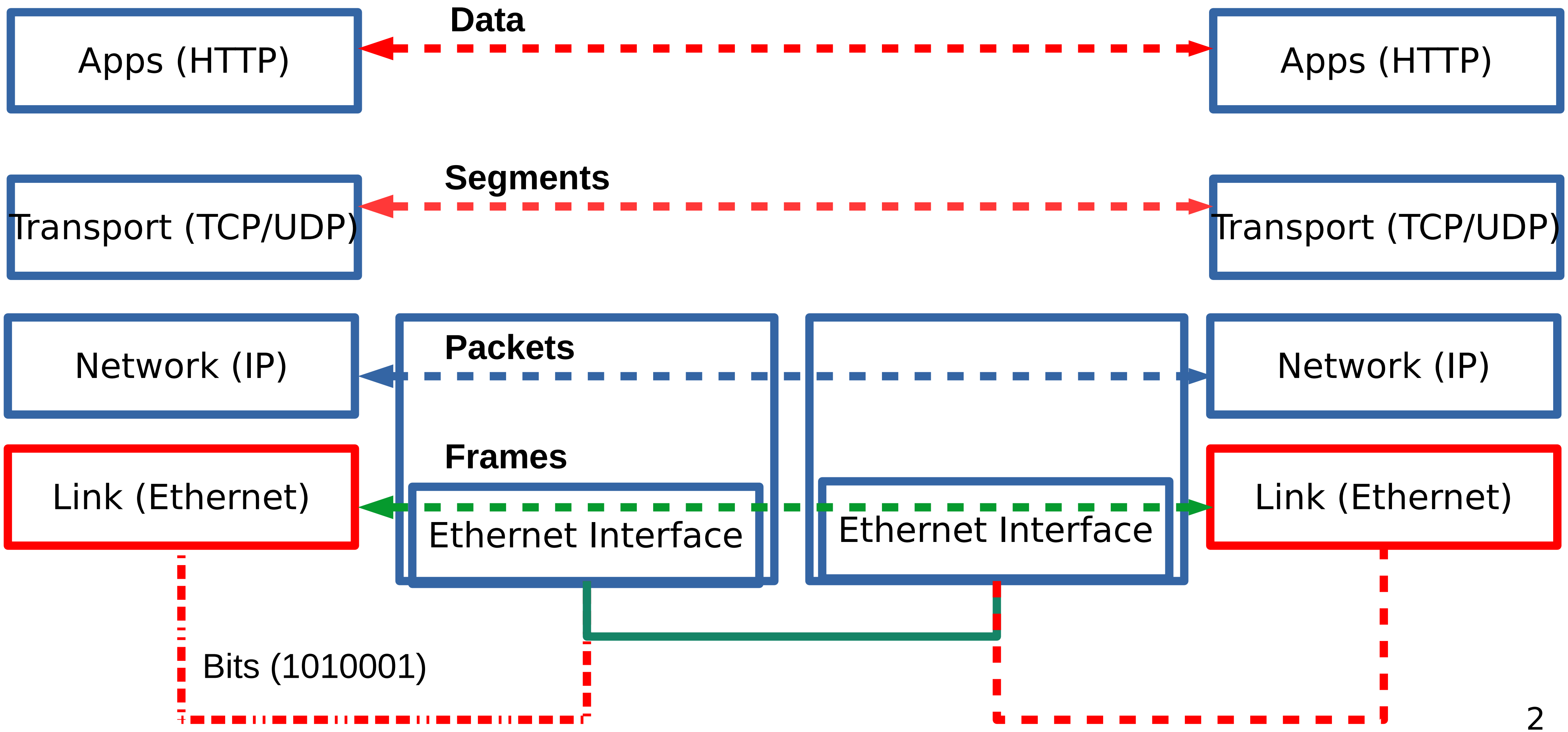
CSC4200/5200 – COMPUTER NETWORKING

RELIABLE DELIVERY – PART 1

Instructor: Susmit Shannigrahi
sshannigrahi@tntech.edu

Tennessee
TECH



Apps (HTTP)

Data

Apps (HTTP)

Transport (TCP/UDP)

Segments

Transport (TCP/UDP)

Network (IP)

Packets

Network (IP)

Link (Ethernet)

Frames

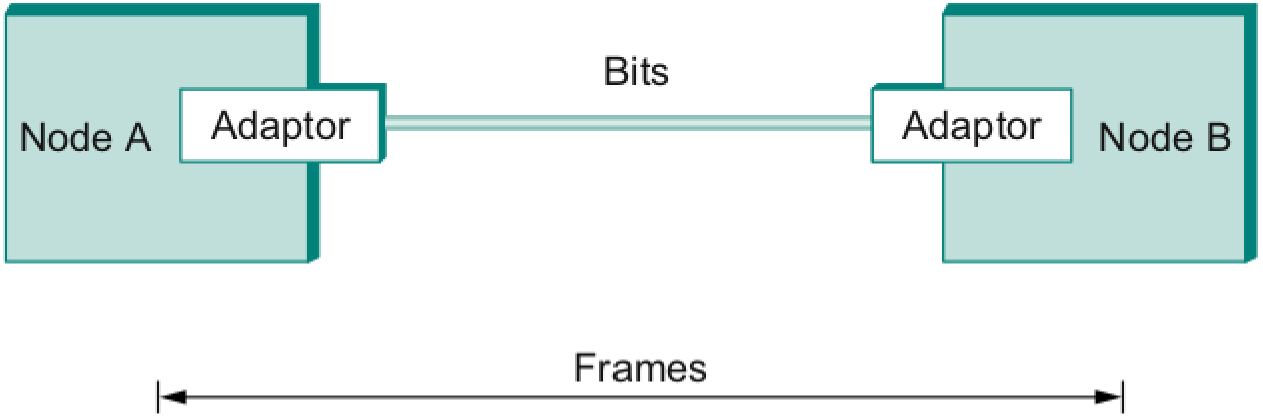
Ethernet Interface

Ethernet Interface

Link (Ethernet)

Bits (1010001)

