



CSC/CPE 138 - Computer Network Fundamentals

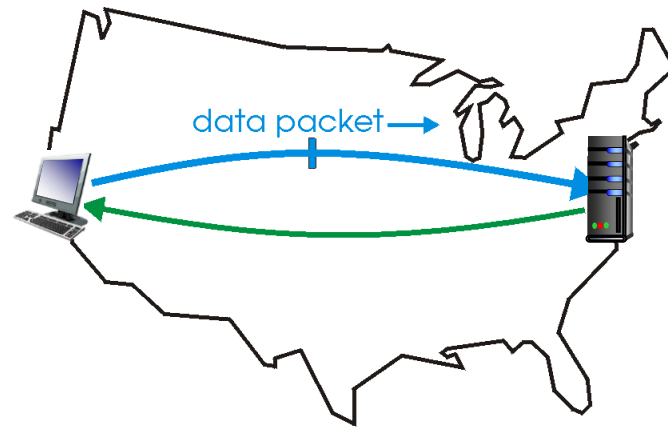
Transport Layer

The presentation was adapted from the textbook: *Computer Networking: A Top-Down Approach* 8th edition Jim Kurose, Keith Ross, Pearson, 2020

Redefine the Possible™

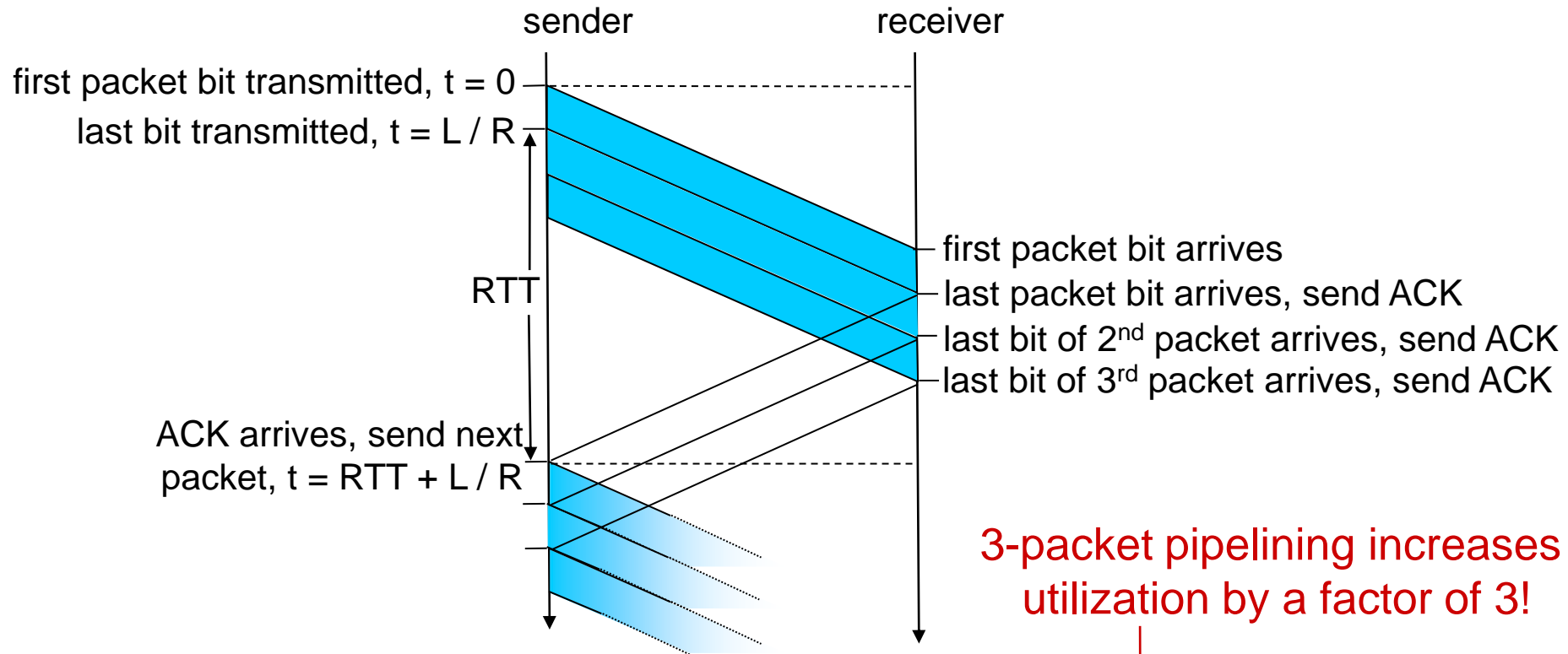
pipelining: sender allows multiple, “in-flight”, yet-to-be-acknowledged packets

