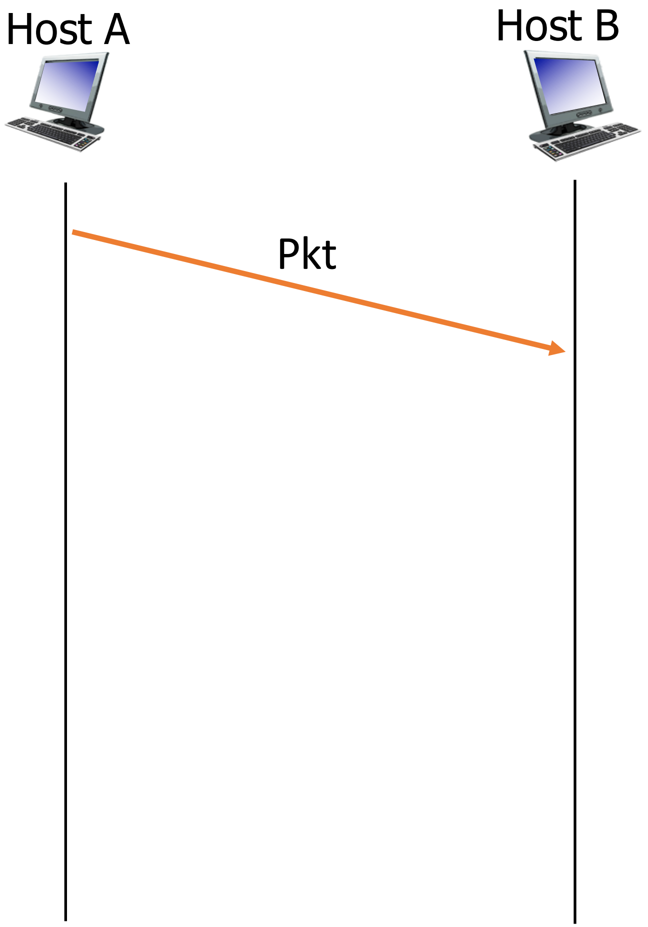
CSE 461: Introduction to Computer Communication Networks

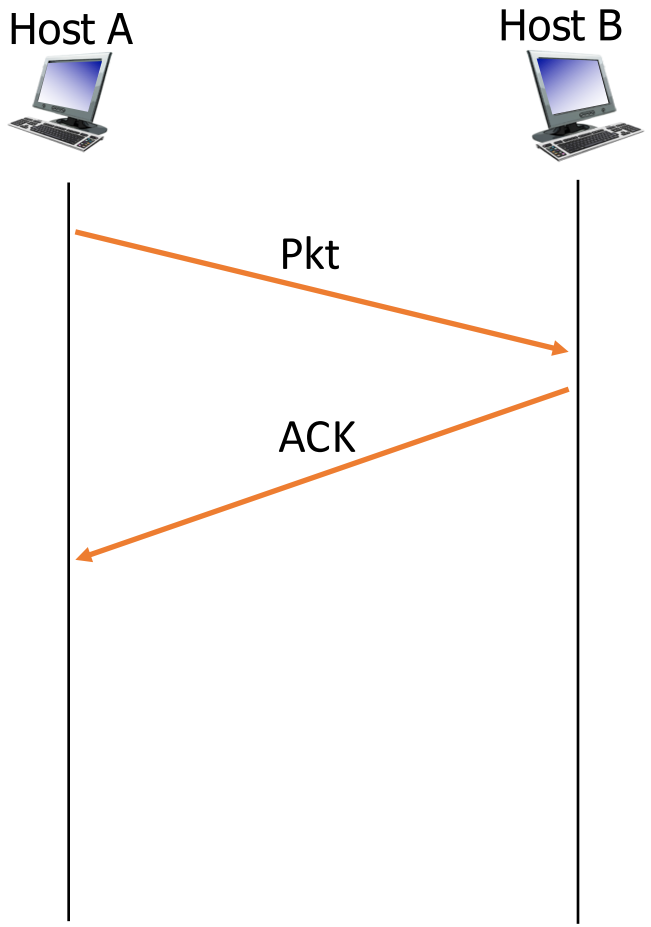
Chunjong Park

Reliable Data Transfer



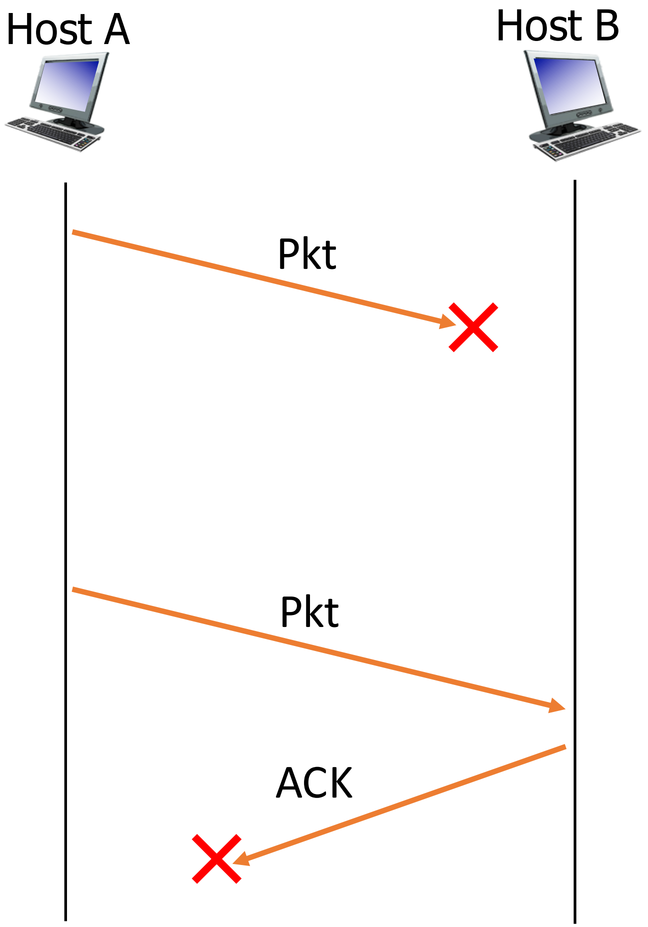
- A sends a packet to B
- Ideally, the packet should arrive at B
- But A does not know whether B receives it
- How could B tell A that the packet is arrived at B?

