CS 43: Computer Networks

Reliable Transport and TCP

October 28, 2026



Today

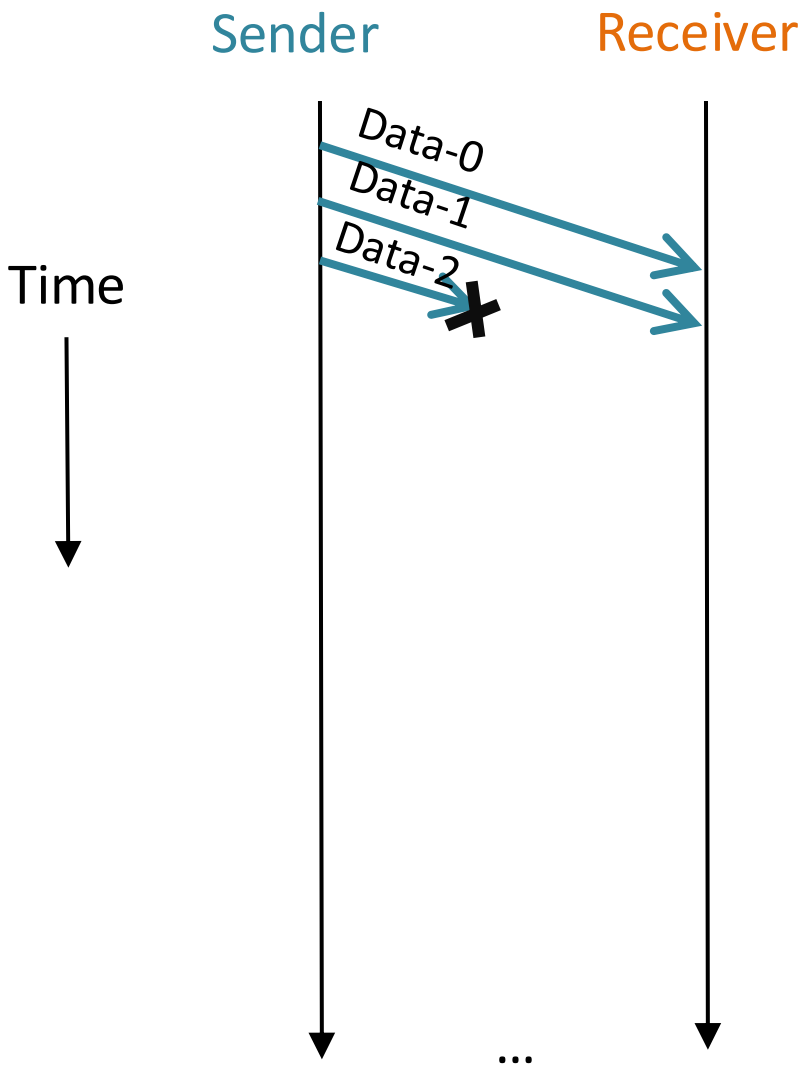
- Go-back N and Selective Repeat
- TCP
  - Packet format
  - Connection establishment & teardown
  - Sequencing
  - Choosing timeout value