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



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

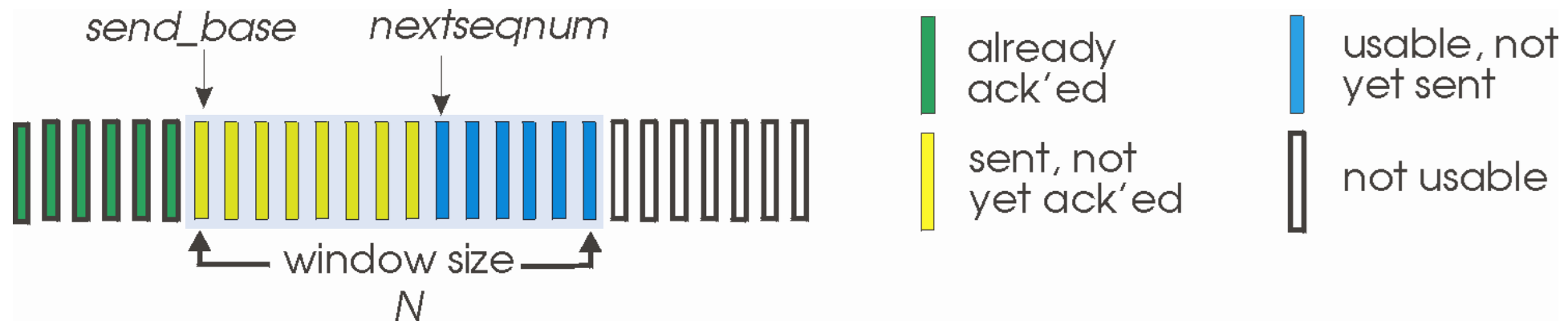
Pipelining: increased utilization



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

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

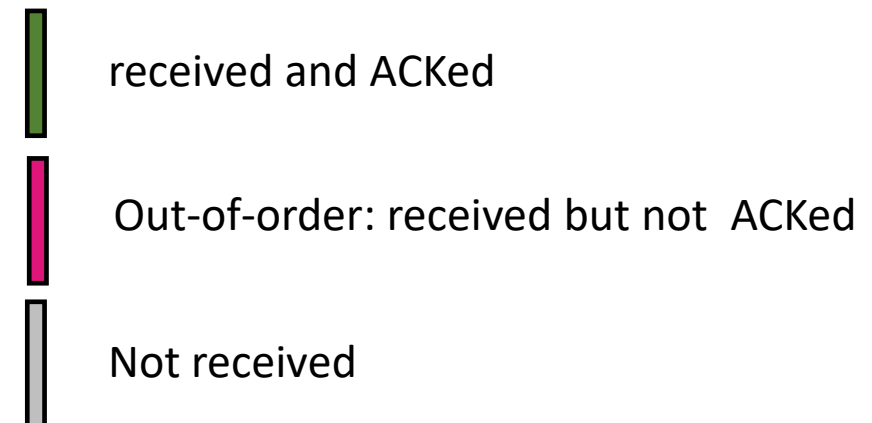
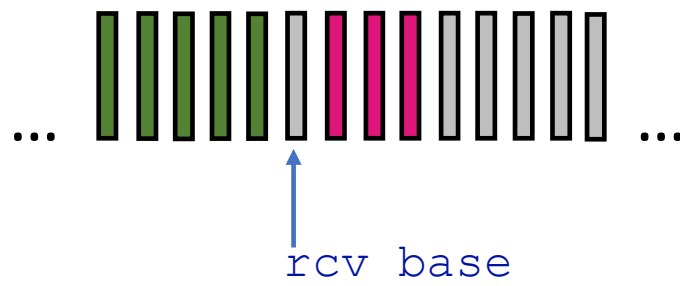
- sender: “window” of up to N , consecutive transmitted but unACKed pkts
 - k -bit seq # in pkt header