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)**

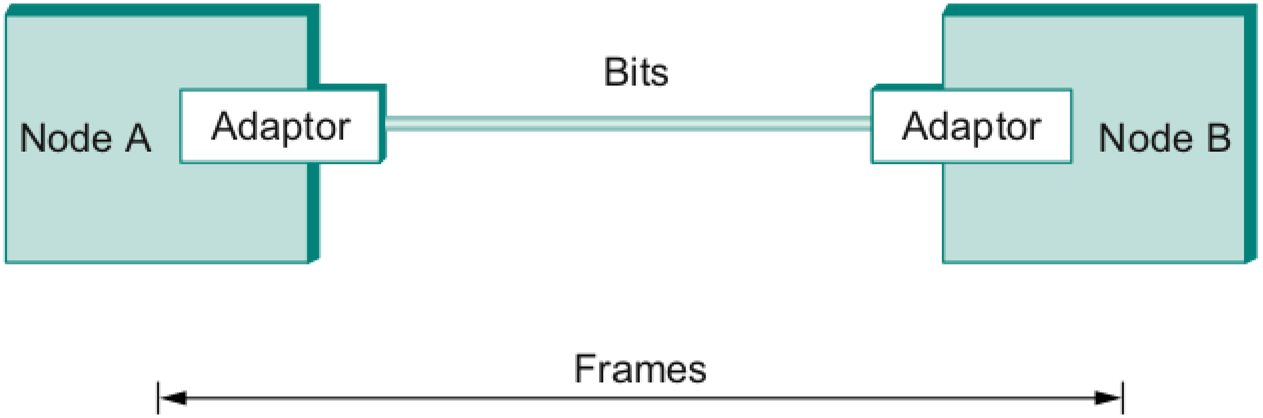
Reliable Delivery

- 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

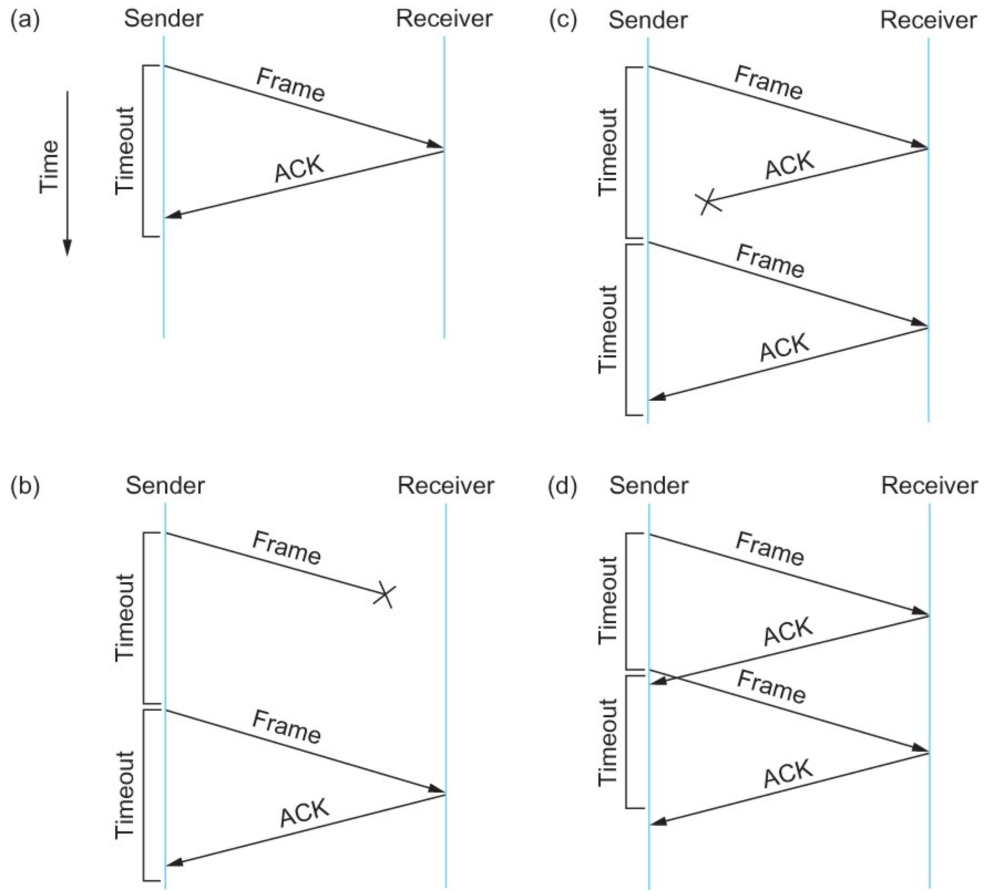
- How about all those error correction stuff we learned?
  - Can we add them to frames?
  - We could, but think of the overhead
  - What happens when the entire frame is lost?

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



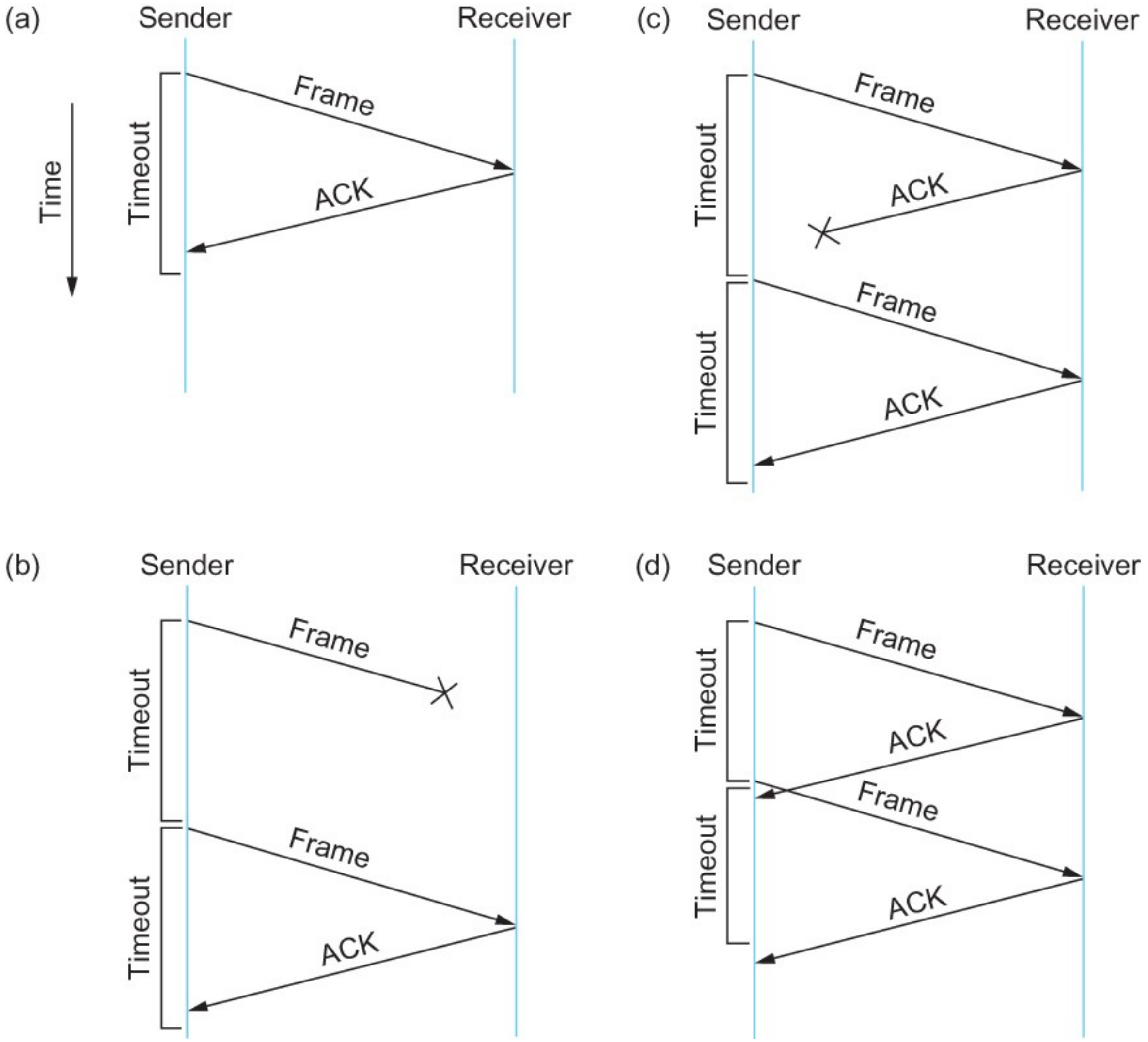
- Sender sends a frame, sets a timeout (e.g., 1 sec)

