is expecting.
- The silence of the receiver causes the timer of the unacknowledged frame to expire.
- Then the 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



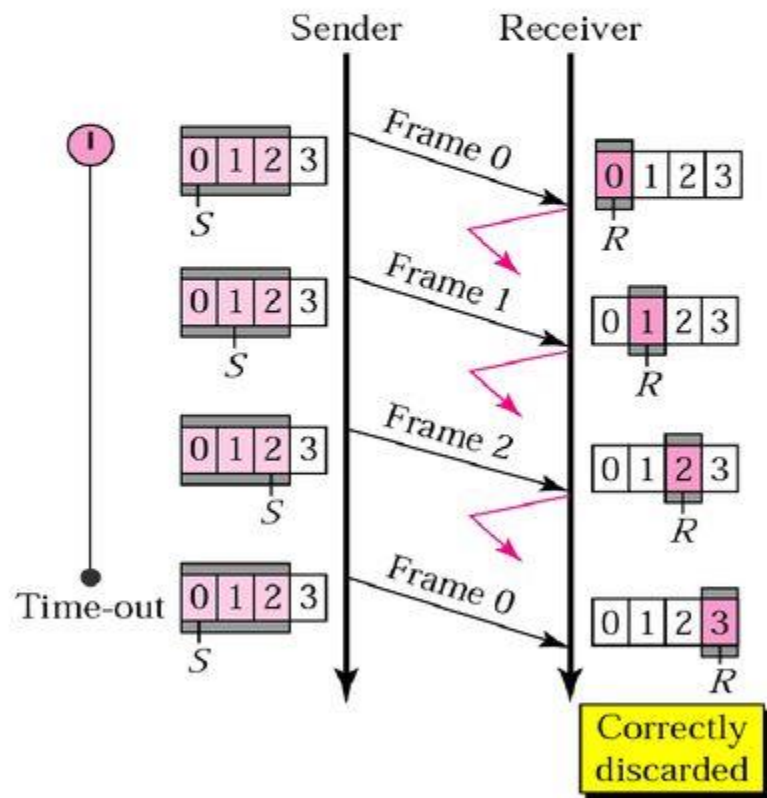
- Frame 2 is lost
- When the receiver receives frame 3, it discards frame 3 as it is expecting frame 2 (according to window).
- After the timer for frame 2 expires at the sender site, the sender sends frame 2 and 3. (go back to 2)

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

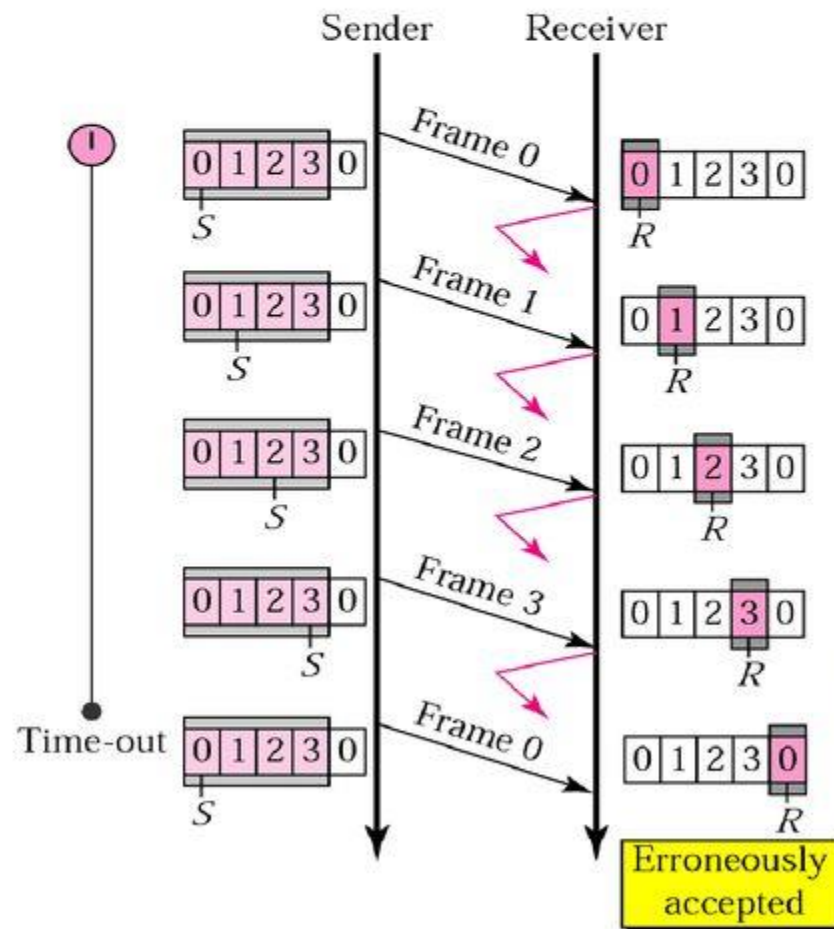
- If an ACK is damaged/lost, we can have two situations:
- If the next ACK arrives before the expiration of any timer, there is no need for retransmission of frames because ACKs are cumulative in this protocol.
- If ACK1, ACK2, and ACK3 are lost, ACK4 covers them if it arrives before the timer expires.
- 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. If $m = 2$, window size = $2^m - 1 = 3$.
- Fig compares a window size of 3 and 4.



