Transport Reliability:

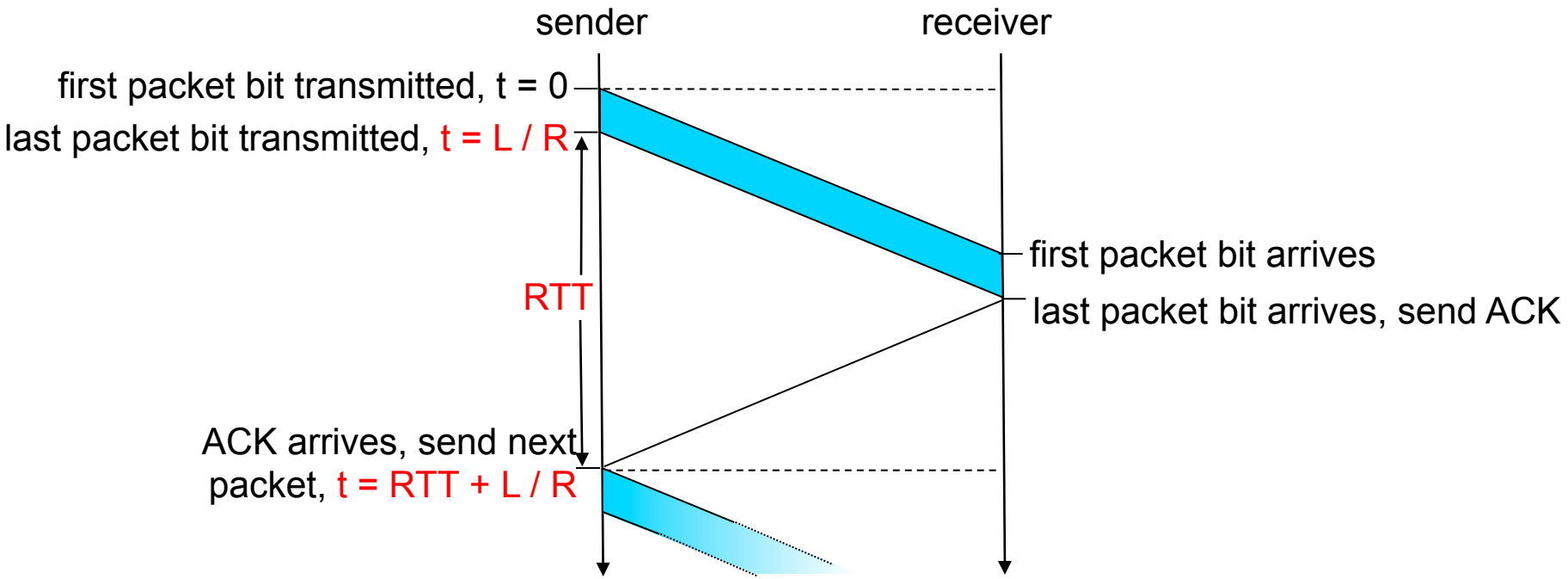
Sliding Window Protocols, Connection Management

Recap: Services Transport Layer Could Provide

- Multiplexing/demultiplexing
- Reliable data transfer
- Flow control
- Congestion control

rdt3.0: Stop-and-Wait Operation

- rdt3.0 works, but performance stinks
- example: 1 Gbps link, 15 ms e-e prop. delay, 1KB packet:



$$U_{sender} = \frac{L / R}{RTT + L / R} = \frac{.008}{30.008} = 0.027\%$$

A Summary of Questions

- How to improve the performance of rdt3.0?
- What if there are reordering and duplication?
- How to determine the “right” timeout value?

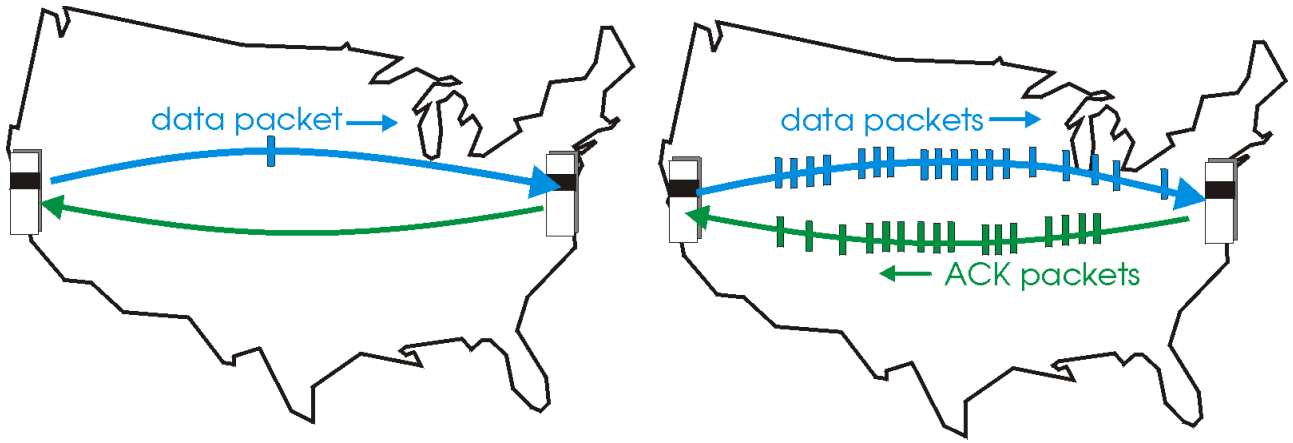
Outline

- Recap
- Sliding window protocols
  - Go-back-n
  - Selective repeat
- Connection management

Sliding Window Protocols: Pipelining

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

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

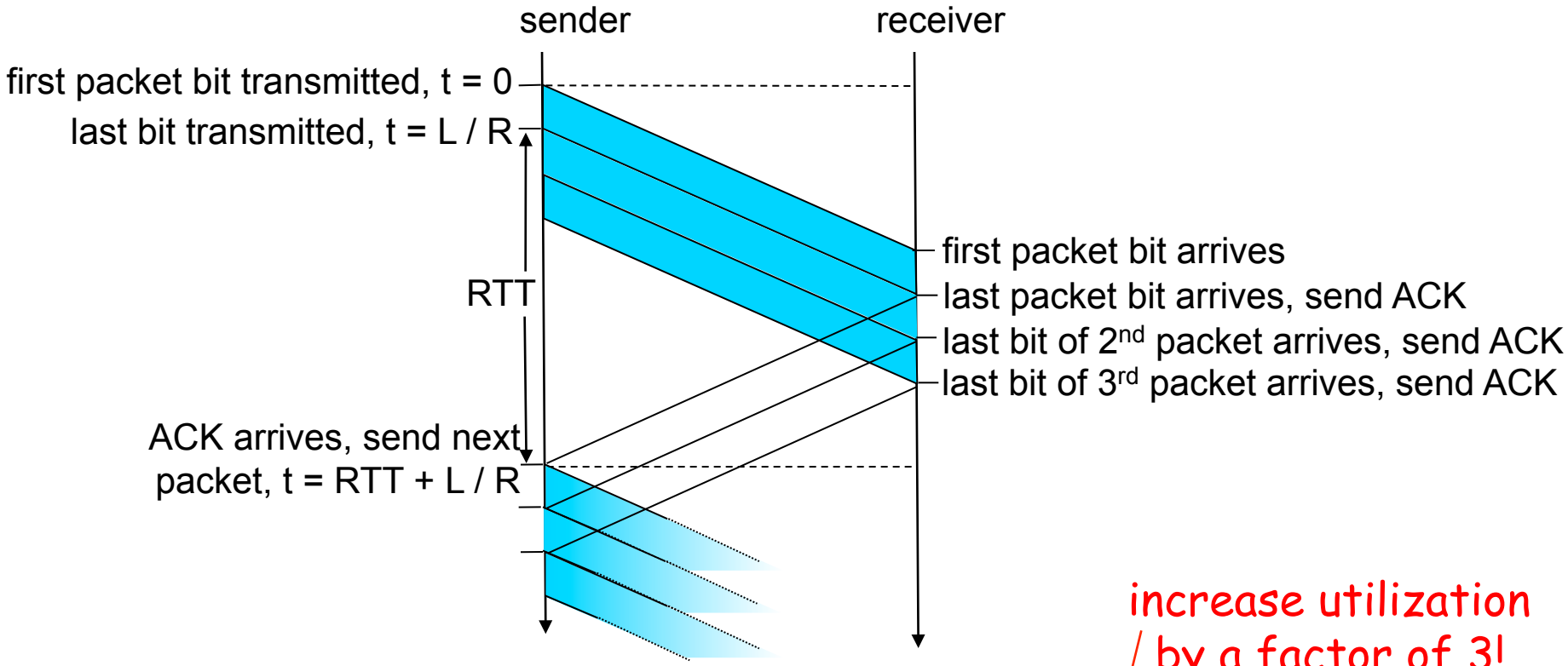


(a) a stop-and-wait protocol in operation

(b) a pipelined protocol in operation

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

Pipelining: Increased Utilization



increase utilization by a factor of 3!

