

CS 43: Computer Networks

TCP

Kevin Webb

Swarthmore College

March 1, 2022

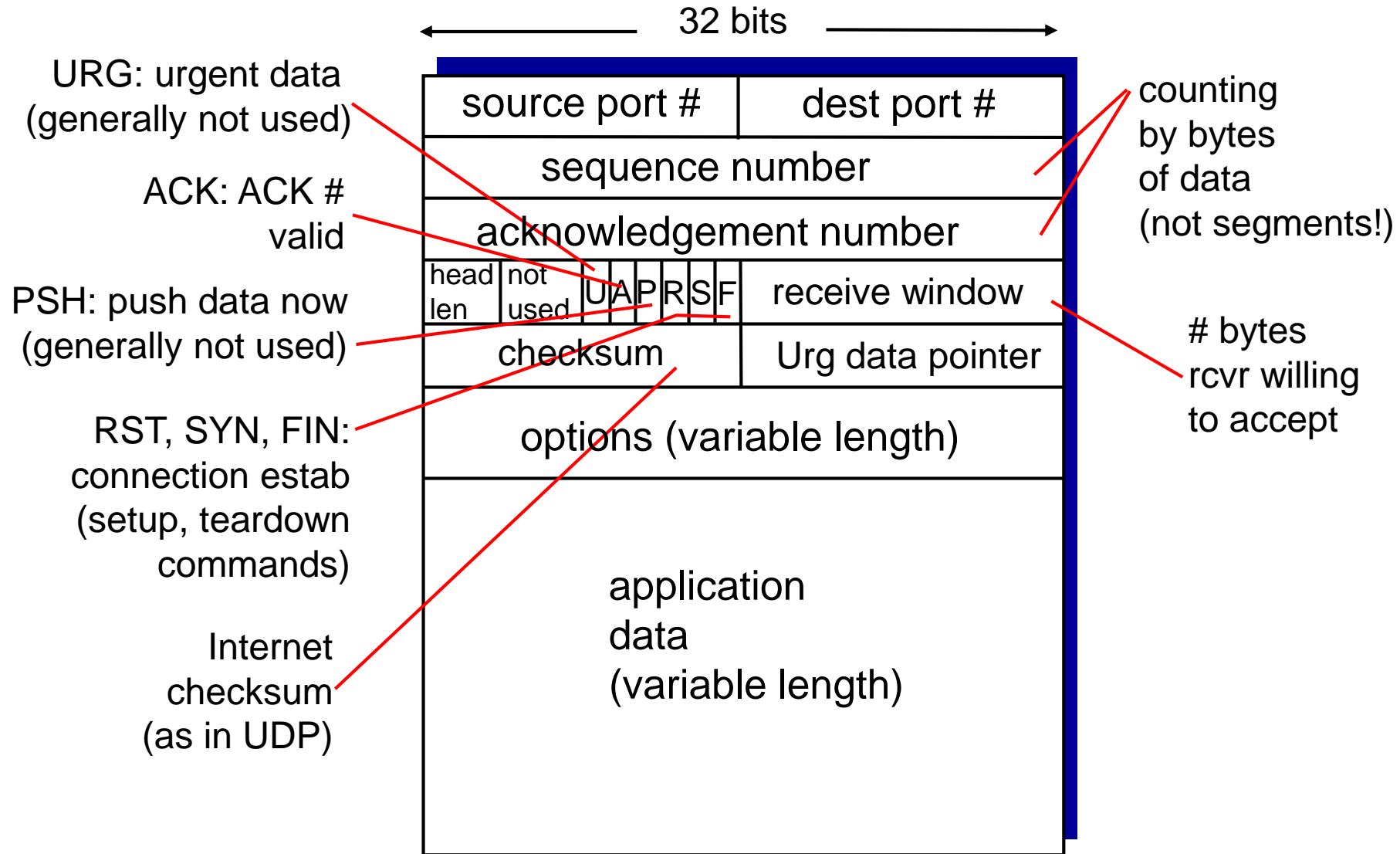
Practical Reliability Questions

- How do the sender and receiver keep track of outstanding pipelined segments?
- How many segments should be pipelined?
- How do we choose sequence numbers?
- What does connection establishment look like?
- How should we choose timeout values?

