

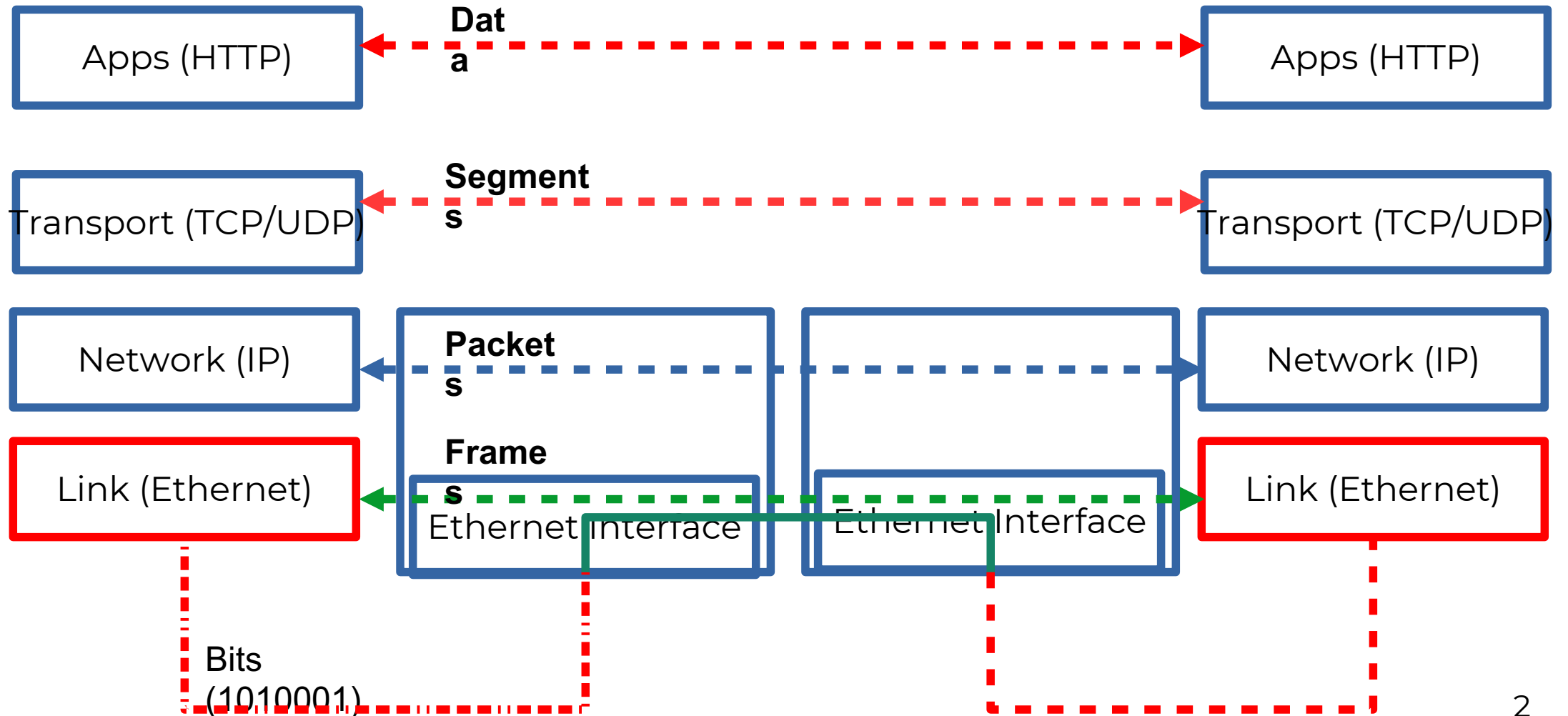
# CSC4200/5200 – COMPUTER NETWORKING

## RELIABLE DELIVERY – PART 1

Instructor: Susmit Shannigrahi

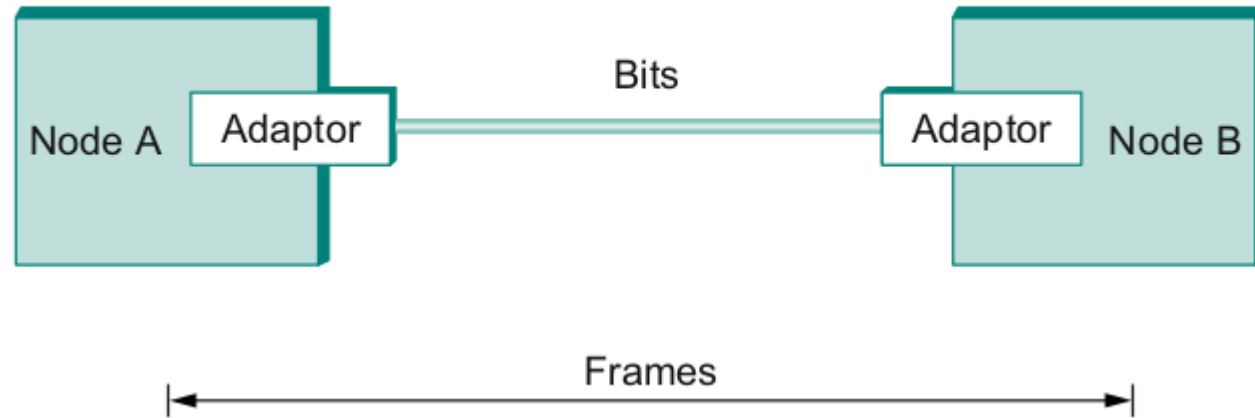
[sshannigrahi@tntech.edu](mailto:sshannigrahi@tntech.edu)





# Frames – bag of bits

---



- Sending side – encapsulation, add error check bits, flow control
- Receiving side – extract frames, check for error, flow control

# Two (2.5) Steps to a Link

---

- Create a physical medium between nodes (**wire, fiber, air!**)
- Make it carry bits
  - **Encoding** bits so that the other end understands (**encoding**)
  - Create bag of bits to create messages (**framing**)
  - Detect errors in frames (**error detection**)
  - Deal with lost frames (**reliable delivery**)
  - Create shared access to link, e.g, WiFi (**media access**)