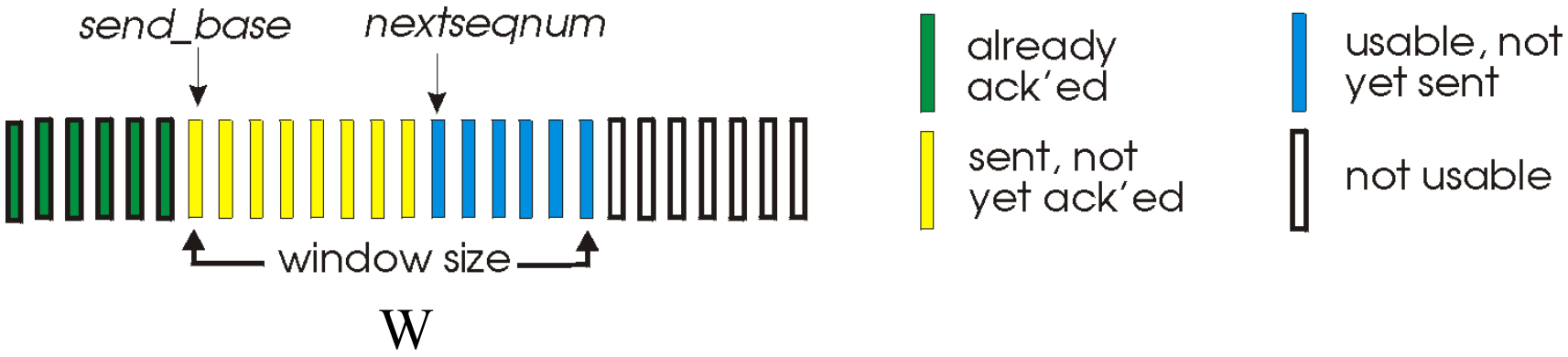
$$U_{sender} = \frac{3 * L / R}{RTT + L / R} = \frac{.024}{30.008} = 0.0008$$

Question: a rule-of-thumb window size?

Go-Back-n

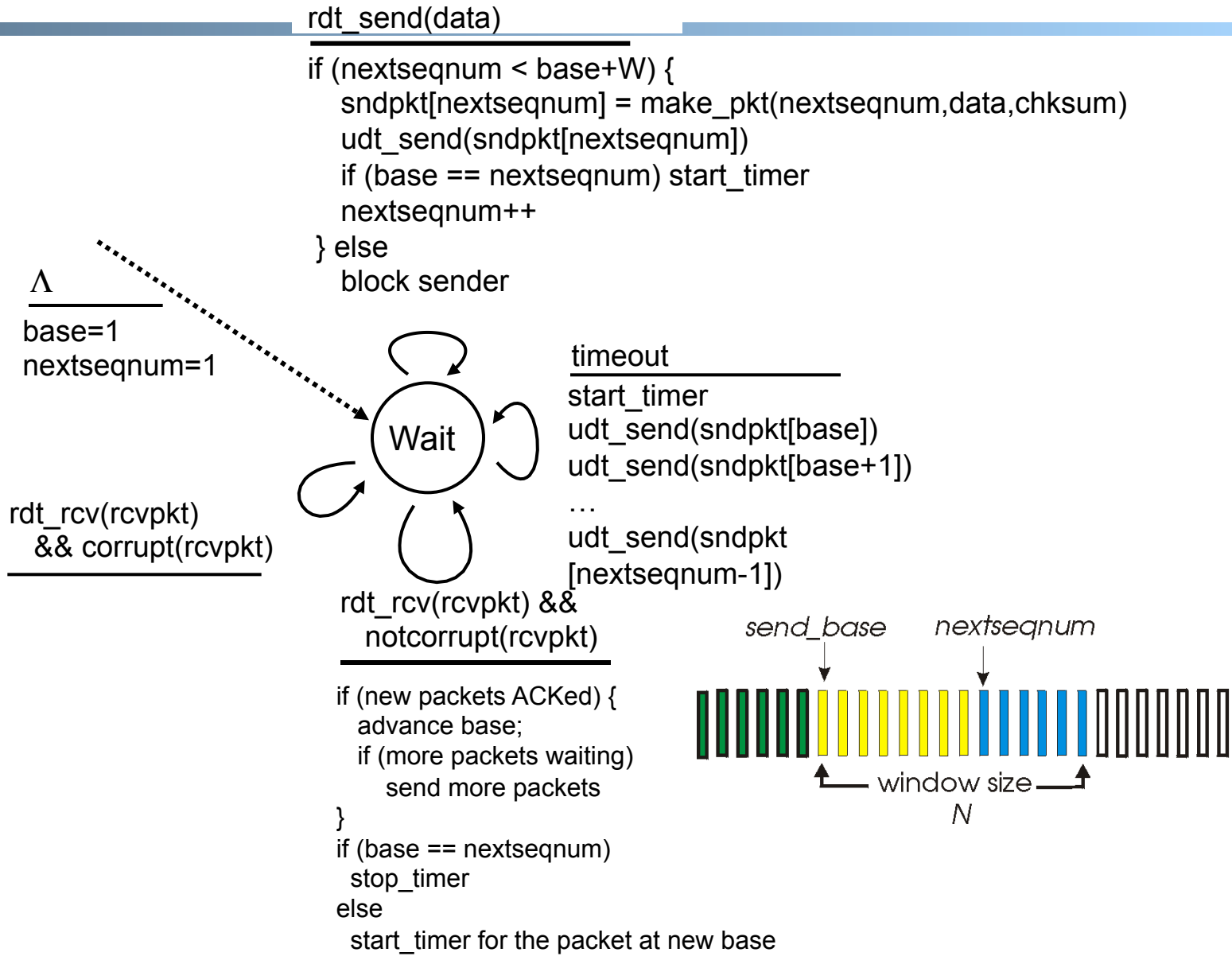
Sender:

- k-bit seq # in pkt header
- "window" of up to W, consecutive unack'ed pkts allowed

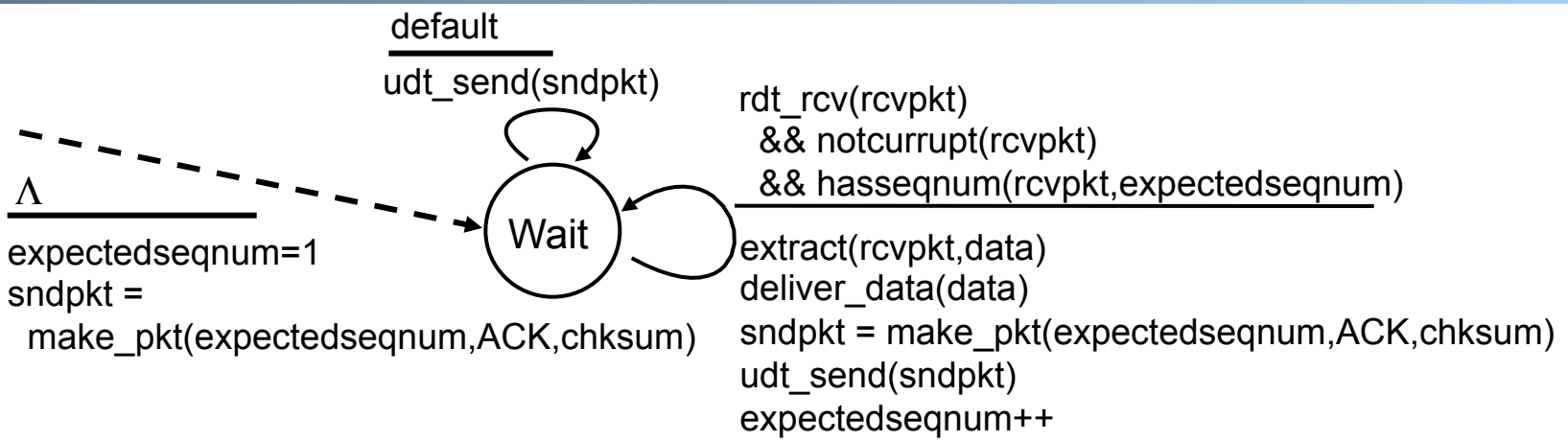


- ACK(n): ACKs all pkts up to, including seq # n - "cumulative ACK"
  - Note: ACK(n) could mean two things: I have received upto and include n, or I am waiting for n+1
- Timer for the packet at base
- *timeout(n):* retransmit pkt n and all higher seq # pkts in window