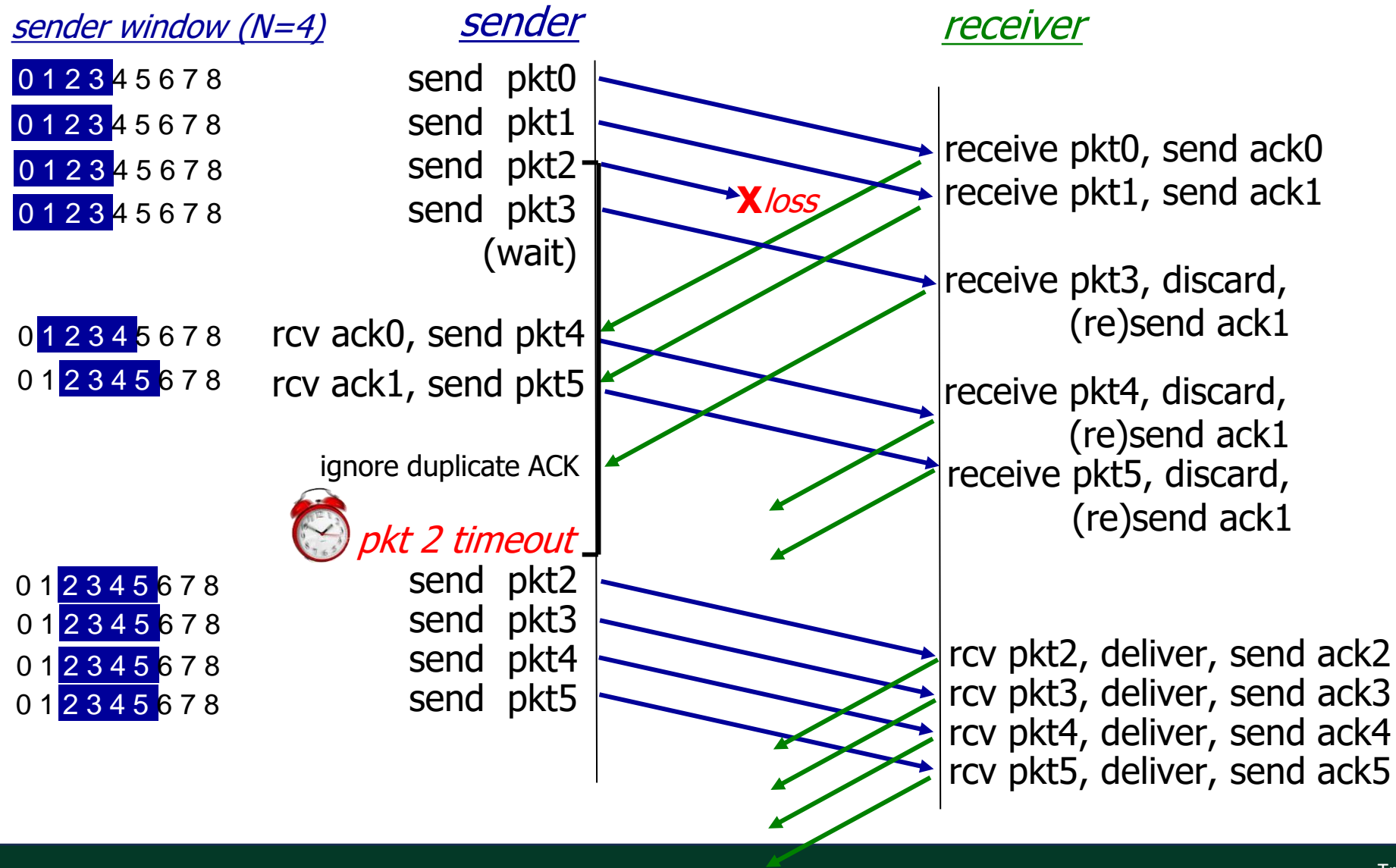
- **cumulative ACK:** $ACK(n)$: ACKs all packets up to, including seq # n
 - on receiving $ACK(n)$: move window forward to begin at $n+1$
- timer for oldest in-flight packet
- $timeout(n)$: retransmit packet n and all higher seq # packets in window