ARQ Broad Classifications

1. Stop-and-wait

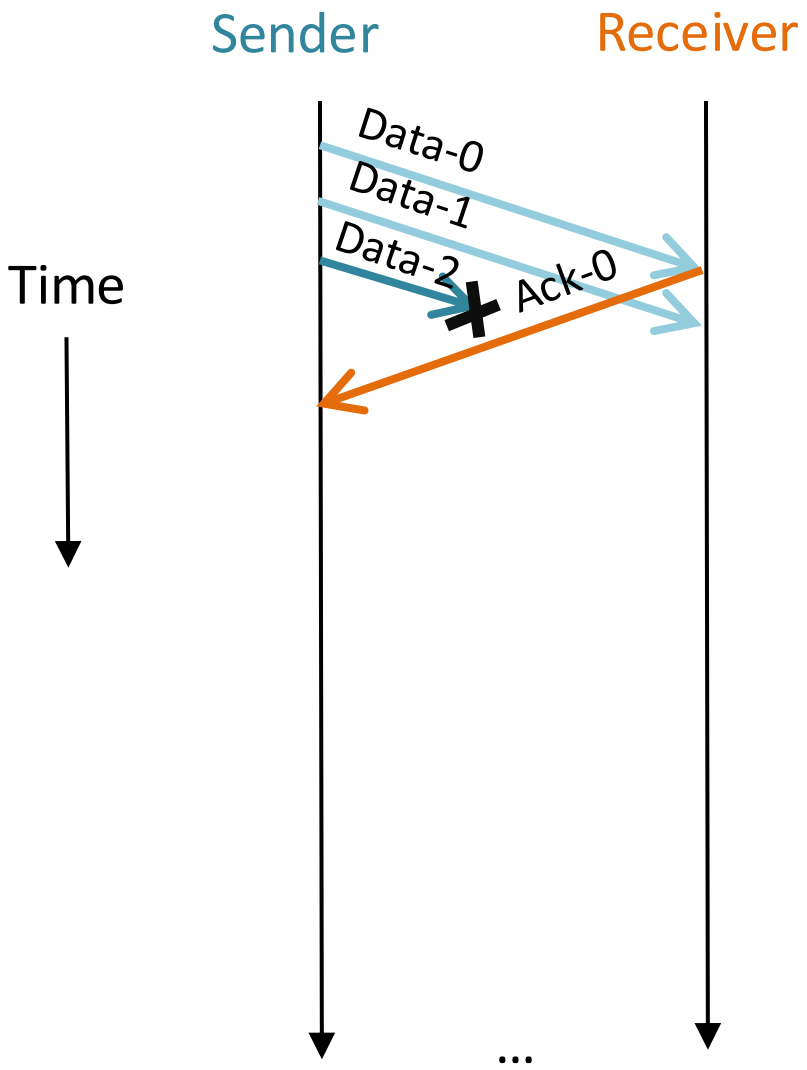
2. Go-back-N

Go-Back-N

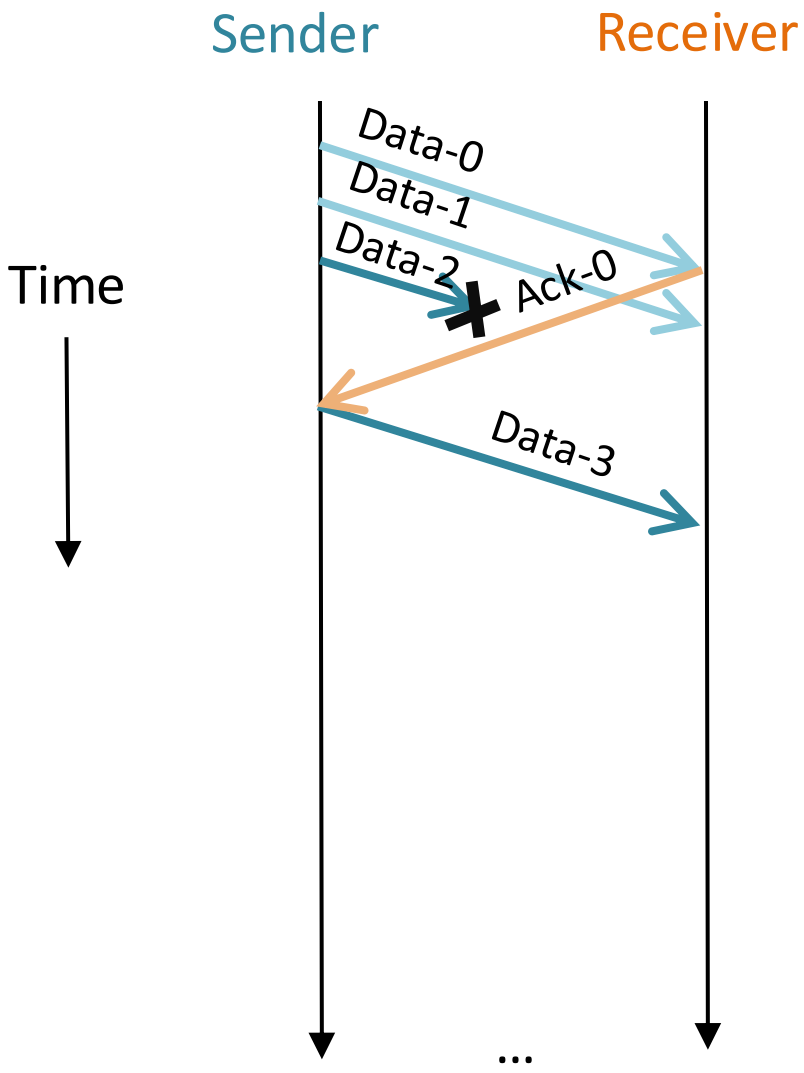


- Retransmit from point of loss
  - Segments between loss event and retransmission are ignored
  - “Go-back-N” if a timeout event occurs

