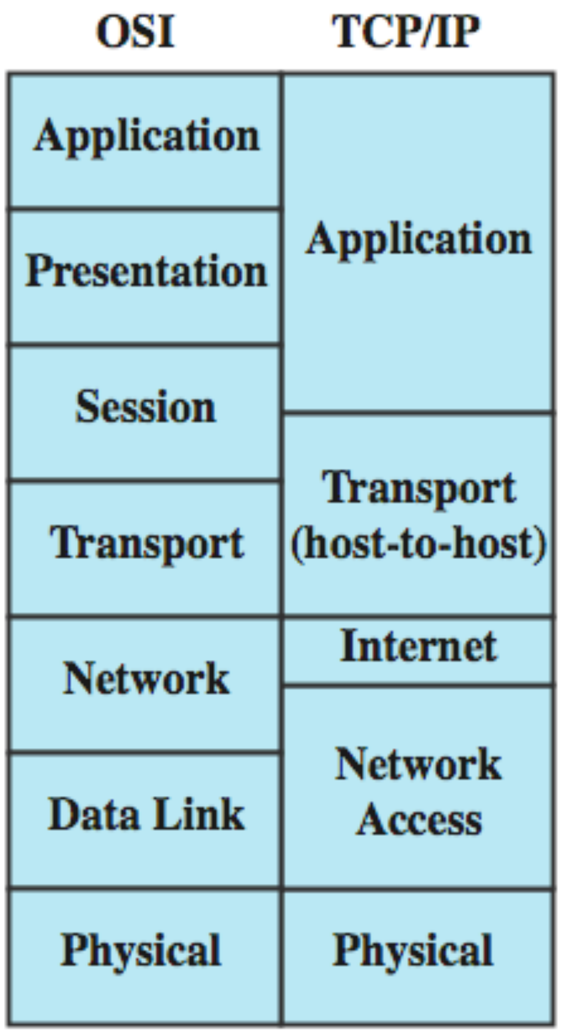
ECE230X Lectures 14-15

Data and Computer Communications Eighth Edition
By William Stallings
Chapter 7 – "Data Link Control Protocols"

D. Richard Brown III
Worcester Polytechnic Institute
Electrical and Computer Engineering Department

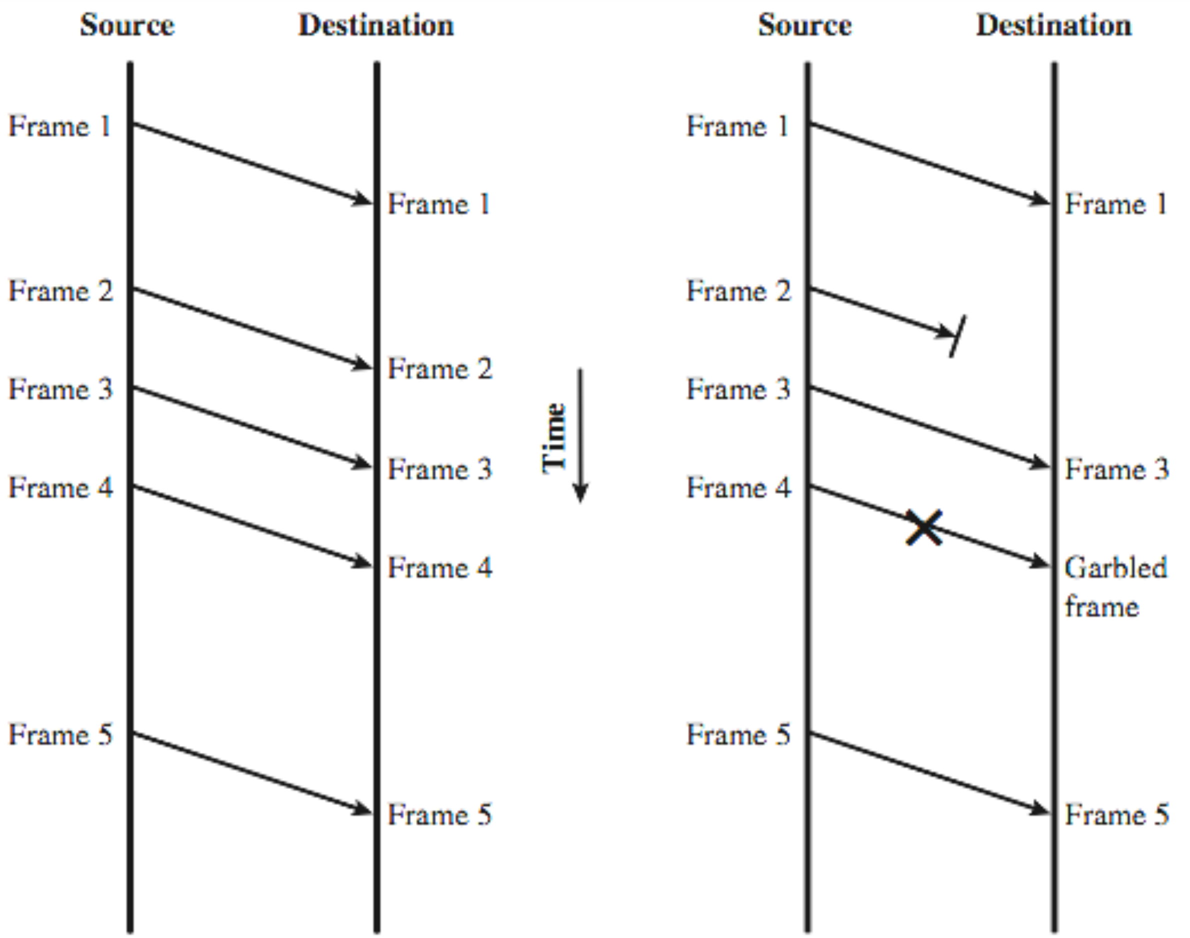
Adapted from Prentice Hall instructor resources