Reliable Data Transfer: ACK



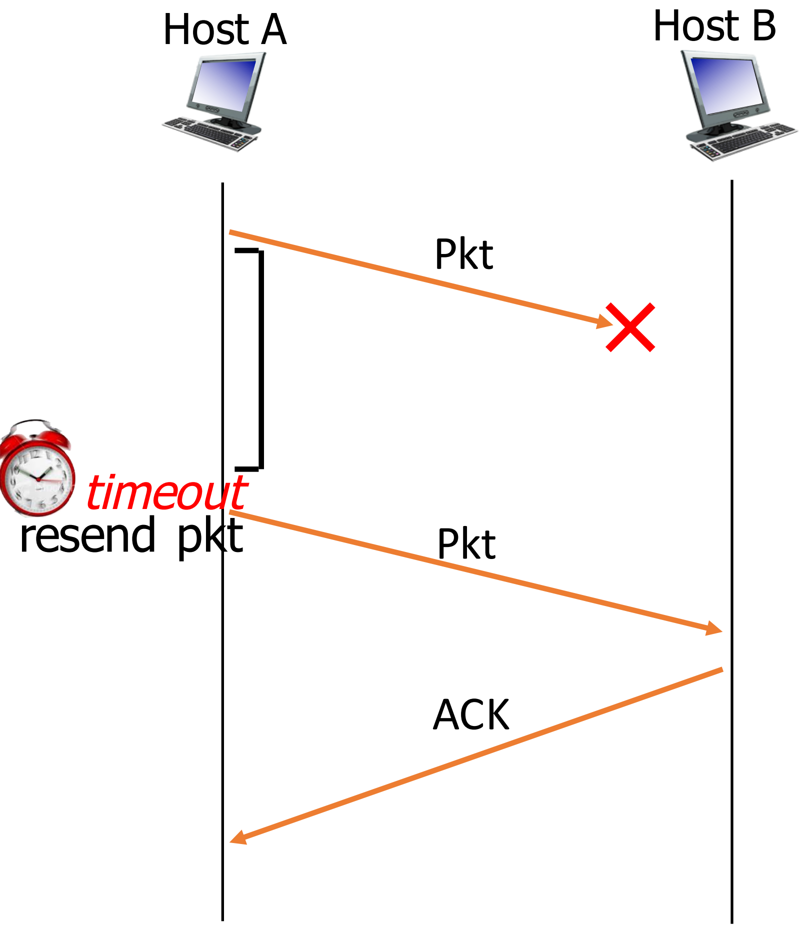
- A sends a packet to B
- The packet arrives at B
- B tells A that the it receives the packet
- A sends out the next packet

Reliable Data Transfer: Packet loss



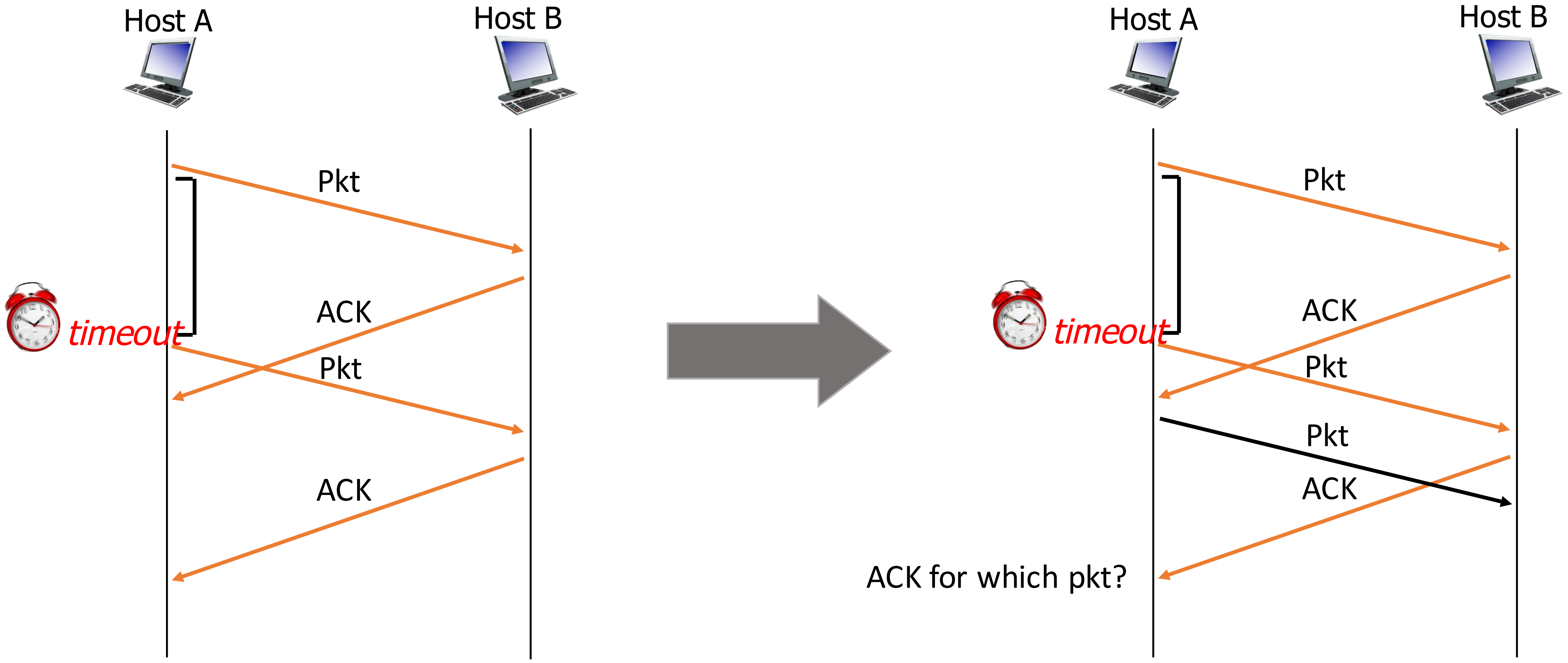
- But what if a packet or an ACK is lost?
- A can't wait for an ACK forever.

