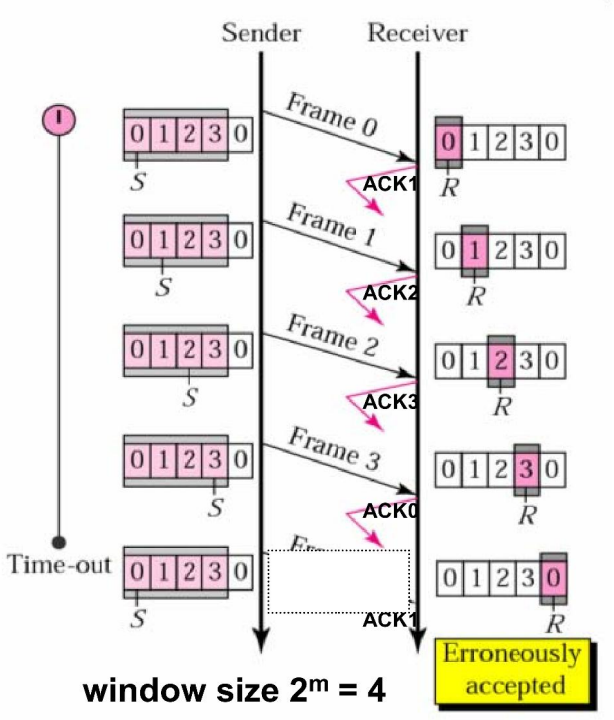
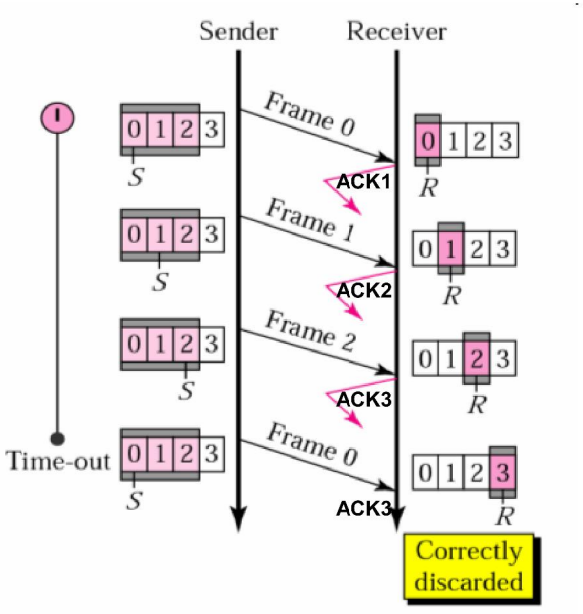
Go-Back-N ARQ (cont.)

Sequence Numbers and Window Size

- **m bits allotted within a header for seq. numbers ⇒ 2^m possible sequence numbers**
  - **how big should the sender window be!?**
  - **W > 2^m cannot be accepted – multiple frames with same seq. # in the window ⇒ ambiguous ACKs**
  - **W = 2^m can still cause some ambiguity – see below**
  - **W = 2^m – 1 acceptable !!!**