GBN: Sender Extended FSM



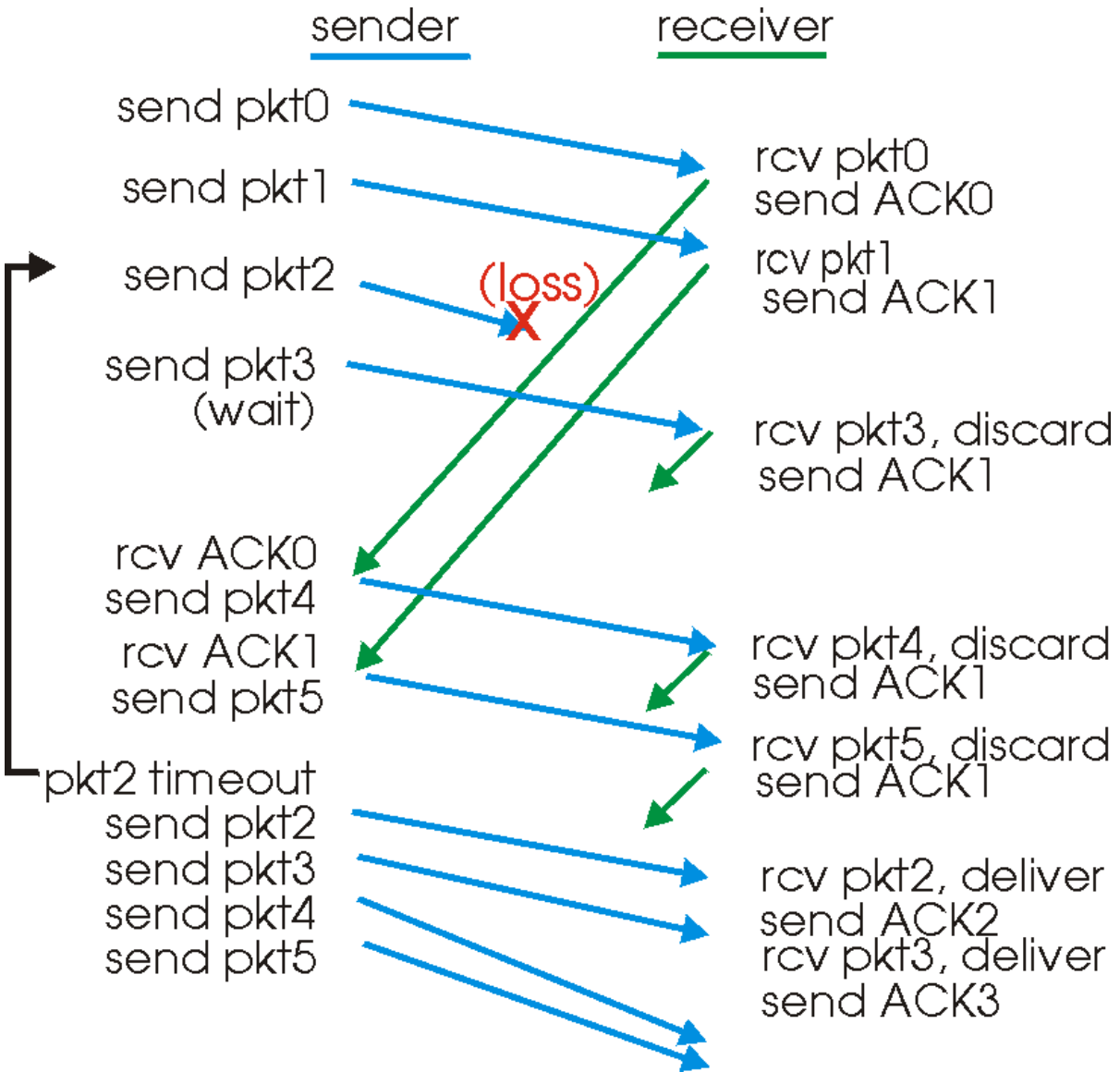
GBN: Receiver Extended FSM



- ACK-only: always send ACK for correctly-received pkt with highest in-order seq #
  - Need only remember **expectedseqnum**
- Out-of-order pkt:
  - Discard (don't buffer) -> no receiver buffering!
  - Re-ACK pkt with highest in-order seq #
  - May generate duplicate ACKs

GBN in Action

window size = 4



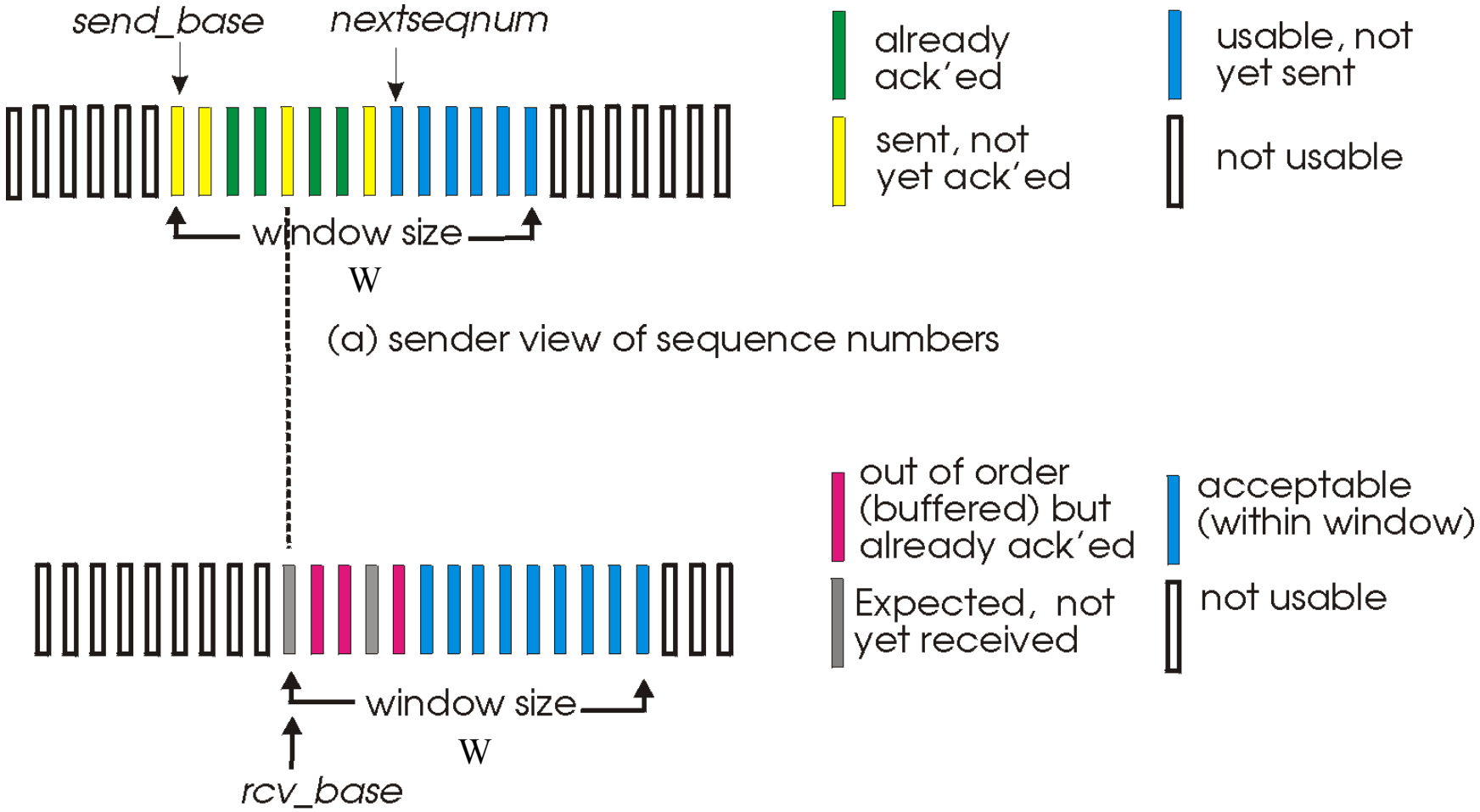
Discussion: Efficiency of Go-Back-n

- Assume window size W
- Assume each packet is lost with probability p
- On average, how many packets do we send for each data packet?
  - At least the packet itself: 1
  - Any one of W failed, we need resend, i.e., p*w
  - Again, any resent packet lost, need resend, i.e. p^2*w
  - …
  - Total: $1 + pw + p^2w + p^3w + \ldots = 1 + pw(1 + p + p^2 + \ldots) = 1 + \frac{pw}{1-p}$

Selective Repeat

- Sender window
  - Window size W: W consecutive unACKed seq #'s
- Receiver *individually* acknowledges correctly received pkts
  - Buffers pkts as needed, for eventual in-order delivery to upper layer
  - ACK(n) means received packet with seq# n only
  - Buffer size at receiver: window size
- Sender only resends pkts for which ACK not received
  - Sender timer for each unACKed pkt

Selective Repeat: Sender, Receiver Windows



Selective Repeat

sender

data from above :

- unACKed packets is less than window size W, send; otherwise block app.

timeout(n):