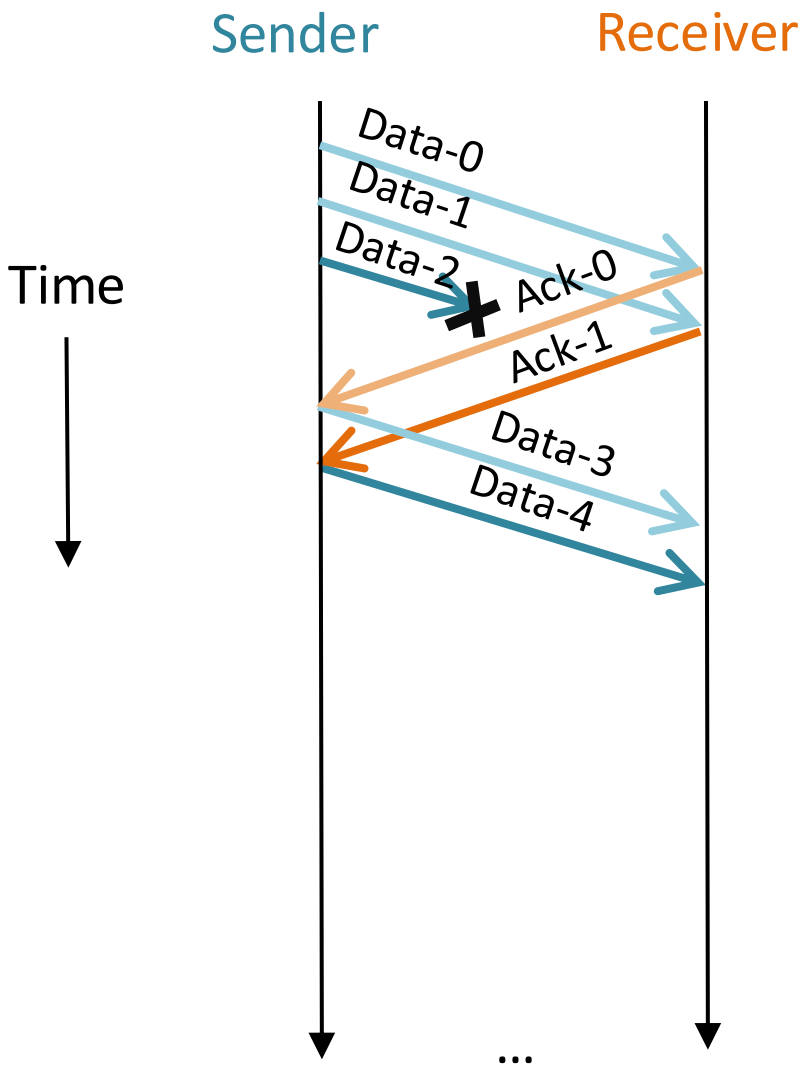
Go-Back-N



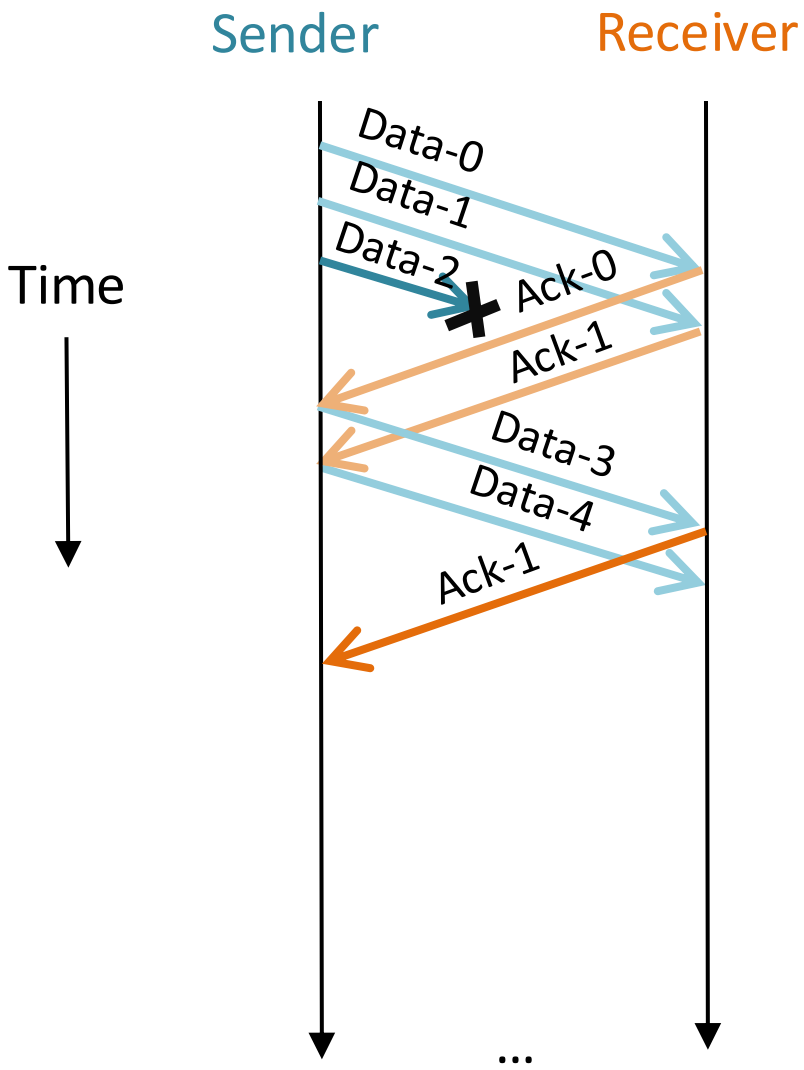
Go-Back-N



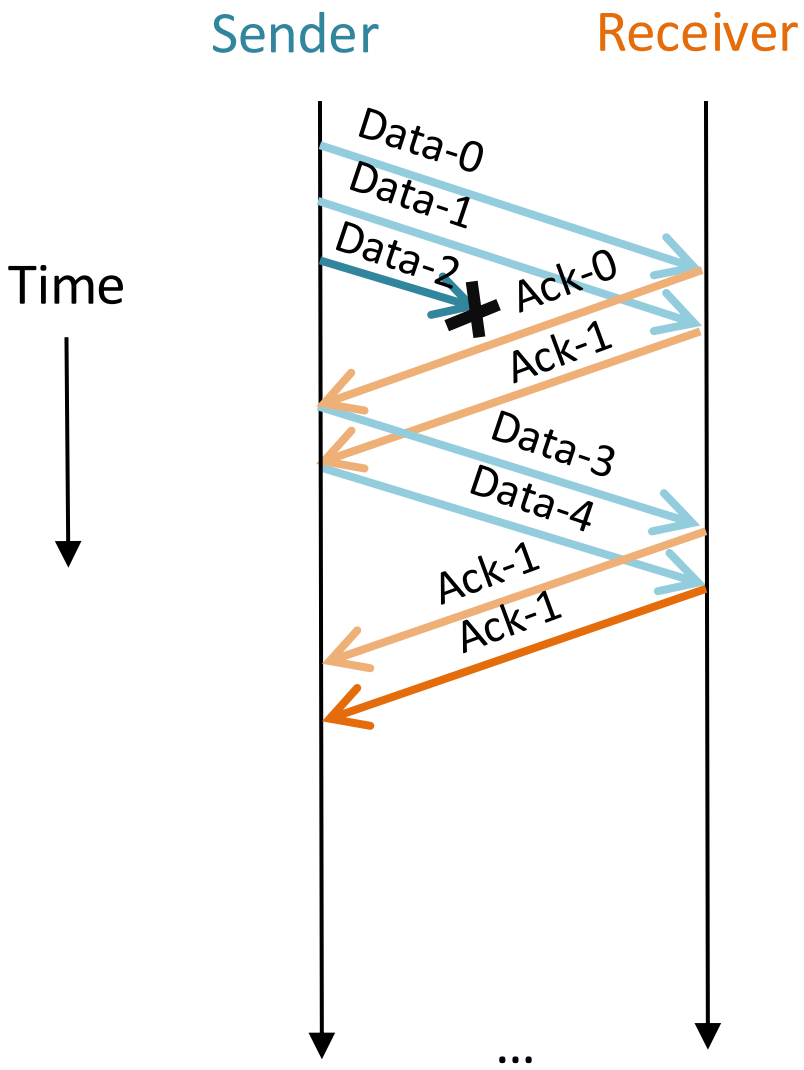
Go-Back-N


Sender
Receiver
Time
Data-0
Data-1
Data-2
Ack-0
Ack-1
Data-3
Data-4
...