Data Link Control Protocols



- Data link (or network access) layer typically handles
  - Frame synchronization (Chap 6.1)
  - Flow control (Chap 7)
  - Error control (Chap 7)
  - Addressing (coming soon)

Model of Frame Transmission



(a) Error-free transmission

(b) Transmission with losses and errors

Flow Control

- Needed to avoid possible buffer overflow at receiver
  - Usually accomplished through acknowledgement (ACK) of good frames by the receiver
- Factors:
  - transmission time: time needed to emit the frame into the medium (t_{frame})
  - propagation time: time needed for the frame to traverse the link (t_{prop})
  - acknowledgement time: time needed to emit the ACK into the medium (t_{ack})
  - processing time: time needed to process received frames/ACKs (t_{proc})

Link Utilization

$$U = \frac{\text{time spent actually transmitting } n \text{ frames of data}}{\text{total time to send } n \text{ frames (and receive ACKs)}}$$

- Total time includes
  - Time to transmit frame from sender to receiver
  - Propagation time from sender to receiver
  - Time to process frame at receiver
  - Time to transmit ACK from receiver to sender
  - Time to process ACK at sender

Flow Control Method 1: Stop and Wait

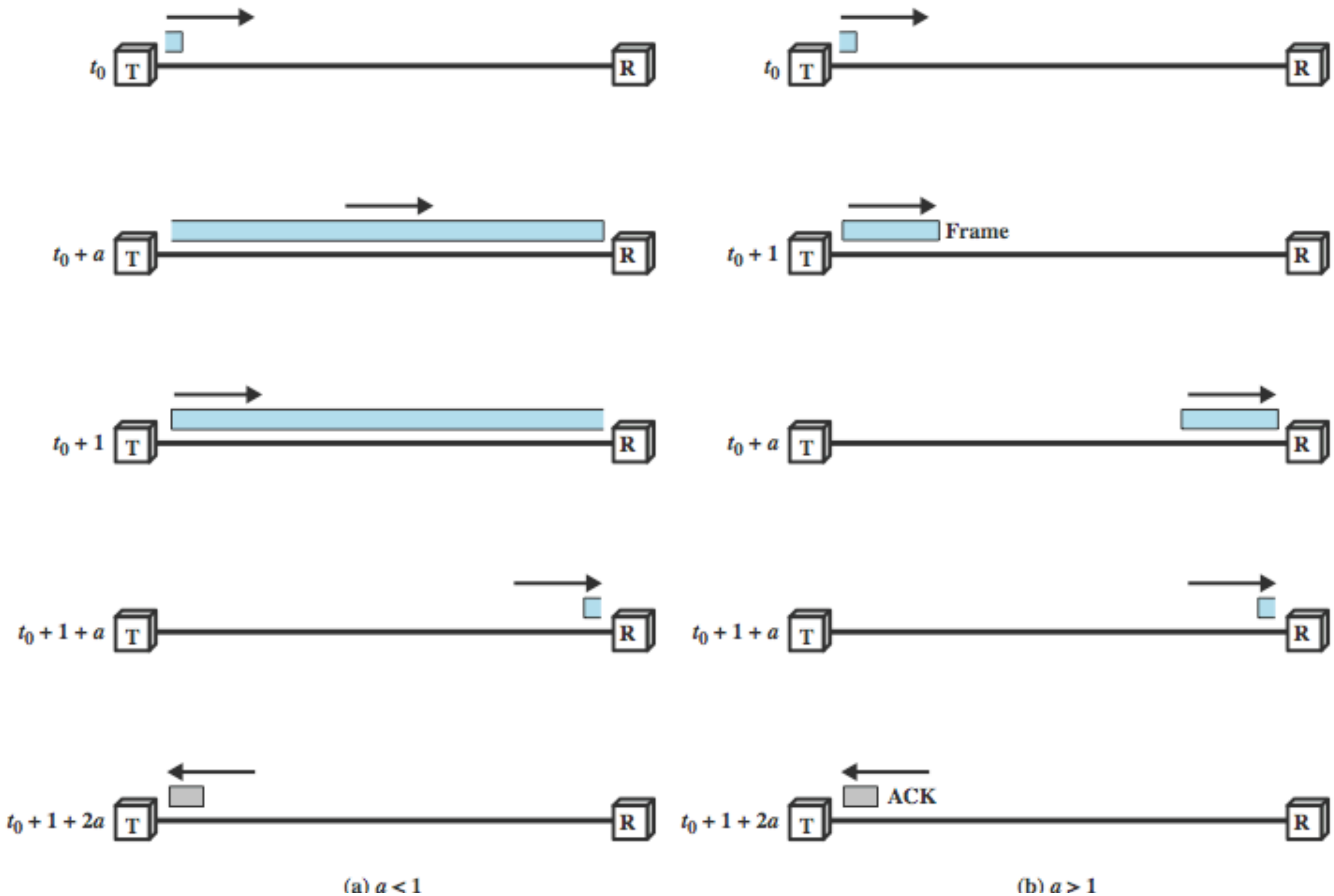
Assume no dropped packets or errors…

1. Source transmits frame
2. Destination receives frame, checks for errors, and replies with acknowledgement (ACK) if frame is ok
3. Source waits for ACK.
4. Go to step 1 if more frames are to be sent.

Remarks:

- Source waits for ACK before sending next frame
- Destination can stop flow by not sending ACK
- Simple and works ok for a few large frames
- Stop and wait becomes inefficient if large block of data is split into small frames

Stop and Wait Link Utilization ($t_{frame}=1$, $t_{prop}=a$)



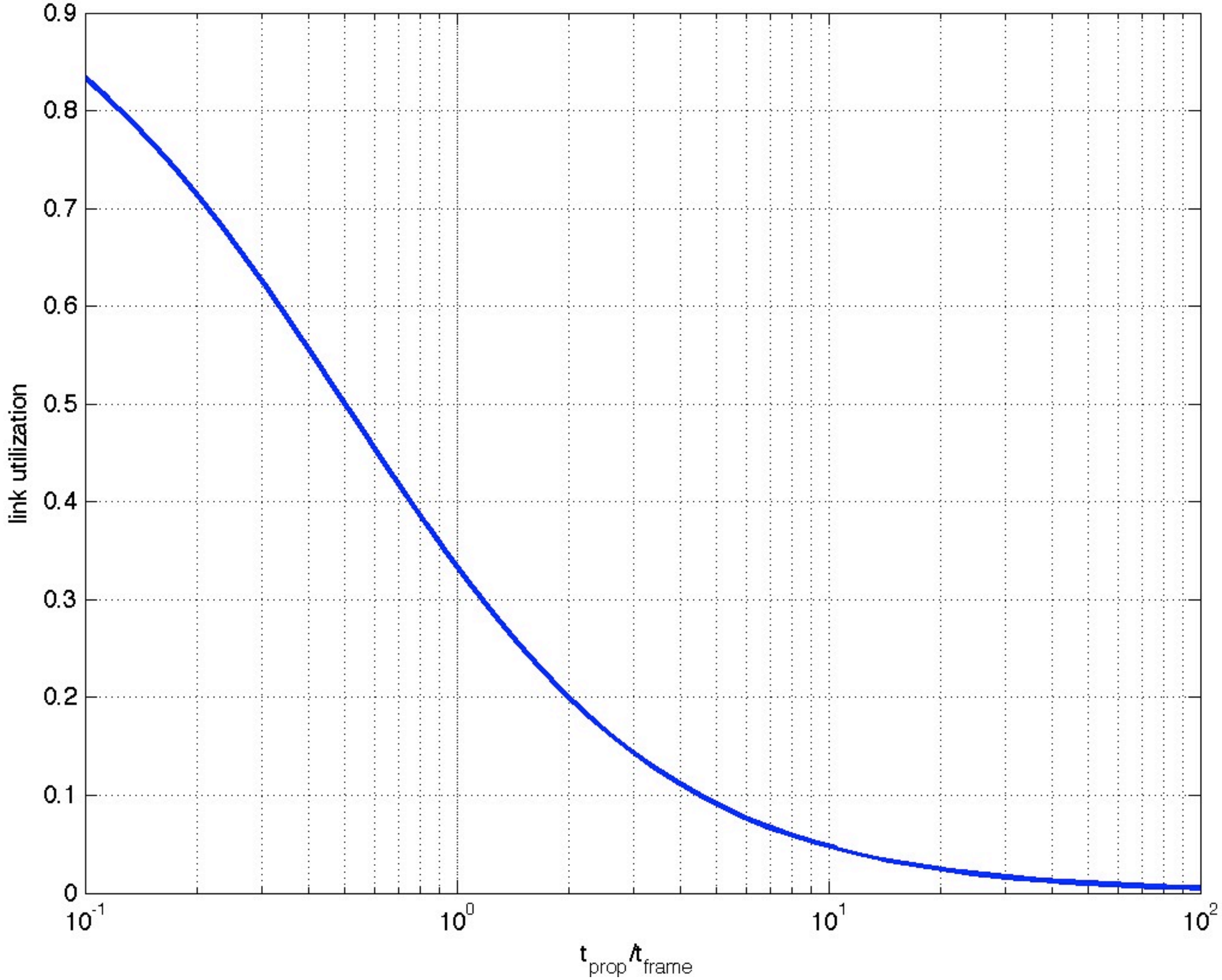
Stop and Wait Link Utilization

- We typically assume that processing time (t_{proc}) and ACK transmission time (t_{ack}) are much smaller than frame transmission time (t_{prop}) and propagation time (t_{prop})