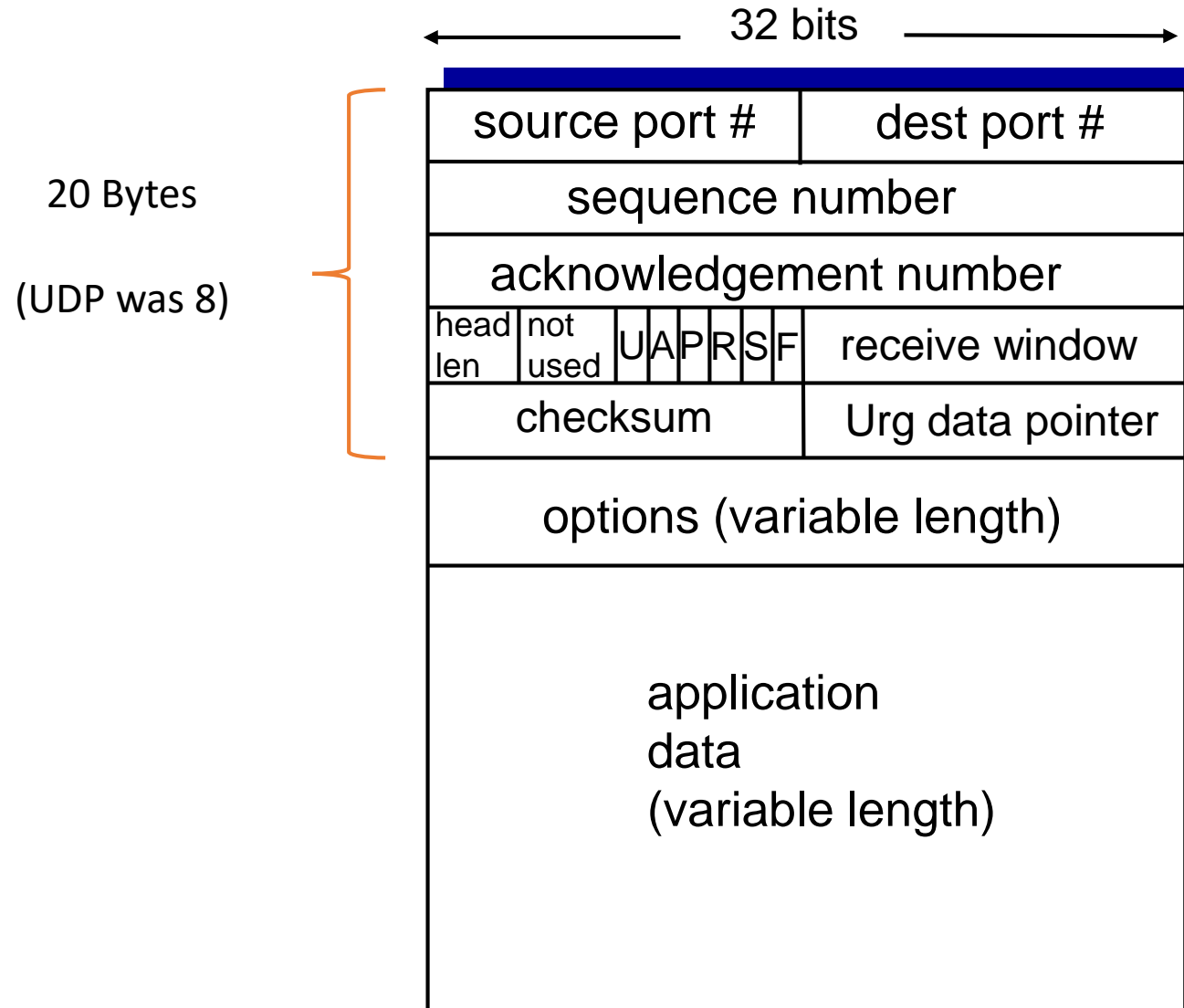
TCP Overview

- Point-to-point, full duplex
 - One pair of hosts
 - Messages in both directions
- Reliable, in-order byte stream
 - No discrete messages
- Connection-oriented
 - Handshaking (exchange of control messages) before data transmitted
- Pipelined
 - Many segments in flight
- Flow control
 - Don't send too fast for the receiver
- Congestion control
 - Don't send too fast for the network

TCP Segments



TCP Segments



Practical Reliability Questions

- How do the sender and receiver keep track of outstanding pipelined segments?
- How many segments should be pipelined?
- How do we choose sequence numbers?
- **What does connection establishment look like?**
- How should we choose timeout values?

A connection...

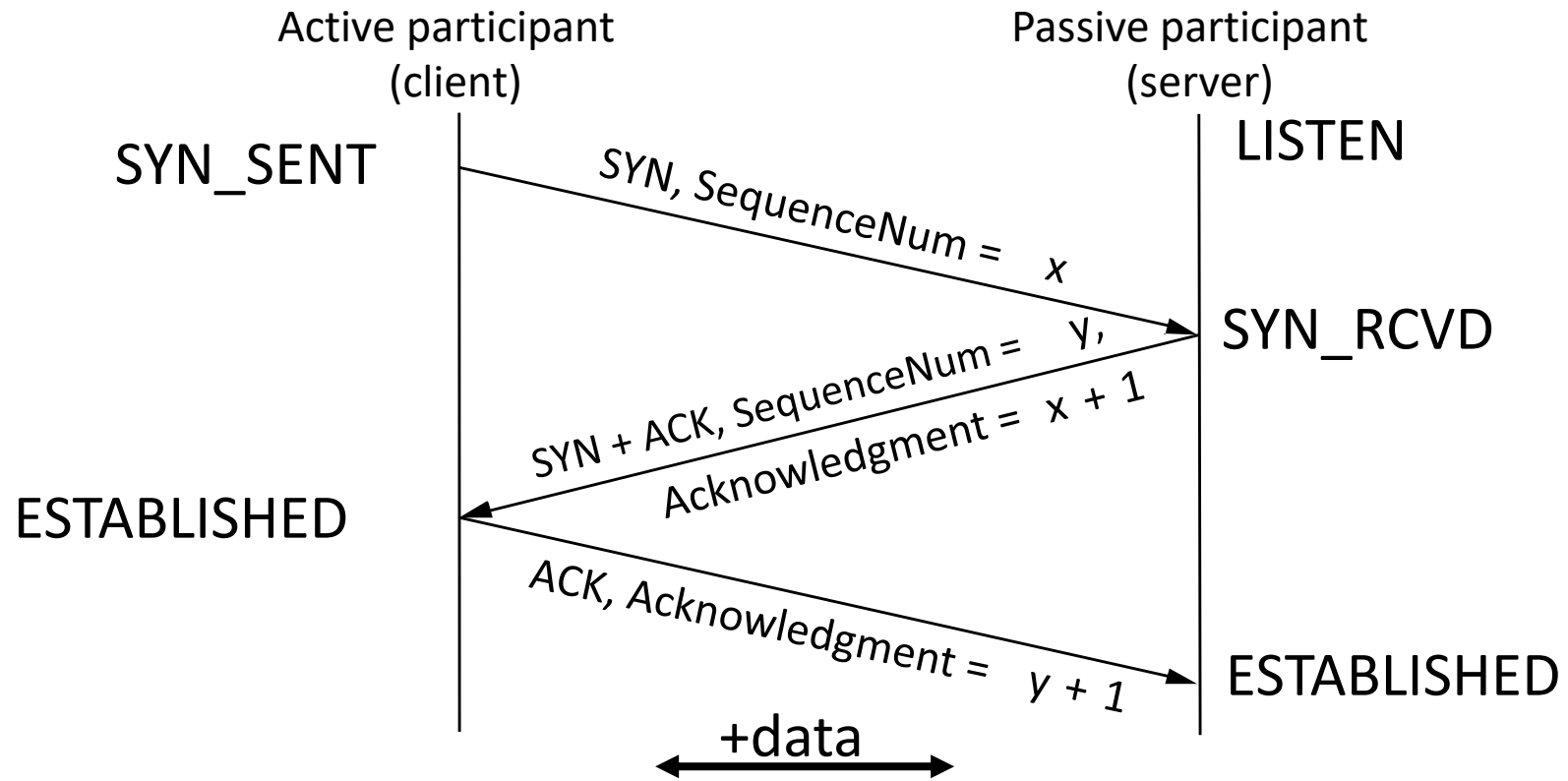
1. Requires stored state at two hosts.
2. Requires stored state within the network.
3. Establishes a path between two hosts.