Reliable Data Transfer: Timeout



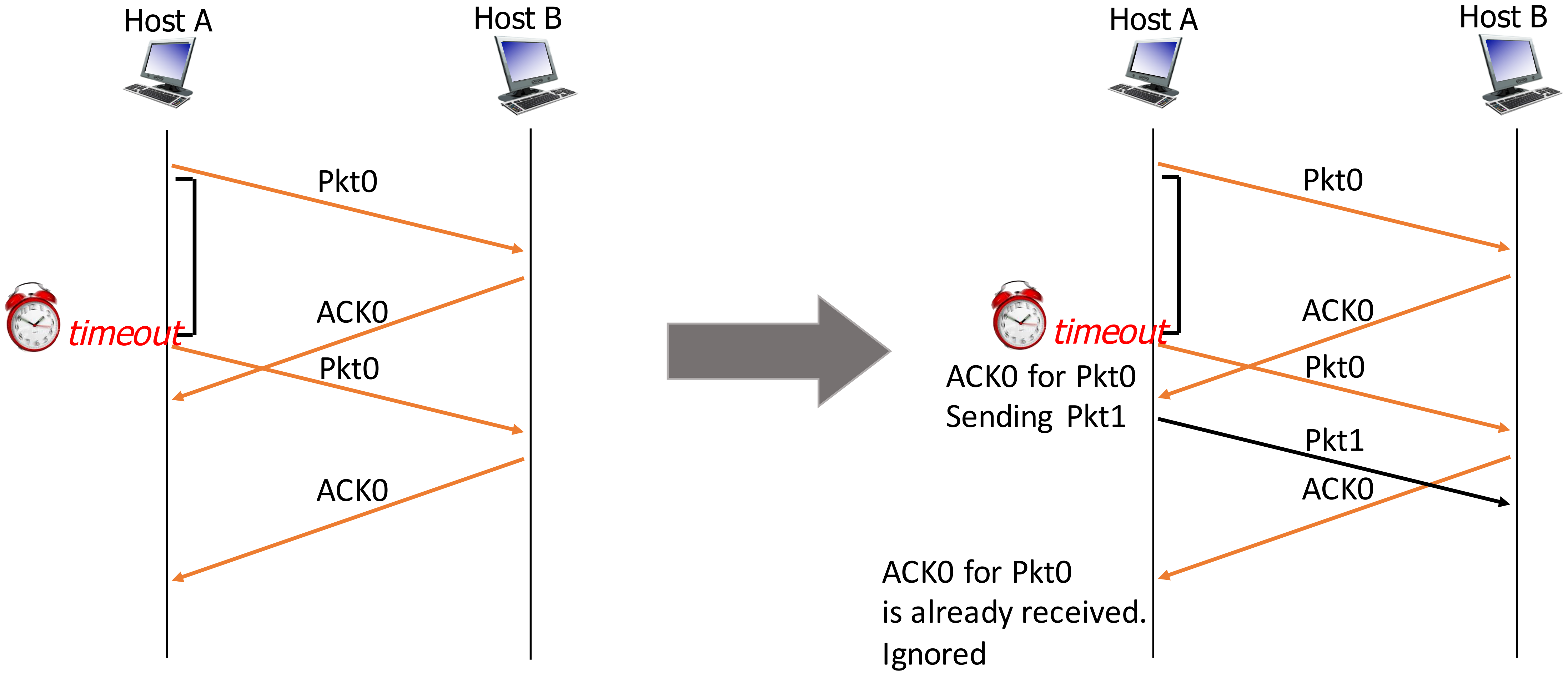
- A only waits for a certain period of time
- When timeout, A resends the packet