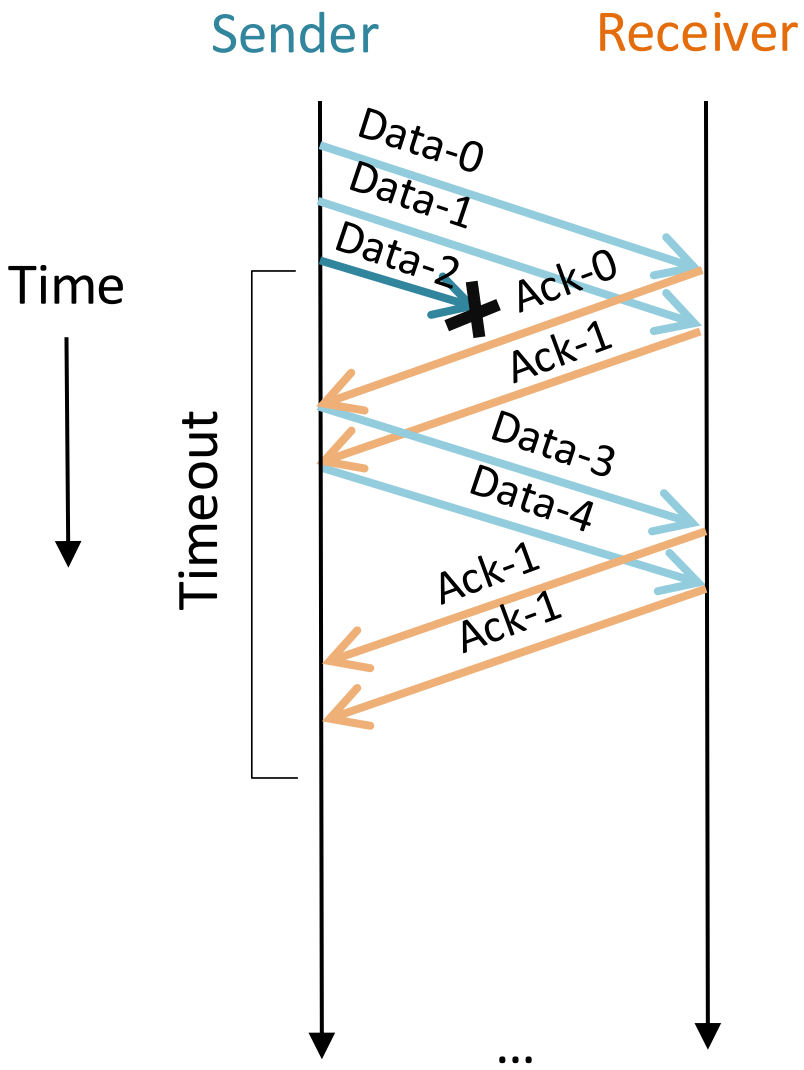
Go-Back-N


Sender
Receiver
Time
Data-0
Data-1
Data-2
Ack-0
Ack-1
Data-3
Data-4
Ack-1
...

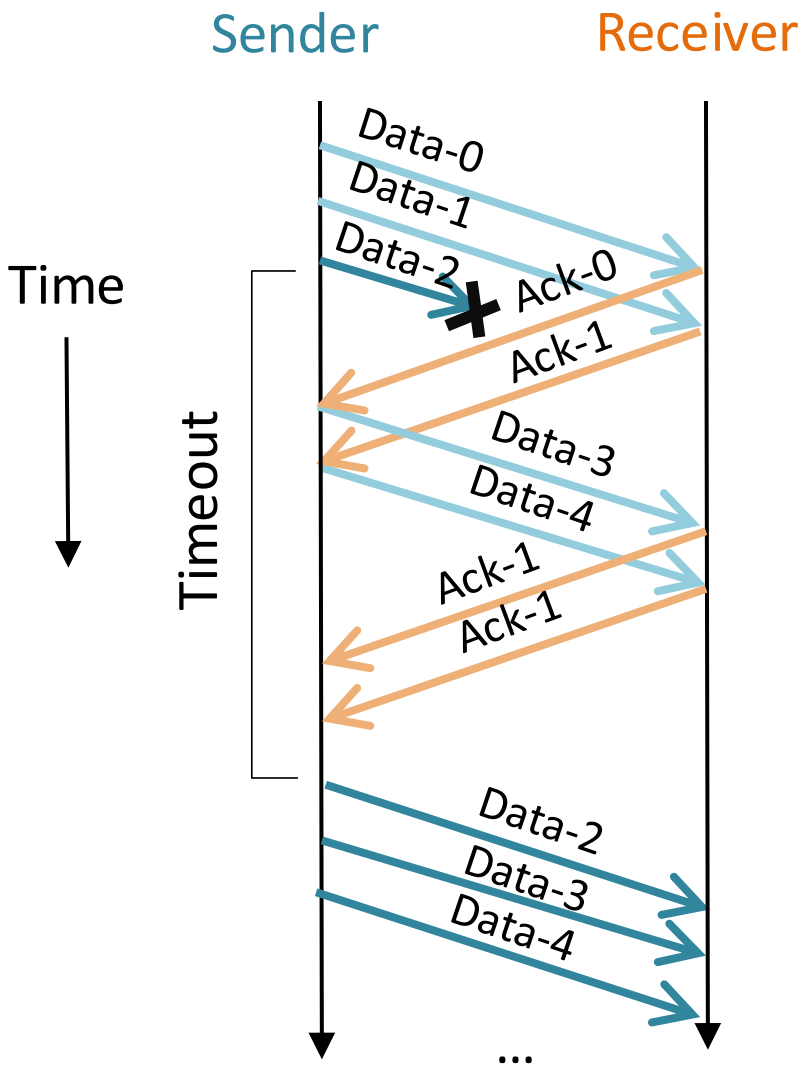
Go-Back-N


Sender
Receiver
Time
Data-0
Data-1
Data-2
Ack-0
Ack-1
Data-3
Data-4
Ack-1
Ack-1
...