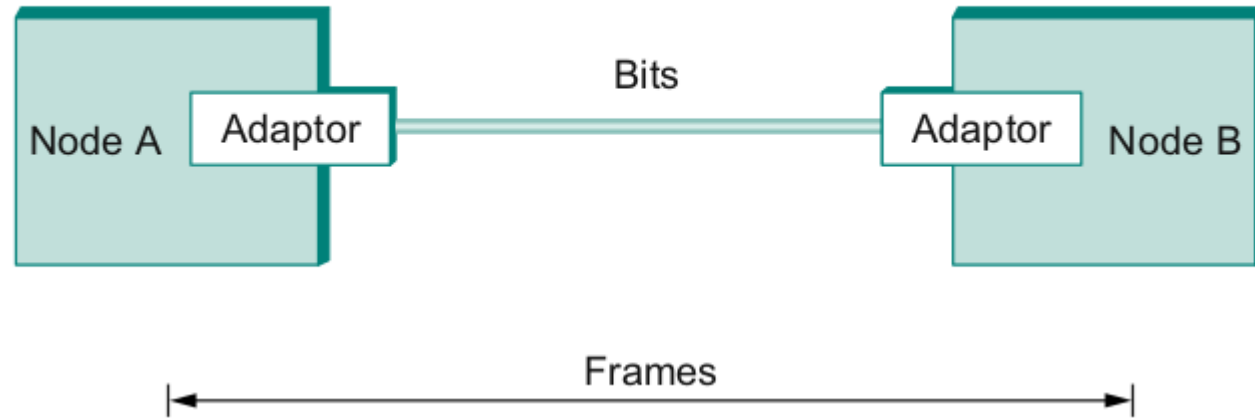
# Reliable Delivery

---

- Frames might get lost
  - Too many bits lost
  - Clock did not sync properly
  - Error detected but the report got lost
- Can we build links that does not have errors?
  - Not possible
- How about all those error correction stuff we learned?
  - Can we add them to frames?
  - We could, but think of the overhead
  - What happens when the entire frame is lost?

# Frames – bag of bits

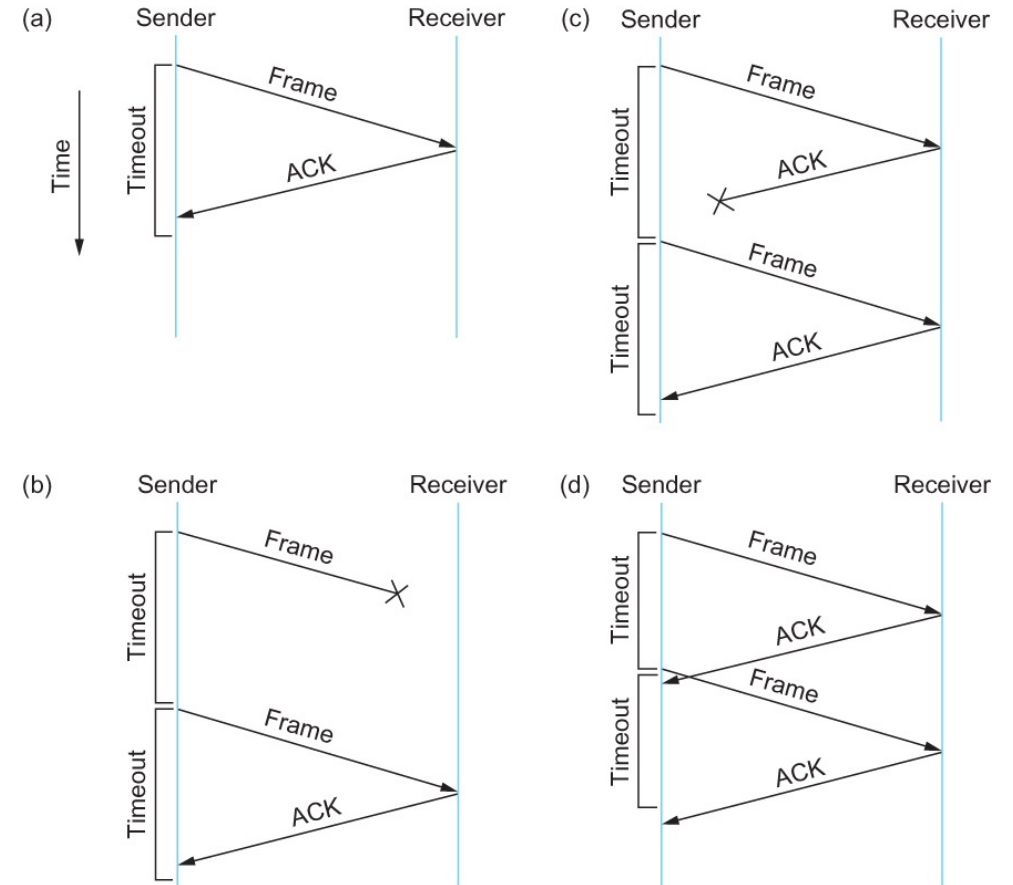
---



- Sending side – encapsulation, add error check bits, flow control
- Receiving side – extract frames, check for error, flow control