- resend pkt n, restart timer

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

- mark pkt n as received
- update sendbase to the first packet unACKed

receiver

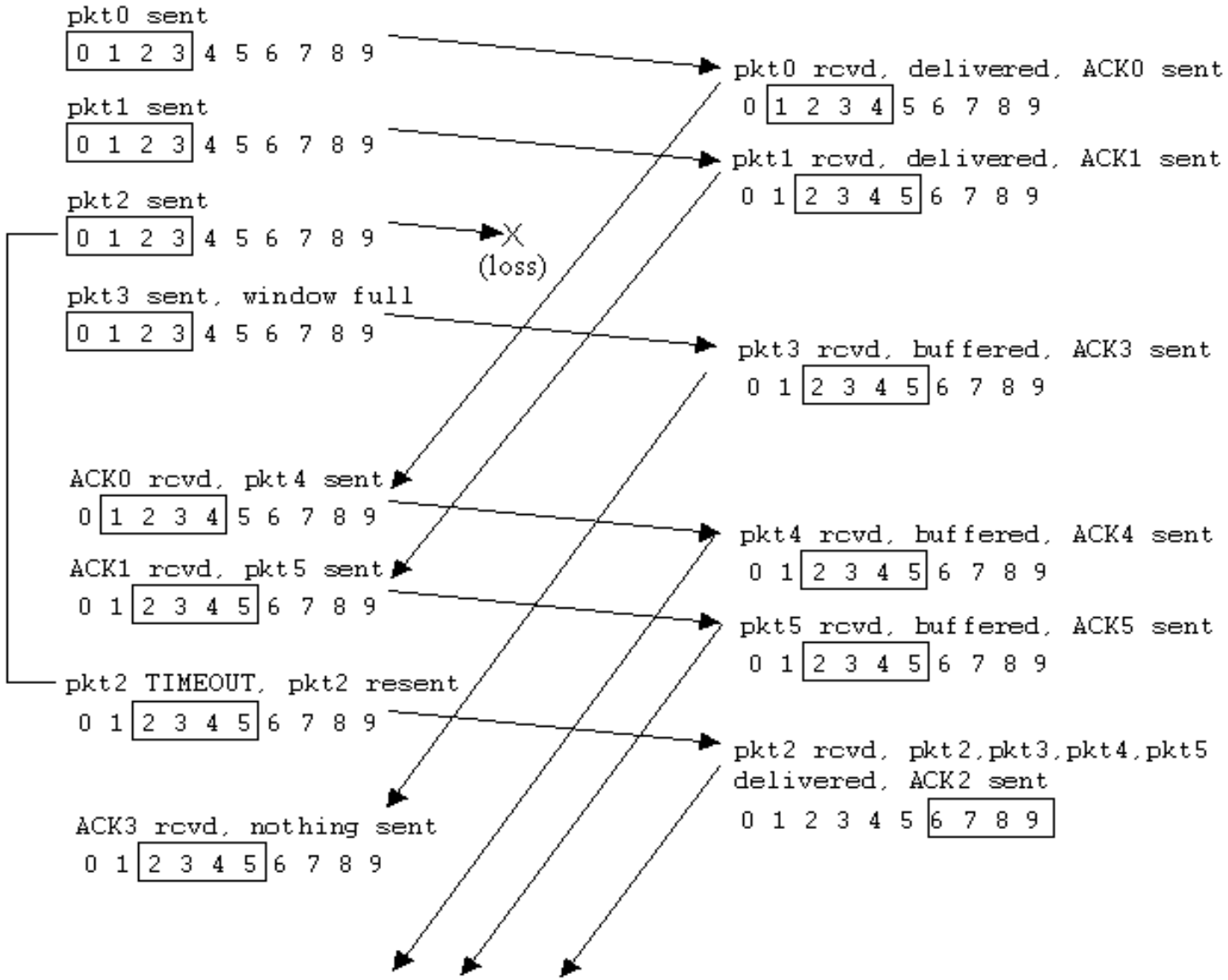
pkt n in [rcvbase, rcvbase+W-1]

- send ACK(n)
- if (out-of-order)
  mark and buffer pkt n
  else /*in-order*/
  deliver any in-order packets

otherwise:

- ignore

Selective Repeat in Action

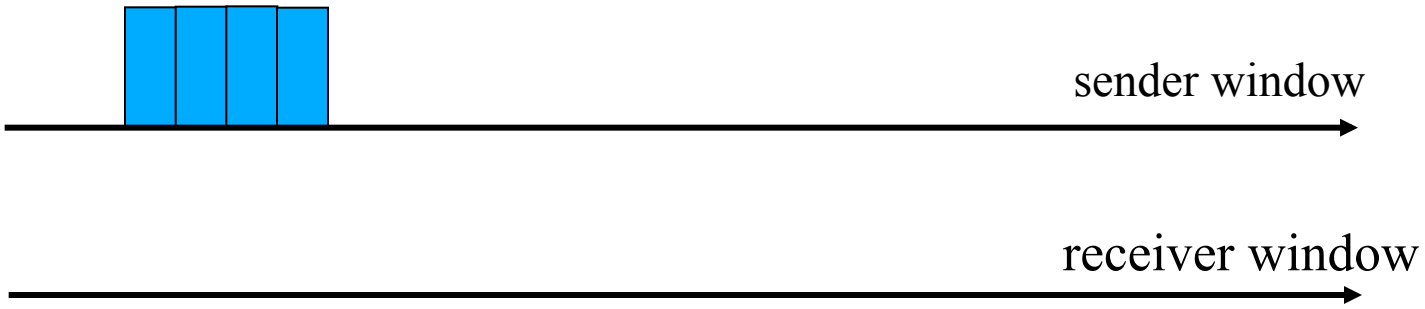


Discussion: Efficiency of Selective Repeat

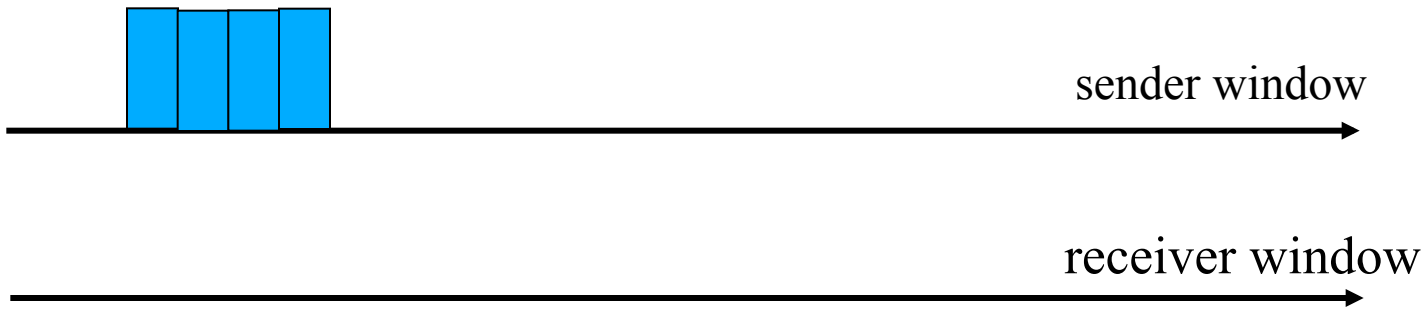
- Assume window size w
- Assume each packet is lost with probability p
- On average, how many packets do we send for each data packet?
  - $1+p+p^2+\ldots=1/(1-p)$

State Invariant: Window Location

- Go-back-n (GBN)



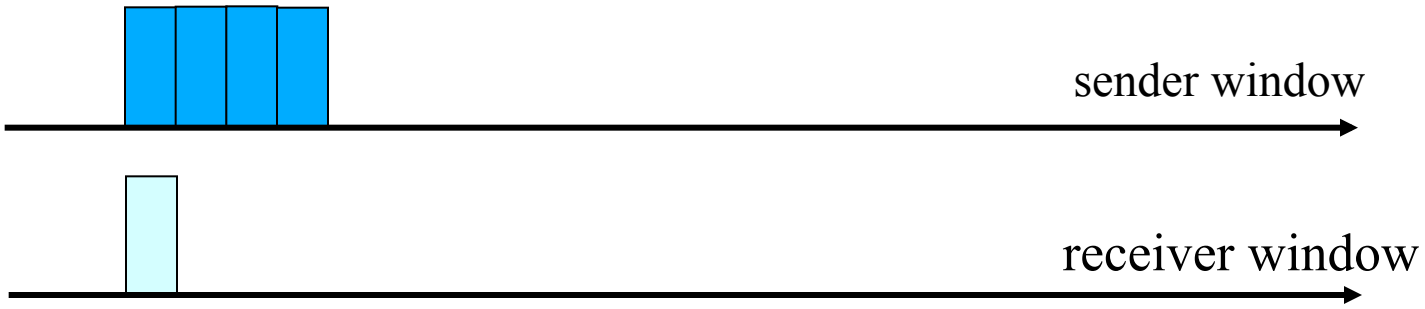
- Selective repeat (SR)