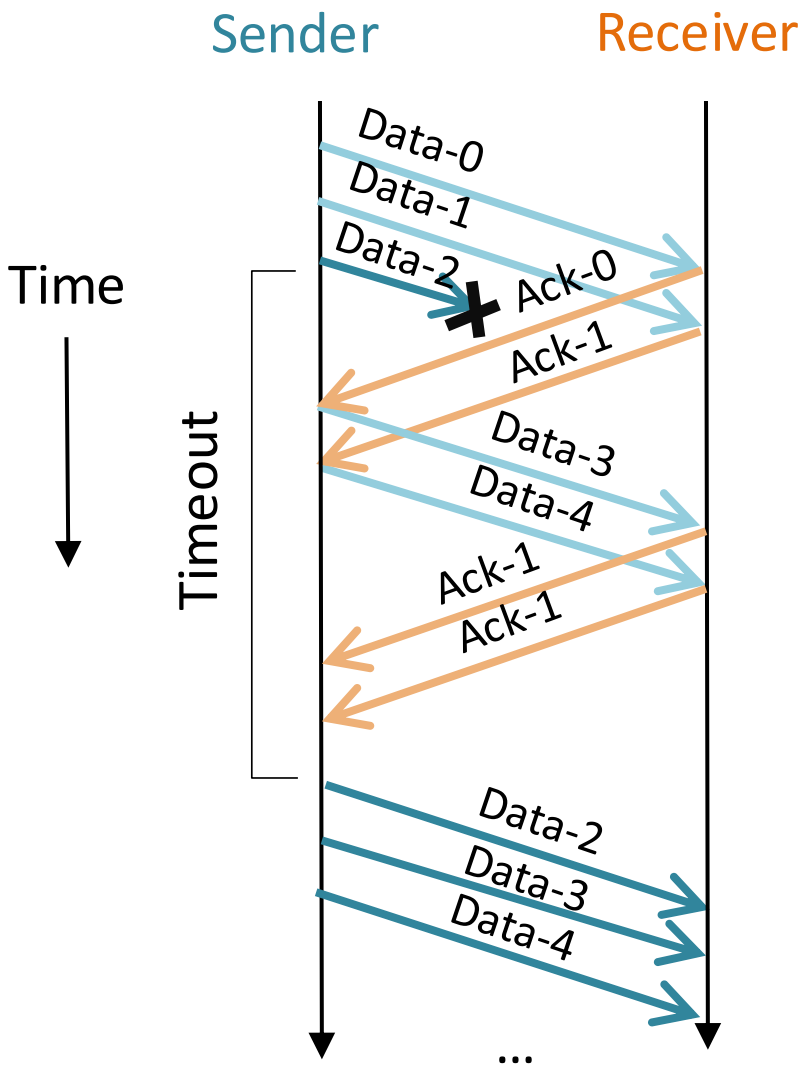
Go-Back-N


Sender
Receiver
Time
Timeout
Data-0
Data-1
Data-2
Ack-0
Ack-1
Data-3
Data-4
Ack-1
Ack-1
...

Go-Back-N

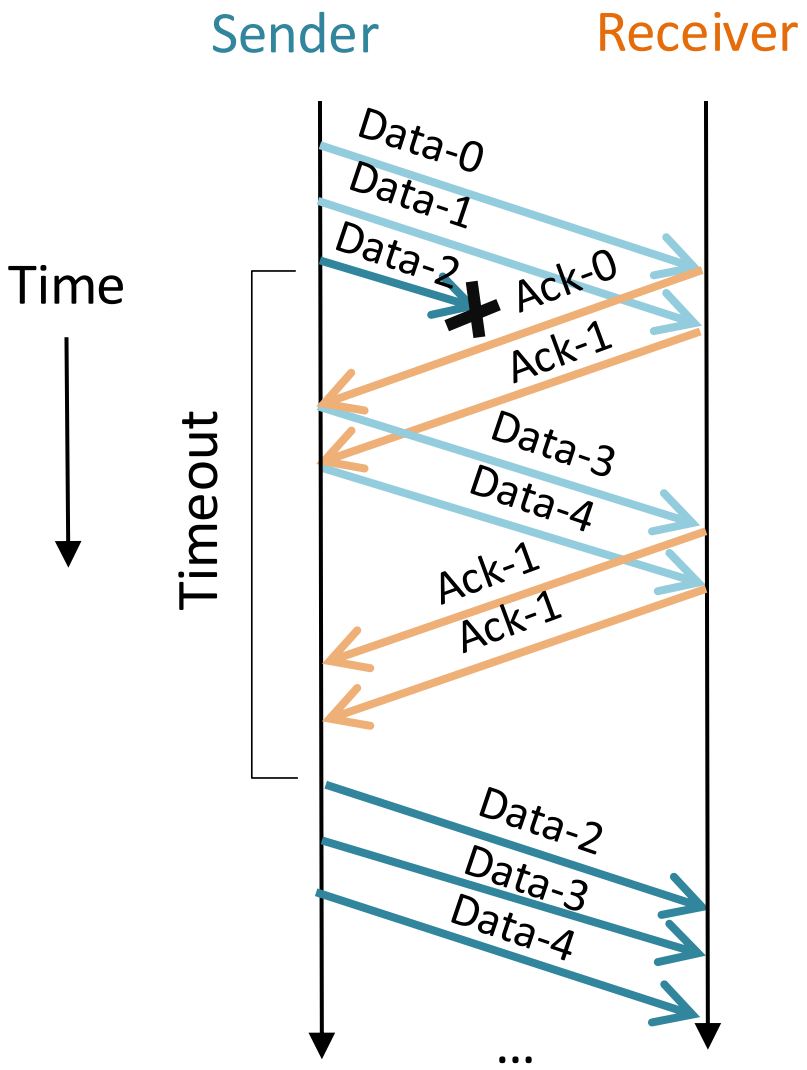

Sender
Receiver
Time
Timeout
Data-0
Data-1
Data-2
Ack-0
Ack-1
Data-3
Data-4
Ack-1
Ack-1
Data-2
Data-3
Data-4
...

Go-Back-N



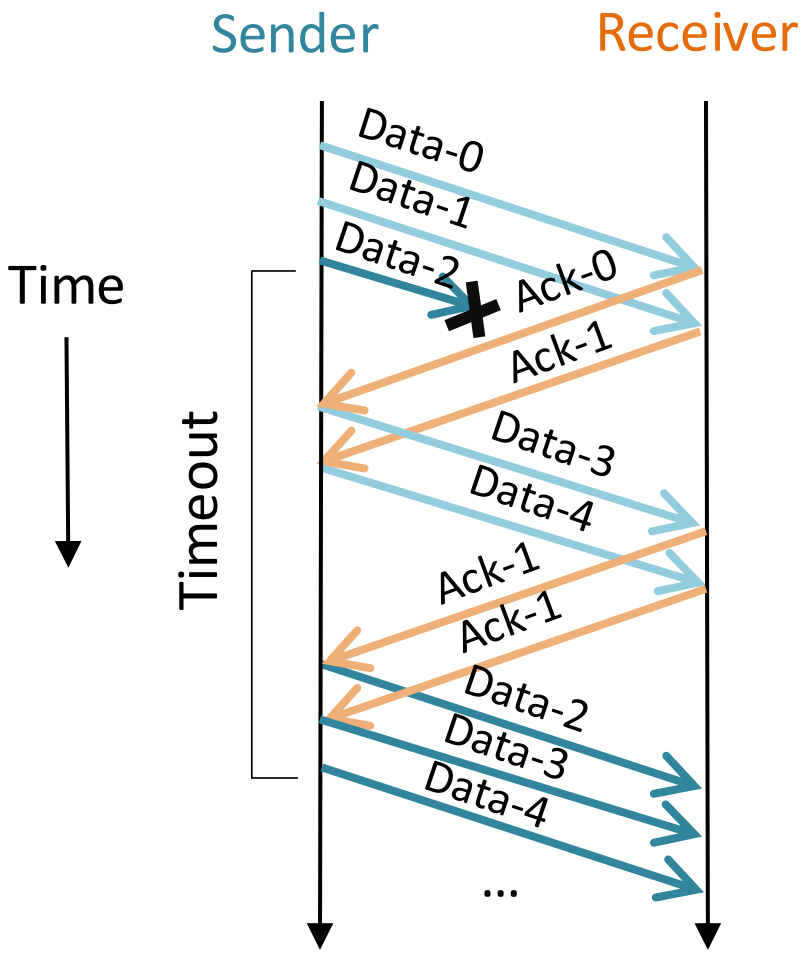
- Retransmit from point of loss
  - Segments between loss event and retransmission are ignored
  - "Go-back-N" if a timeout event occurs

Go-Back-N Performance Optimization



- Can we optimize the performance? Yes!

