
MONTANA
STATE UNIVERSITY

Performance of rdt3.0

- rdt3.0 correct, but performance stinks
- Calculate:
  - Transmission delay of 8000bit packet on 1Gbps link

$$D_t x = \frac{L}{R} = \frac{8000b}{10^9 bps} = 0.008ms$$

  - Link *utilization*, or the fraction of time sender is transmitting, assuming 15ms propagation delay and negligible size ACK packets

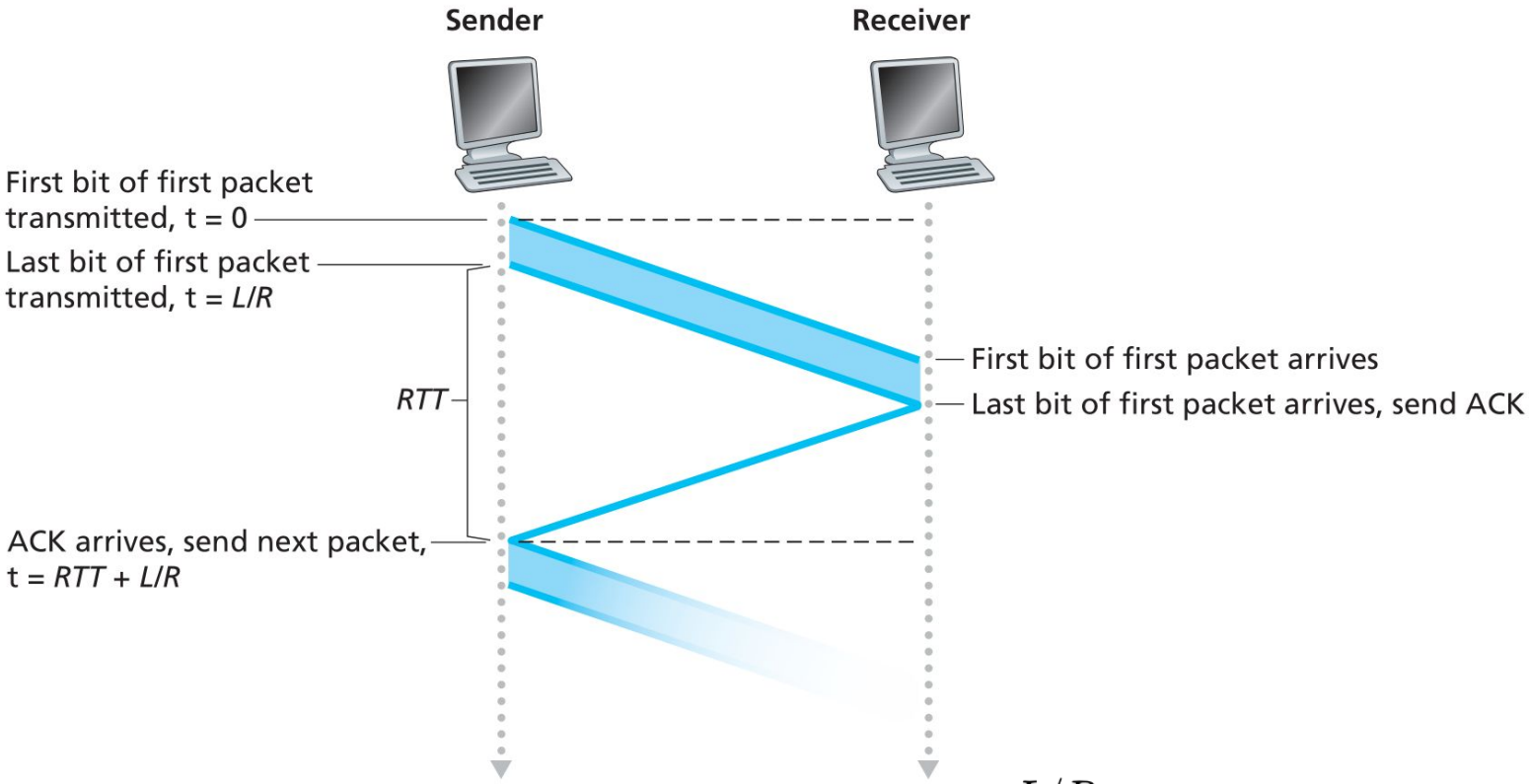
$$U_{snd} = \frac{L/R}{RTT+L/R} = \frac{0.008}{30.008} = 0.00027$$

  - Link goodput, or rate of data delivery over time

$$8000b/30.008ms = 267kbps$$

On a 1Gbps link!