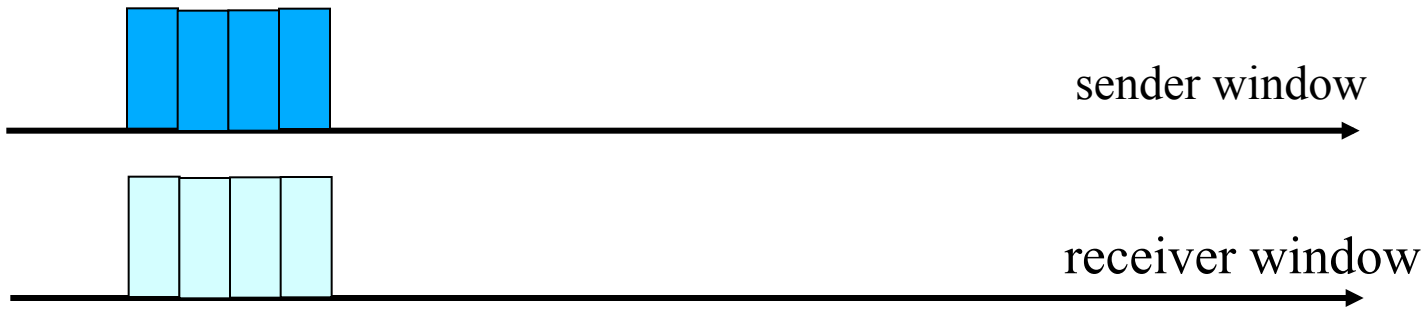
Window Location

Q: what relationship between seq # size and window size?

- Go-back-n (GBN)



- Selective repeat (SR)

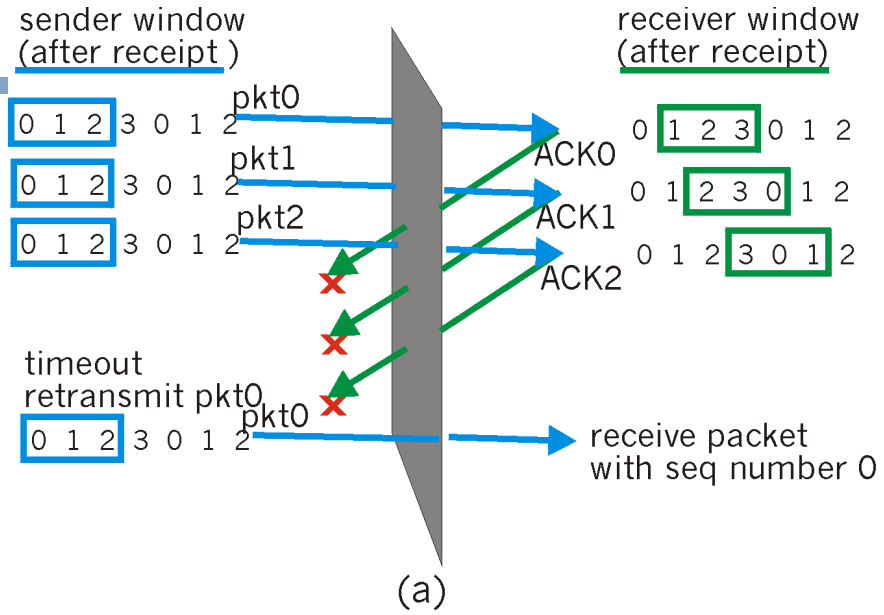


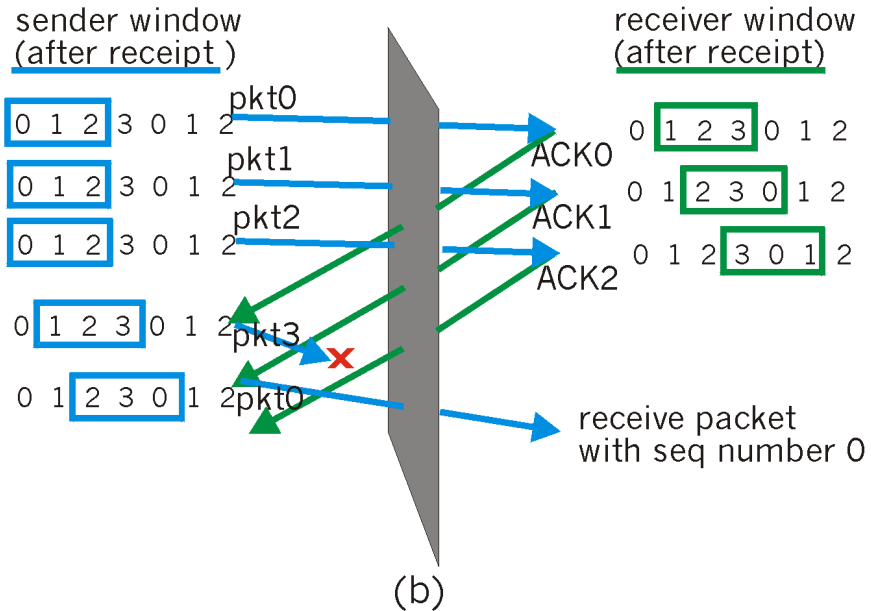
Selective Repeat: Seq# Ambiguity

Example:

- seq #'s: 0, 1, 2, 3
- window size=3

- receiver sees no difference in two scenarios!
- incorrectly passes duplicate data as new in (a)