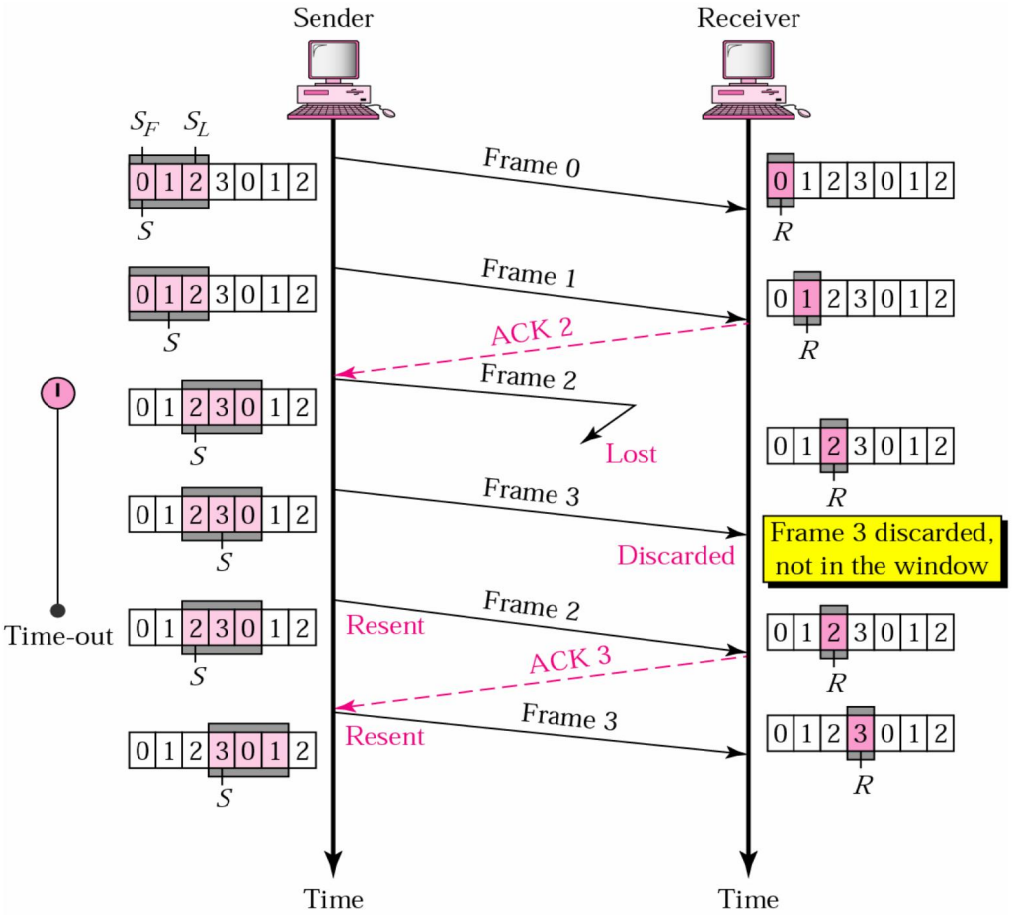
window size 2^m = 4



window size 2^m-1 = 3

Go-Back-N ARQ (cont.)

Example **[lost frame in Go-Back-N with time-out]**



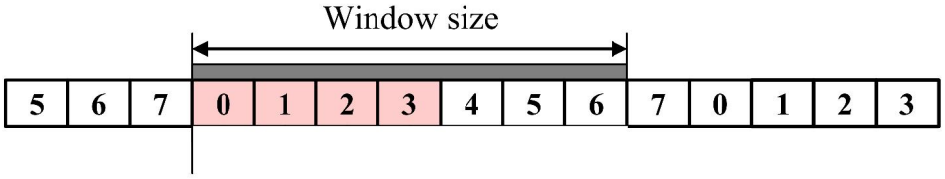
Note:

- **ACKs number always defines the number of the next expected frame !!!**
- **in Go-Back-N, receiver does not have to acknowledge each frame received – it can send one cumulative ACK for several frames**

Go-Back-N ARQ (cont.)

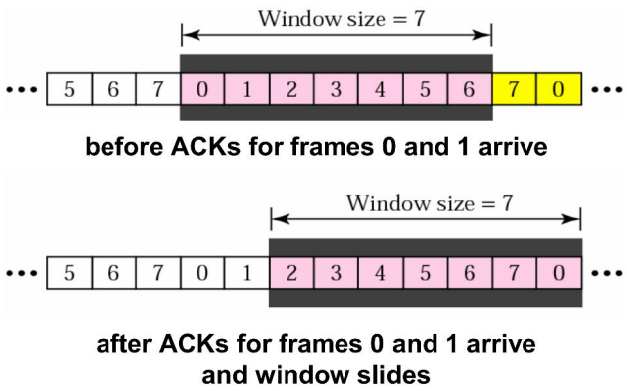
Problems with Go-Back-N (Go-Back-N with Timeout)

- **Go-Back-N works correctly (retransmission of damaged frames gets triggered) as long as the sender has an unlimited supply of packets that need to be transmitted**
  - **but, in case when packets arrive sporadically, there may not be W_s-1 subsequent transmissions ⇒ window will not be exhausted, retransmissions will not be triggered**
  - **this problem can be resolved by modifying Go-Back-N such that:**
    1) **set a timer for each sent frame**
    2) **resend all outstanding frames either when window gets full or when the timer of first frame expires**



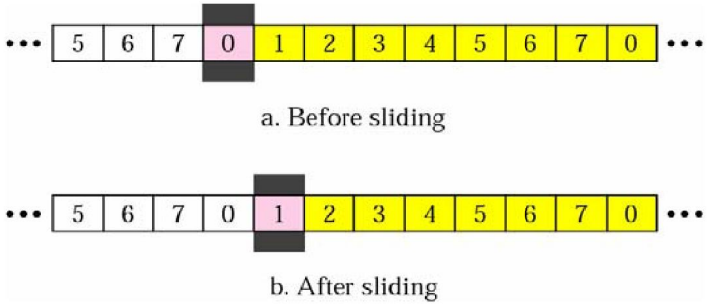
Go-Back-N ARQ (cont.)