rdt3.0: stop-and-wait operation



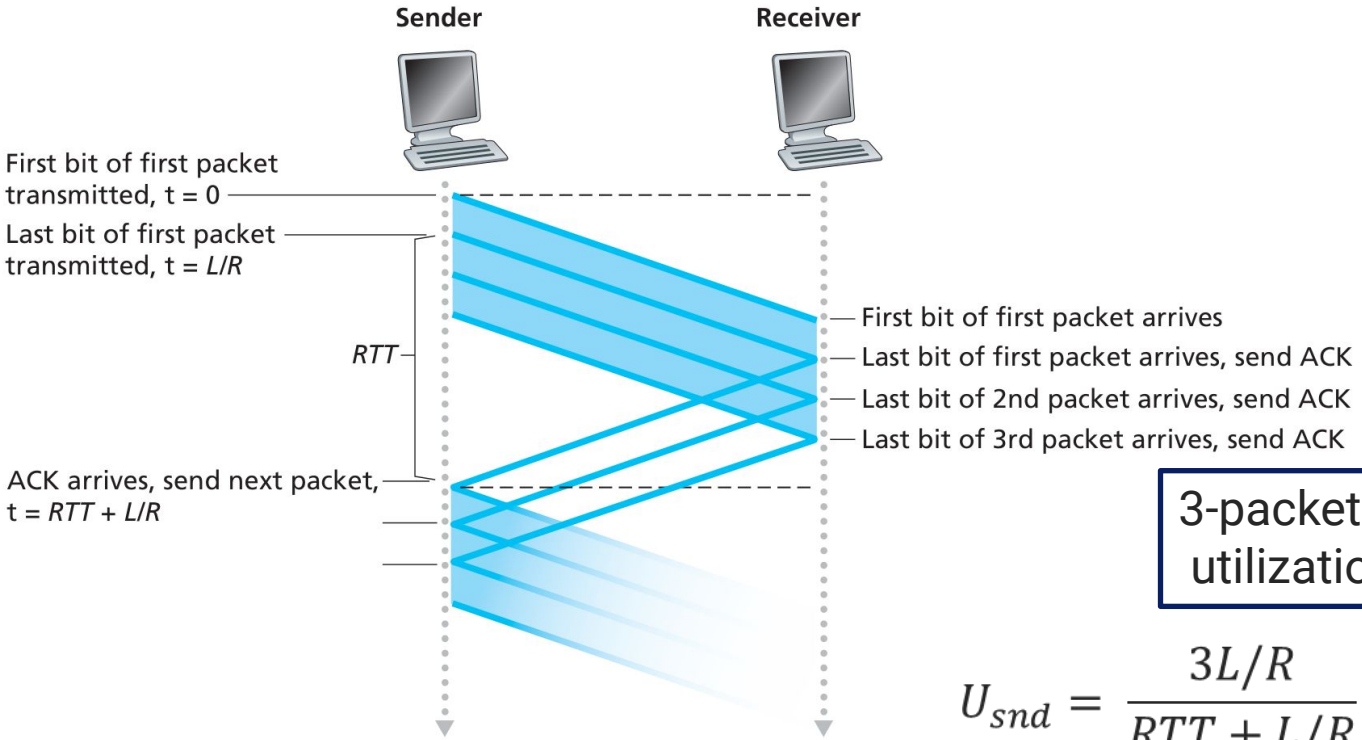
a. Stop-and-wait operation

$$U_{snd} = \frac{L/R}{RTT+L/R} = \frac{0.008}{30.008} = 0.00027$$

Pipelining: increased utilization

Pipelining: Sender allows multiple, "in-flight", yet-to-be-acknowledged packets

- Range of sequence numbers must be increased
- Buffering at sender and/or receiver



3-packet pipelining increases utilization by a factor of 3!

$$U_{snd} = \frac{3L/R}{RTT + L/R} = \frac{0.0024}{30.008} = 0.00081$$

Pipelined protocols: Overview

Go-back-N (GBN)

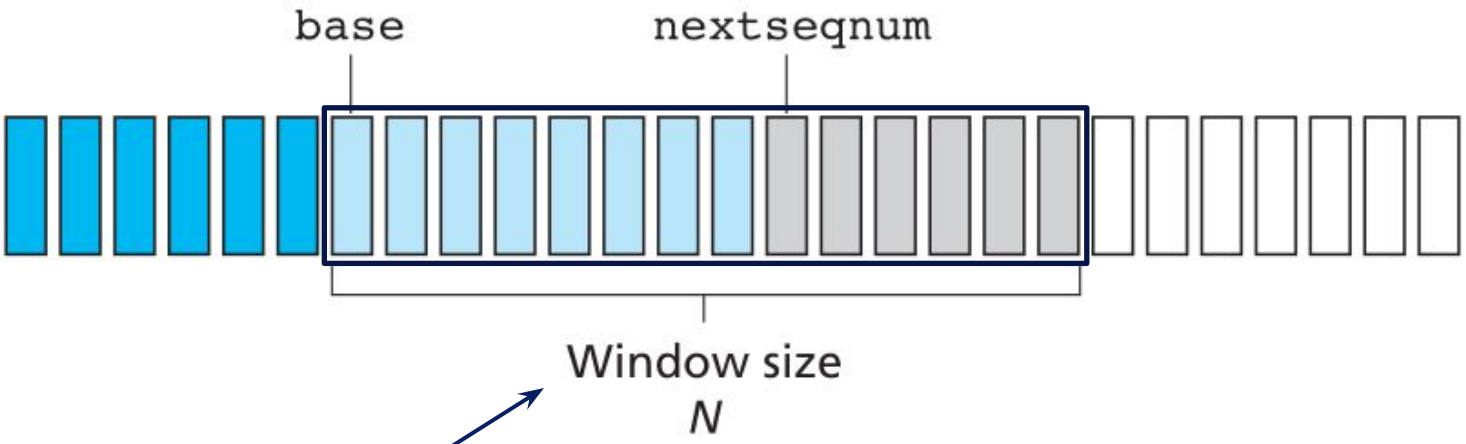
- Sender can have up to N unacked packets in pipeline
- Receiver only sends cumulative ack
  - ACK for last contiguous packet
  - No ACK for new packets past a sequence number gap
- Sender has timer for oldest unacked packet
  - When timer expires, retransmit all unacked packets

Selective Repeat (SR)

- Sender can have up to N unacked packets in pipeline
- Receiver sends individual ACK for each packet
- Sender maintains timer for each unacked packet
  - When timer expires, retransmit only that unacked packet

What are the relative advantages of each protocol?