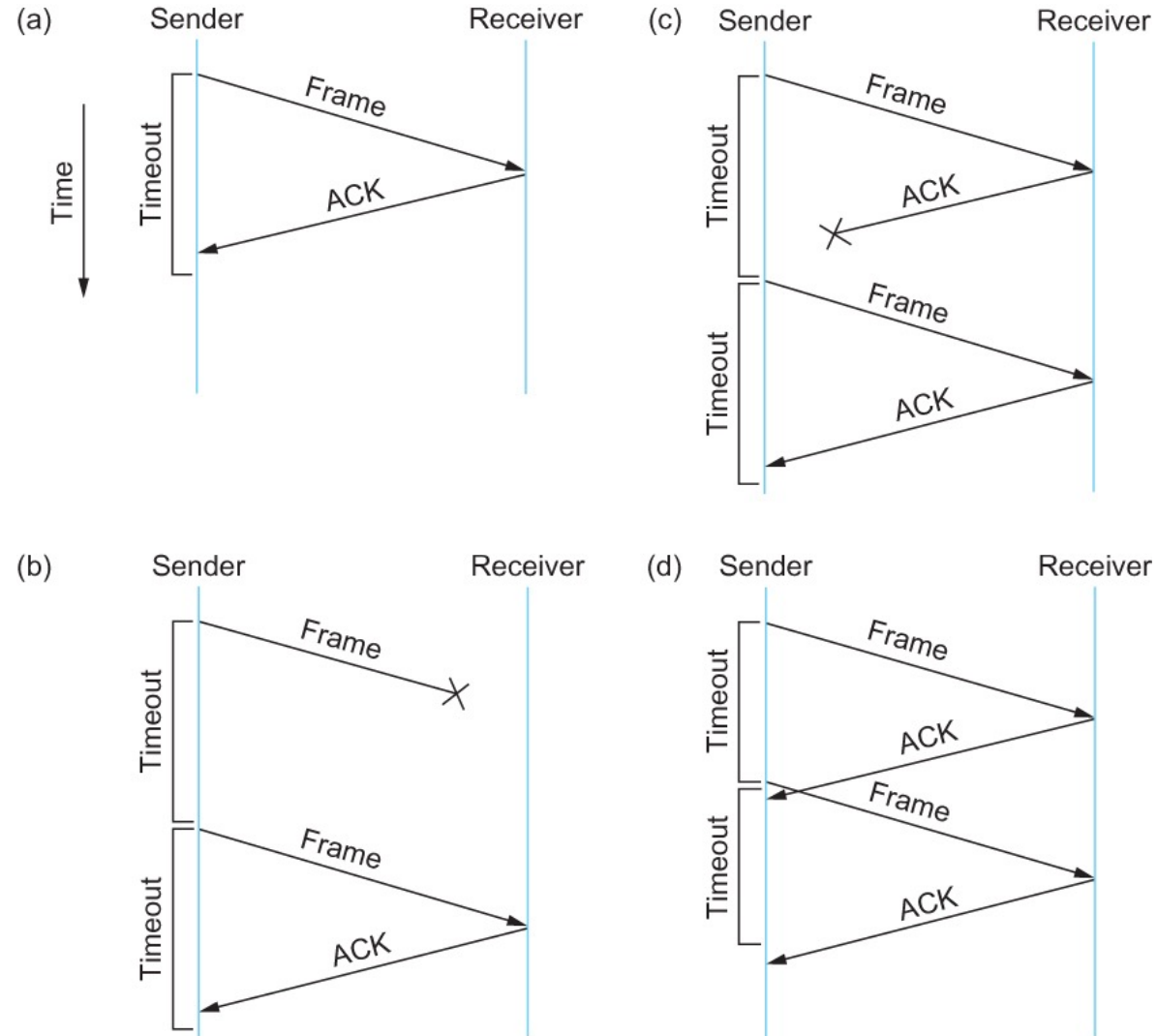
# Stop and Wait

- Sender sends a frame, sets a timeout (e.g., 1 sec)
- Receiver receives the frame, sends an ACK
- Sender
  - sends the next frame on ACK
  - retransmits the same frame if timeout happens



- **Spot the bugs in the protocol**

# Stop and Wait – Bugs (C and D)



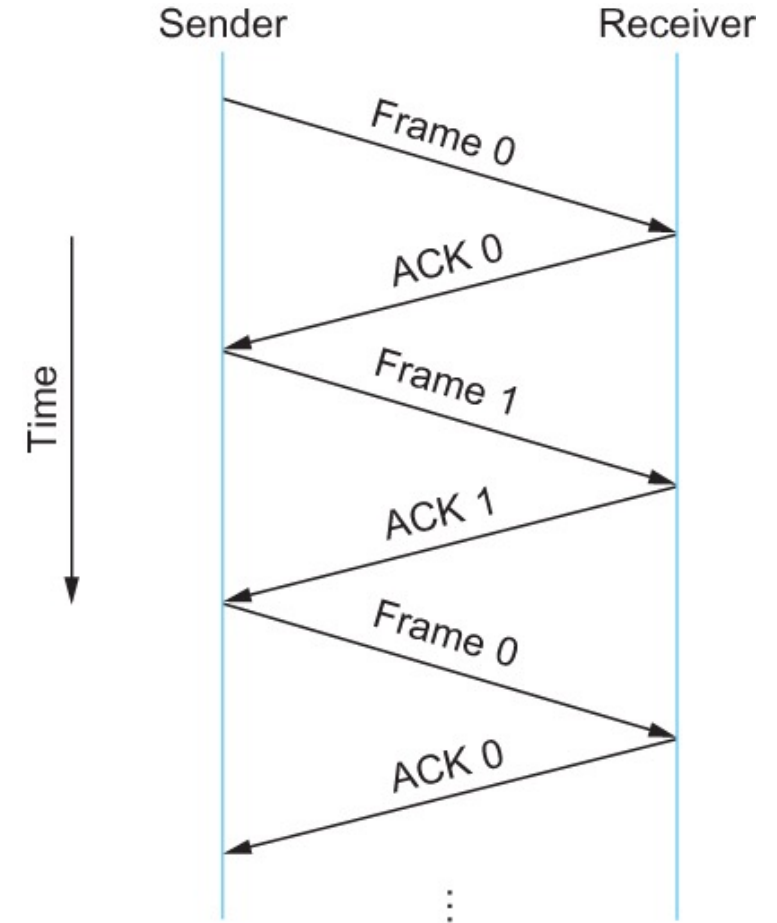


# Stop and Wait – How to fix the bug?

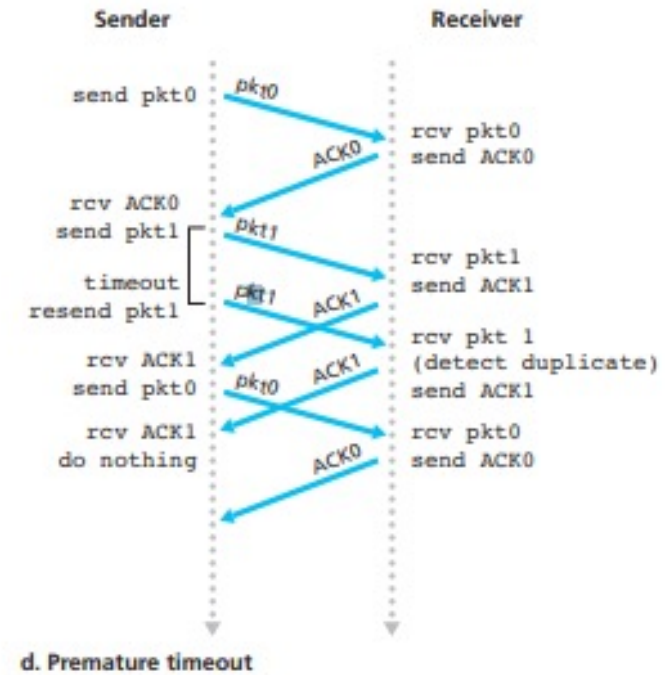
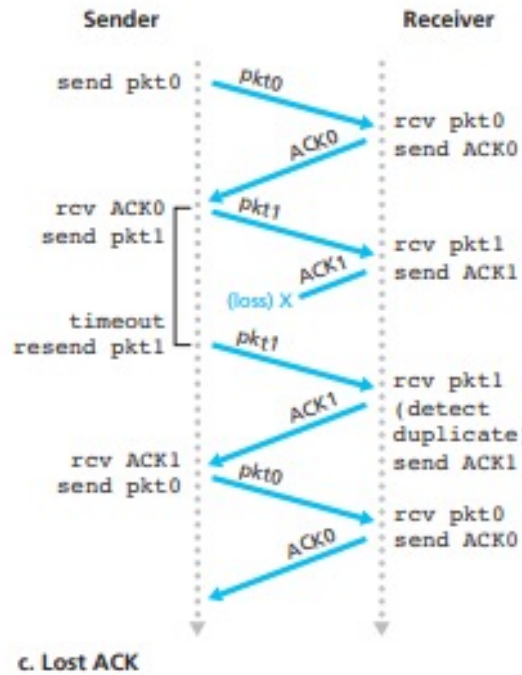
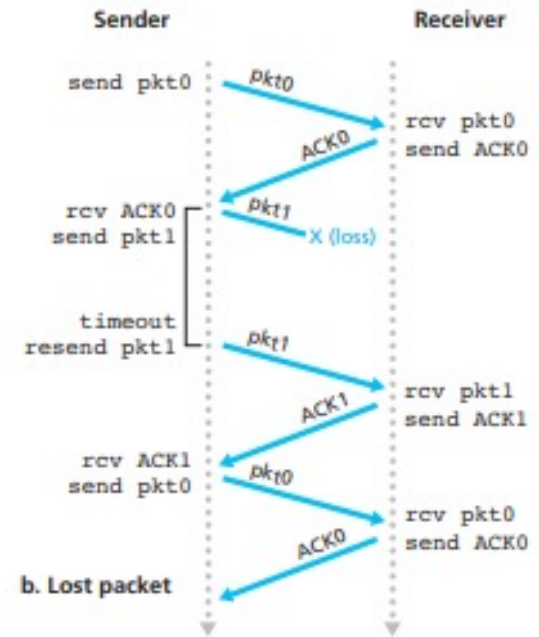
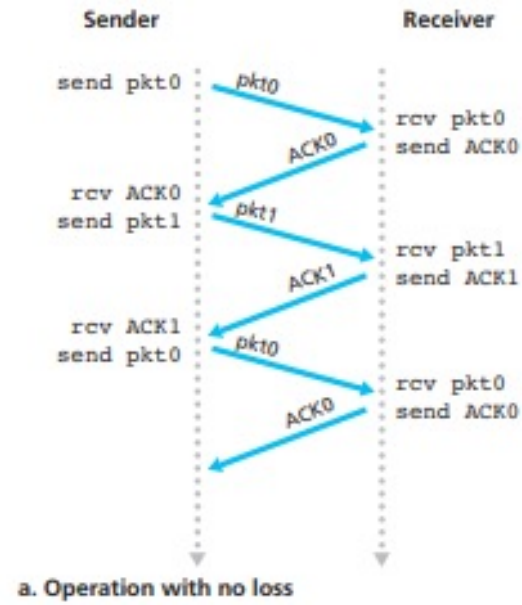
Hint: Uniquely identify each packet