- ACK-only: always send ACK for correctly-received packet so far, with highest *in-order* seq #
 - may generate duplicate ACKs
 - need only remember `rcv_base`
- on receipt of out-of-order packet:
 - can discard (don't buffer) or buffer: an implementation decision
 - re-ACK pkt with highest in-order seq #

Receiver view of sequence number space:

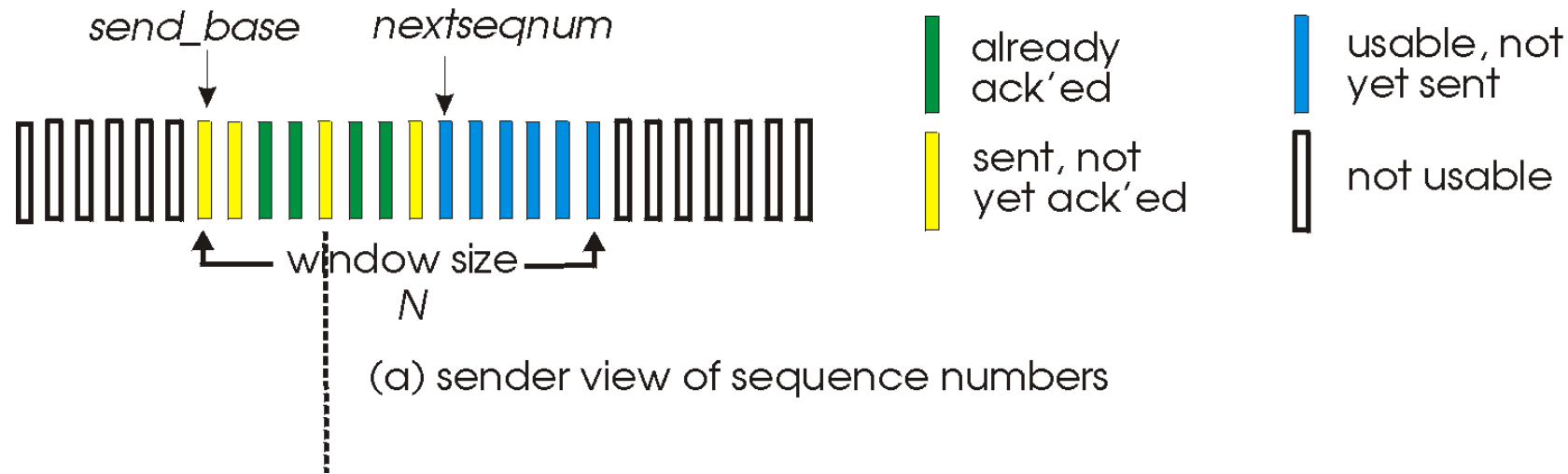


Go-Back-N in action



- receiver *individually* acknowledges all correctly received packets
 - buffers packets, as needed, for eventual in-order delivery to upper layer
- sender times-out/retransmits individually for unACKed packets
 - sender maintains timer for each unACKed pkt
- sender window
 - N consecutive seq #s
 - limits seq #s of sent, unACKed packets