$$U \approx \frac{nt_{frame}}{n(2t_{prop} + t_{frame})} = \frac{t_{frame}}{2t_{prop} + t_{frame}} = \frac{1}{2a + 1}$$

where $a = t_{prop}/t_{frame}$

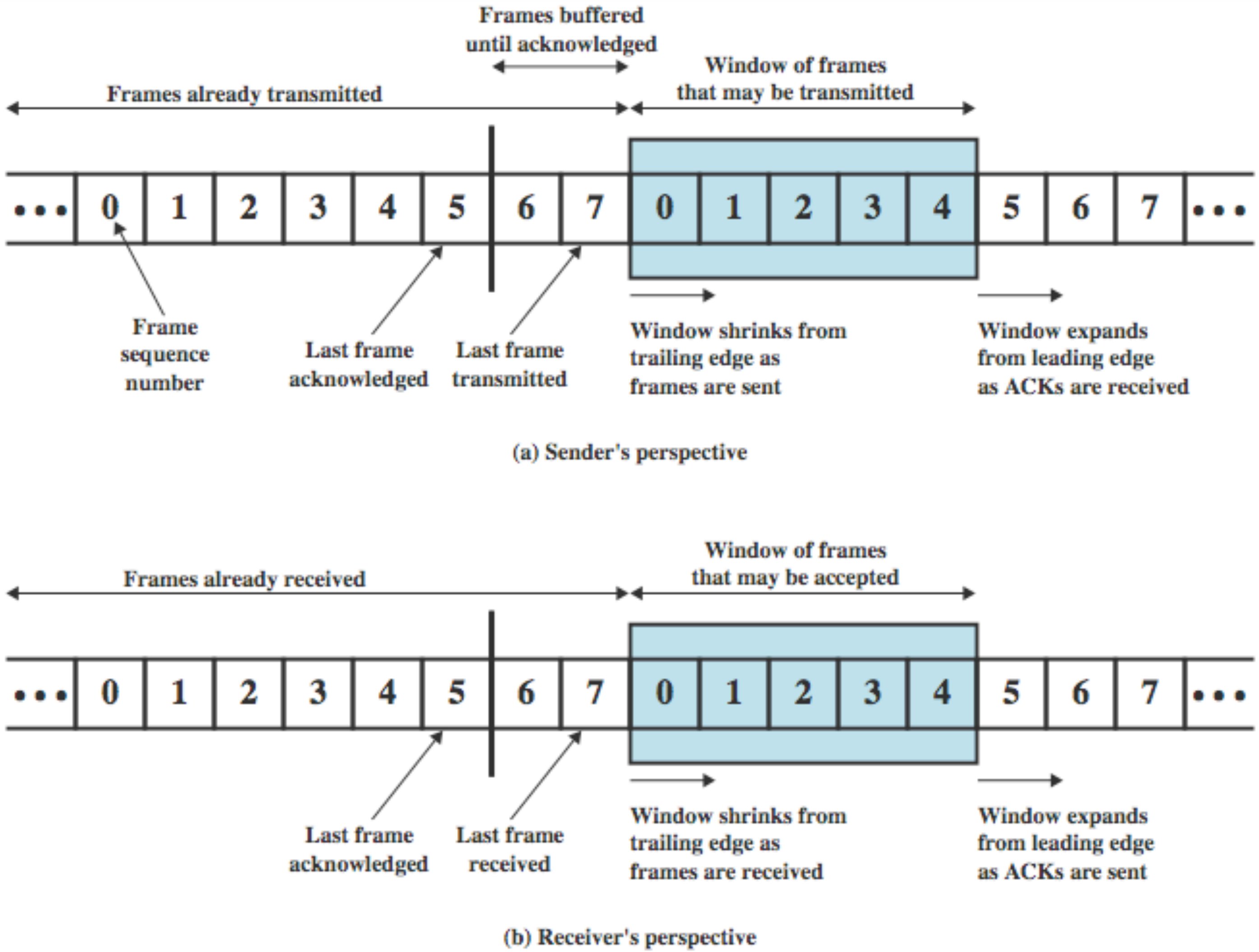
Stop and Wait Link Utilization



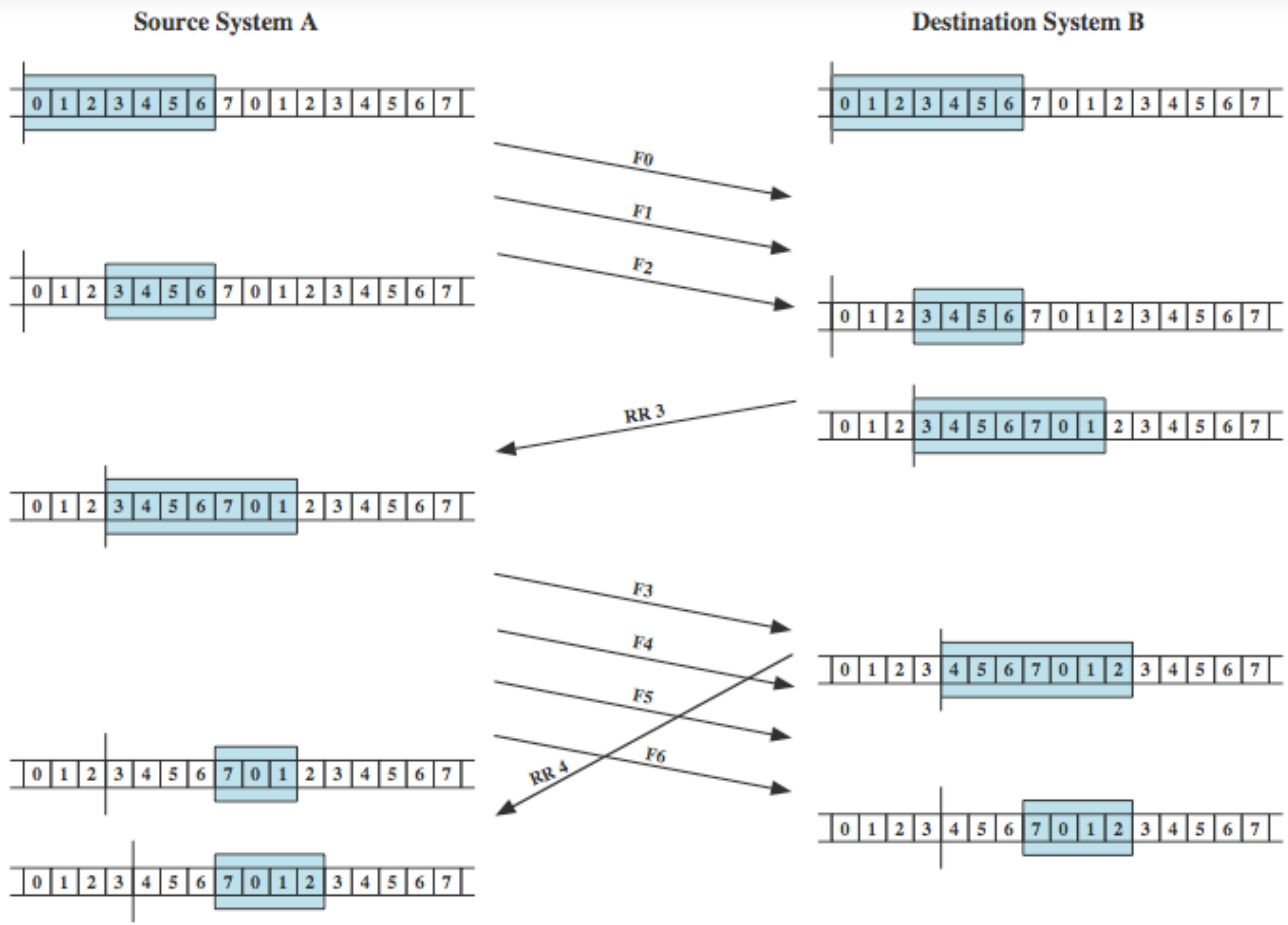
Flow Control Method 2: Sliding-Window

- Basic idea:
  - Add sequence number (0 to W-1) to all frames
  - Transmit multiple frames (up to W) in sequence without waiting for ACK
  - Receiver occasionally sends a "**Cumulative Acknowledgement**" ("receive ready" (RR)) with the next expected frame number
  - Improves link utilization by not waiting for ACK/RR after each frame.
- Receiver can also halt transmission ("receive not ready" (RNR))
  - Must send a normal RR to resume

Sliding Window Diagram With 3-bit Sequence Number (0-7)