- A. 1
- B. 1 & 3
- C. 1, 2 & 3
- D. 2
- E. 2 & 3

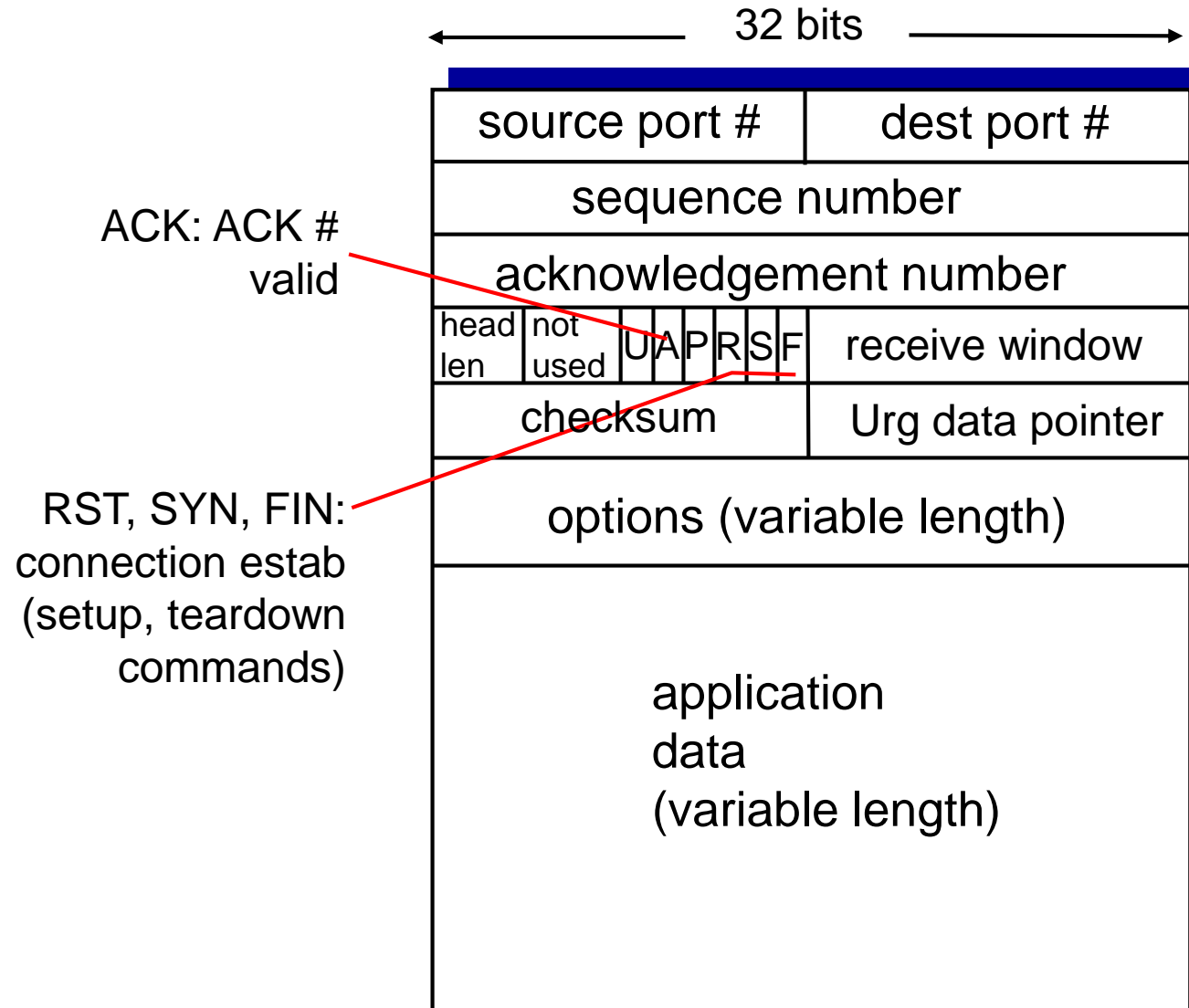
Connections

- In TCP, hosts must establish a connection prior to communicating.
- Opportunity to exchange initial protocol state.
 - Which sequence numbers to use.
 - What the maximum segment size should be.
 - Initial window sizes, etc. (several parameters)

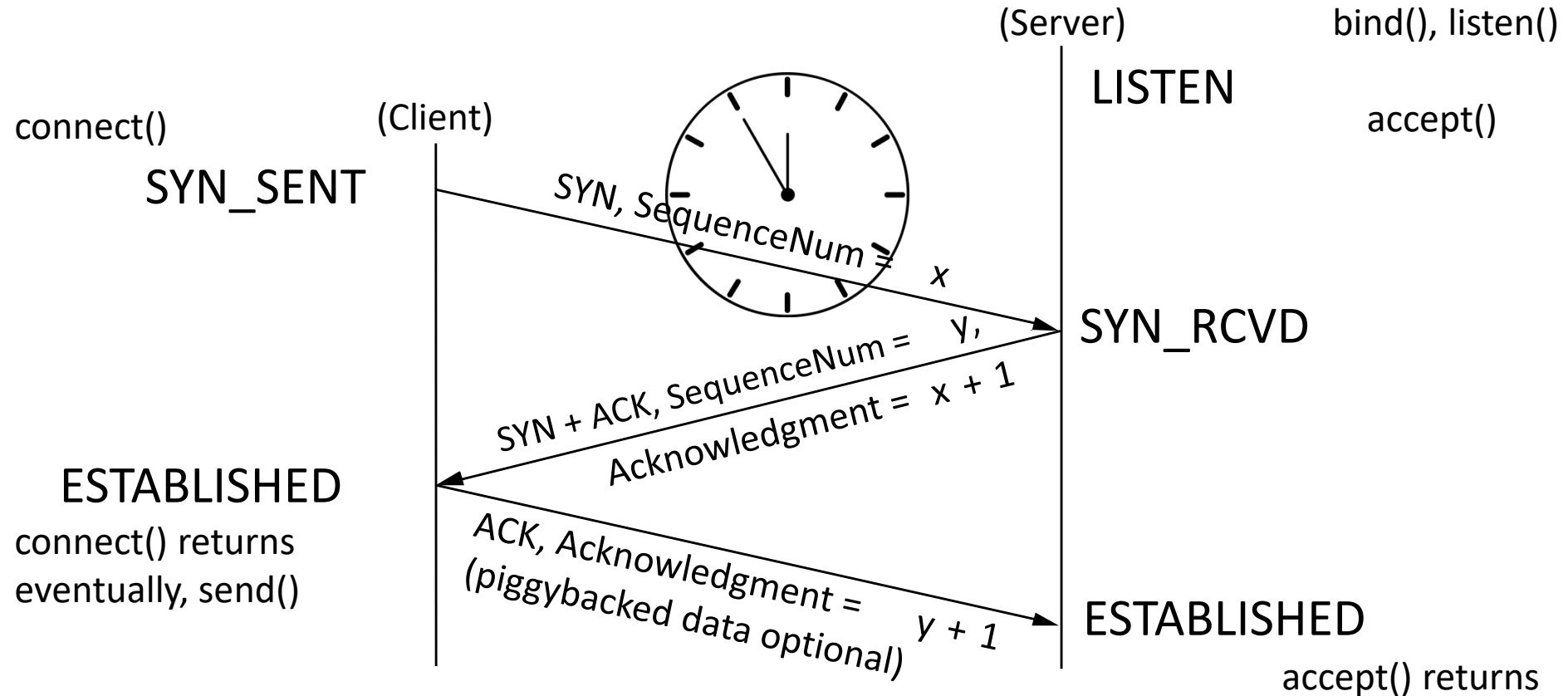
Connection Establishment (three-way handshake)