- 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)

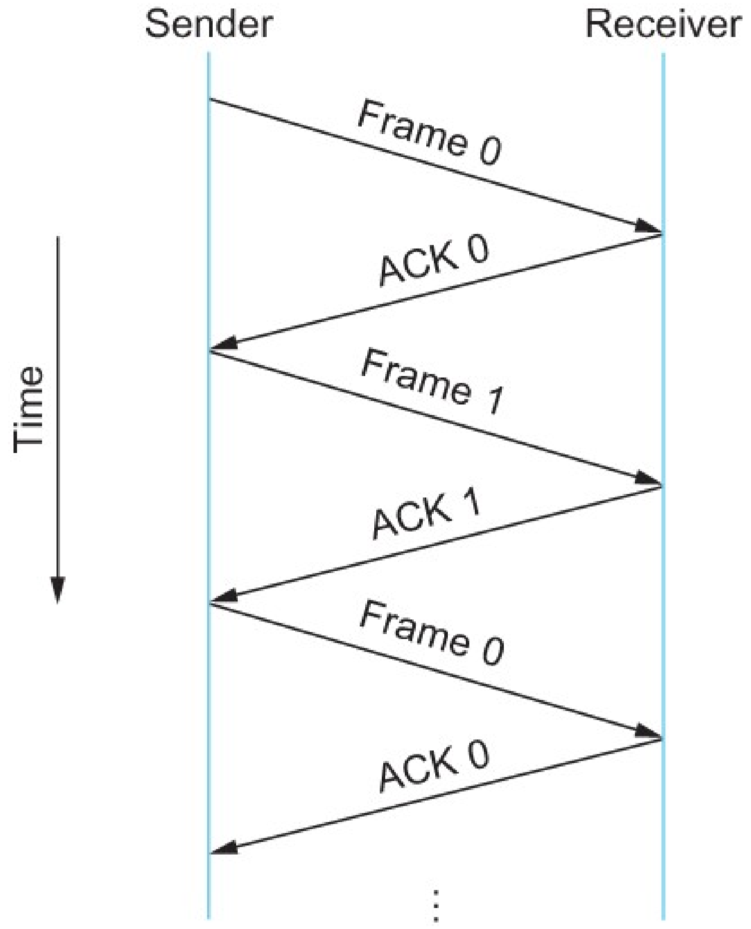


Stop and Wait – How to fix the bug?