Sliding Window Example



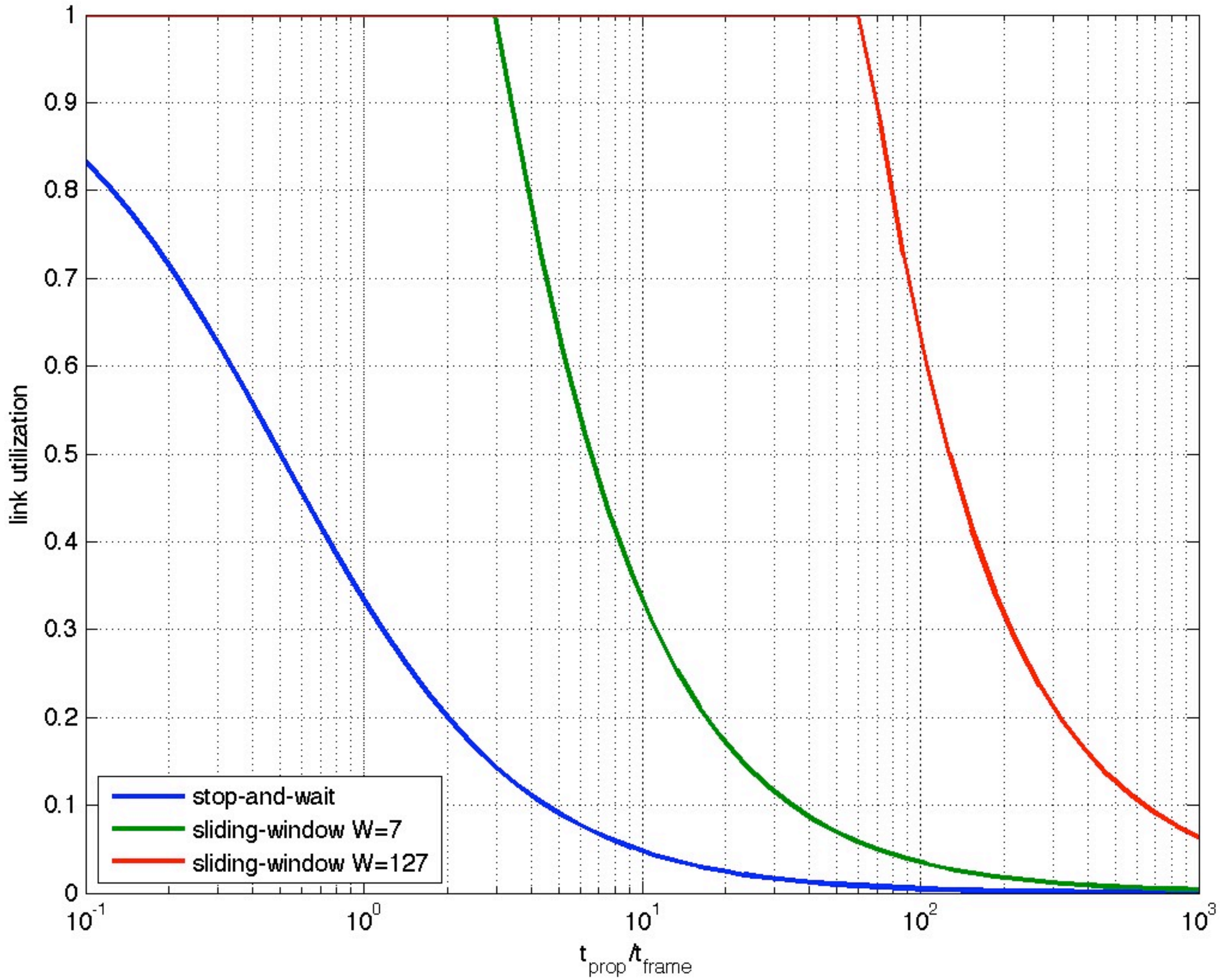
Sliding Window Link Utilization

- Same assumptions as before: processing time (t_{proc}) and ACK/RR transmission time (t_{ack}) are much smaller than frame transmission time (t_{frame}) and propagation time (t_{prop})
- Also assume full-duplex link (RRs can be sent while frames are also being sent)

$$U \approx \begin{cases} 1 & W \geq 2a+1 \\ \frac{W}{2a+1} & W < 2a+1 \end{cases}$$

where a= t_{prop}/t_{frame}

Sliding Window Link Utilization



Summary of Flow Control

- Primary purpose: avoid buffer overflow at receiver
- How? Receiver transmits acknowledgements to sender
  - Permits sender to transmit more packets or tells transmitter to stop
- Stop-and-wait flow control
  - Simple but low link utilization
- Sliding-window flow control
  - Sequence numbering and occasional ACK/RRs used to improve link utilization
  - More complicated to implement
  - W=1 sliding-window is the same as stop-and-wait
  - Higher values of W typically achieve better link utilization

Error Control

- We already know all about error detection and correction
  - What do we do when we have a damaged frame?
  - What do we do if the frame is mysteriously lost?
- "Error control" = protocol to handle damaged and lost frames.
- Common techniques:
  - positive acknowledgment by receiver
    - R: "I got it"
  - retransmission by sender after timeout
    - S: "I didn't hear back from you. Here it is again."
  - negative acknowledgement & retransmission
    - R: "I'm missing a frame"
    - S: "Ok, here it is again"

Automatic Repeat Request (ARQ)