GBN: Sender window

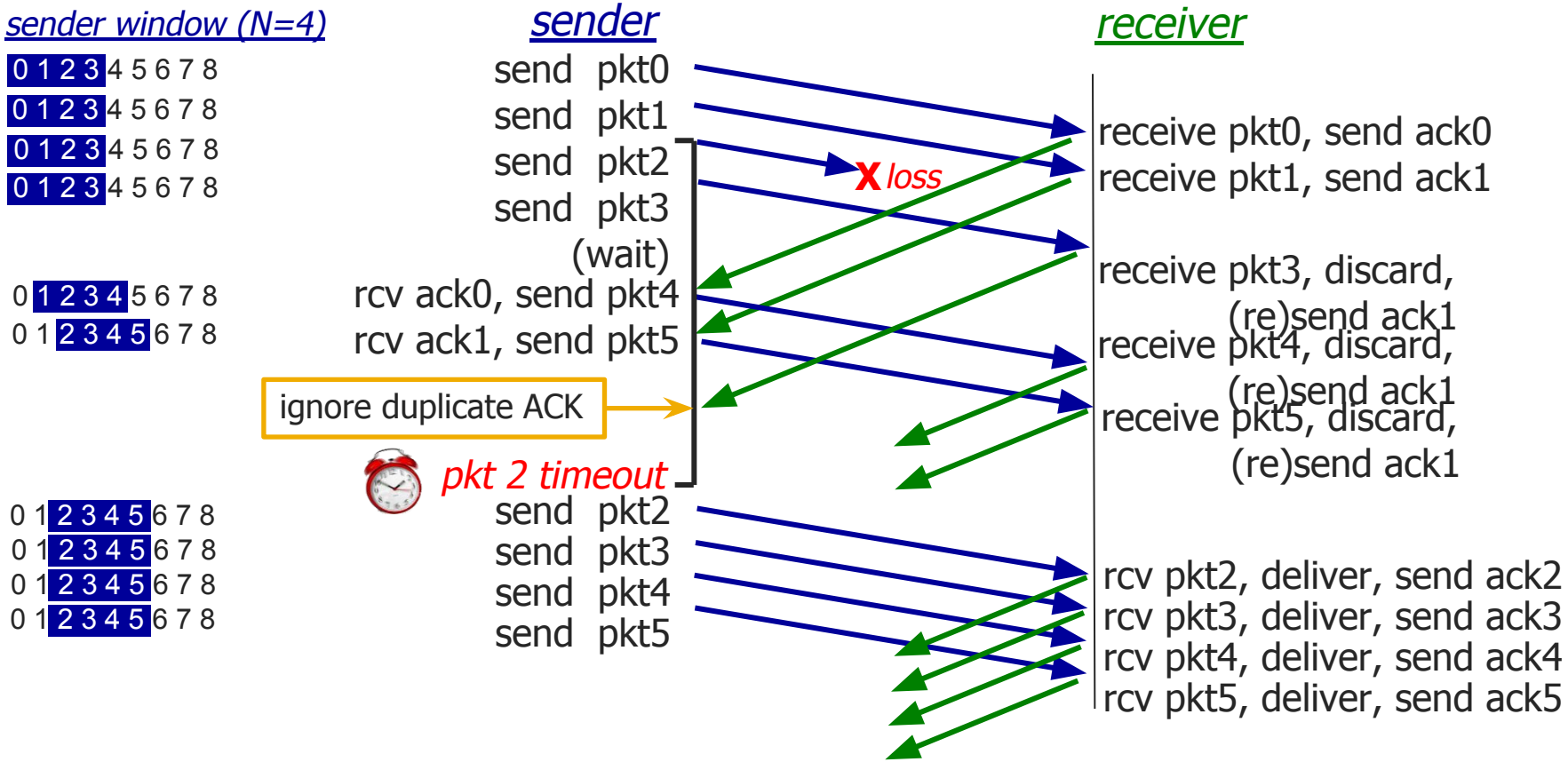


"Window" of up to *N*, consecutive unacknowledged packets allowed

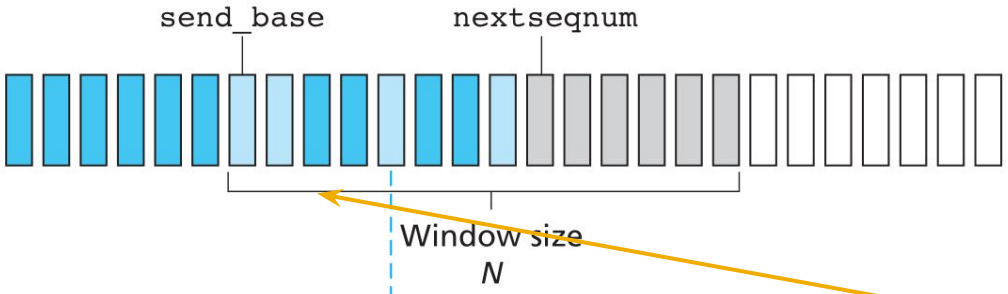
Key:

- Already ACK'd
- Usable, not yet sent
- Sent, not yet ACK'd
- Not usable

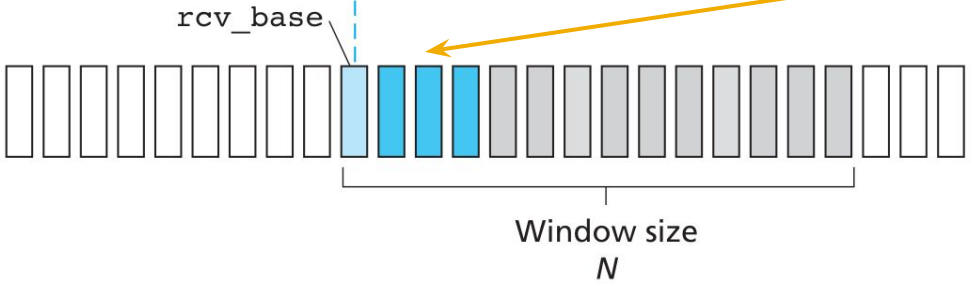
GBN in action



Selective Repeat (SR)



a. Sender view of sequence numbers



b. Receiver view of sequence numbers

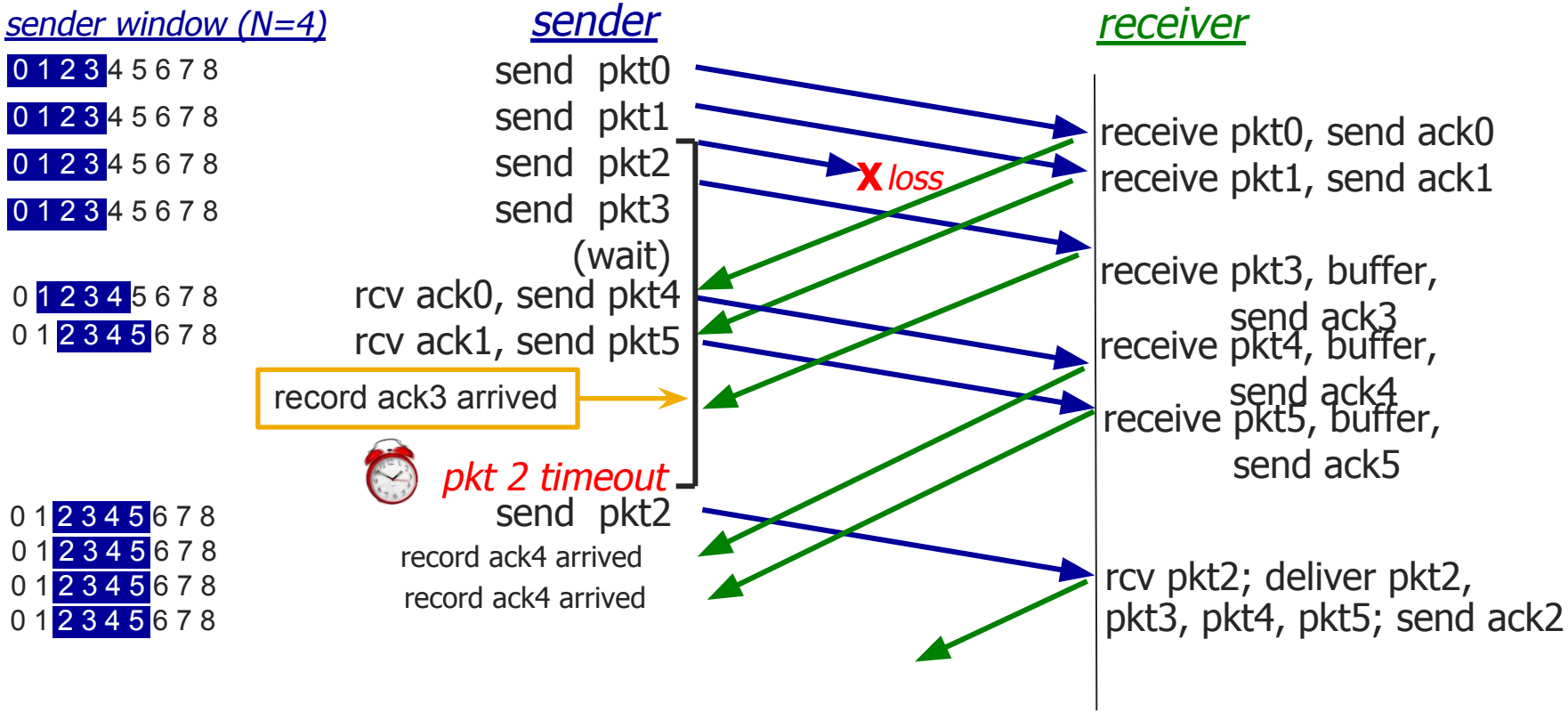
Key:
- Already ACK'd
- Usable, not yet sent
- Sent, not yet ACK'd
- Not usable

Sender only resends pkts for which ACK not received

Receiver *individually* ACKs all correctly received pkts

Key:
- Out of order (buffered) but already ACK'd
- Acceptable (within window)
- Expected, not yet received
- Not usable