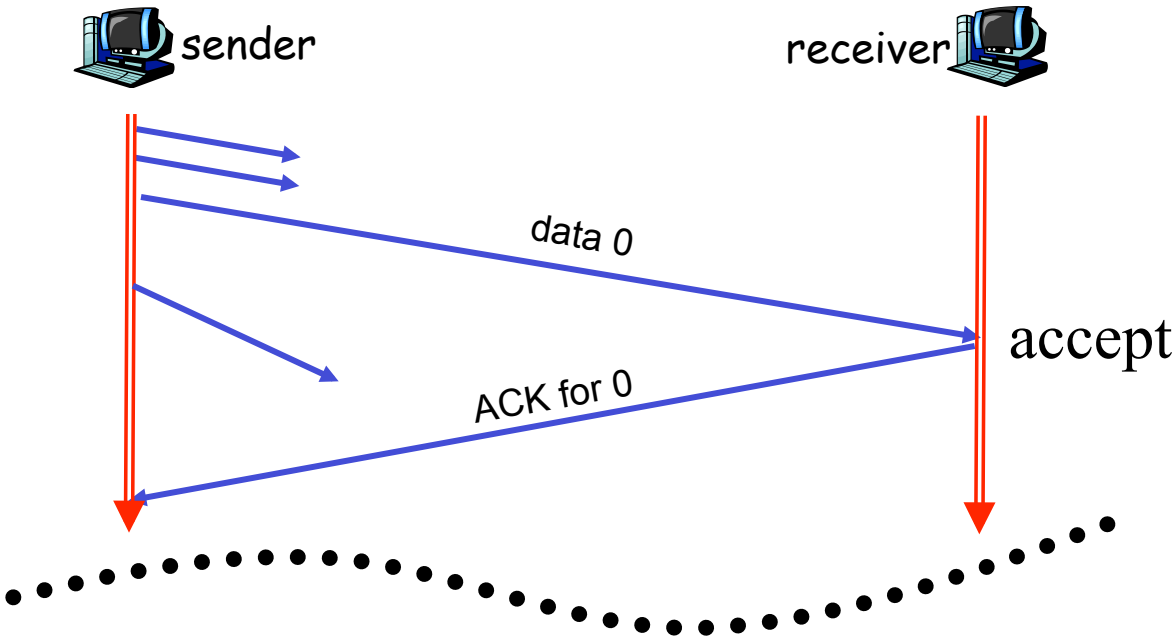


Sliding Window Protocols: Go-back-n and Selective Repeat

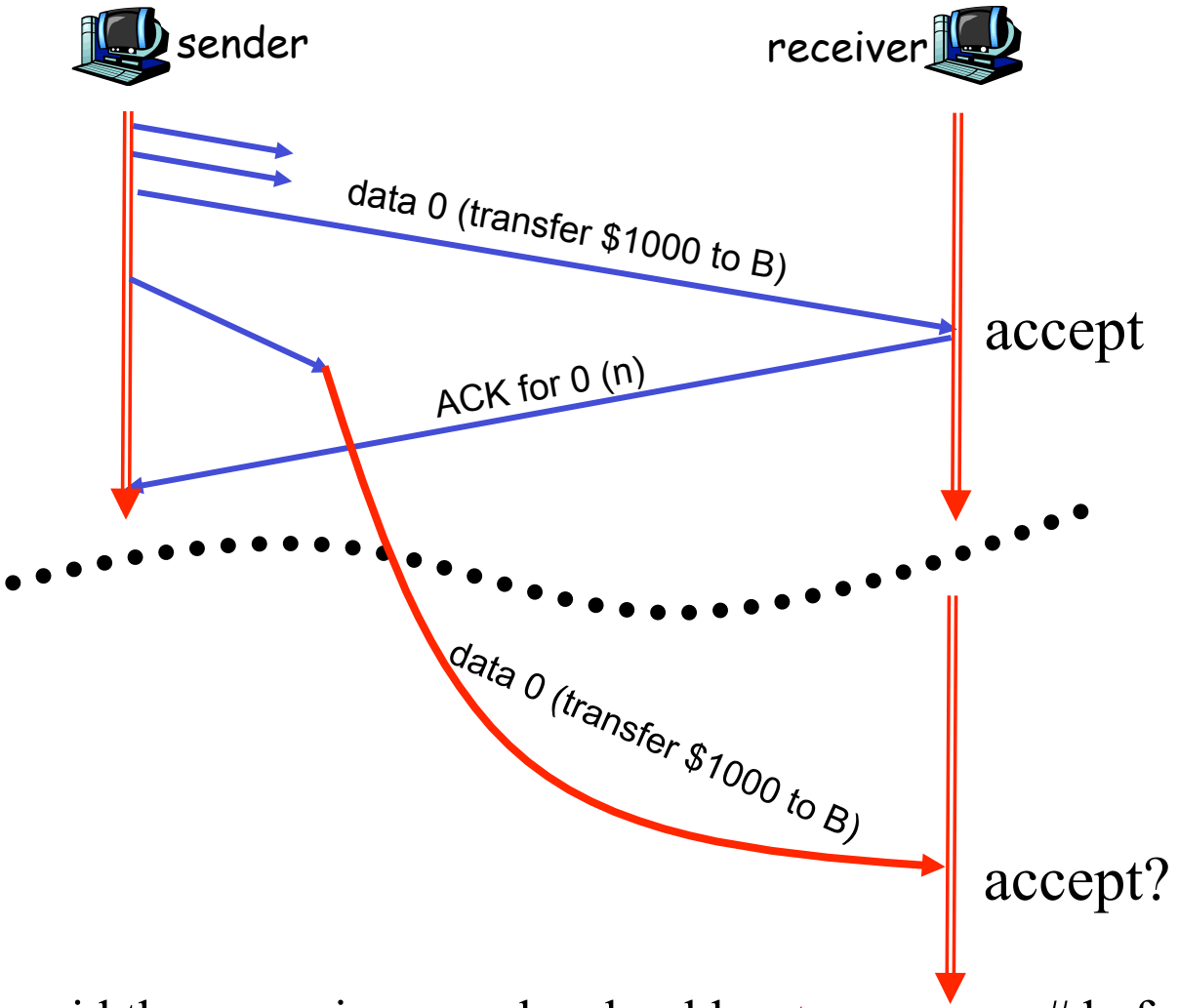
	Go-back-n	Selective Repeat
data bandwidth: sender to receiver (avg. number of times a pkt is transmitted)	Less efficient $\frac{1-p+pw}{1-p}$	More efficient $\frac{1}{1-p}$
ACK bandwidth (receiver to sender)	More efficient	Less efficient
Relationship between M (the number of seq#) and W (window size)	$M > W$	$M \geq 2W$
Buffer size at receiver	1	W
Complexity	Simpler	More complex

p: the loss rate of a packet; M: number of seq# (e.g., 3 bit M = 8); W: window size

Question: What is Initial Seq#?



Question: What is Initial Seq#?



- To avoid the scenario, a sender should *not* reuse a seq# before it is sure the packet has left the network

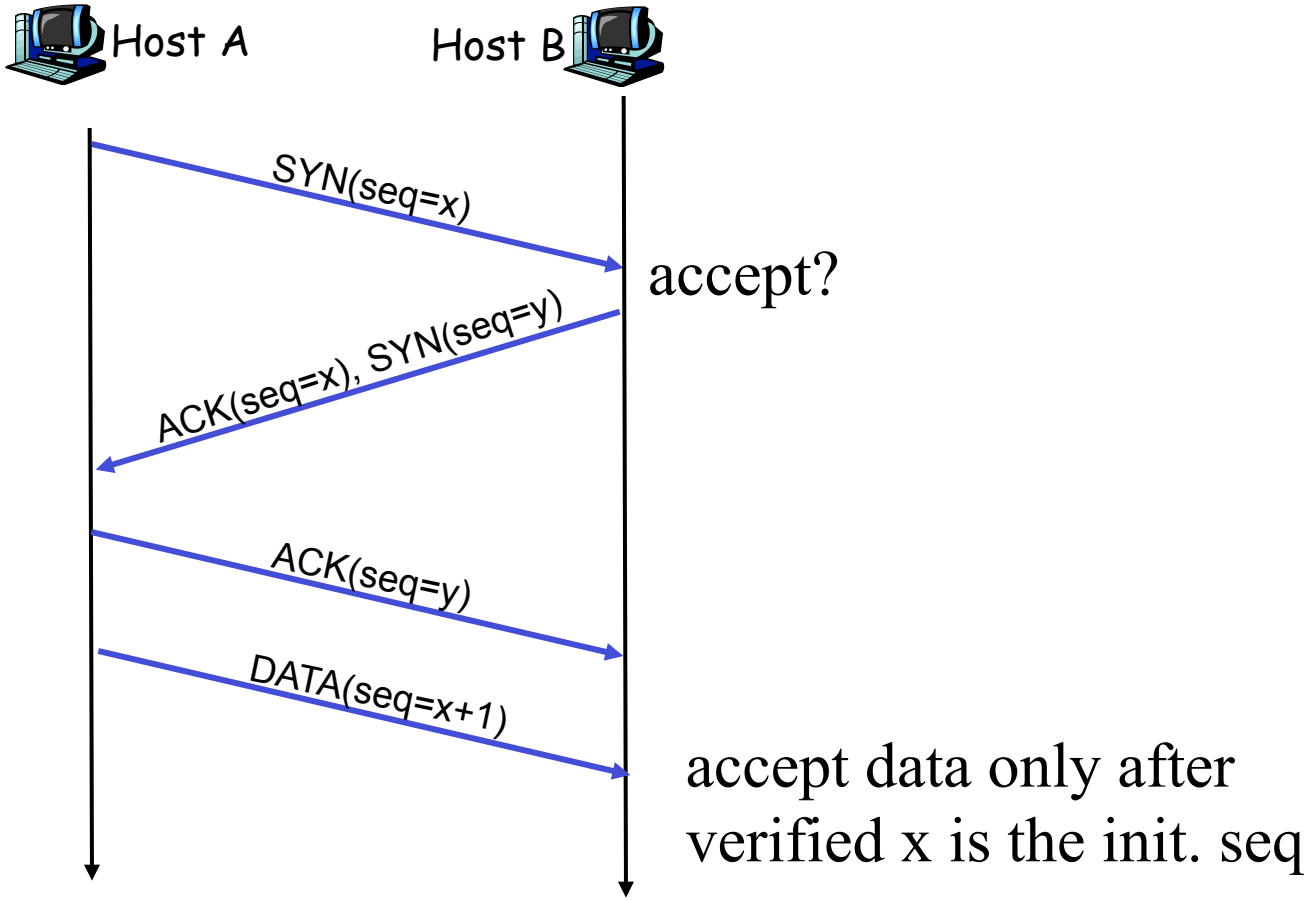
Outline

- Recap
- Sliding window protocols
  - Go-back-n
  - Selective repeat
- Connection management