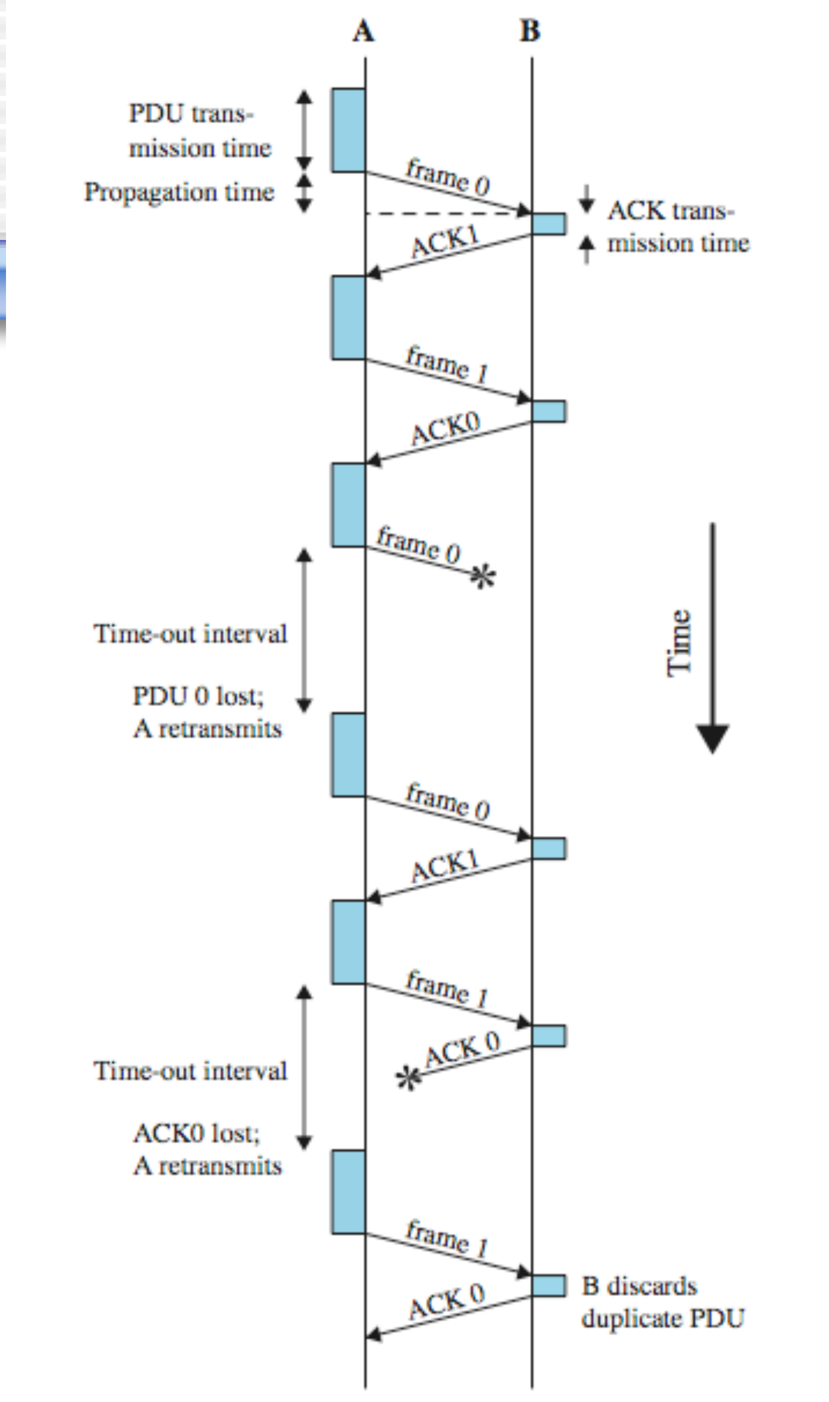
- "ARQ" = collective name for error control protocols including:
  - stop and wait
  - go back N
  - selective reject (selective retransmission)

Stop and Wait

1. Sender transmits a single frame
2. Receiver checks frame for uncorrectable errors
   1. If received frame is ok, receiver sends an ACK
   2. If received frame damaged, receiver discards it and doesn't send an ACK
3. Sender waits for ACK until timeout
   1. If no ACK within timeout, sender retransmits frame
   2. If ACK is damaged or lost, sender retransmits frame (note that receiver will get two copies of the frame in this case)
   3. Need 1-bit frame sequence number and different ACKs:
      - Frame 1 correctly received: Send ACK0
      - Frame 0 correctly received: Send ACK1
4. After successful ACK, repeat

Stop and Wait

- Super simple
- Super inefficient (low link utilization)



Go Back N

Based on sliding-window flow control

1. Receiver has successfully received frame i-1
2. Frame i arrives at receiver
3. Receiver checks frame i for uncorrectable errors
    1. If frame i is ok, receiver sends RRs as usual
    2. If frame i is damaged/lost, receiver discards it and does nothing until either
        - Frame i+1 or any later frame arrives undamaged. The receiver sends a "REJ i" message to tell Sender to retransmit frame i and all subsequent frames.
        - It receives a message from the Sender asking "What is the next frame that you are expecting?" The receiver sends an "RR i" message to tell Sender to send frame i and all subsequent frames.