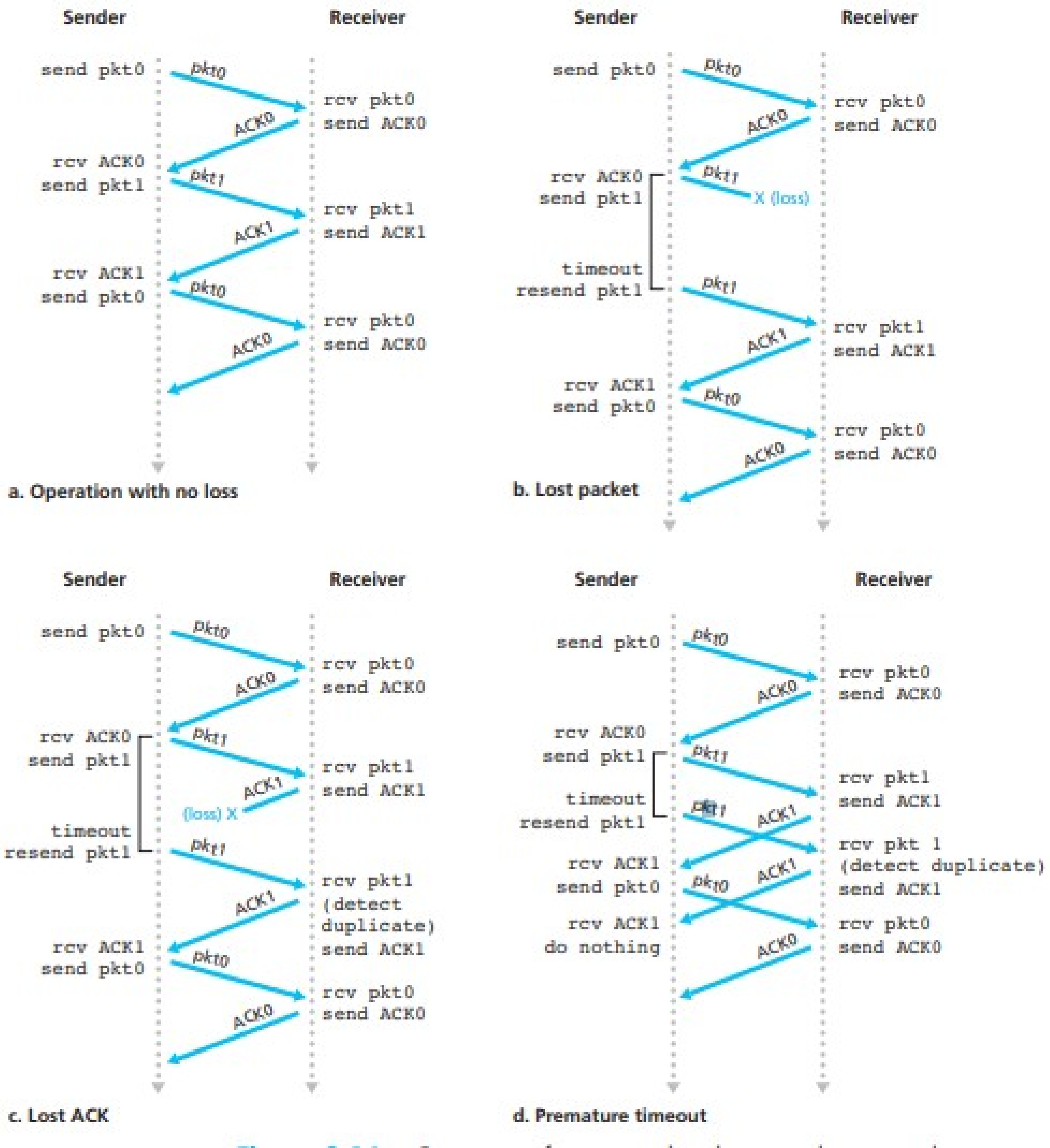
Hint: Uniquely identify each packet

1 bit sequence number - 0 or 1

Alternate between 0 and 1



Stop and Wait v2



a. Operation with no loss

b. Lost packet

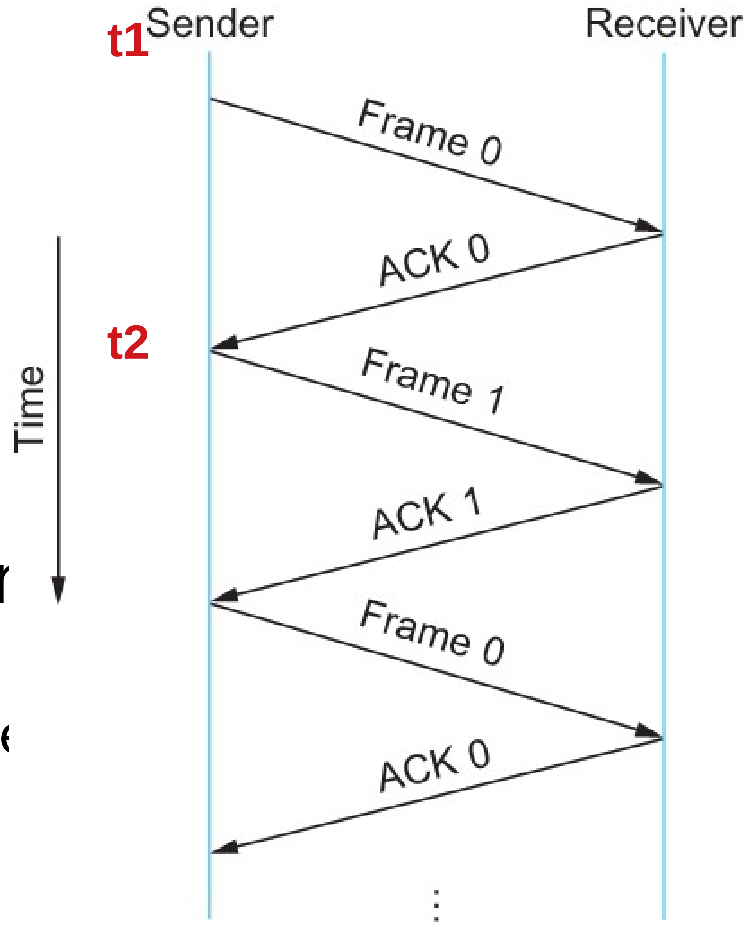
c. Lost ACK

d. Premature timeout

Stop and Wait - V2 Problems

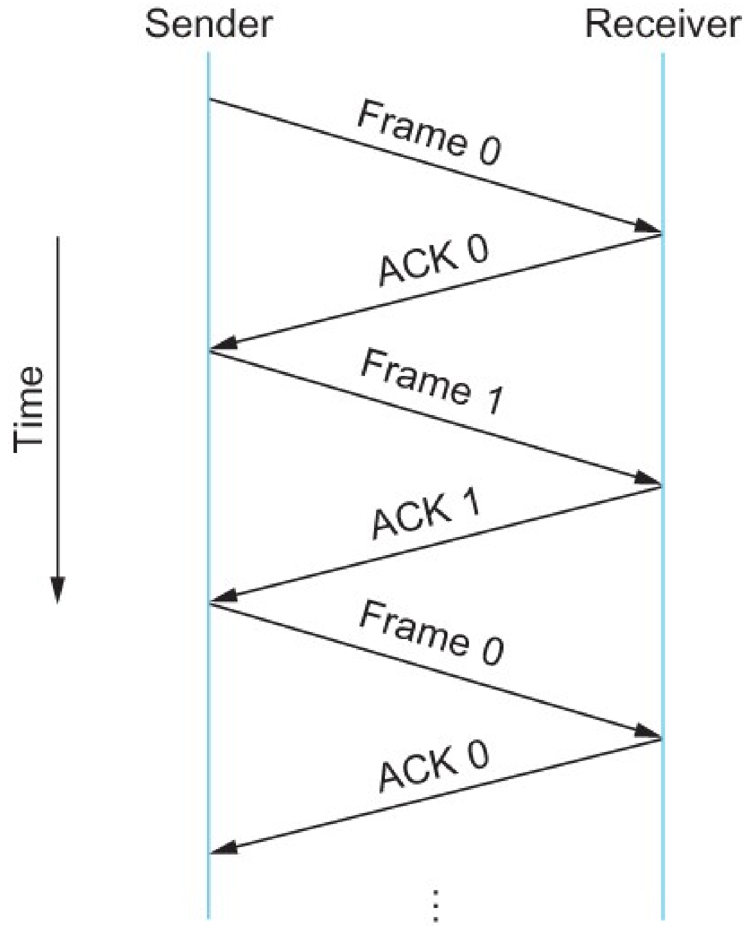
- Sender sets a timeout to wait for an ACK
  - Too small – retransmissions
  - Too large – long wait if frames are lost

- Solution:
  - Keep a running average of Round Trip Tir
  - EstimatedRTT = (1 – α) • EstimatedRTT + α • Sample
  - Timeout = 2*EstimatedRTT
  - Value of α = 0.125
  - Where does α come from? RFC 6928 (for now)



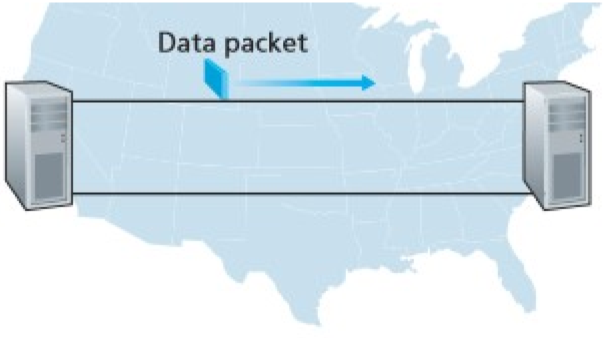
Stop and Wait – How to fix the bug?

Hint: Uniquely identify each packet



Stop and Wait – How does it perform?

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

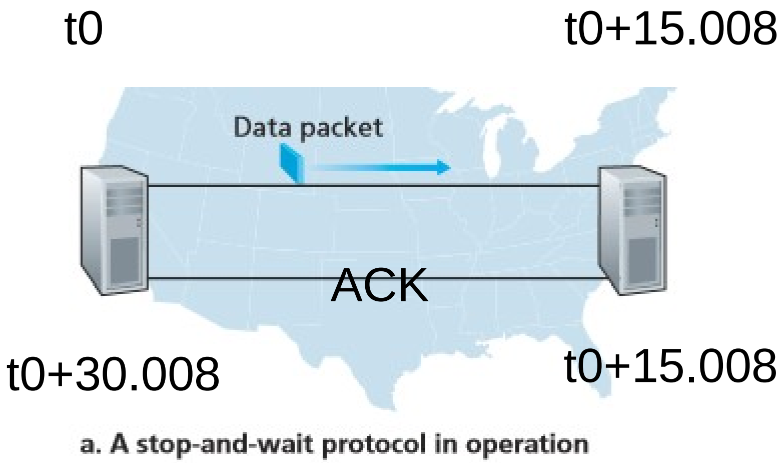


a. A stop-and-wait protocol in operation

Kurose/Ross

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!