Selective repeat: sender, receiver windows



Selective repeat: sender and receiver



sender

data from above:

- if next available seq # in window, send packet

timeout(n):

- resend packet n , restart timer

ACK(n) in [sendbase, sendbase+N-1]:

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

receiver

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

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

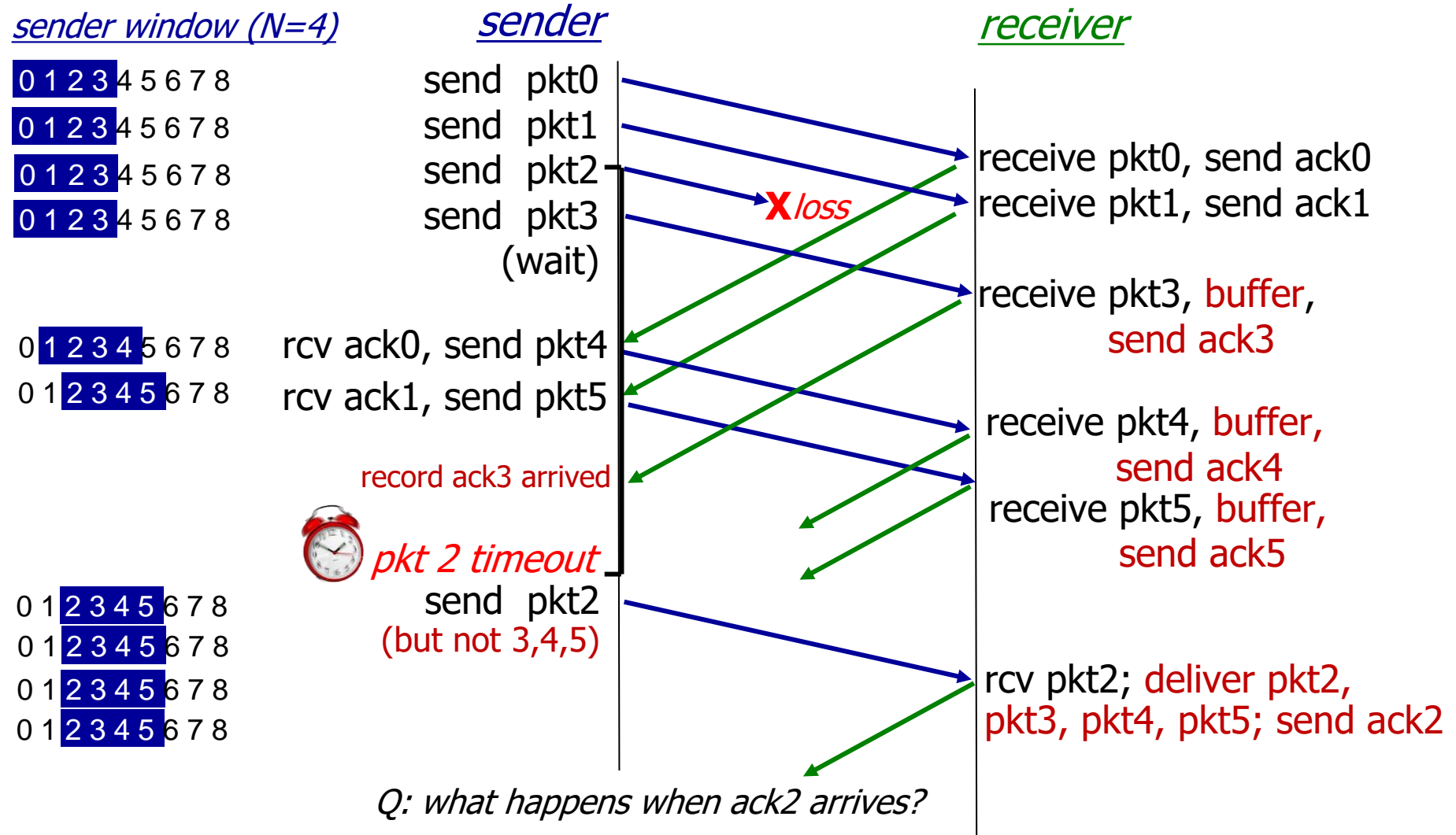
packet n in [rcvbase-N, rcvbase-1]