TCP Segments

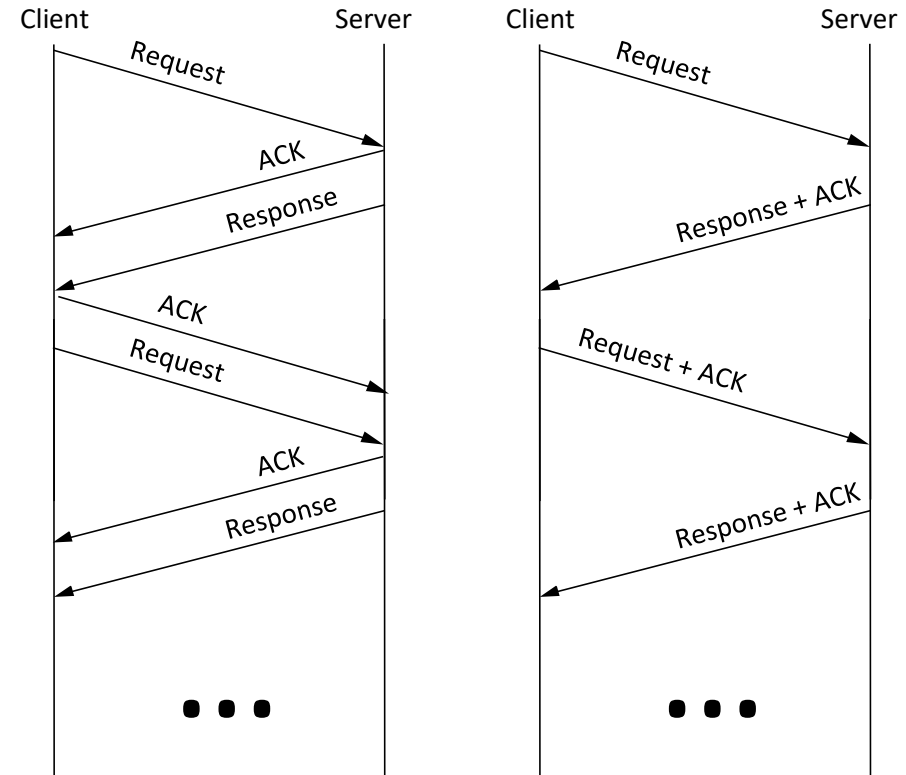


Connection Establishment (three-way handshake)



Piggybacking

- So far, we've assumed distinct "sender" and "receiver" roles
- In reality, usually both sides of a connection send some data
 - request/response is a common pattern