1 bit sequence number  
- 0 or 1

Alternate between 0 and  
1

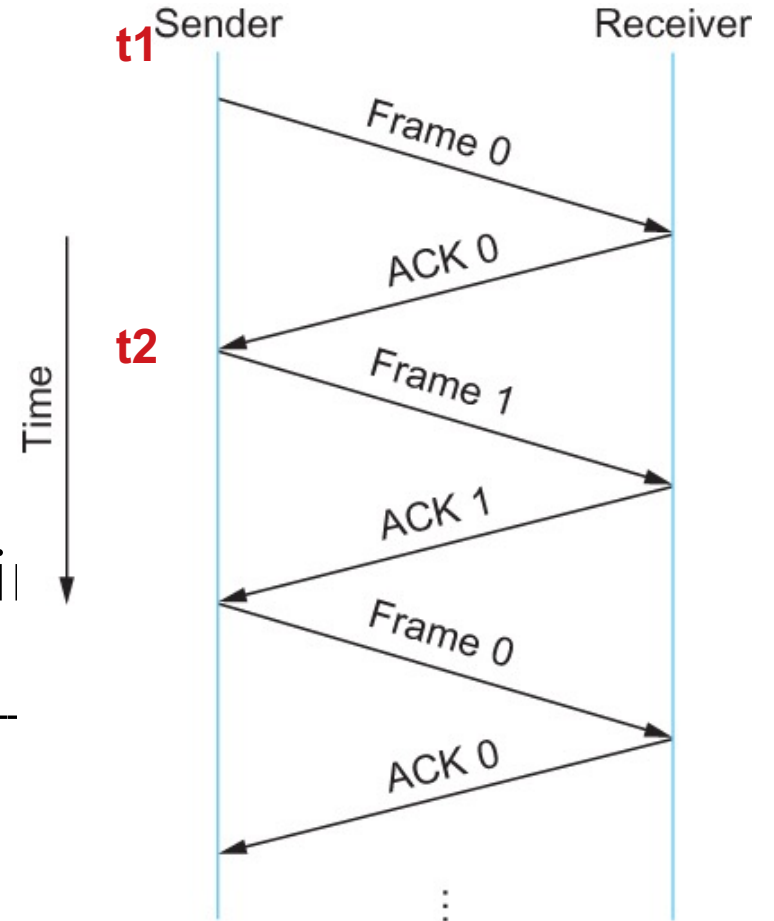


# Stop and Wait v2



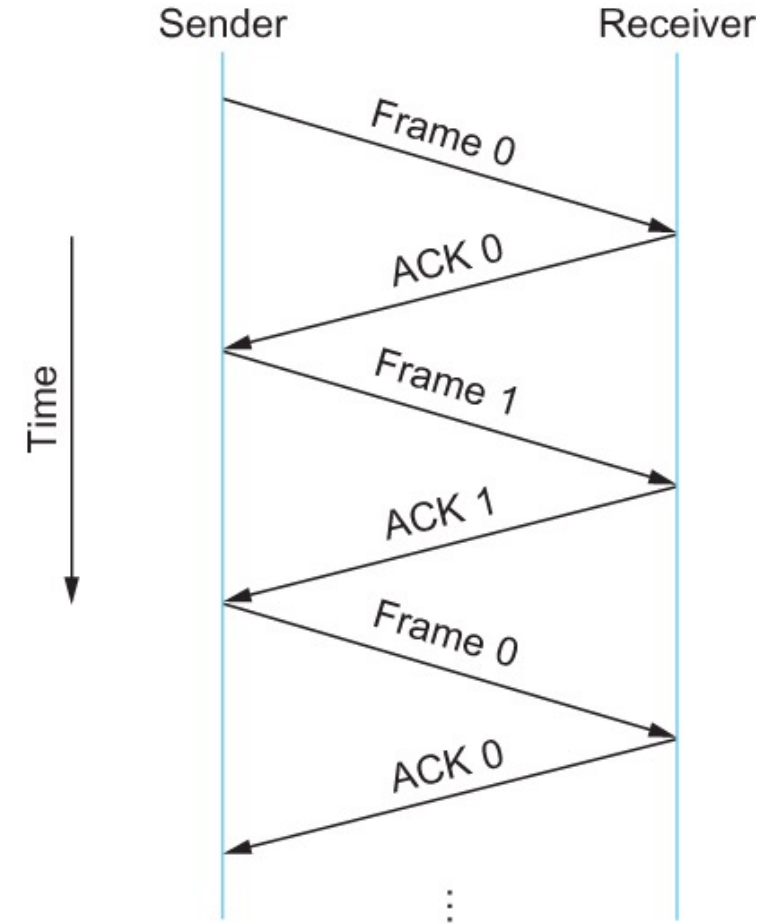
# Stop and Wait - V2 Problems

- Sender sets a timeout to wait for an ACK
  - Too small – retransmissions
  - Too large – long wait if frames are lost
- Solution:
  - Keep a running average of Round Trip Time
  - $\text{EstimatedRTT} = (1 - \alpha) \cdot \text{EstimatedRTT} + \alpha \cdot \text{SampleRTT}$
  - $\text{Timeout} = 2 \cdot \text{EstimatedRTT}$
  - Value of  $\alpha = 0.125$
  - Where does  $\alpha$  come from? RFC 6928 (for now)