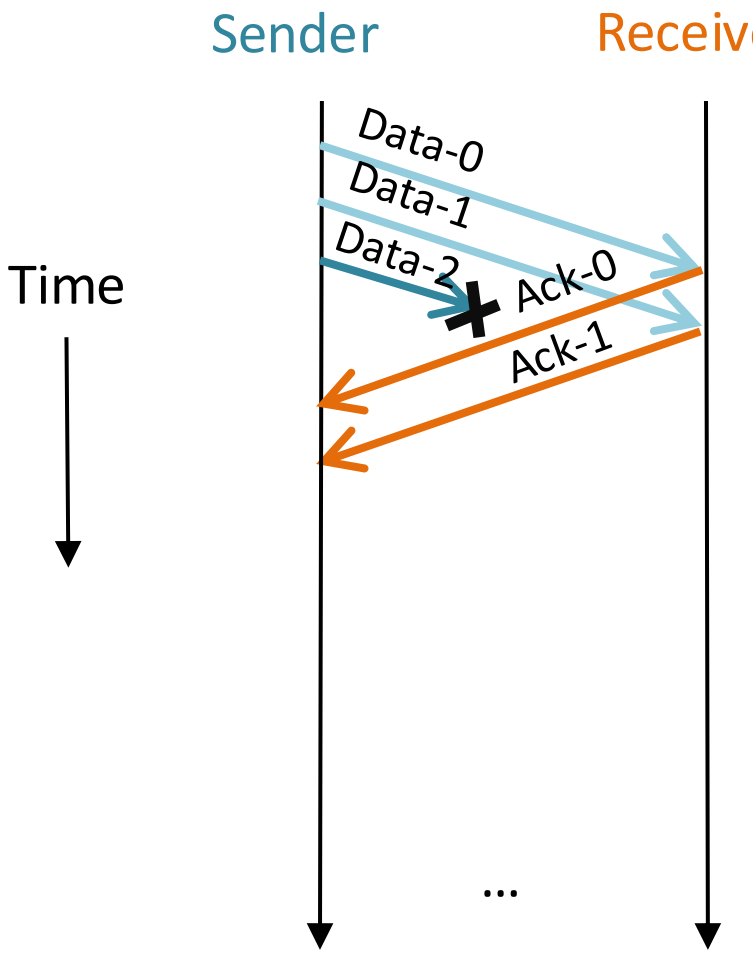
Go-Back-N: Performance Optimization



Animation

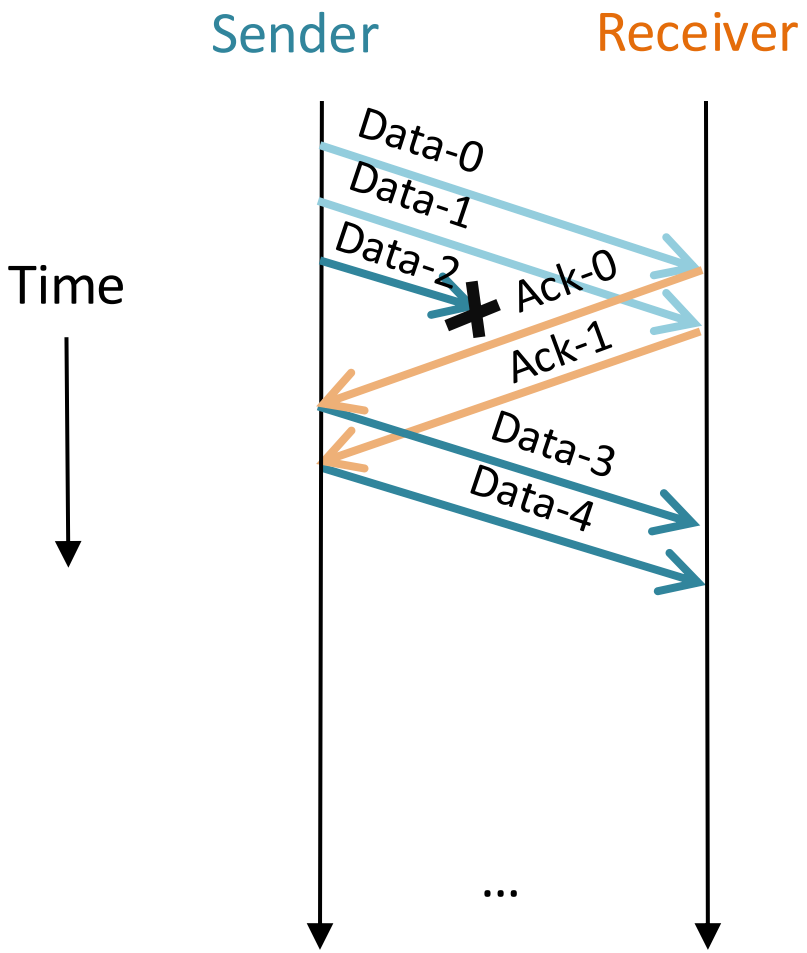
- https://media.pearsoncmg.com/ph/esm/ecs_kurose_compnetwork_8/cw/content/interactiveanimations/go-back-n-protocol/index.html