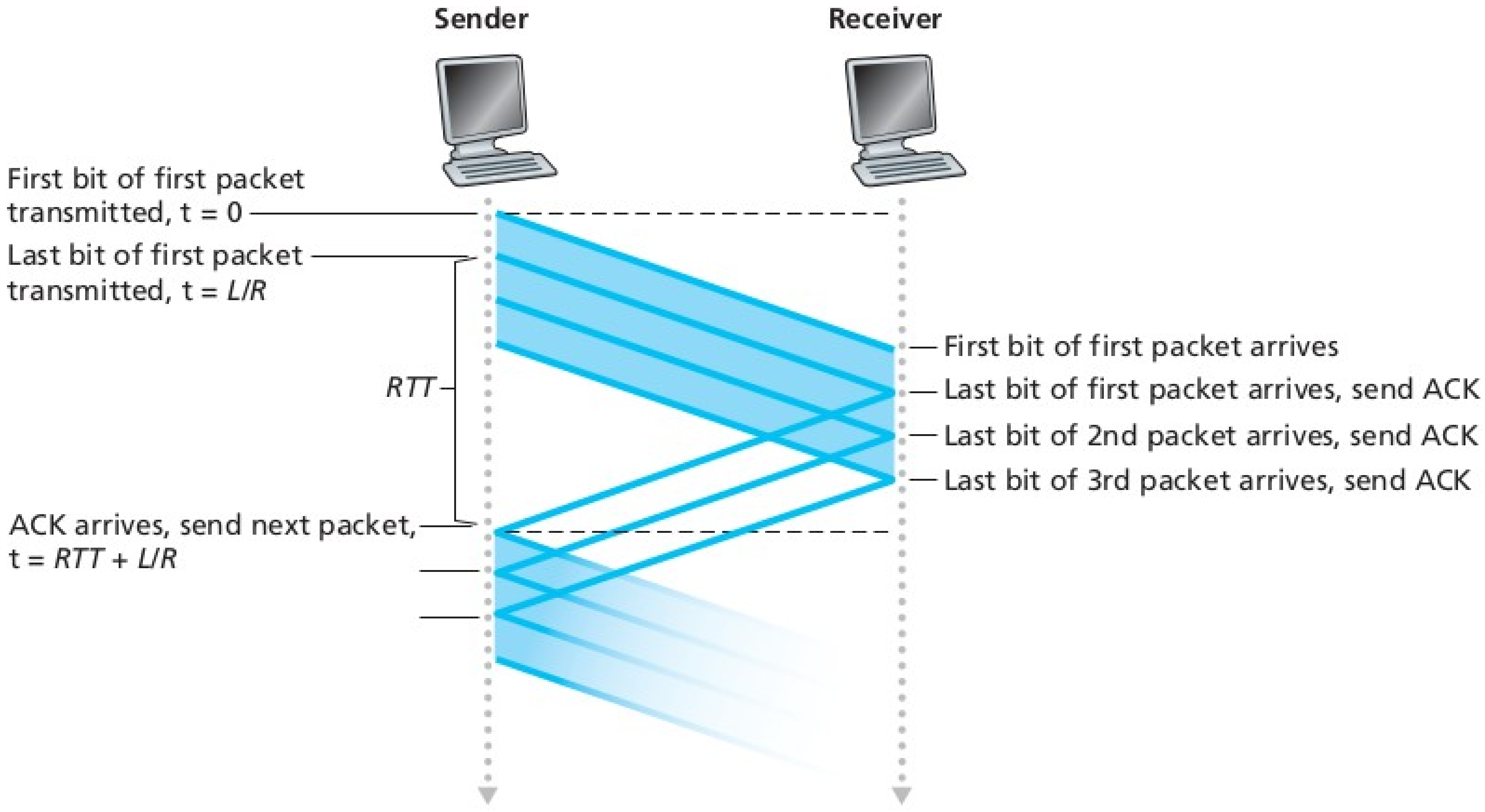


a. A stop-and-wait protocol in operation

Kurose/Ross

Sliding window to the rescue!

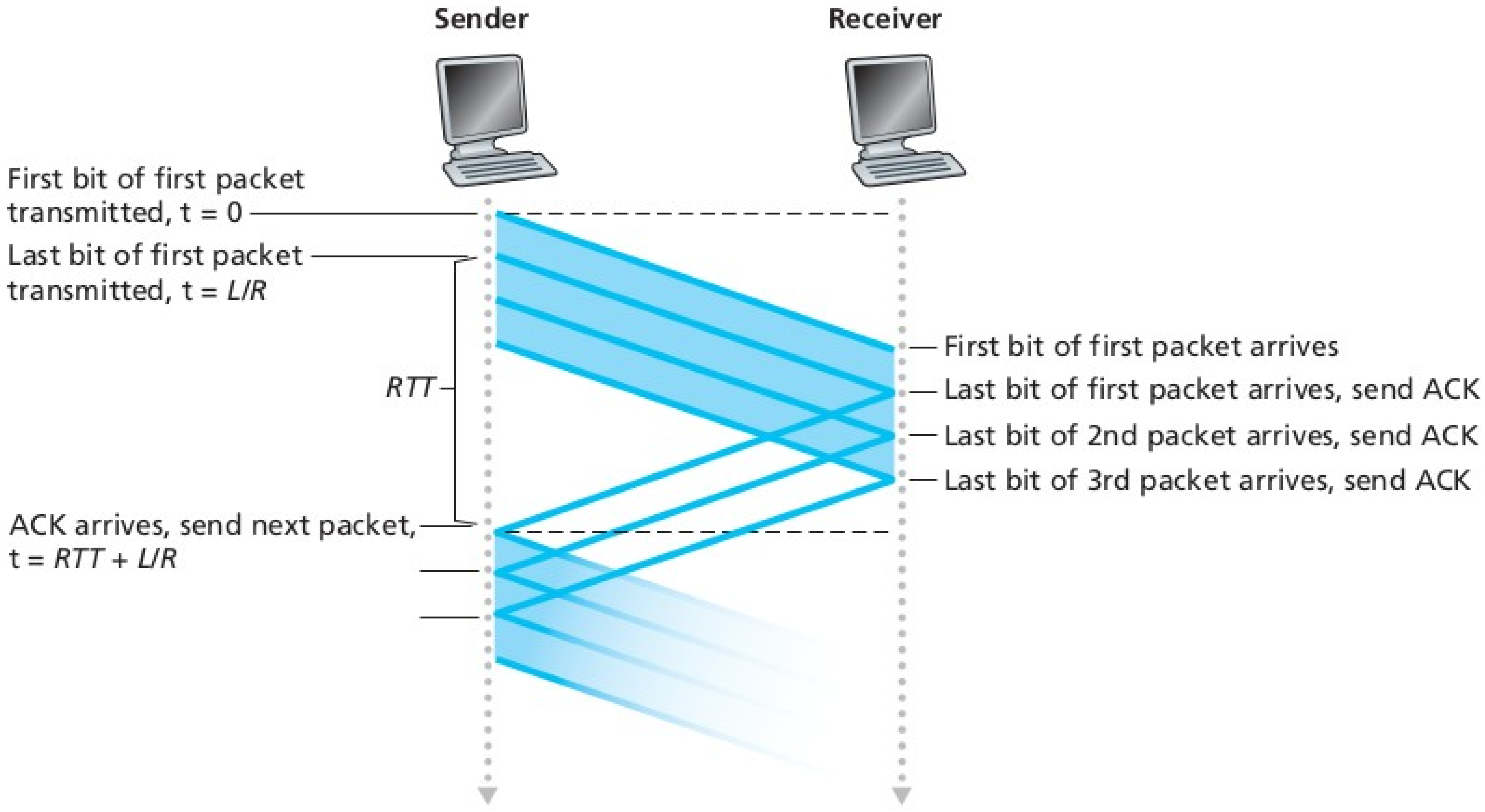
Utilization = 0.008*3/30.008 = 0.00079 (3 times increase)