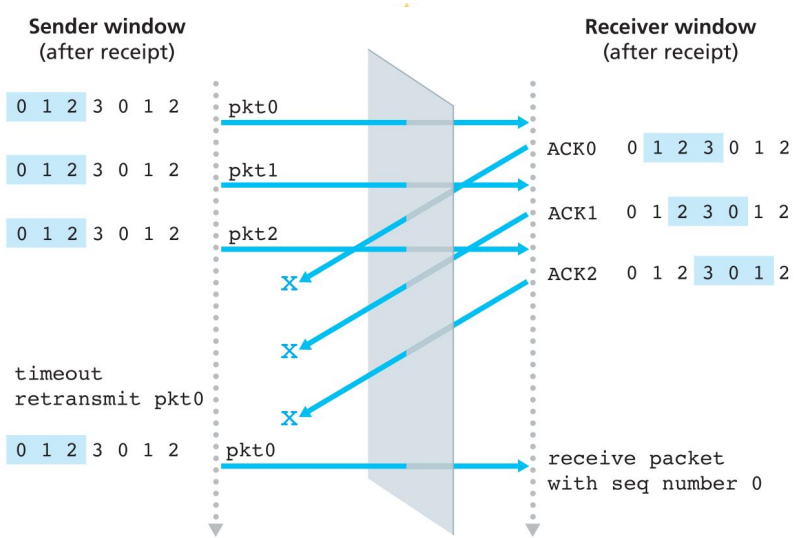
SR in action

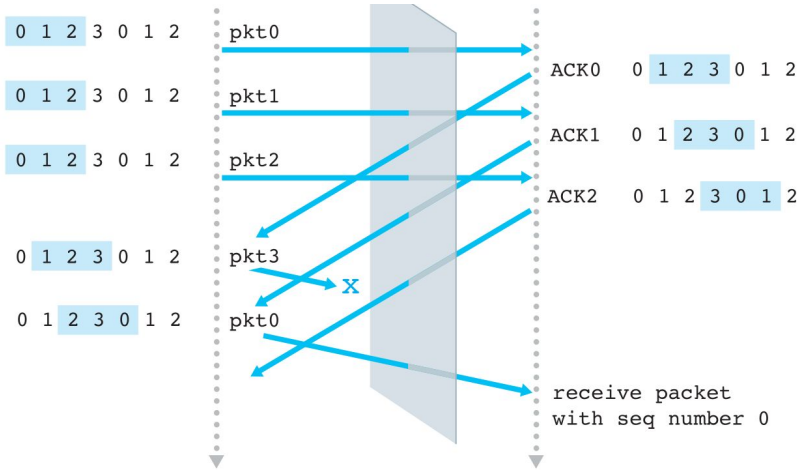

sender window (N=4)
0 1 2 3 4 5 6 7 8
0 1 2 3 4 5 6 7 8
0 1 2 3 4 5 6 7 8
0 1 2 3 4 5 6 7 8
0 1 2 3 4 5 6 7 8
0 1 2 3 4 5 6 7 8
0 1 2 3 4 5 6 7 8
0 1 2 3 4 5 6 7 8
0 1 2 3 4 5 6 7 8
sender
receiver
send pkt0
send pkt1
send pkt2
send pkt3
(wait)
rcv ack0, send pkt4
rcv ack1, send pkt5
record ack3 arrived
pkt 2 timeout
send pkt2
record ack4 arrived
record ack4 arrived
X loss
receive pkt0, send ack0
receive pkt1, send ack1
receive pkt3, buffer,
send ack3
receive pkt4, buffer,
send ack4
receive pkt5, buffer,
send ack5
rcv pkt2; deliver pkt2,
pkt3, pkt4, pkt5; send ack2

Selective Repeat Dilemma

- Receiver sees no difference in the two scenarios!
- Duplicate data accepted as new in the second scenario!

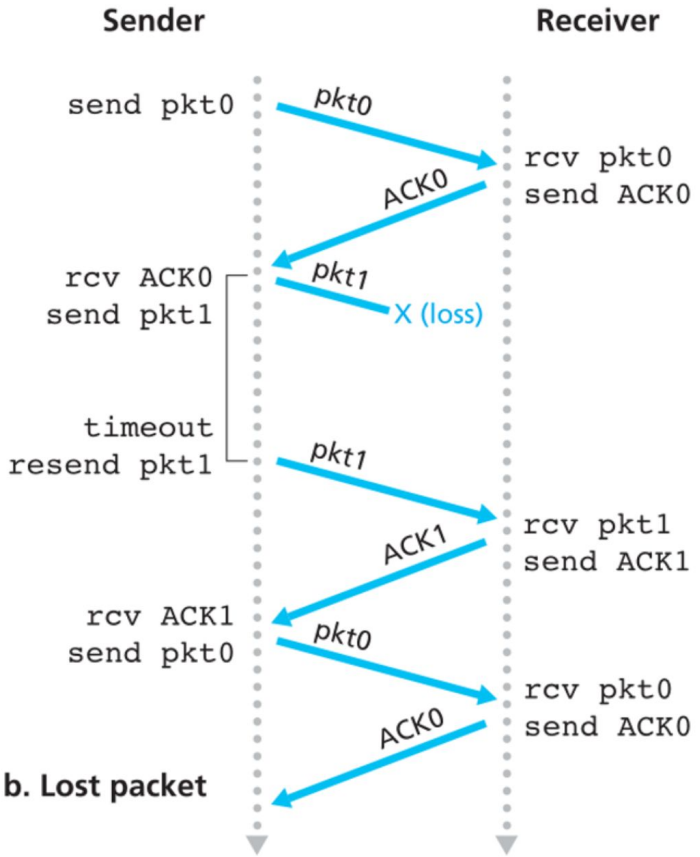
Q: What should be the relationship between window size and sequence numbers to avoid the problem?