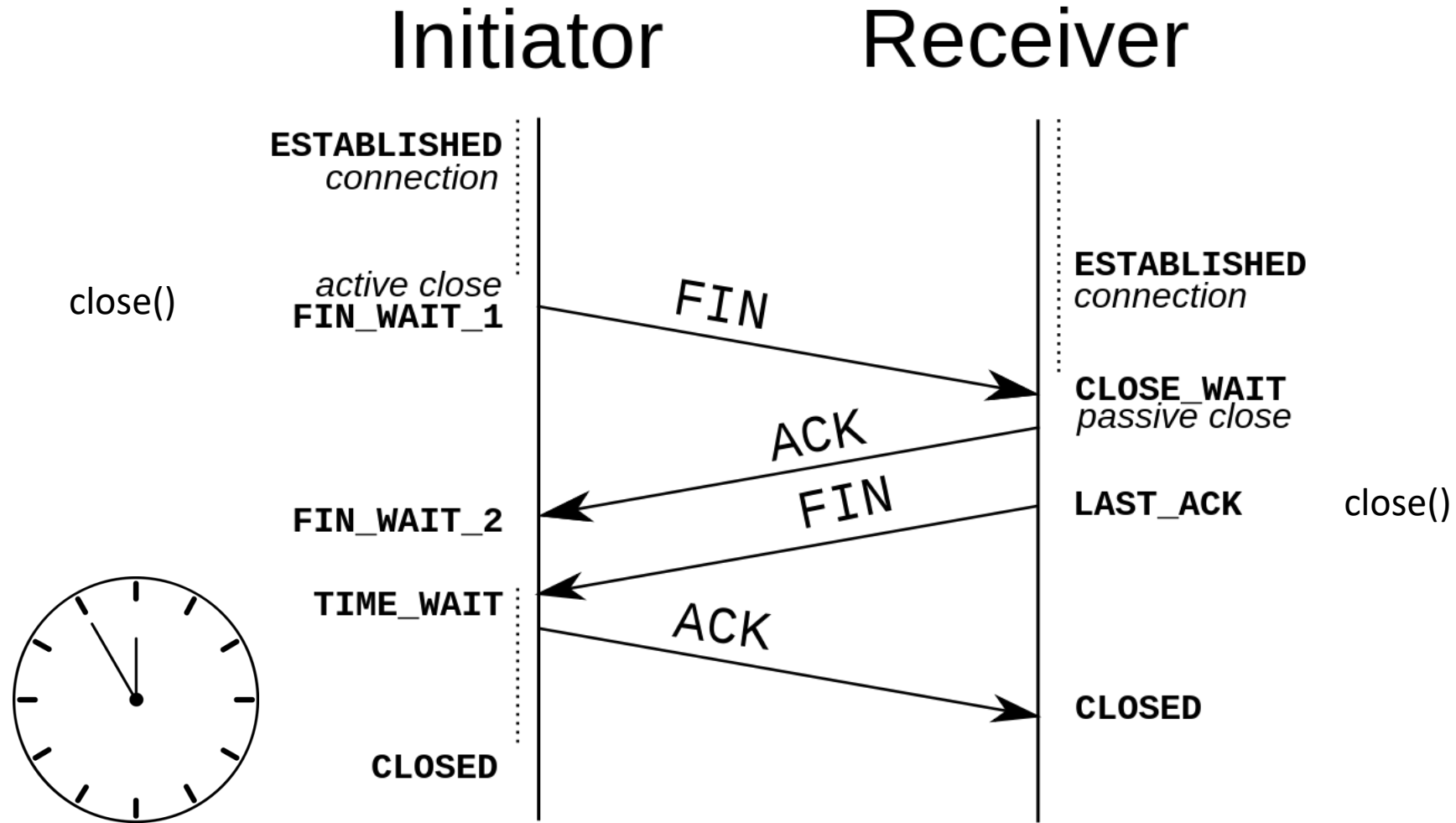
Without
Piggybacking

With
Piggybacking

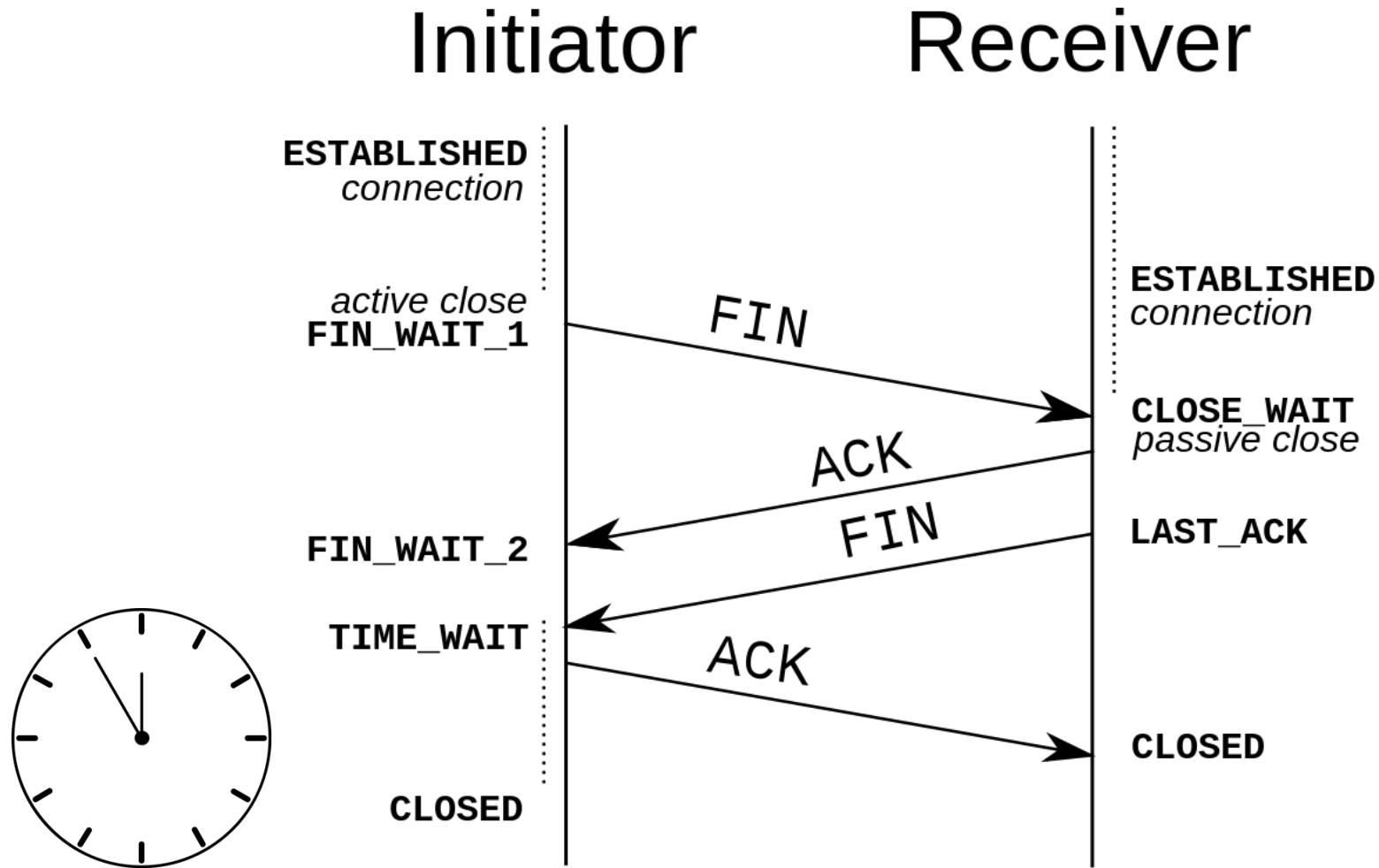
Connection Teardown

- Orderly release by sender and receiver when done
 - Delivers all pending data and “hangs up”
- Cleans up state in sender and receiver
- Each side may terminate independently

TCP Connection Teardown



Why does one side need to wait before transitioning to CLOSED state?



The TIME_WAIT State

- We wait $2*MSL$ (maximum segment lifetime) before completing the close. The MSL is arbitrary (usually 60 sec)
- ACK might have been lost and so FIN will be resent
 - Could interfere with a subsequent connection
- This is why we used `SO_REUSEADDR` socket option in lab 2
 - Says to skip this waiting step and immediately abort the connection

Practical Reliability Questions

- How do the sender and receiver keep track of outstanding pipelined segments?
- How many segments should be pipelined?
- **How do we choose sequence numbers?**
- What does connection establishment look like?
- How should we choose timeout values?

How should we choose the initial sequence number?