Sender
Receiver
First bit of first packet transmitted, t = 0
Last bit of first packet transmitted, t = L/R
RTT
ACK arrives, send next packet, t = RTT + L/R
First bit of first packet arrives
Last bit of first packet arrives, send ACK
Last bit of 2nd packet arrives, send ACK
Last bit of 3rd packet arrives, send ACK

b. Pipelined operation

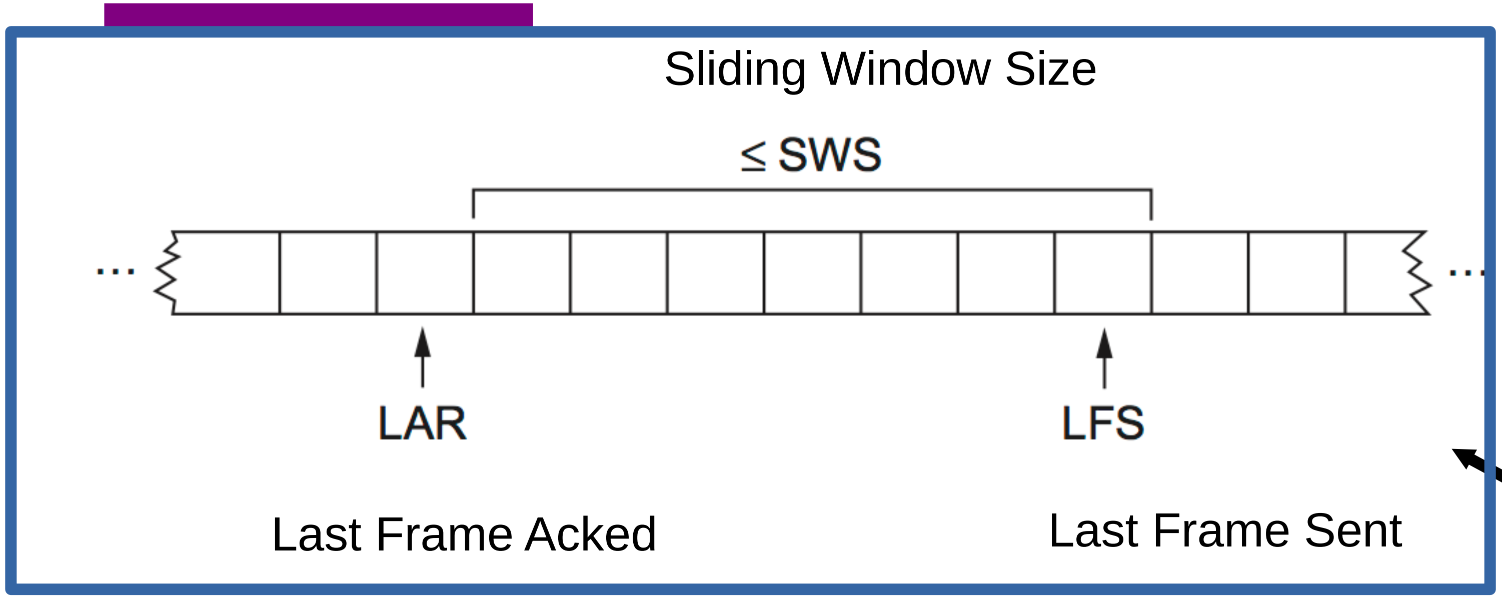
Sliding window to the rescue!

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


Sender
Receiver
First bit of first packet transmitted, t = 0
Last bit of first packet transmitted, t = L/R
RTT
ACK arrives, send next packet, t = RTT + L/R
First bit of first packet arrives
Last bit of first packet arrives, send ACK
Last bit of 2nd packet arrives, send ACK
Last bit of 3rd packet arrives, send ACK

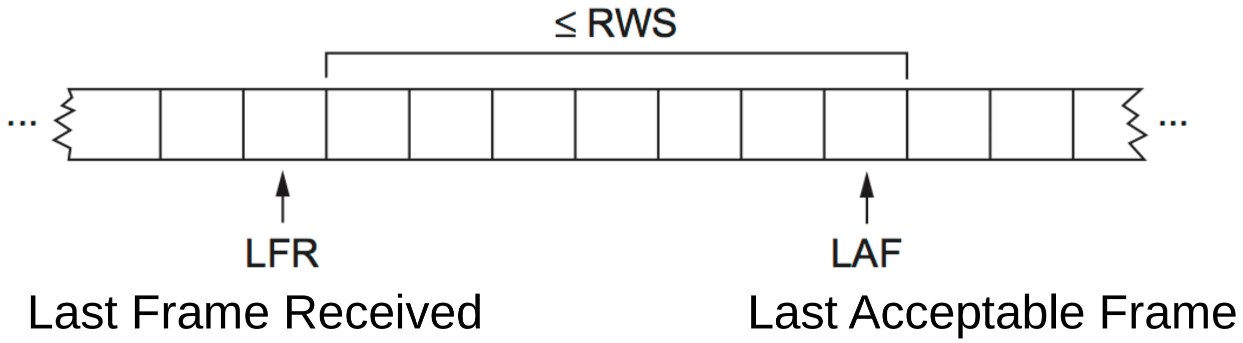
b. Pipelined operation

Sliding window – How does this work?



```
Outstanding
requests → SWS
LFS - LAR <= SWS
```