# Stop and Wait – How to fix the bug?

Hint: Uniquely identify each packet



# Stop and Wait – How does it perform?

- Bandwidth (R) = 1Gbps
- Packet size (L) = 1000 bytes
- RTT = 30ms
- $T_{\text{trans}} = L/R = 8000\text{bits}/10^9\text{bits/sec} = 8\text{microsecond}$
- $T_{\text{prop}} = 15\text{ms}$
- Total Delay = 15.008 ms

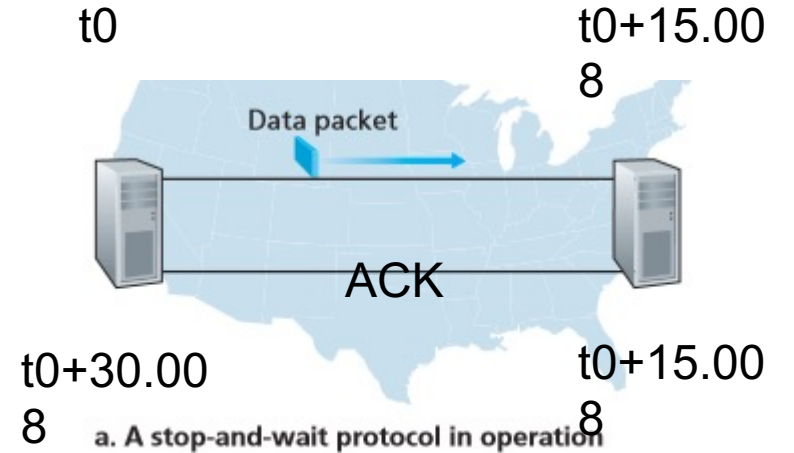


a. A stop-and-wait protocol in operation

Kurose/Ros  
s

# Stop and Wait – How does it perform?

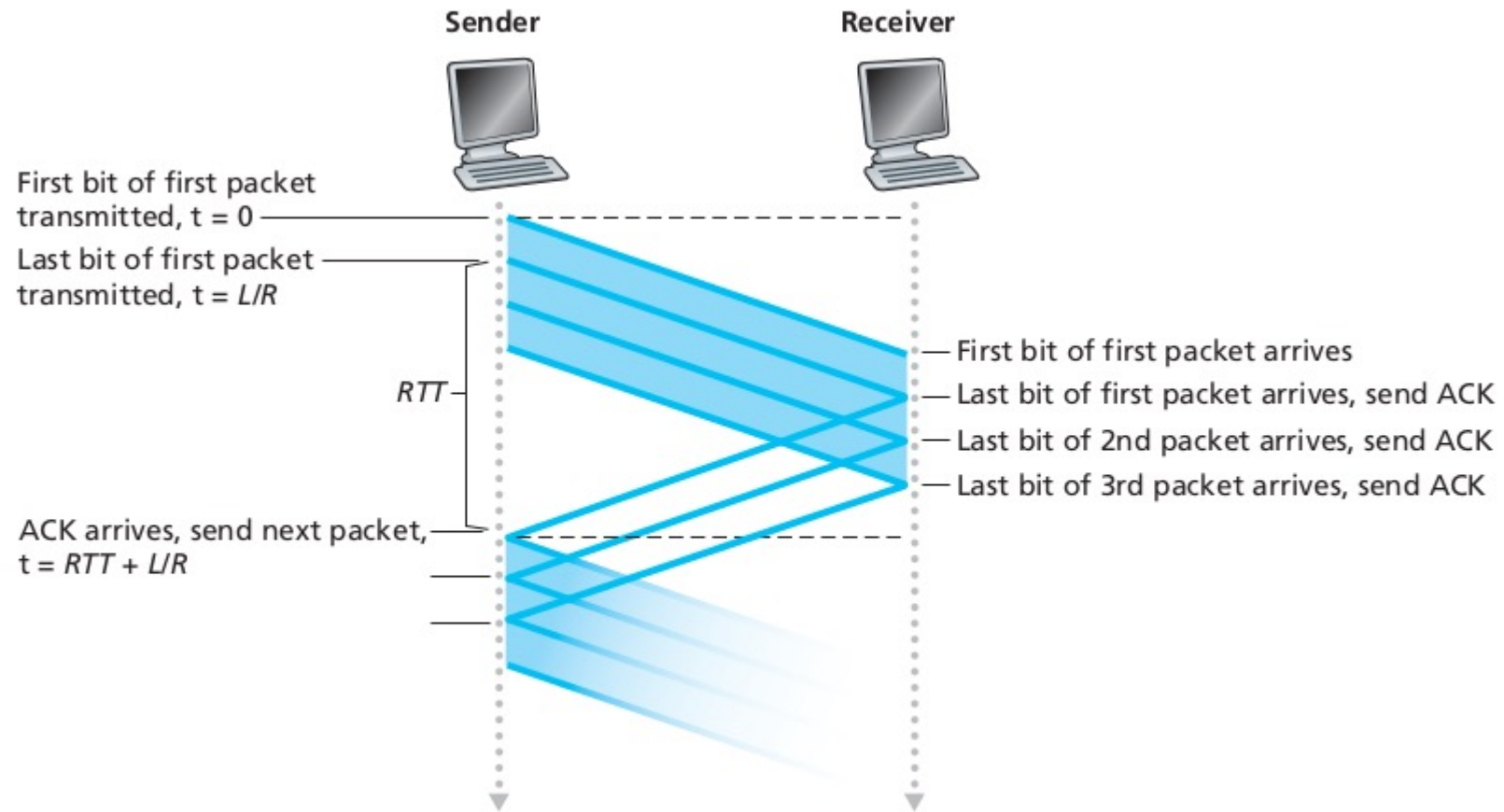
- Sender transmits for only 0.008 ms in 30.008ms
- Utilization =  $0.008/30.008 = 0.00027$
- One bit at a time
- Worse when loss happens!