a. Window size $< 2^m$

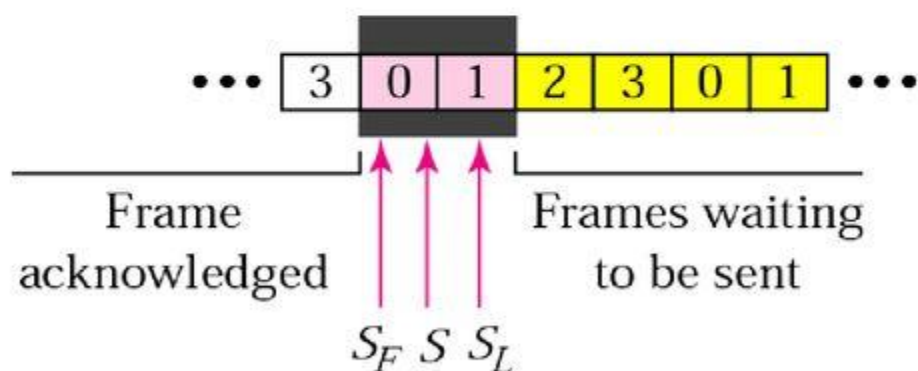


b. Window size $= 2^m$

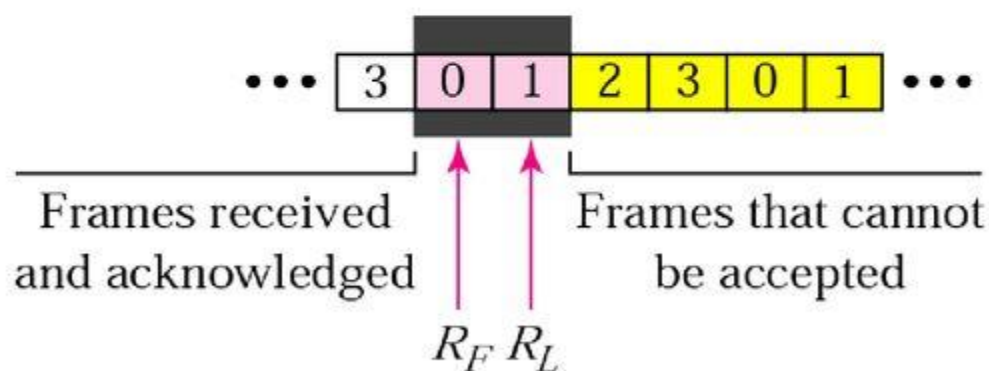
Accepts as the 1st frame in the next cycle-an error

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. It bandwidth inefficient and slows down the transmission.
- 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.