Connection Management: Objective

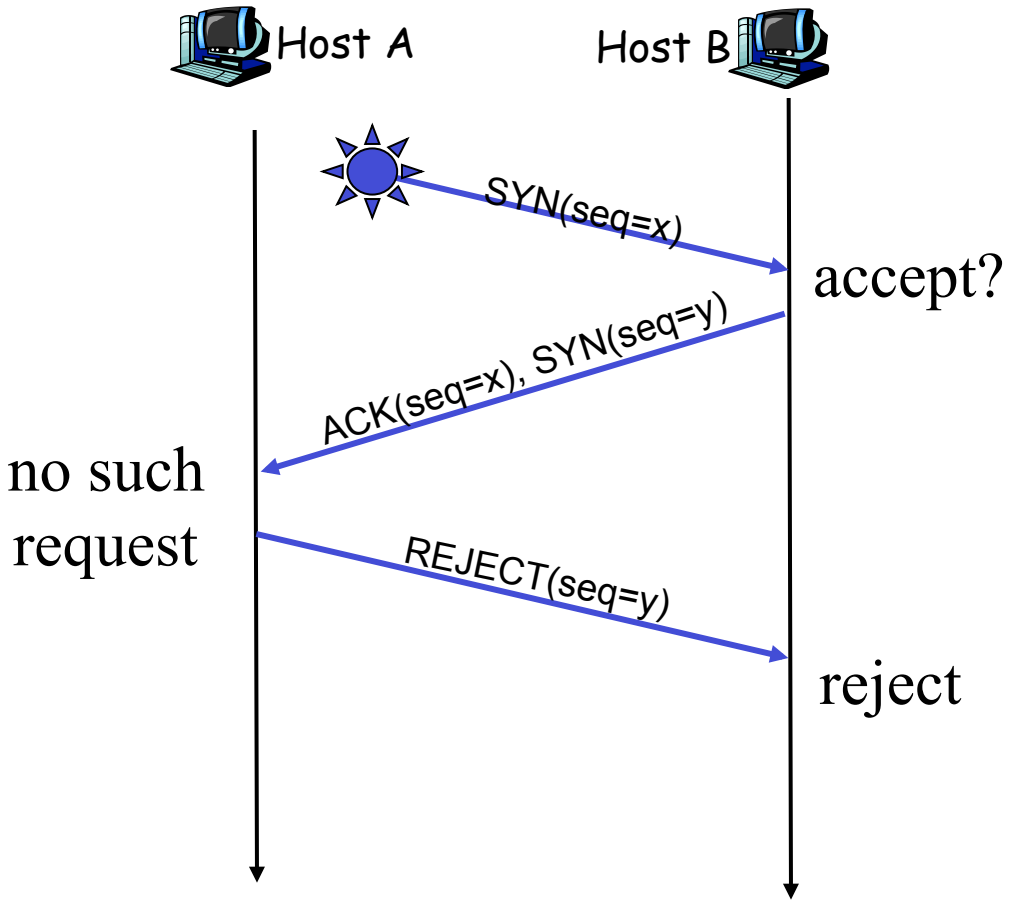
- Agree on initial sequence numbers
  - A sender will not reuse a seq# before it is sure that all packets with the seq# are purged from the network
    - The network guarantees that a packet too old will be purged from the network: network bounds the life time of each packet
  - To avoid waiting for the seq# to start a session, use a larger seq# space
    - Needs connection setup so that the sender tells the receiver initial seq#
- Agree on other initial parameters

Three Way Handshake (TWH) [Tomlinson 1975]



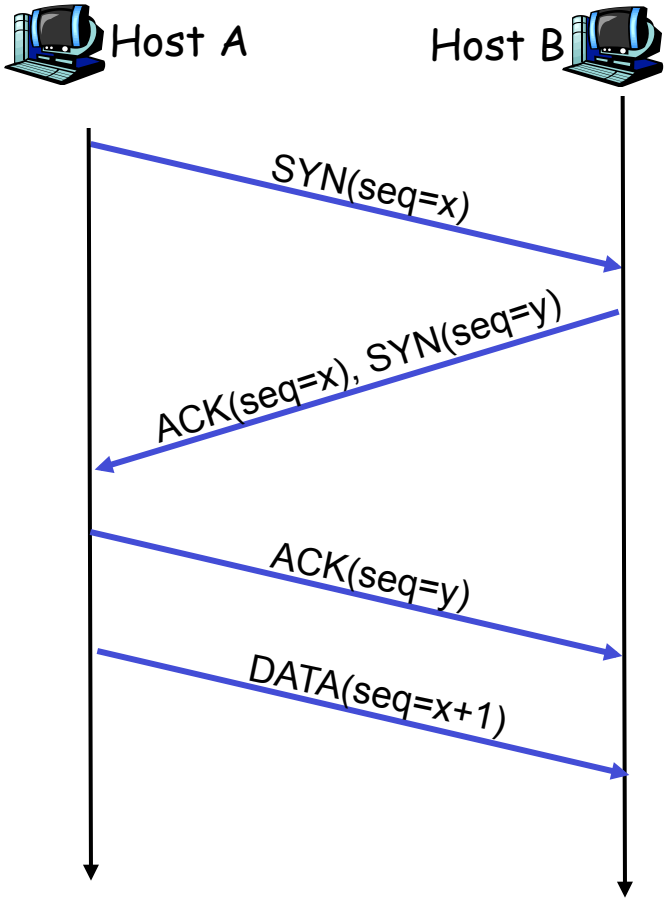
SYN: indicates connection setup

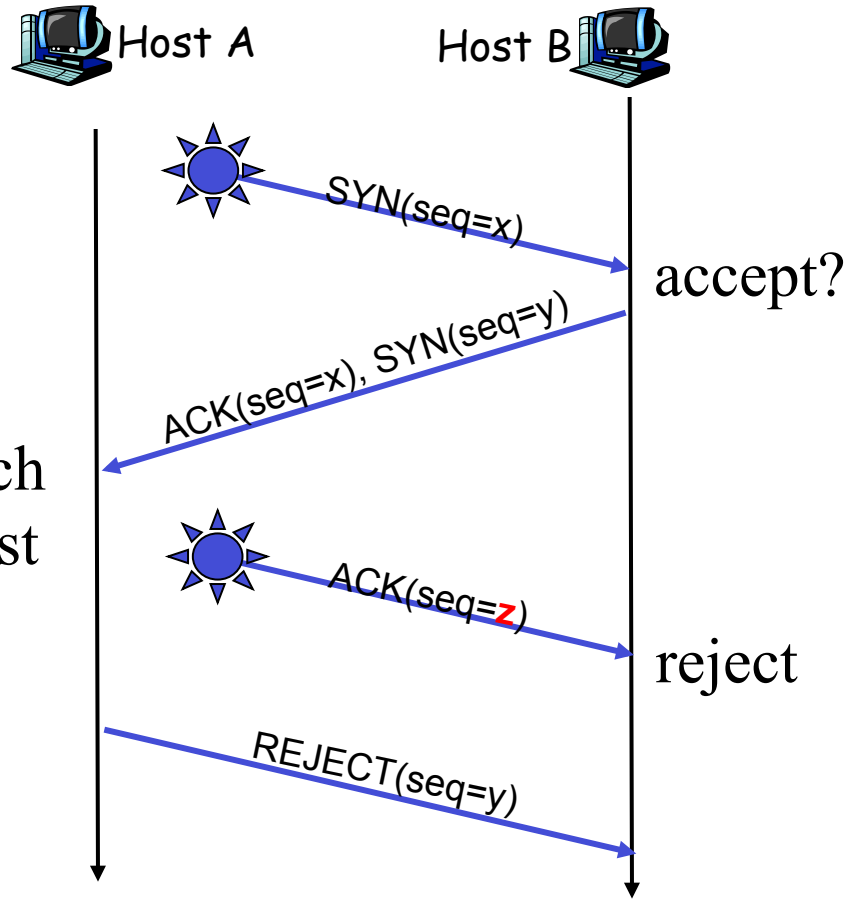
Scenarios with Duplicate Request



Three Way Handshake (TWH) [Tomlinson 1975]

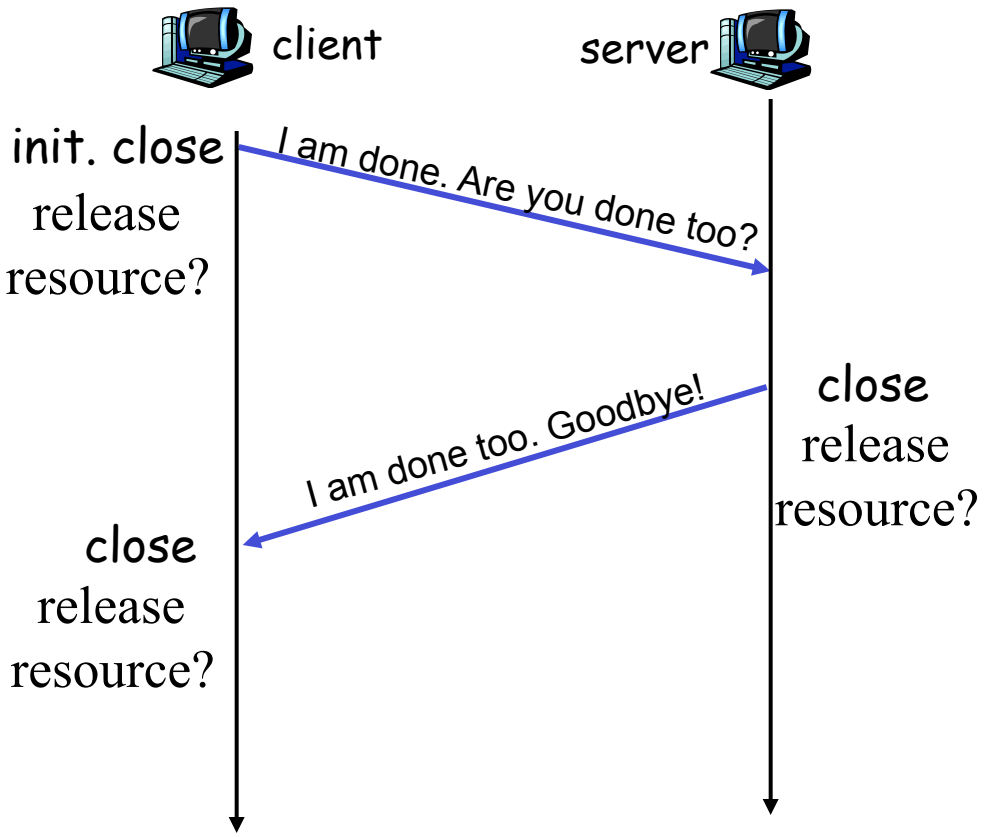
- To ensure that the other side does want to send a request