Premature Timeout

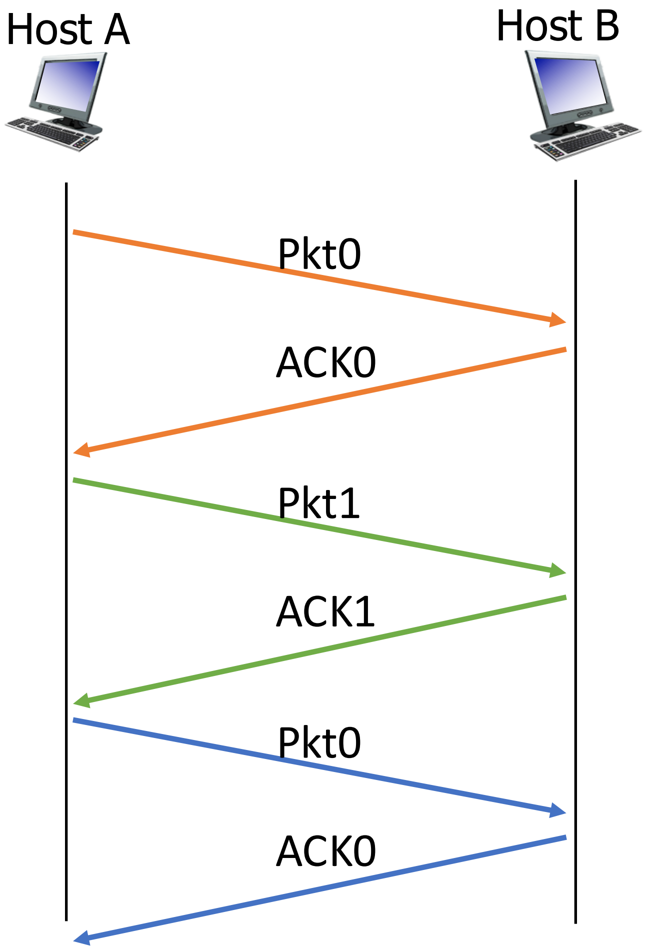


ACK for which pkt?

Sequence


Host A
Host B
Pkt0
ACK0
timeout
Pkt0
ACK0
Host A
Host B
Pkt0
ACK0
timeout
ACK0 for Pkt0
Sending Pkt1
Pkt0
Pkt1
ACK0
ACK0 for Pkt0
is already received.
Ignored

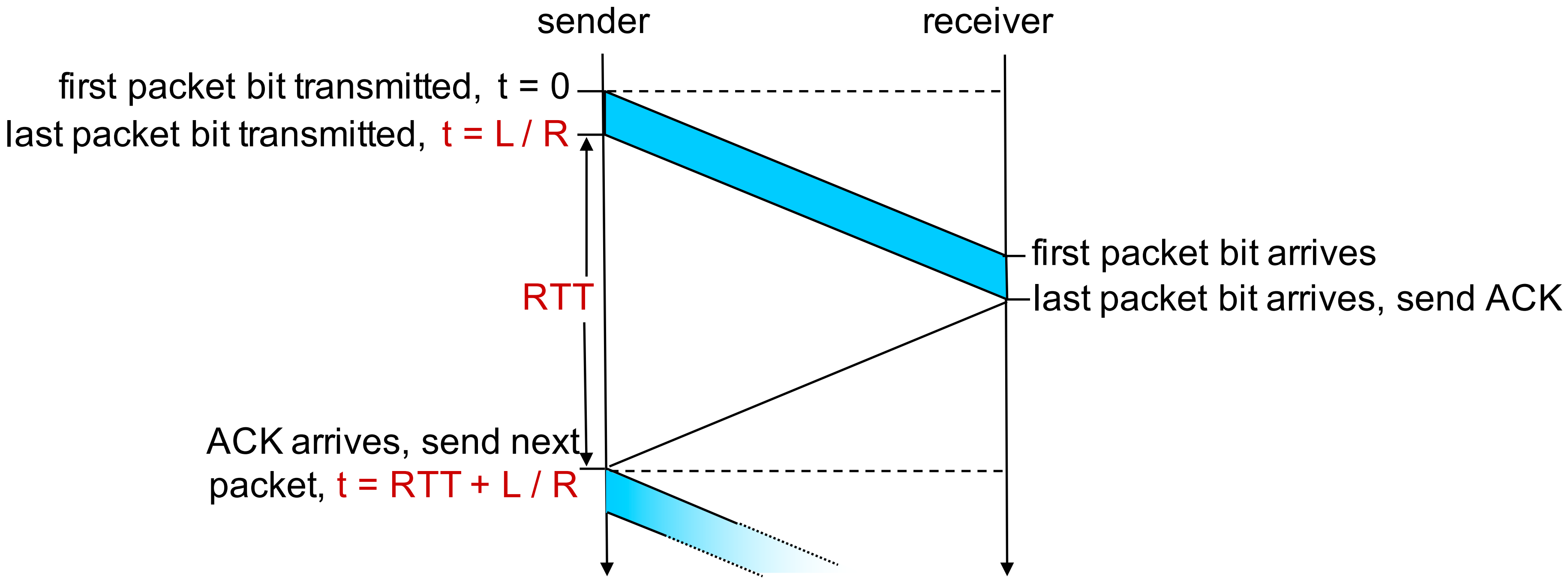
Stop-and-wait



- A sender sends only a single packet before it receives the corresponding ACK
- Only needs 0/1 for sequence number
  - Just needs to distinguish two consecutive pkts
- Physical link is underutilized!

Stop-and-wait

- R = 1 Gbps link, RTT = 15 ms prop. delay, L = 8000 bit packet



sender
receiver
first packet bit transmitted, t = 0
last packet bit transmitted, t = L / R
RTT
first packet bit arrives
last packet bit arrives, send ACK
ACK arrives, send next packet, t = RTT + L / R