Bottom line: Sender starts over at the first damaged frame, even if subsequent frames were correctly received

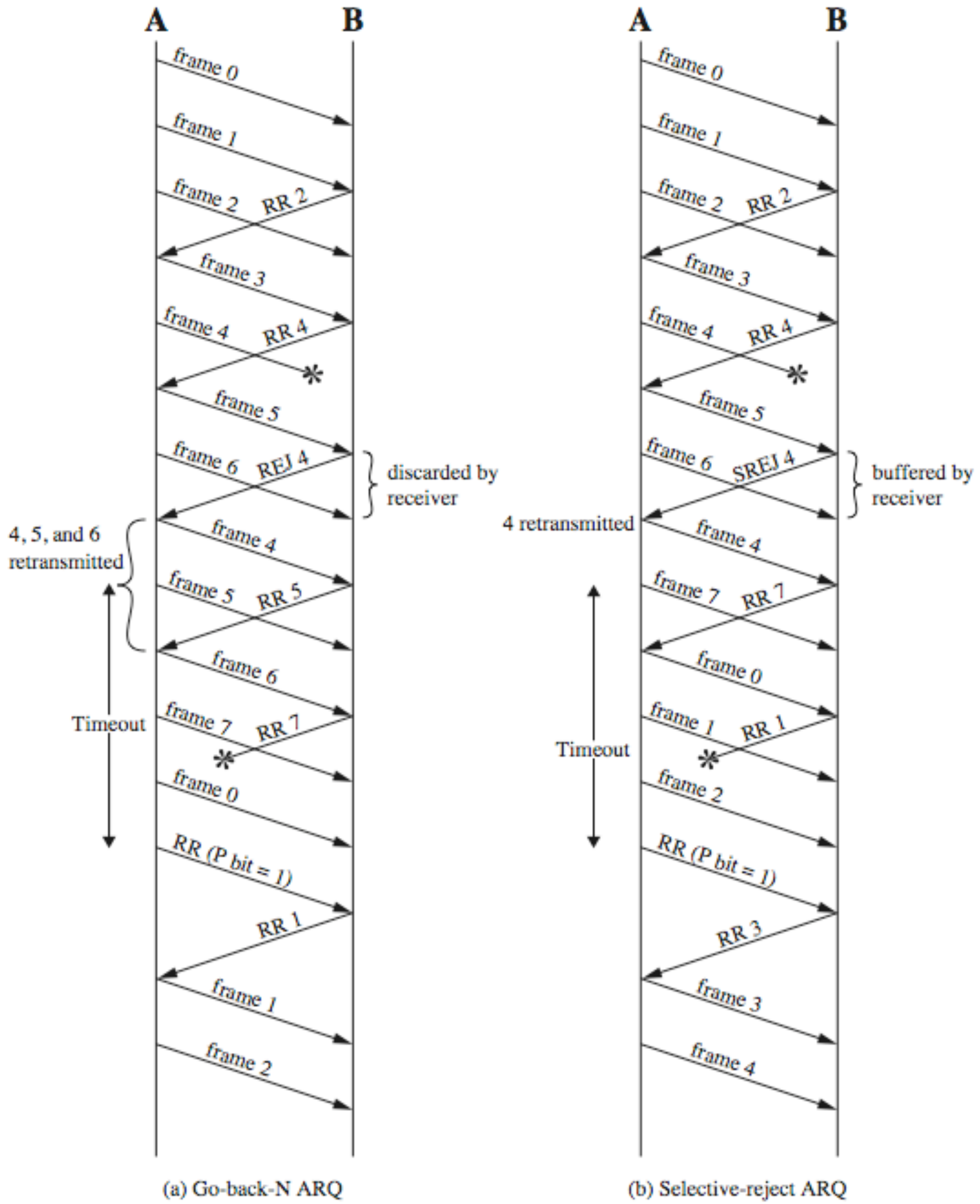
Go Back N – Damaged or Lost Acknowledgements/Rejections

- Damaged or lost RR (acknowledge)
  - Receiver correctly receives frame *i*, sends RR (*i+1*) which is damaged/lost
  - Two things can happen:
    - A later RR (e.g. RR (*i+5*)) may arrive before sender times out on frame *i.* In this case, the lost RR doesn't matter.
    - The sender times out waiting for RR on frame i. It asks "What is the next frame that you are expecting?" and restarts transmission at this frame number.
- Damaged REJ (rejection)
  - This leads to an RR timeout and is handled as discussed above.

Selective Reject

- Similar to "Go Back N" except that only rejected frames are retransmitted
  - SREJ messages are sent from receiver to sender to fill in gaps caused by damaged/lost packets
  - Good packets are never discarded
- Most efficient technique but
  - receiver must maintain a large buffer to insert out-of-order frames
  - more complex logic in transmitter
- Stallings says "Go Back N" is more popular except in scenarios with very large t_{prop} (e.g. satellite link)

"Go Back N" vs "Selective Reject"



Summary of Error Control

- Primary purpose: ensure data integrity at receiver
  - Make sure all frames are correctly received
- Stop-and-wait ARQ
  - Simple but low link utilization
- Go back N ARQ
  - Good tradeoff between complexity and link utilization
- Selective Reject ARQ
  - Highest complexity but also highest link utilization