Kurose/Ros  
S

# Sliding window to the rescue!

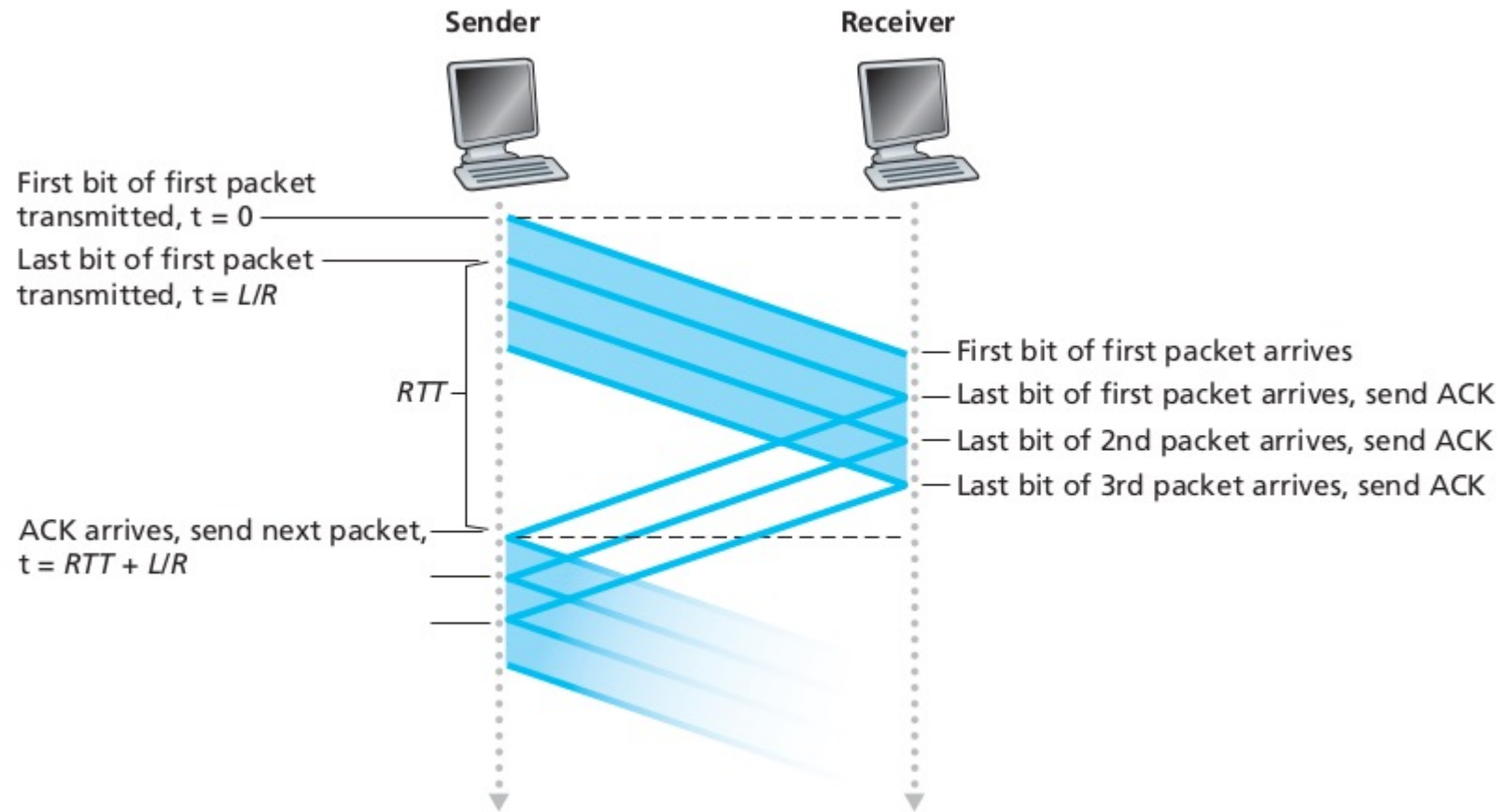
Utilization =  $0.008 * 3 / 30.008 = 0.00079$  (3 times increase)



b. Pipelined operation

# Sliding window to the rescue!

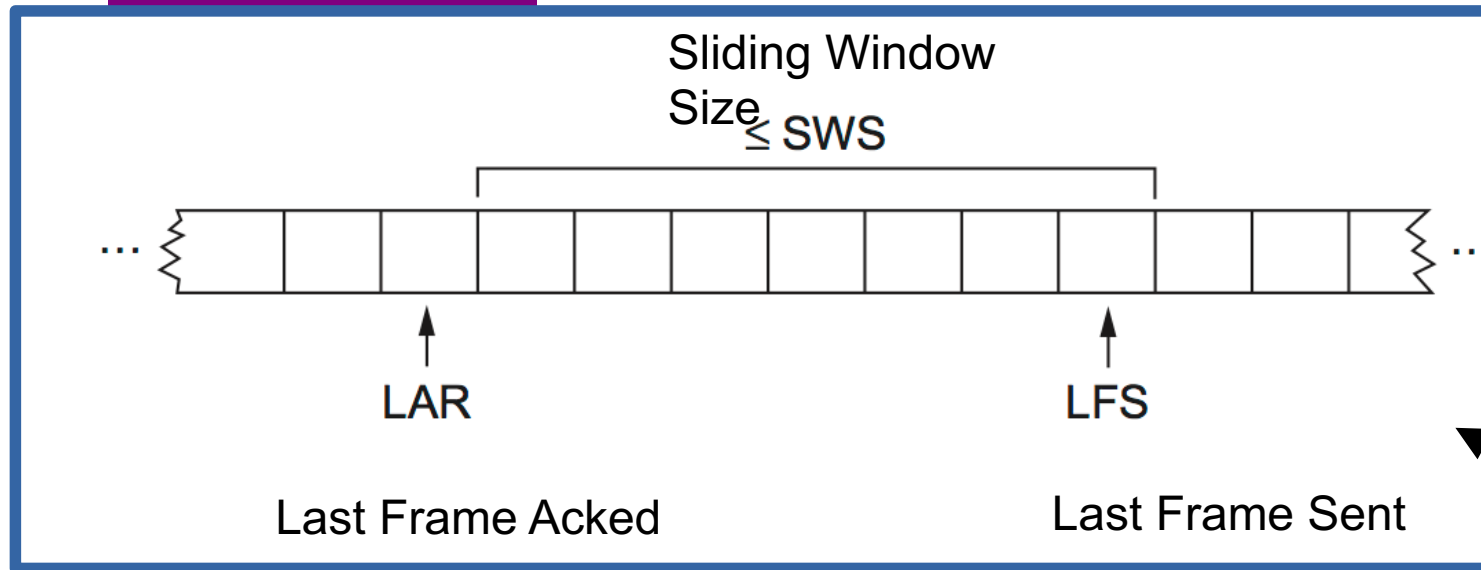
Utilization =  $0.008 * 3 / 30.008 = 0.00079$  (3 times increase)