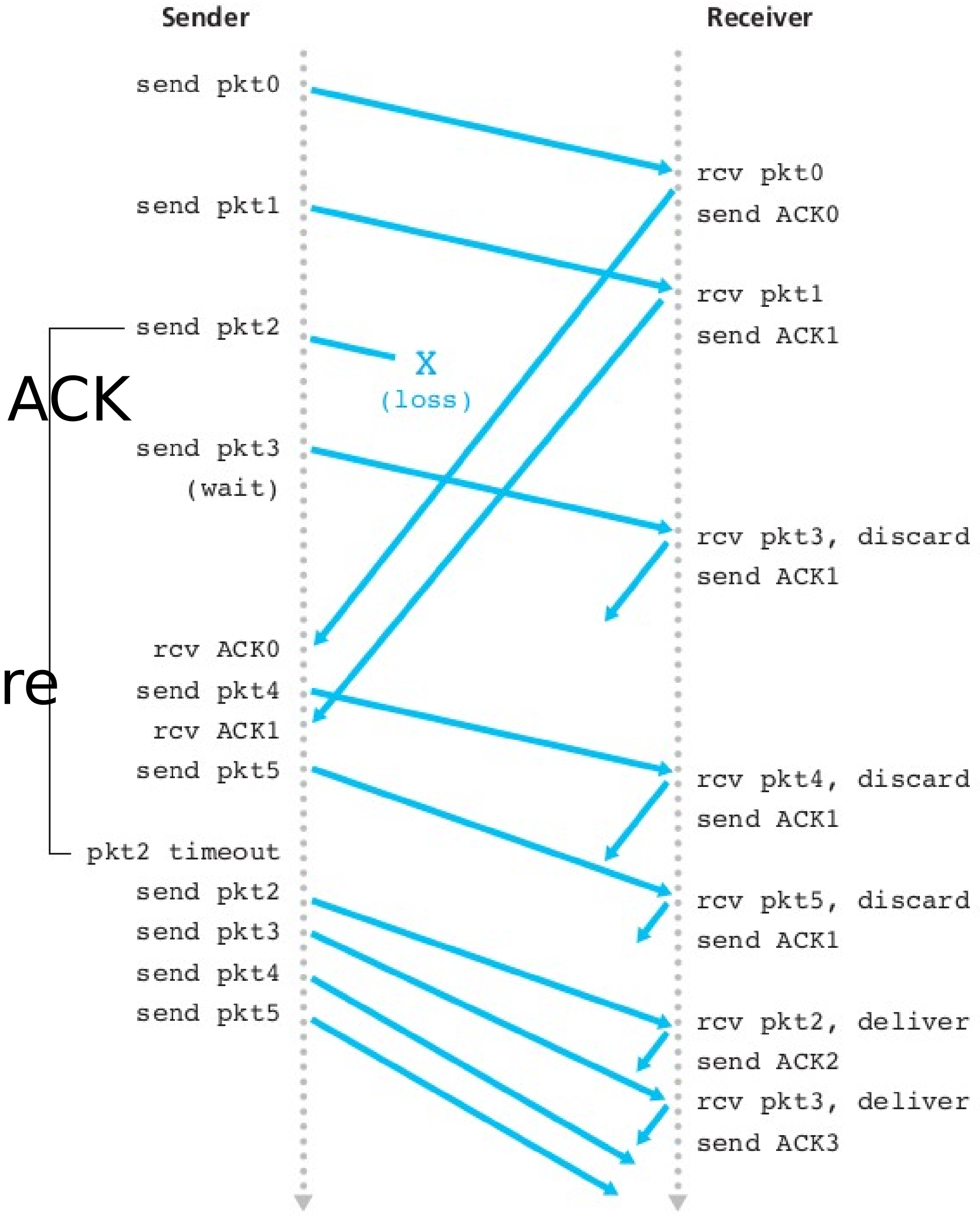
Sender



Sliding window - Go-Back-N

- Can transmit N bits before ACK

- See the problem?
- Can not move forward until all previous packets are acknowledged