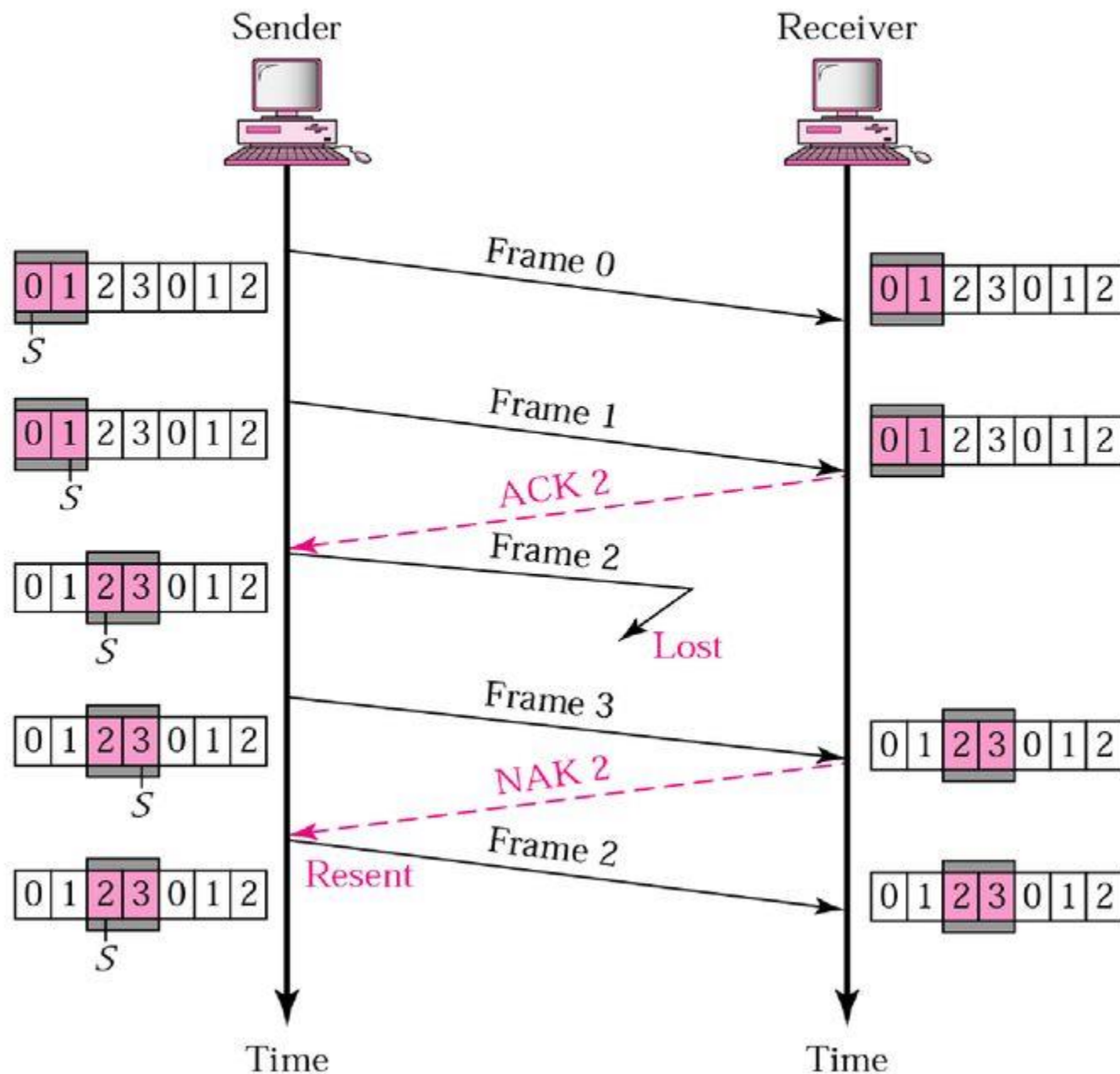


a. Sender window



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 (part of the window), so accepts frame 0, as the 1st frame of the next cycle – an **error**.