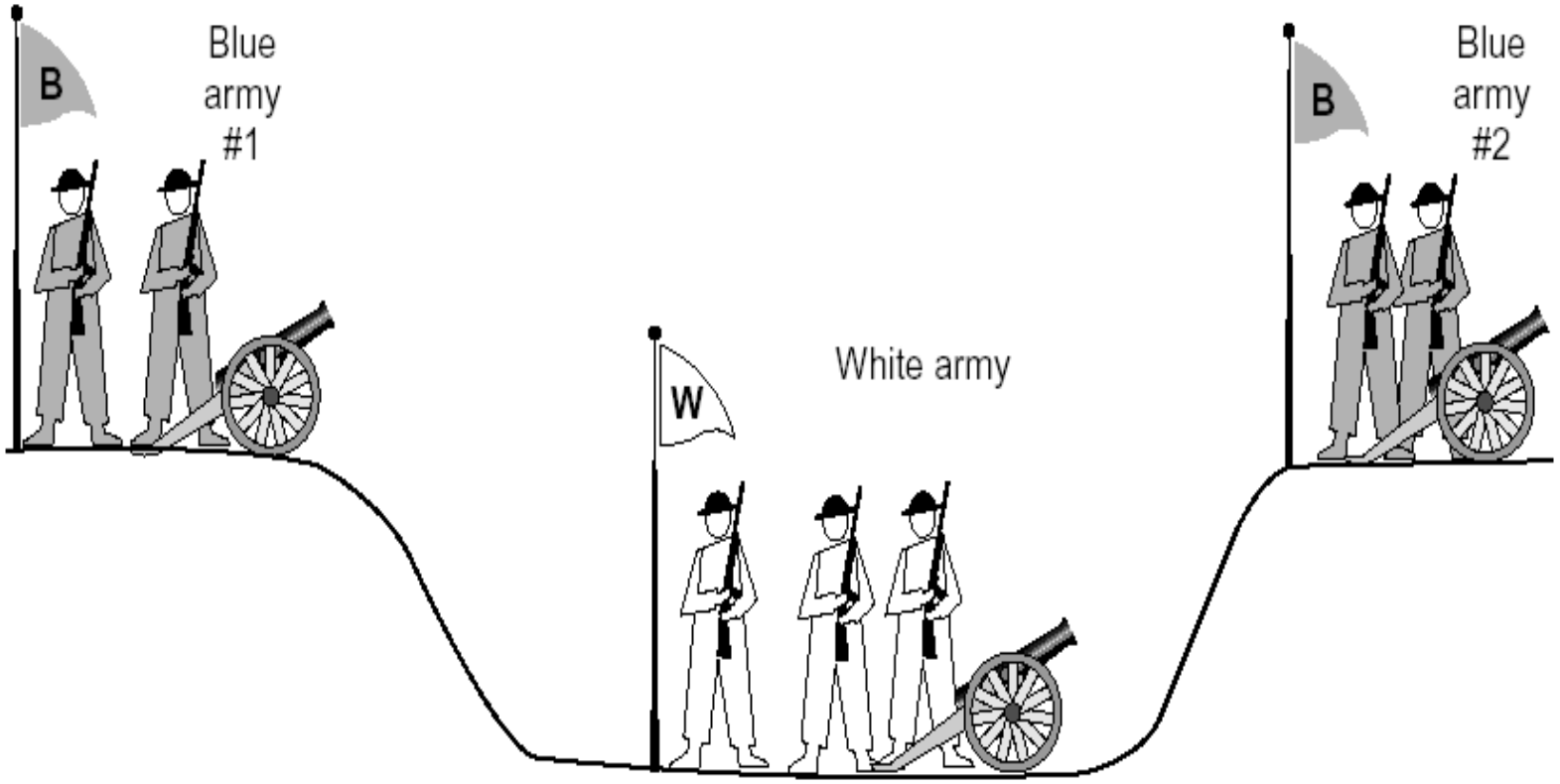


Connection Close

- Objective of closure handshake:
  - each side can release resource and remove state about the connection

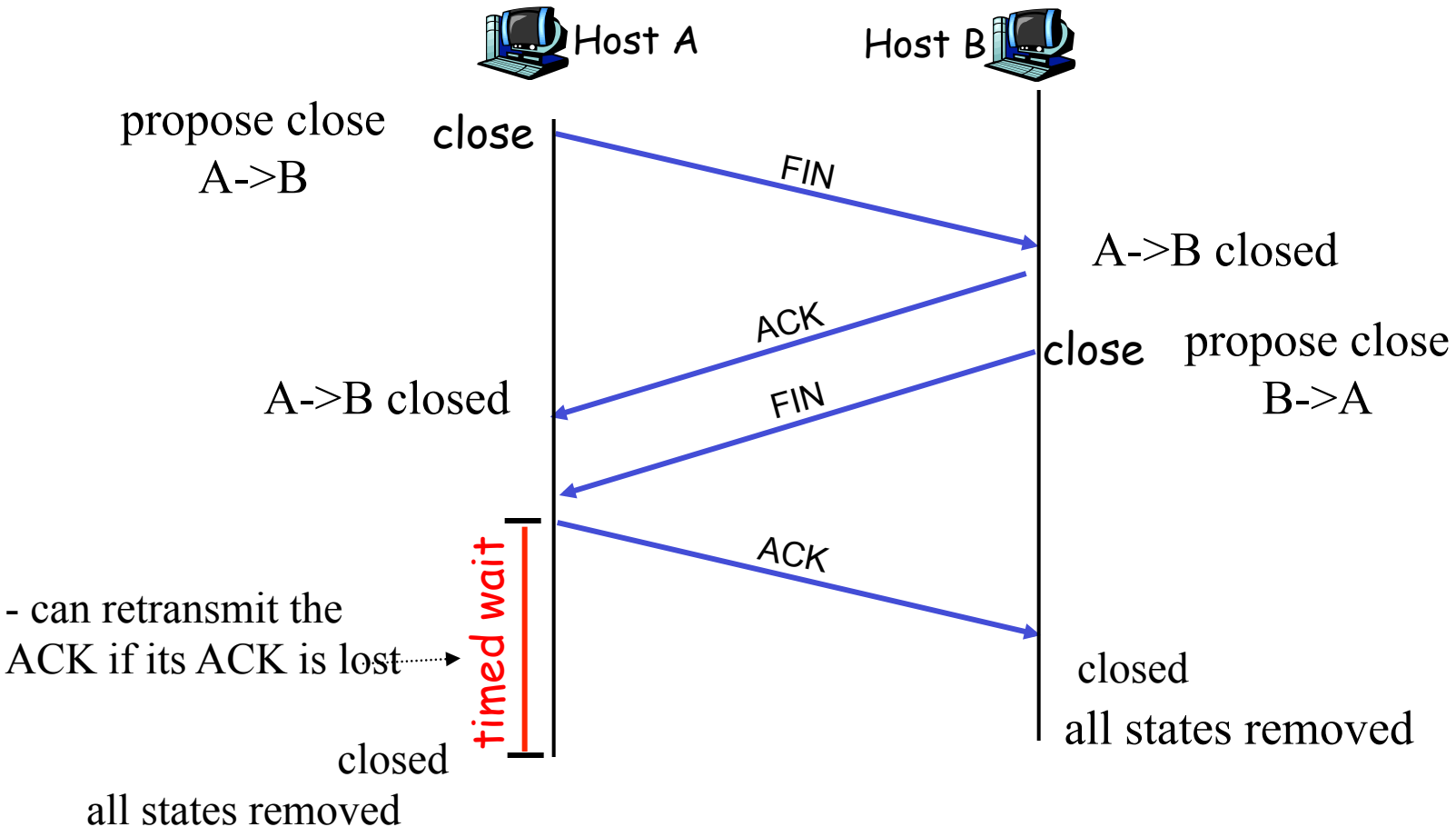


General Case: The Two-Army Problem



The two blue armies need to agree on whether or not they will attack the white army. They achieve agreement by sending messengers to the other side. If they both agree, attack; otherwise, no. Note that a messenger can be captured!

Four Way Teardown



A Summary of Questions

- How to improve the performance of rdt3.0?
  - Sliding window protocols
- What if there are duplication and reordering?
  - Network guarantee: max packet life time
  - Transport guarantee: not reuse a seq# before life time
  - Seq# management and connection management
- How to determine the "right" parameters?