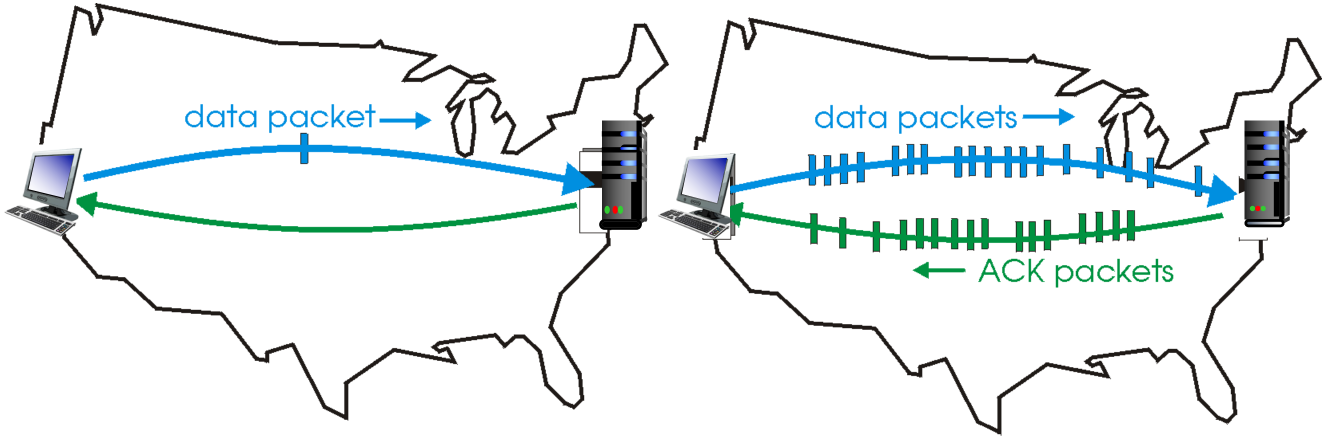
$$U_{sender} = \frac{L / R}{RTT + L / R} = \frac{.008}{30.008} = 0.00027$$

Borrowed from Kurose’s slides

Pipelined protocols (Sliding Window)

pipelining: sender allows multiple, "in-flight", yet-to-be-acknowledged pkts

- **range of sequence numbers must be increased**
- **buffering at sender and/or receiver**