Sender Sliding Window



- all frames are stored in a buffer, outstanding frames are enclosed in a window
  - frames to the left of the window are already ACKed and can be purged
  - frames to the right of the window cannot be sent until the window slides over them
  - whenever a new ACK arrives, the window slides to include new unsent frames
  - once the window gets full (max # of outstanding frames is reached), entire window gets resent

Receiver Sliding Window



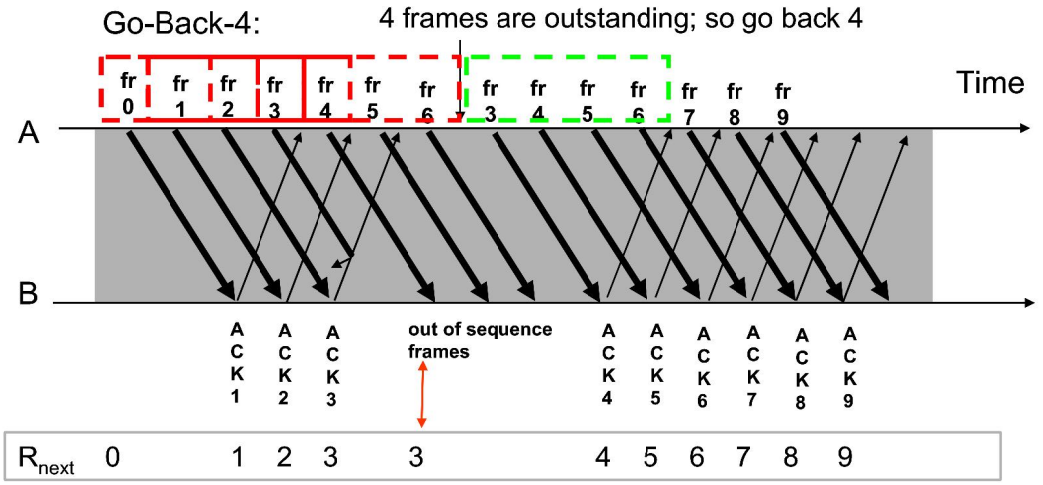
- the size of receiver window is always 1
  - receiver is always looking for a specific frame to arrive in a specific order
  - any frame arriving out of order is discarded and needs to be resent

The complexity of the receiver in Go-Back-N is the same as that of Stop-and-Wait!!!
Only the complexity of the transmitter increases.

Go-Back-N ARQ

Go-Back-N ARQ – **overcomes inefficiency of Stop-and-Wait ARQ – sender continues sending enough frames to keep channel busy while waiting for ACKs**

- **a window of W_s outstanding frames is allowed**
- **m-bit sequence numbers are used for both - frames and ACKs, and $W_s = 2^m - 1$**



Assume: $W_s = 4$

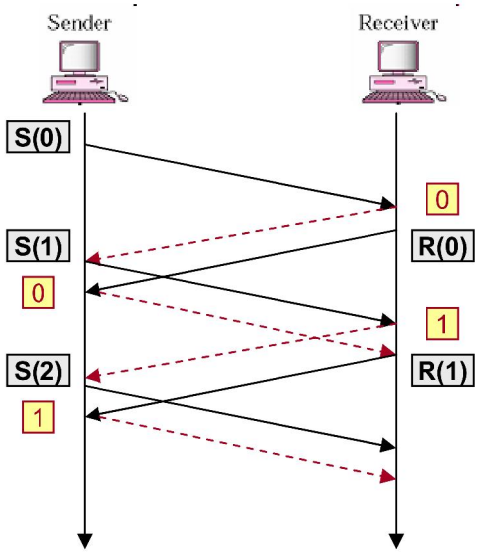
1) **sender sends frames one by one**
2) **frame 3 undergoes transmission error – receiver ignores frame 3 and all subsequent frames**
3) **sender eventually reaches max number of outstanding frames, and takes following action:**
   - **go back $N=W_s$ frames and retransmit all frames from 3 onwards**

(2) Go-Back-N ARQ

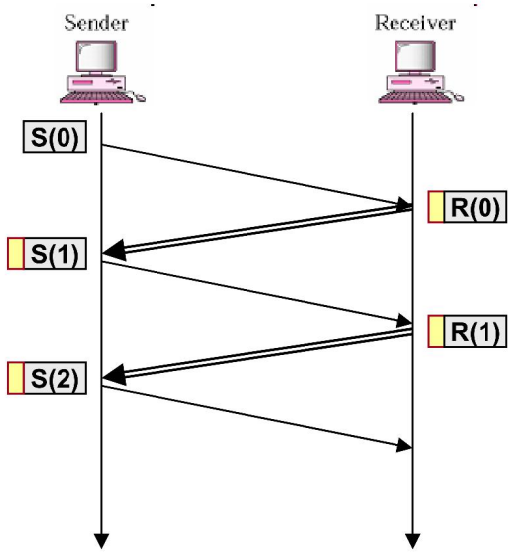
Stop-and-Wait ARQ (cont.)

Piggybacking

- Stop-and-Wait discussed so far was 'unidirectional'
- in 'bidirectional' communications, both parties send & acknowledge data, i.e. both parties implement flow control
- piggybacking method: outstanding ACKs are placed in the header of information frames
- piggybacking can save bandwidth since the overhead from a data frame and an ACK frame (addresses, CRC, etc) can be combined into just one frame



without piggybacking



with piggybacking