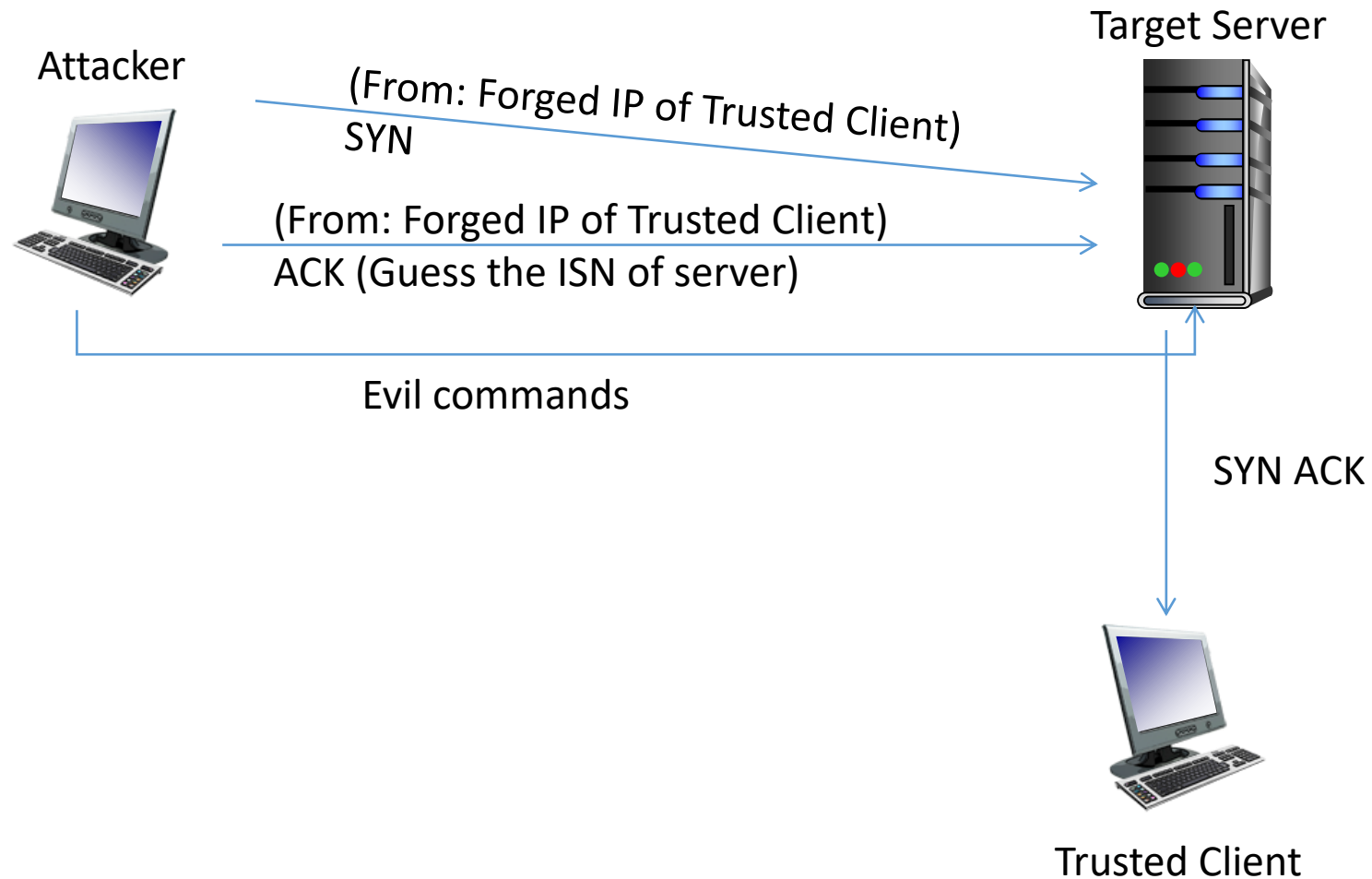
- A. Start from zero
- B. Start from one
- C. Start from a random number
- D. Start from some other value (such as...?)

What can go wrong with sequence numbers?
-How they're chosen?
-In the course of using them?

Sequencing

- Initial sequence numbers (ISN) chosen at random
 - Does not start at 0 or 1 (anymore).
 - Helps to prevent against forgery attacks.
- TCP sequences bytes rather than segments
 - Example: if we're sending 1500-byte segments
 - Randomly choose ISN (suppose we picked 1150)
 - First segment (sized 1500) would use number 1150
 - Next would use 2650

Sequence Prediction Attack (1996)



Practical Reliability Questions

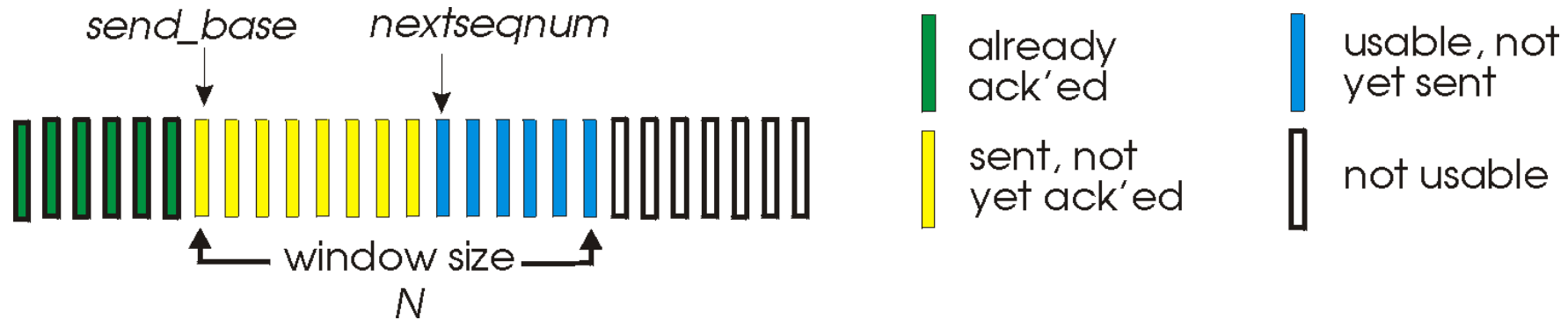
- How do the sender and receiver keep track of outstanding pipelined segments?
- How many segments should be pipelined?
- How do we choose sequence numbers?
- What does connection establishment look like?
- How should we choose timeout values?

Windowing (Sliding Window)

- At the sender:
 - What's been ACKed
 - What's still outstanding
 - What to send next
- At the receiver:
 - Go-back-N
 - Highest sequence number received so far.
 - (Selective repeat)
 - Which sequence numbers received so far.
 - Buffered data.

Go-back-N

- At the sender:

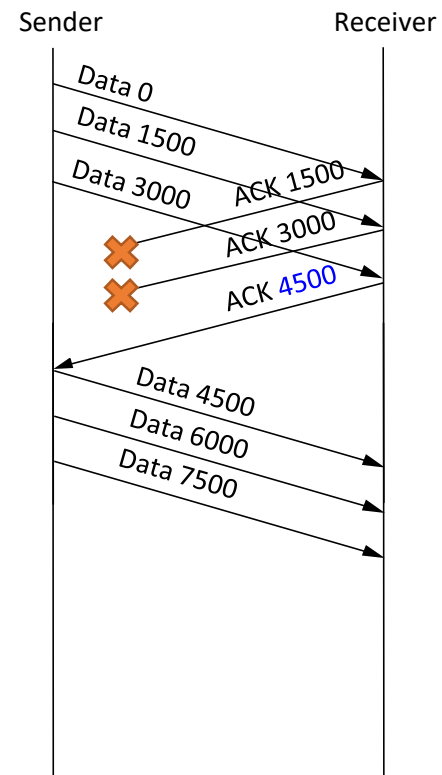


- At the receiver:

- Keep track of largest sequence number seen.
- If it receives ANYTHING, sends back ACK for largest sequence number seen so far. (Cumulative ACK)

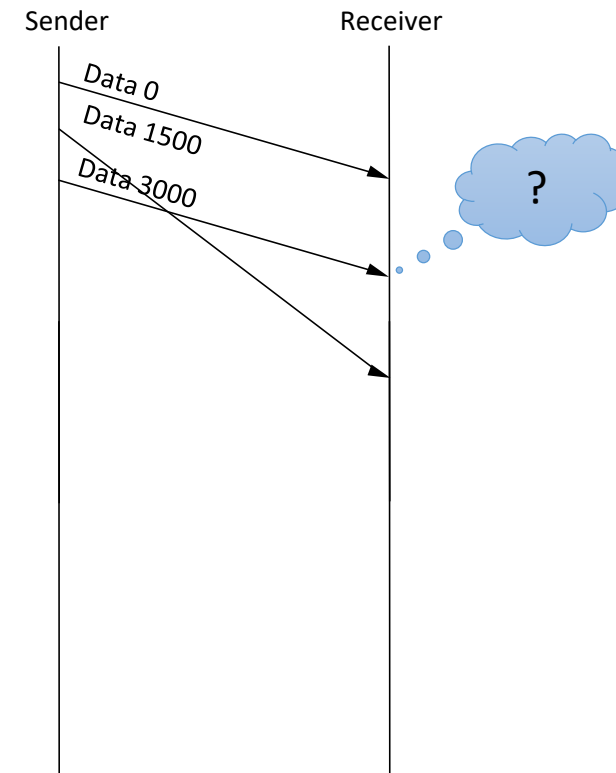
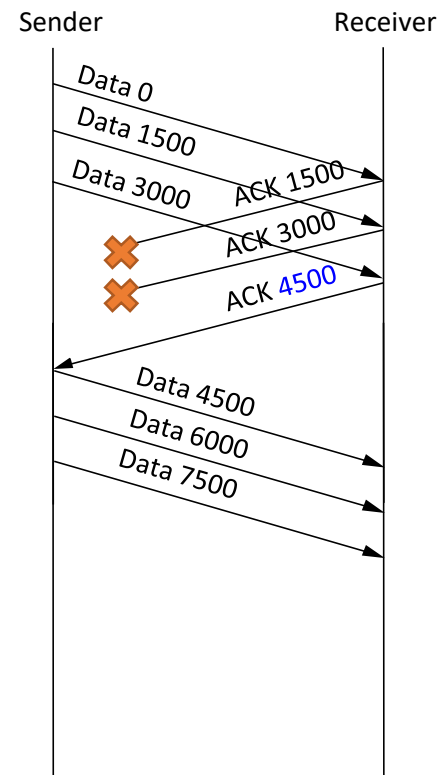
Cumulative Acknowledgements

- An ACK for sequence number N implies that all data prior to N has been received.



Cumulative Acknowledgements

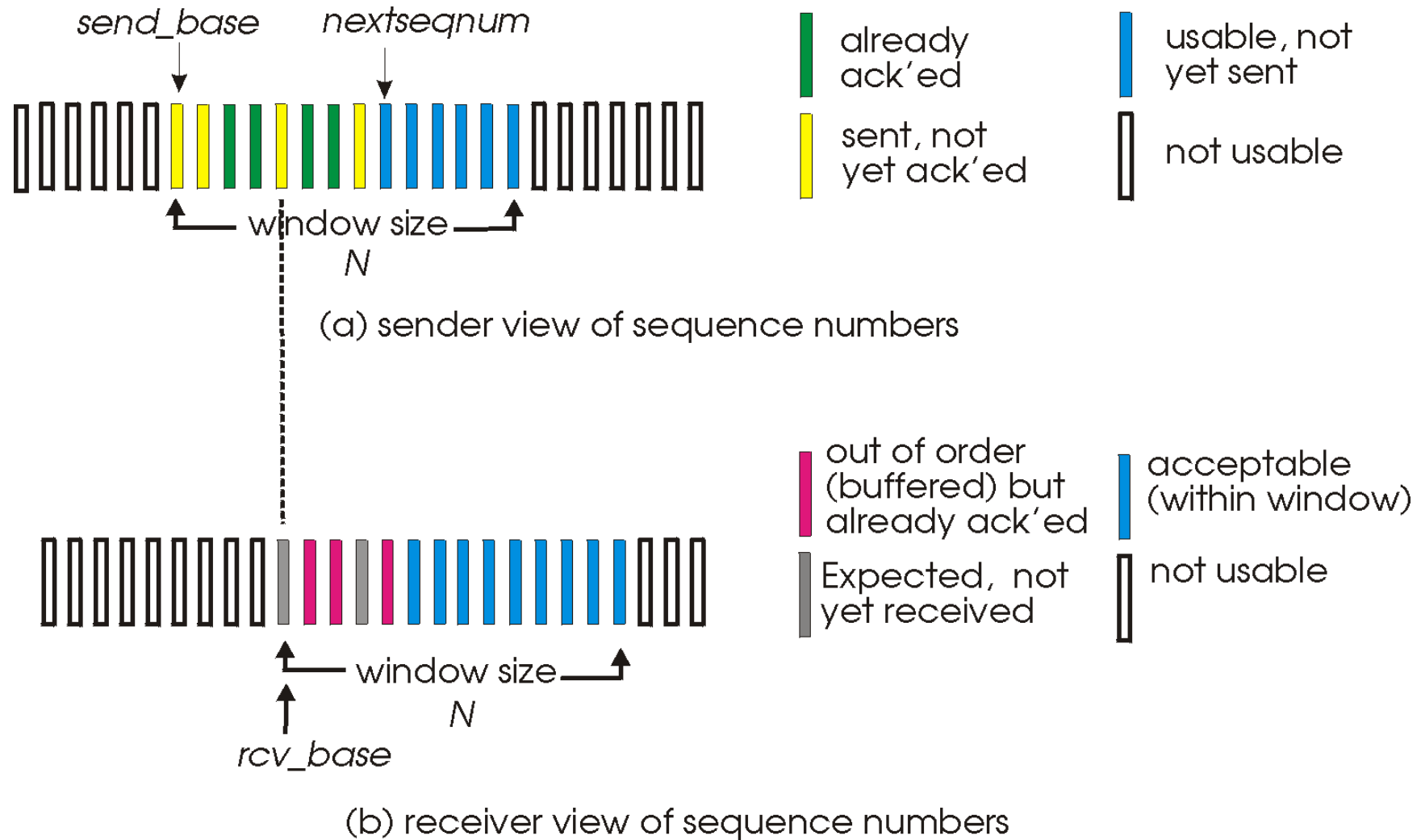
- An ACK for sequence number N implies that all data prior to N has been received.