Selective Repeat

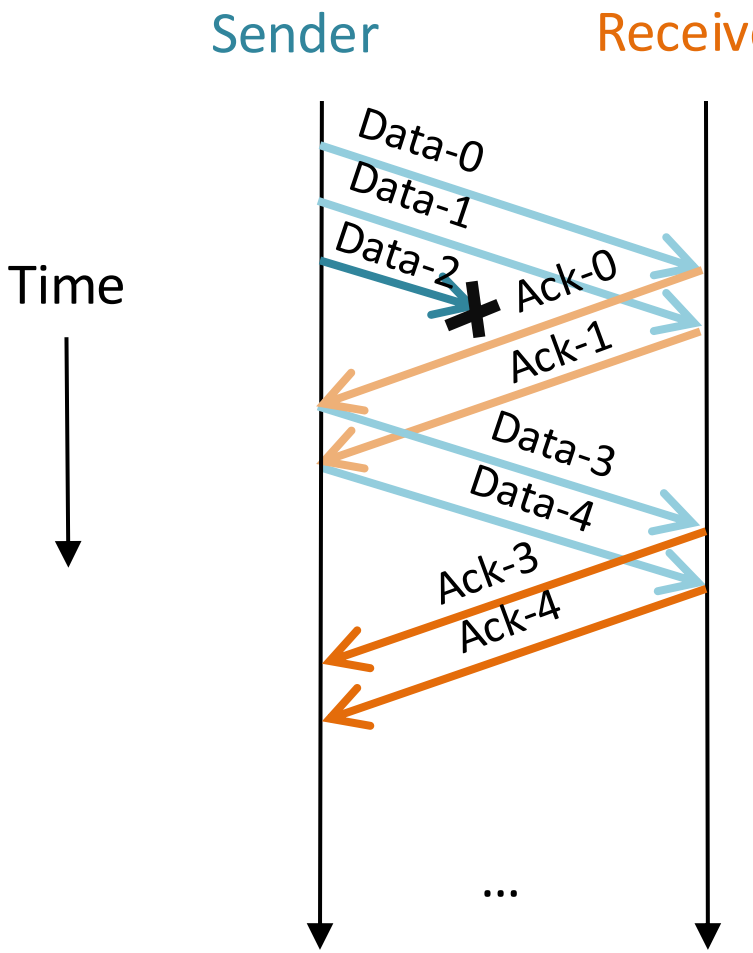


- Receiver ACKs each segment individually (not cumulative)
- Sender only resends those not ACKed

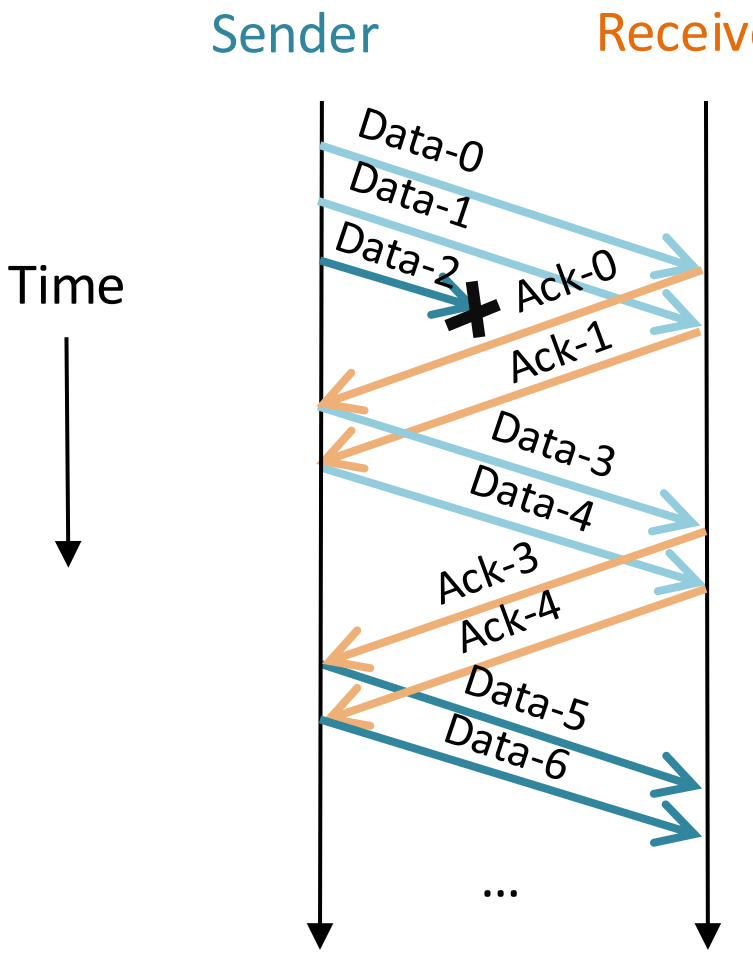
Selective Repeat



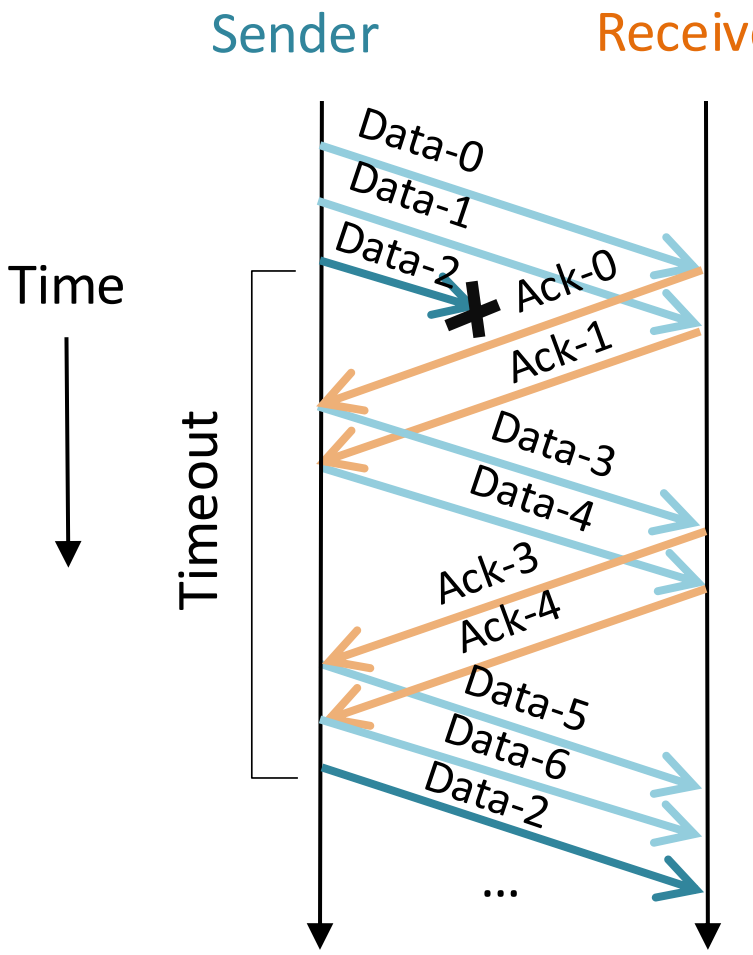
Selective Repeat



Selective Repeat



Selective Repeat



- Receiver ACKs each segment individually (not cumulative)
- Sender only resends those not ACKed