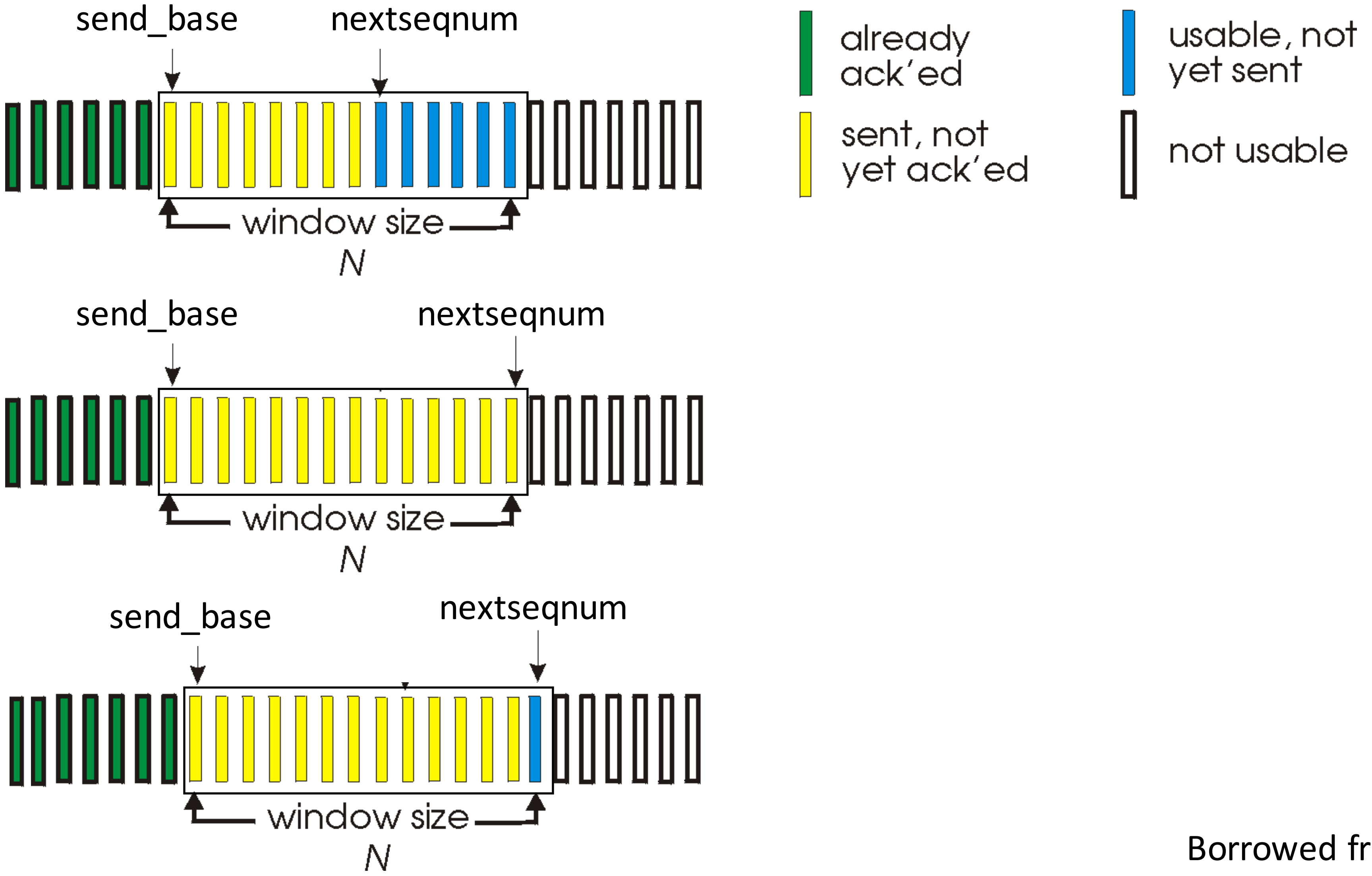


(a) a stop-and-wait protocol in operation (b) a pipelined protocol in operation

❖two generic forms of pipelined protocols: *go-Back-N, **selective repeat***

Borrowed from Kurose's slides

Sliding Window: Sender



send_base

nextseqnum

window size

N

send_base

nextseqnum

window size

N

send_base

nextseqnum

window size

N

🟩 already ack'ed

🟦 usable, not yet sent

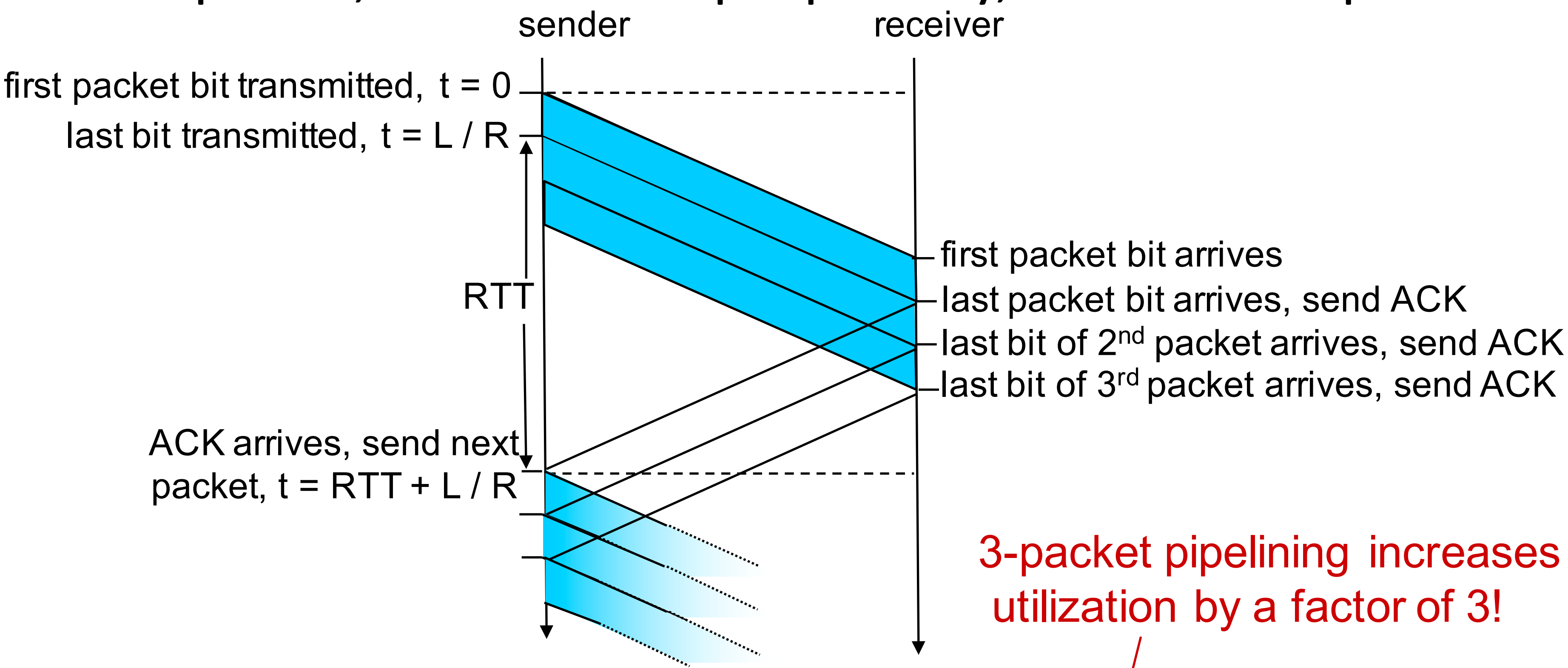
🟨 sent, not yet ack'ed

▯ not usable

Borrowed from Kurose's slides

Sliding Window: Increased Utilization

- R = 1 Gbps link, RTT = 15 ms prop. delay, L = 8000 bit packet



sender

receiver

first packet bit transmitted, t = 0

last bit transmitted, t = L / R

RTT

ACK arrives, send next packet, t = RTT + L / R

first packet bit arrives

last packet bit arrives, send ACK

last bit of 2nd packet arrives, send ACK

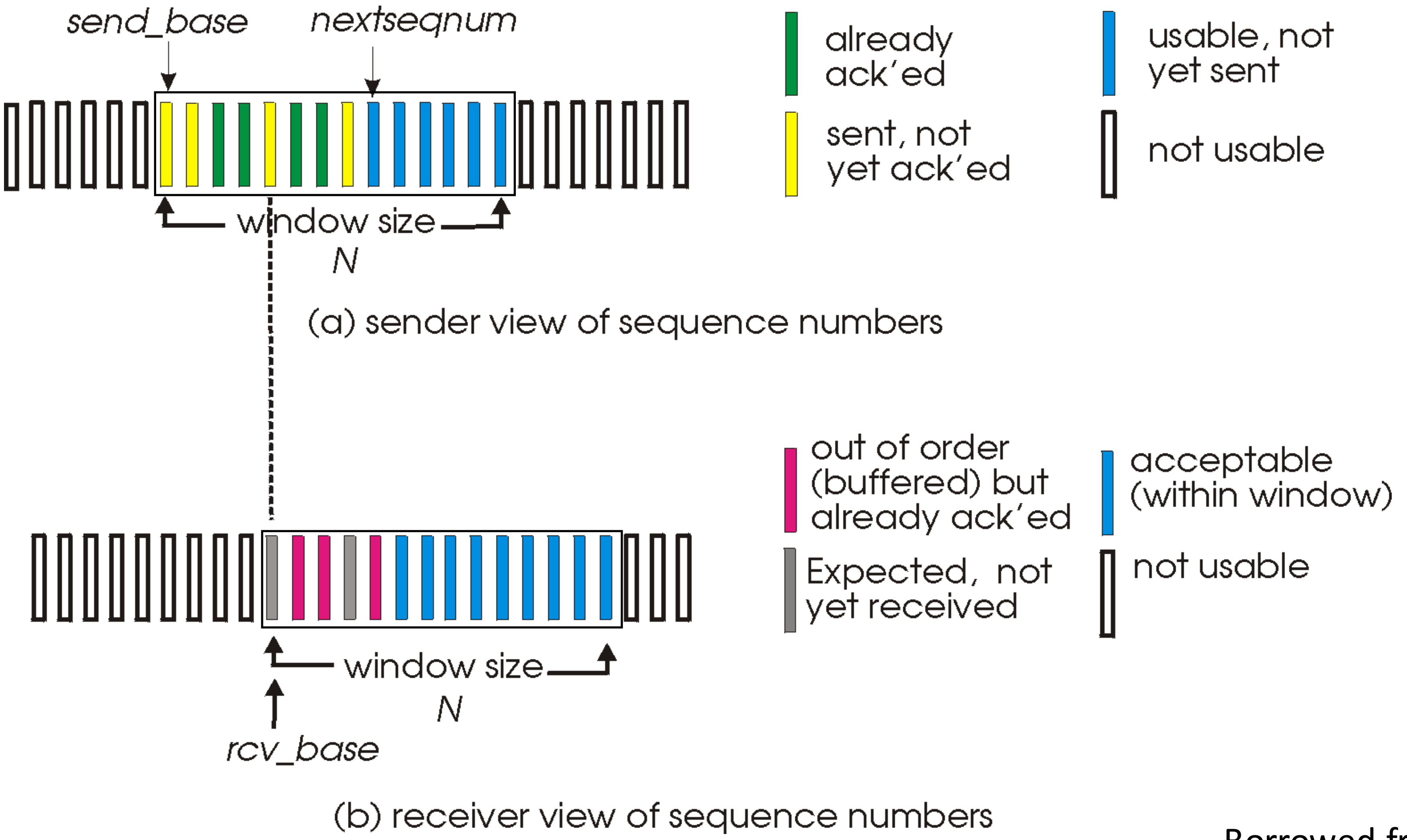