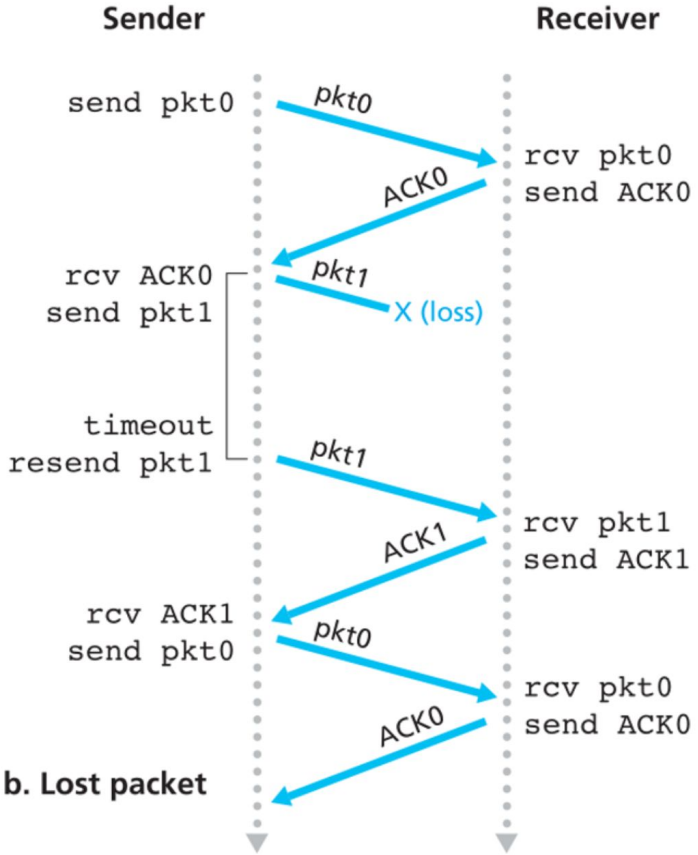
TCP round trip time, timeout

- Q: How to set TCP timeout value?
  - Too short: premature timeout, unnecessary retransmissions
  - Too long: slow reaction to segment loss
  - Longer than RTT?
    - but RTT varies



TCP round trip time, timeout

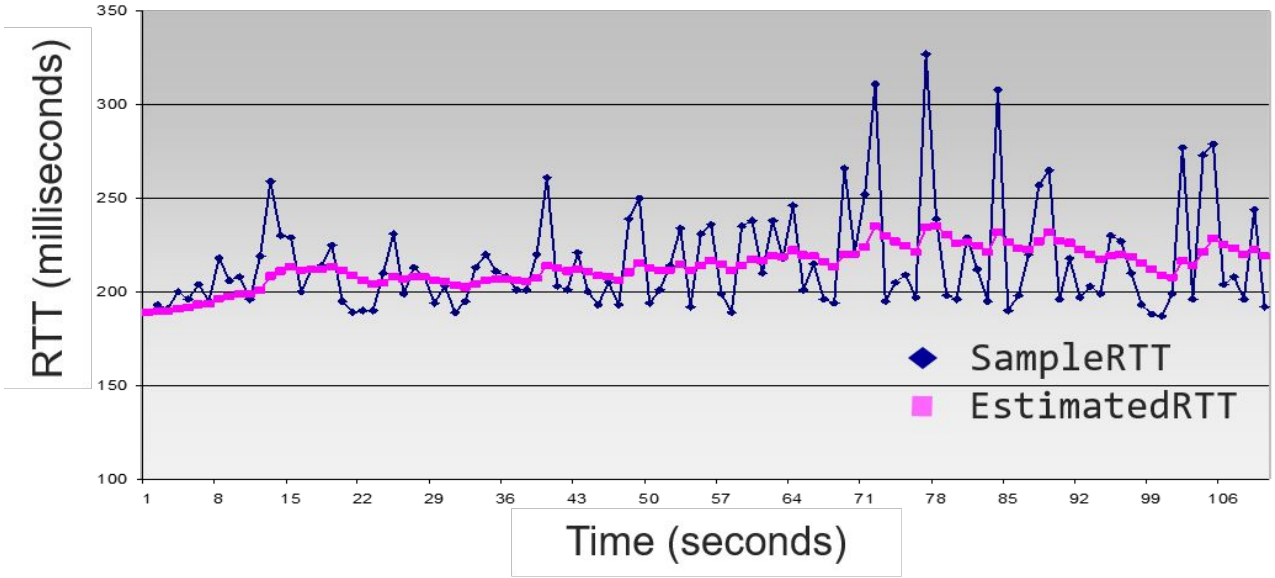
- Q: how to estimate RTT?
  - SampleRTT: measured time from segment transmission until ACK receipt
  - SampleRTT will vary, want estimated RTT "smoother"
    - Average several recent measurements, not just current SampleRTT



TCP round trip time, timeout

- Exponential weighted moving average
  - `EstimatedRTT = (1-α)*EstimatedRTT + α*SampleRTT`
  - `Typical value: α=0.125`

RTT: gaia.cs.umass.edu to fantasia.eurecom.fr



`TimeoutInterval = EstimatedRTT ?`

TCP round trip time, timeout

- Timeout interval:
  - `EstimatedRTT` plus "safety margin"
  - large variation in `EstimatedRTT` → larger safety margin
- Estimate SampleRTT deviation from EstimatedRTT:

$$DevRTT = (1-\beta)*DevRTT + \beta*|SampleRTT-EstimatedRTT|$$

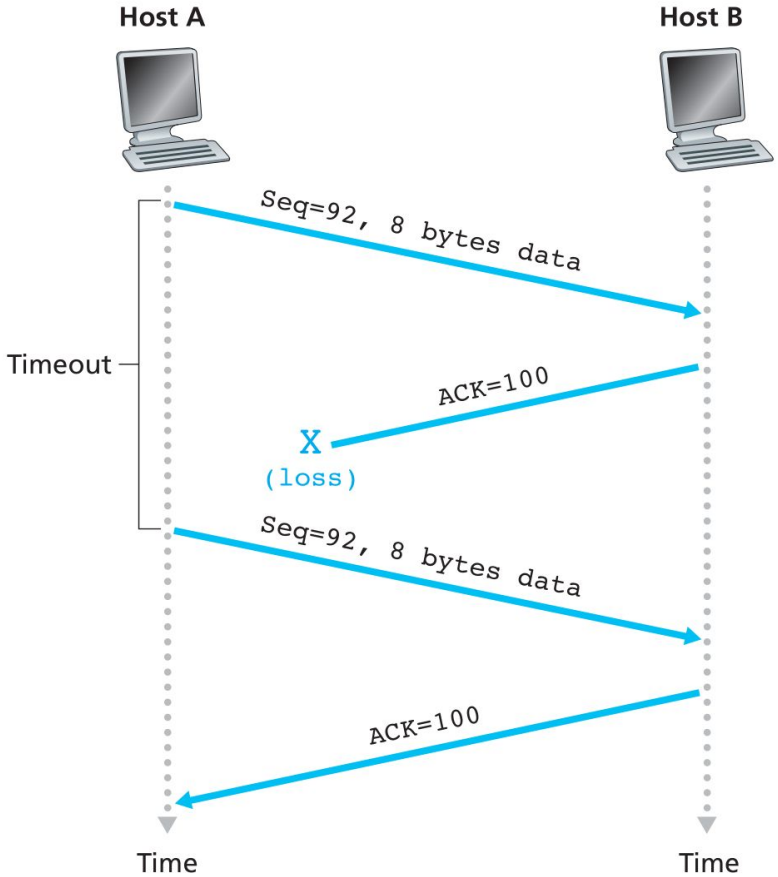
$$TimeoutInterval = EstimatedRTT + 4*DevRTT$$

Estimated RTT

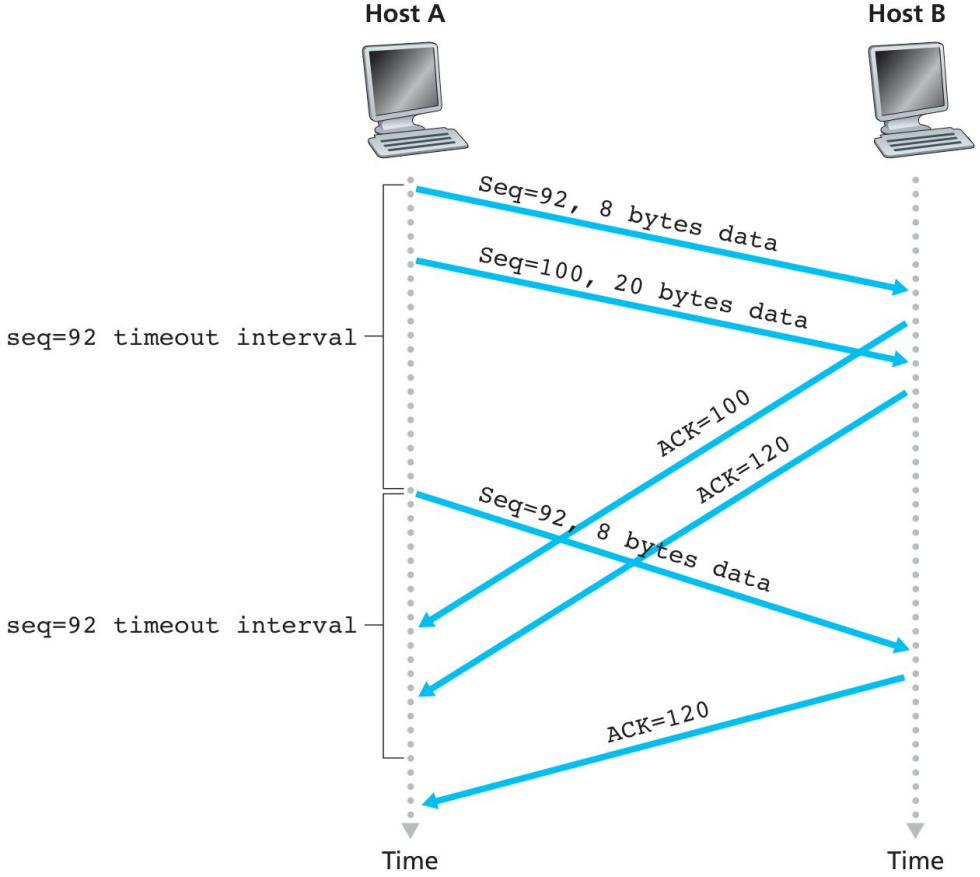
"safety margin"