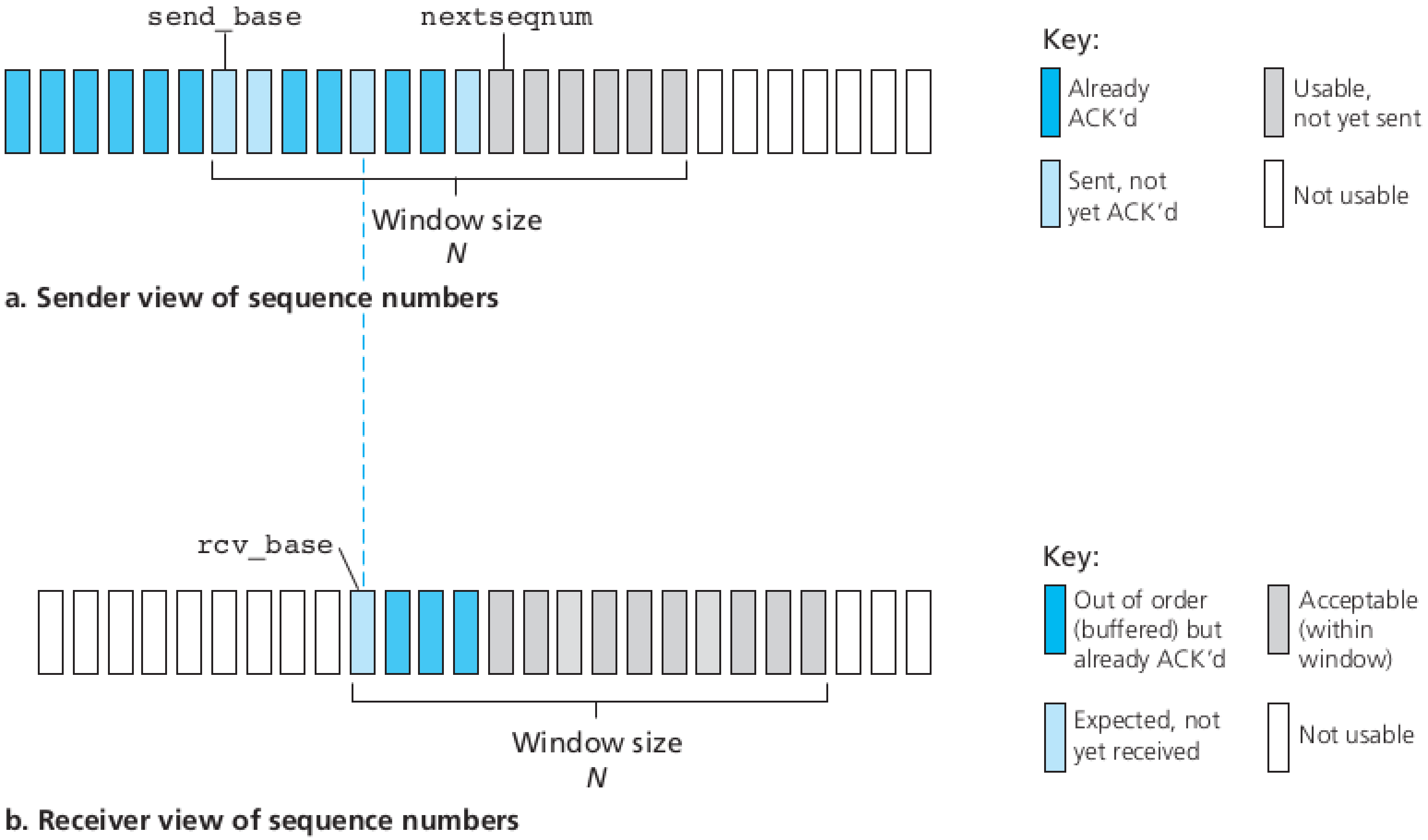
Sliding Window - Selective Repeat

http://www.exa.unicen.edu.ar/catedras/comdat1/material/Filminas3_Practico3.swf

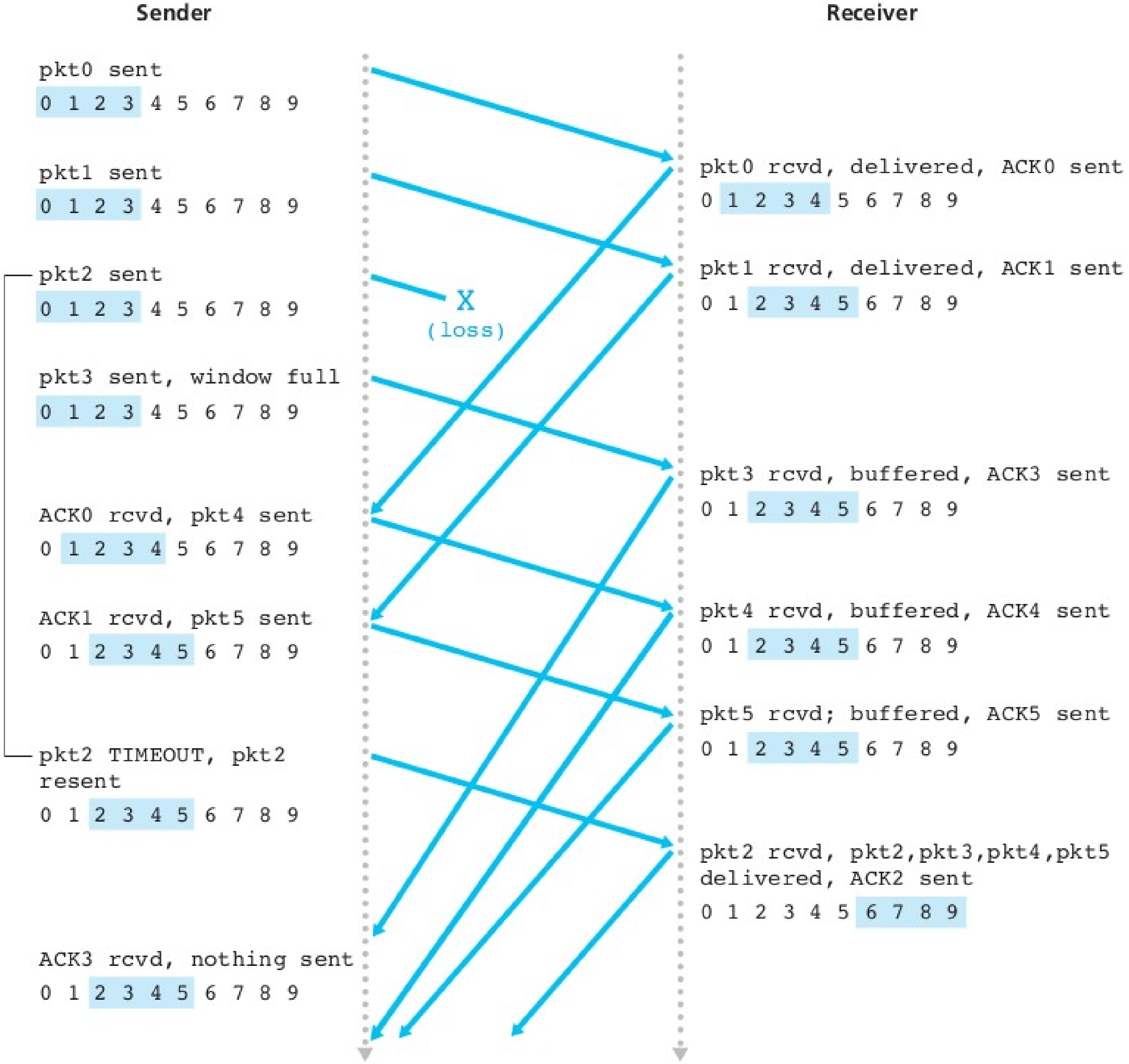
- Receiver:
  - Individually acks all packets
  - Buffers packets as necessary
  - Buffer packets until lost packets are received

- Sender:
  - Resend packets ***(only)*** for which ACK not received
  - Timer for each unACKed packet
  - Can send only n packets

Sliding window - Selective Repeat


send_base
nextseqnum
Window size
N
a. Sender view of sequence numbers
Key:
Already ACK'd
Usable, not yet sent
Sent, not yet ACK'd
Not usable
rcv_base
Window size
N
b. Receiver view of sequence numbers
Key:
Out of order (buffered) but already ACK'd
Acceptable (within window)
Expected, not yet received
Not usable

Sliding window - Selective Repeat



Sender

pkt0 sent
0 1 2 3 4 5 6 7 8 9

pkt1 sent
0 1 2 3 4 5 6 7 8 9

pkt2 sent
0 1 2 3 4 5 6 7 8 9

pkt3 sent, window full
0 1 2 3 4 5 6 7 8 9

ACK0 rcvd, pkt4 sent
0 1 2 3 4 5 6 7 8 9

ACK1 rcvd, pkt5 sent
0 1 2 3 4 5 6 7 8 9

pkt2 TIMEOUT, pkt2 resent
0 1 2 3 4 5 6 7 8 9

ACK3 rcvd, nothing sent
0 1 2 3 4 5 6 7 8 9

Receiver

pkt0 rcvd, delivered, ACK0 sent
0 1 2 3 4 5 6 7 8 9

pkt1 rcvd, delivered, ACK1 sent
0 1 2 3 4 5 6 7 8 9

pkt3 rcvd, buffered, ACK3 sent
0 1 2 3 4 5 6 7 8 9

pkt4 rcvd, buffered, ACK4 sent
0 1 2 3 4 5 6 7 8 9

pkt5 rcvd; buffered, ACK5 sent
0 1 2 3 4 5 6 7 8 9

pkt2 rcvd, pkt2,pkt3,pkt4,pkt5 delivered, ACK2 sent
0 1 2 3 4 5 6 7 8 9

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 recev_base, recv_base + N - 1, send ack (n)
  - Out of order: buffer
  - In-order or closes gap – deliver to application
  - Packet within <recv_base-N, recv_base -1>, 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:
  - https://book.systemsapproach.org/direct/reliable.html#reliable-transmission

    You may skip the coding part
    - **About 20 minutes**
  - https://en.wikipedia.org/wiki/Go-Back-N_ARQ
    - 5 minutes