last bit of 3rd packet arrives, send ACK

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

$$U_{sender} = \frac{3L / R}{RTT + L / R} = \frac{.0024}{30.008} = 0.00081$$

Borrowed from Kurose’s slides

Sliding Window: Selective Repeat



(a) sender view of sequence numbers

(b) receiver view of sequence numbers

Borrowed from Kurose's slides

Selective Repeat

sender

data from above:

- if next available seq # in window, send pkt

timeout(n):

- resend pkt n, restart timer

ACK(n) in [sendbase,sendbase+N]:

- mark pkt n as received
- if n smallest unACKed pkt, advance window base to next unACKed seq #

receiver

pkt n in [rcvbase, rcvbase+N-1]

- send ACK(n)
- out-of-order: buffer
- in-order: deliver (also deliver buffered, in-order pkts), advance window to next not-yet-received pkt

pkt n in [rcvbase-N,rcvbase-1]

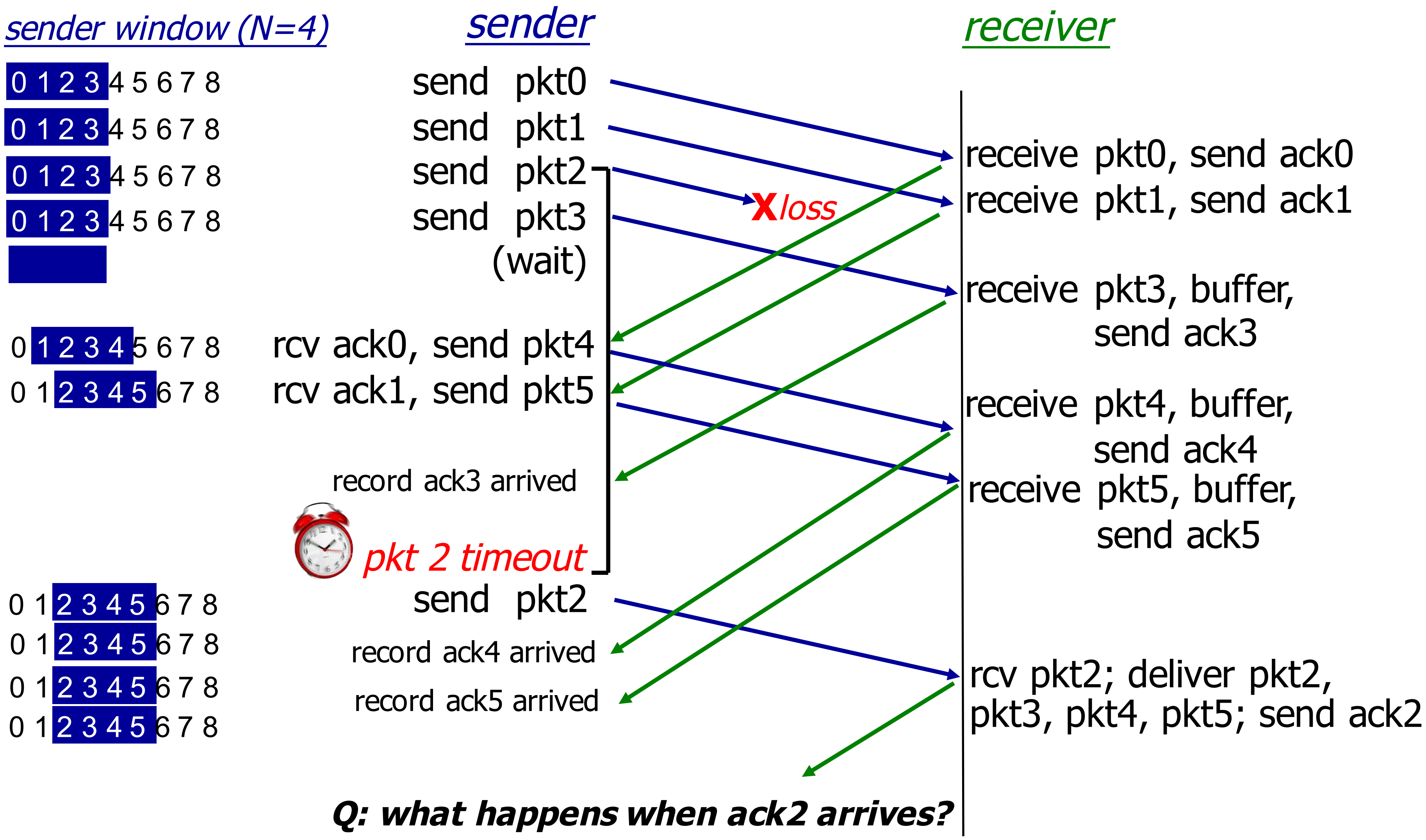
- ACK(n)