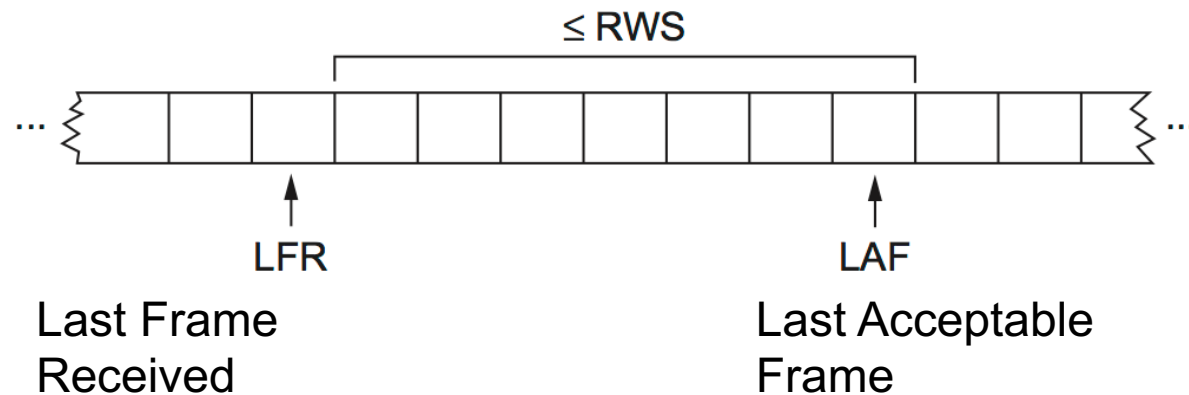
b. Pipelined operation



# Sliding window – How does this work?

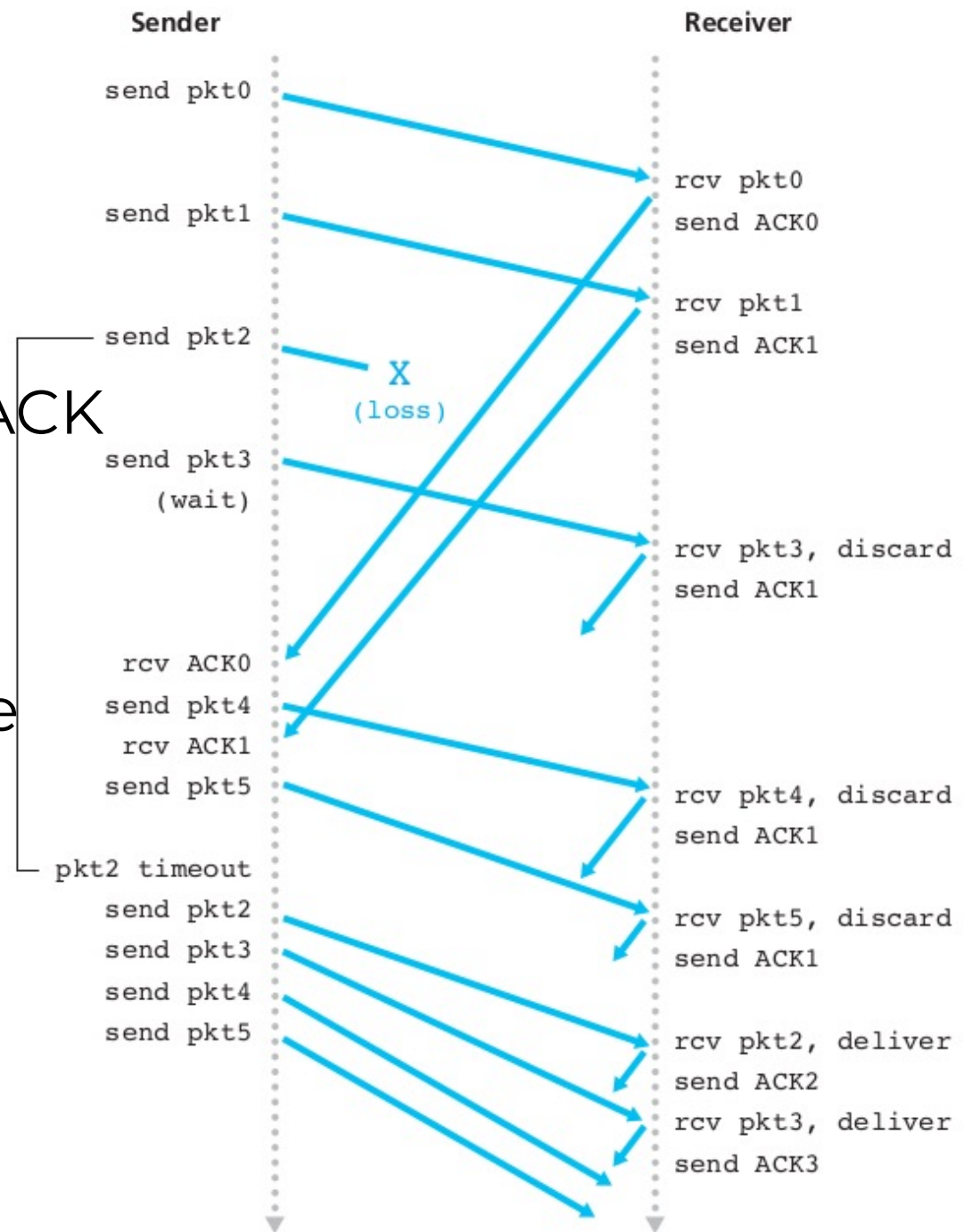


Outstanding requests  $\rightarrow$  SWS  
 $LFS - LAR \leq SWS$



# Sliding window - Go-Back-N

- Can transmit N bits before ACK
- See the problem?
- Can not move forward until all previous packets are acknowledged



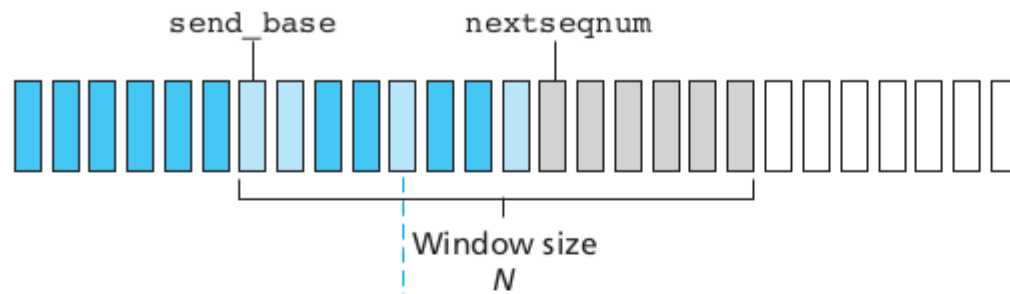
# Sliding Window - Selective Repeat

---

[http://www.exa.unicen.edu.ar/catedras/omdat1/material/Filminas3\\_Practico3.swf](http://www.exa.unicen.edu.ar/catedras/omdat1/material/Filminas3_Practico3.swf)