TCP retransmission scenarios

Lost ACK scenario

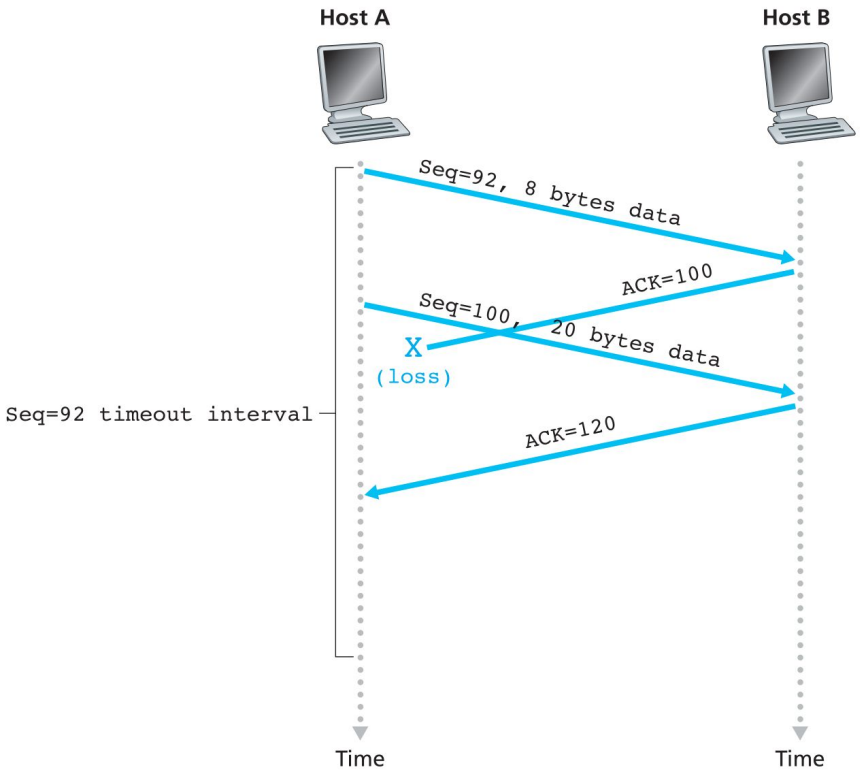


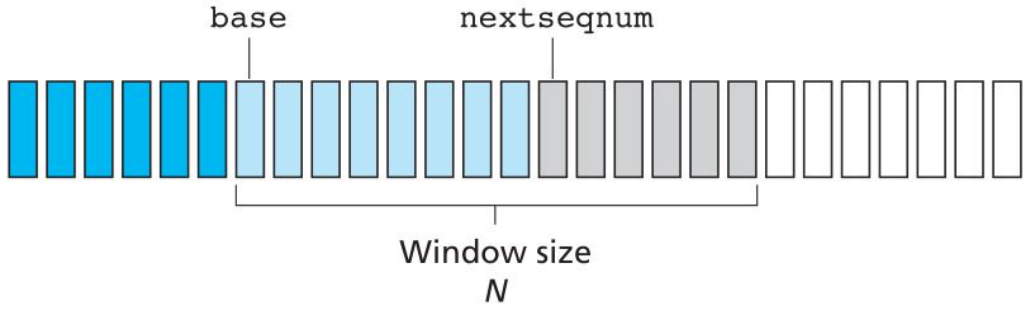
Premature timeout



TCP: retransmission scenarios

Cumulative ACK



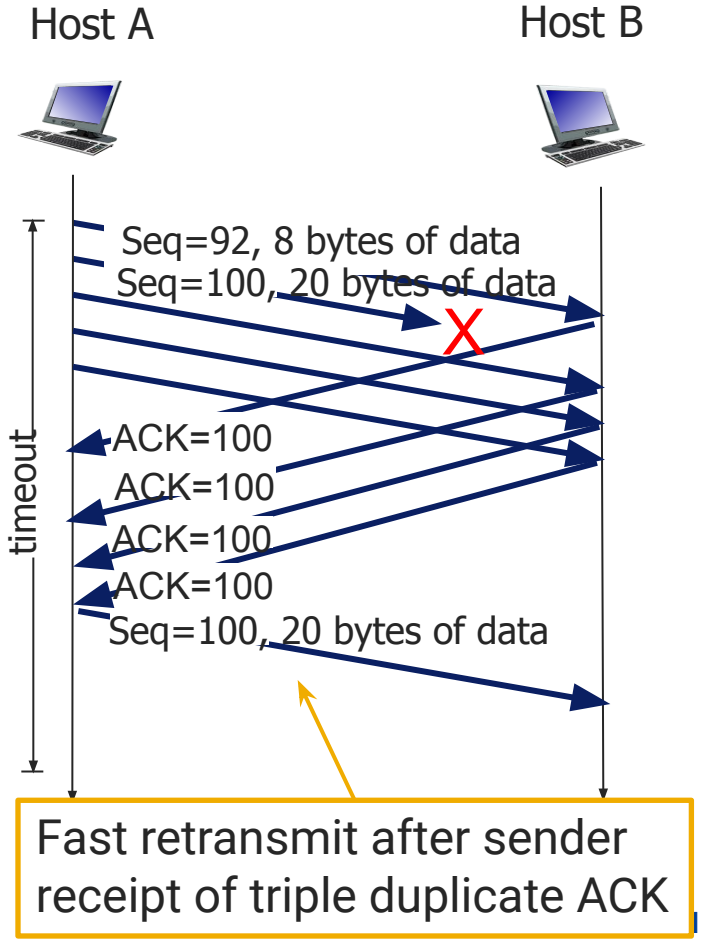


TCP ACK generation

Event at receiver	TCP receiver action
arrival of in-order segment with expected seq #. All data up to expected seq # already ACKed	delayed ACK. Wait up to 500ms for next segment. If no next segment, send ACK
arrival of in-order segment with expected seq #. One other segment has ACK pending	immediately send single cumulative ACK, ACKing both in-order segments
arrival of out-of-order segment higher-than-expect seq. # . Gap detected	immediately send *duplicate ACK*, indicating seq. # of next expected byte
arrival of segment that partially or completely fills gap	immediate send ACK, provided that segment starts at lower end of gap

TCP *fast retransmit*

- Time-out period often relatively long before packet loss detected
- Detect lost segments via duplicate ACKs
  - Many packets in pipeline
  - If segment lost, many duplicate ACKs generated
- TCP fast retransmit
  - Resend packet after **3 duplicate ACKs**
  - Do not wait for timeout – packet probably already lost




MONTANA
STATE UNIVERSITY