otherwise:

- ignore

Borrowed from Kurose's slides

Selective Repeat 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
0 1 2 3 4 5 6 7 8

sender

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 ack5 arrived

receiver

receive pkt0, send ack0
receive pkt1, send ack1

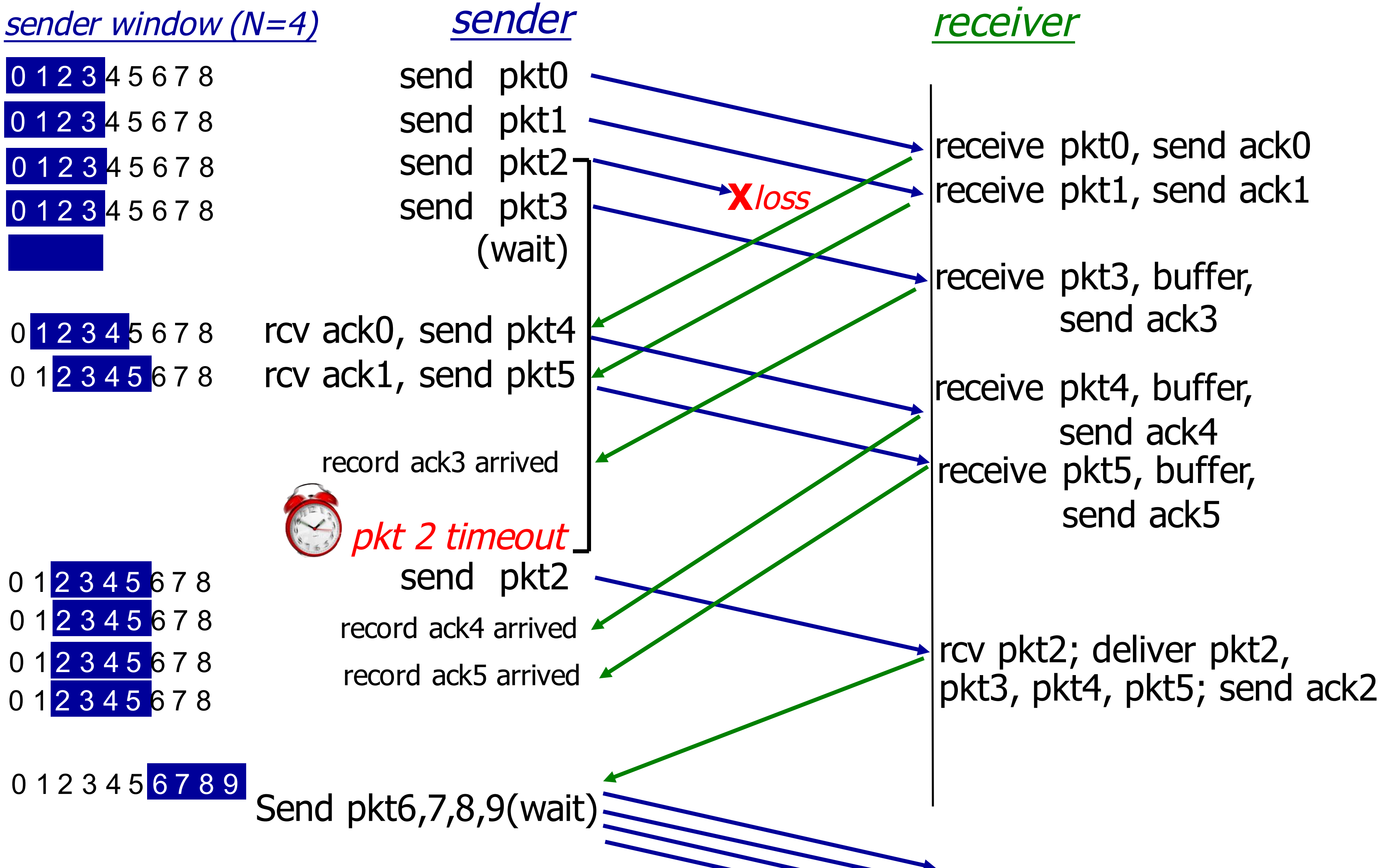
receive pkt3, buffer,
send ack3

receive pkt4, buffer,
send ack4
receive pkt5, buffer,
send ack5

rcv pkt2; deliver pkt2,
pkt3, pkt4, pkt5; send ack2

Q: what happens when ack2 arrives?

Selective Repeat 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
0 1 2 3 4 5 6 7 8

0 1 2 3 4 5 6 7 8 9

sender

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 ack5 arrived

Send pkt6,7,8,9(wait)

X loss

receiver

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