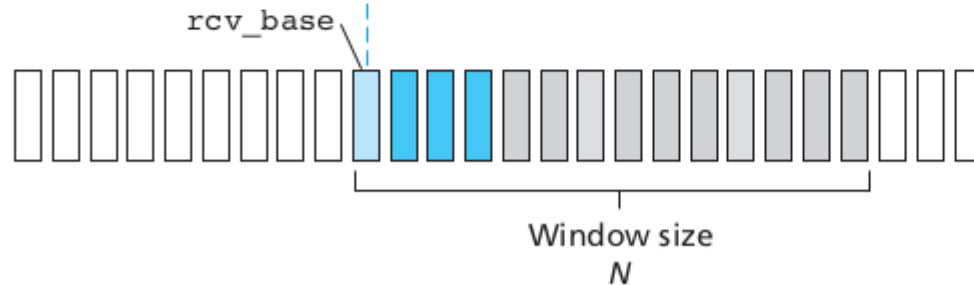
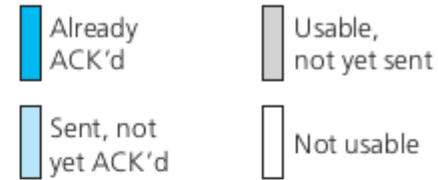
- Receiver:
  - Individually acks all packets
  - Buffers packets as necessary
  - Buffer packets until lost packets are received
- Sender:
  - Resend packets (**only**) for which ACK not received
  - Timer for each unACKed packet
  - Can send only n packets

# Sliding window - Selective Repeat



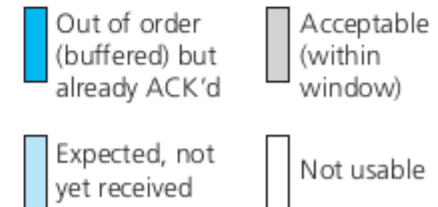
a. Sender view of sequence numbers

Key:

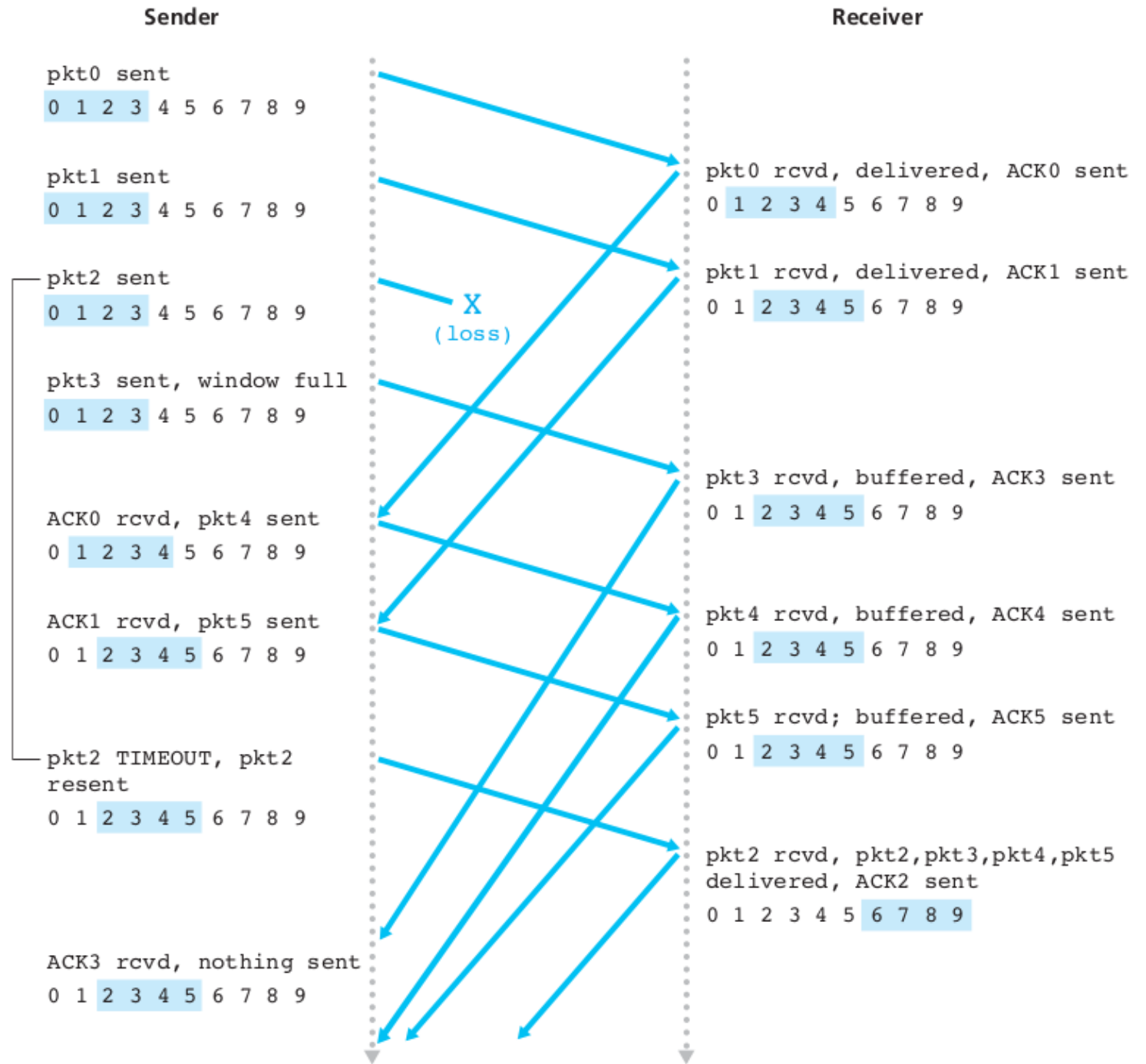


b. Receiver view of sequence numbers

Key:



# Sliding window - Selective Repeat



# Sliding window - Selective Repeat - LOSS

---

- **Sender:**

- Data received, if next to-be-sent-packet's seq # within window, send. Else, buffer or return to application.
- Timeout: Each packet has its own timer. resend the packet
- ACK received: Mark received, Advance window to next unacked seq # if ack for send\_base

- **Receiver**, packet (n)

- Sequence between  $\text{recv\_base}$ ,  $\text{recv\_base} + N - 1$ , send ack (n)
- Out of order: buffer
- In-order or closes gap – deliver to application
- Packet within  $\langle \text{recv\_base} - N, \text{recv\_base} - 1 \rangle$ , ACK(n)
- Otherwise: Ignore

# Issues with Sliding Window Protocol

---

- When timeout occurs, the amount of data in transit decreases
  - Since the sender is unable to advance its window
- When the packet loss occurs, this scheme is no longer keeping the pipe full
  - The longer it takes to notice that a packet loss has occurred, the more severe the problem becomes
- How to improve this
  - Negative Acknowledgement (NAK)
  - Additional Acknowledgement
  - Selective Acknowledgement (SAK)

# Next Steps

---

- Reading Material:
  - Chapter 2.5
  - [https://en.wikipedia.org/wiki/Go-Back-N\\_ARQ](https://en.wikipedia.org/wiki/Go-Back-N_ARQ)
    - 5 minutes