- ACK(n)

otherwise:

- ignore

Selective Repeat in action



Selective repeat: a dilemma!

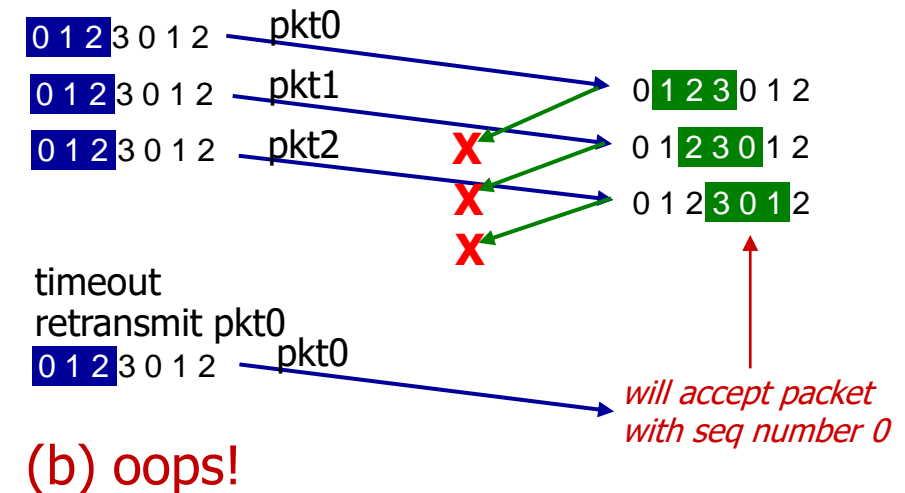
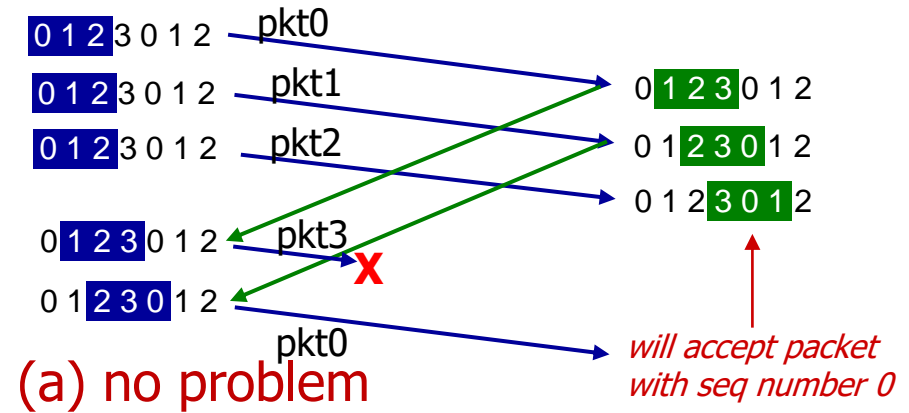


example:

- seq #s: 0, 1, 2, 3 (base 4 counting)
- window size=3

sender window
(after receipt)

receiver window
(after receipt)



Selective repeat: a dilemma!



example:

- seq #s: 0, 1, 2, 3 (base 4 counting)
- window size=3

Q: what relationship is needed between sequence # size and window size to avoid problem in scenario (b)?