What should we do with an out-of-order segment at the receiver?

- A. Drop it.
- B. Save it and ACK it.
- C. Save it, don't ACK it.
- D. Something else (explain).

Selective Repeat



If you were building a transport protocol, which would you use?

- A. Go-back-N
- B. Selective repeat
- C. Something else (explain)

Practical Reliability Questions

- How do the sender and receiver keep track of outstanding pipelined segments?
- How many segments should be pipelined?
- How do we choose sequence numbers?
- What does connection establishment look like?
- **How should we choose timeout values?**

Timeouts

- How long should we wait before timing out and retransmitting a segment?
- Too short: needless retransmissions
- Too long: slow reaction to losses
- Should be (a little bit) longer than the RTT

Estimating RTT

- Problem: RTT changes over time
 - Routers buffer packets in queues
 - Queue lengths vary
 - Receiver may have varying load
- Sender takes measurements
 - Use statistics to decide future timeouts for sends
 - Estimate RTT and variance
- Apply “smoothing” to account for changes

Estimating RTT

- For each segment that did not require a retransmit (ACK heard without a timeout)
 - Consider the time between segment sent and ACK received to be a sample of the current RTT
 - Use that, along with previous history, to update the current RTT estimate
- Exponentially Weighted Moving Average (EWMA)

EWMA

$$\text{EstimatedRTT} = (1 - a) * \text{EstimatedRTT} + a * \text{SampleRTT}$$

a is usually 1/8.

In other words, our current estimate is a blend of 7/8 of the previous estimate plus 1/8 of the new sample.

$$\text{DevRTT} = (1 - B) * \text{DevRTT} + B * | \text{SampleRTT} - \text{EstimatedRTT} |$$

B is usually 1/4

Example

- Suppose EstimateRTT = 64, Dev = 8

Latest sample: 120

$$\text{New estimate} = 7/8 * 64 + 1/8 * 120 = 56 + 15 = 71$$

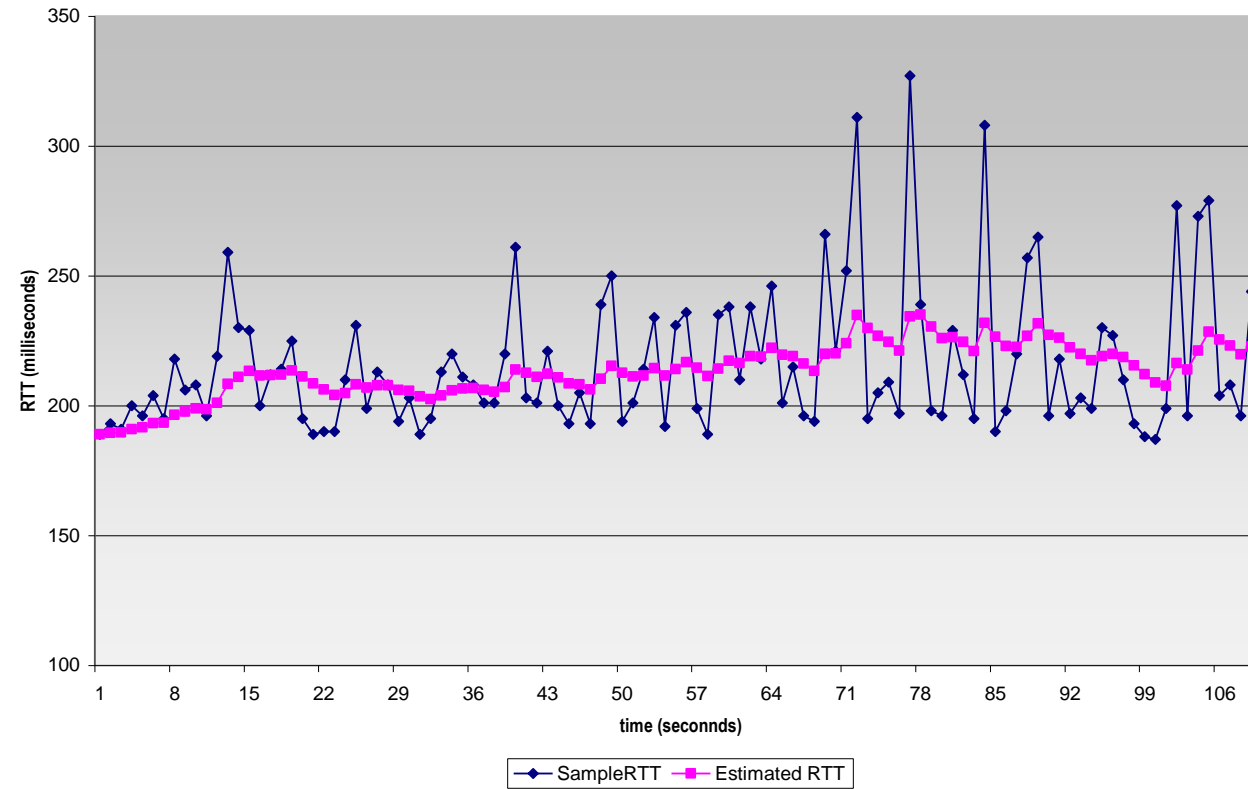
$$\text{New dev} = 3/4 * 8 + 1/4 * | 120 - 71 | = 6 + 12 = 18$$

- Another sample: 400

$$\text{New estimate} = 7/8 * 71 + 1/8 * 400 = 62 + 50 = 112$$


$$\text{New dev} = 3/4 * 18 + 1/4 * | 400 - 112 | = 13 + 72 = 85$$

Book Example (Smoothing)



TCP Timeout Value

$$\text{TimeoutInterval} = \text{EstimatedRTT} + 4 * \text{DevRTT}$$

 ↑ ↑

estimated RTT “safety margin”

Practical Reliability Questions

- How do the sender and receiver keep track of outstanding pipelined segments?
- How many segments should be pipelined?
- How do we choose sequence numbers?
- What does connection establishment look like?
- How should we choose timeout values?

Next time...

- How do the sender and receiver keep track of outstanding pipelined segments?
- **How many segments should be pipelined?**
- How do we choose sequence numbers?
- What does connection establishment look like?
- How should we choose timeout values?