Animation

- https://media.pearsoncmg.com/ph/esm/ecs_kurose_compnetwork_8/cw/content/interactiveanimations/selective-repeat-protocol/index.html

ARQ Alternatives

- Can’t afford the RTT’s or timeouts?
- When?
  - Broadcasting, with lots of receivers
  - Very lossy or long-delay channels (e.g., space)
- Use redundancy – send more data
  - Simple form: send the same message N times
  - More efficient: use “erasure coding”
  - For example, encode your data in 10 pieces such that the receiver can piece it together with any subset of size 8.

Practical Reliability Questions

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

TCP Overview

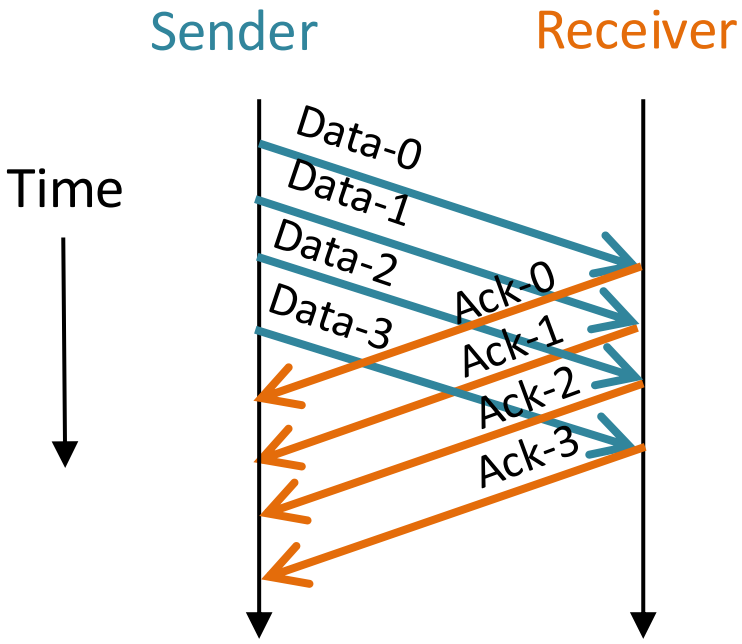
- Point-to-point, full duplex
  - One pair of hosts
  - Messages in both directions
- Reliable, in-order byte stream
  - No discrete message
- Connection-oriented
  - Handshaking (exchange of control messages) before data transmitted
- Pipelined
  - Many segments in flight
- Flow control
  - Don’t send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

Practical Reliability Questions

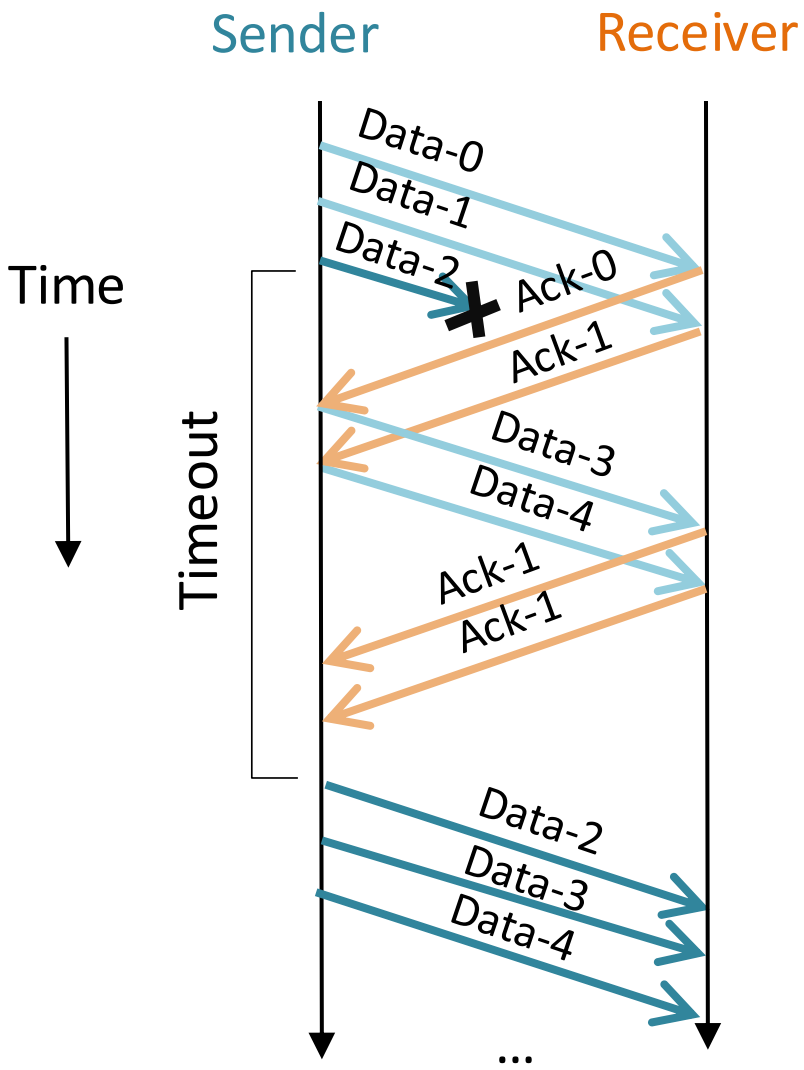
- What does connection establishment look like?
- How do we choose sequence numbers?
- How should we choose timeout values?
- How do the sender and receiver keep track of outstanding pipelined segments?
- How many segments should be pipelined?

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.



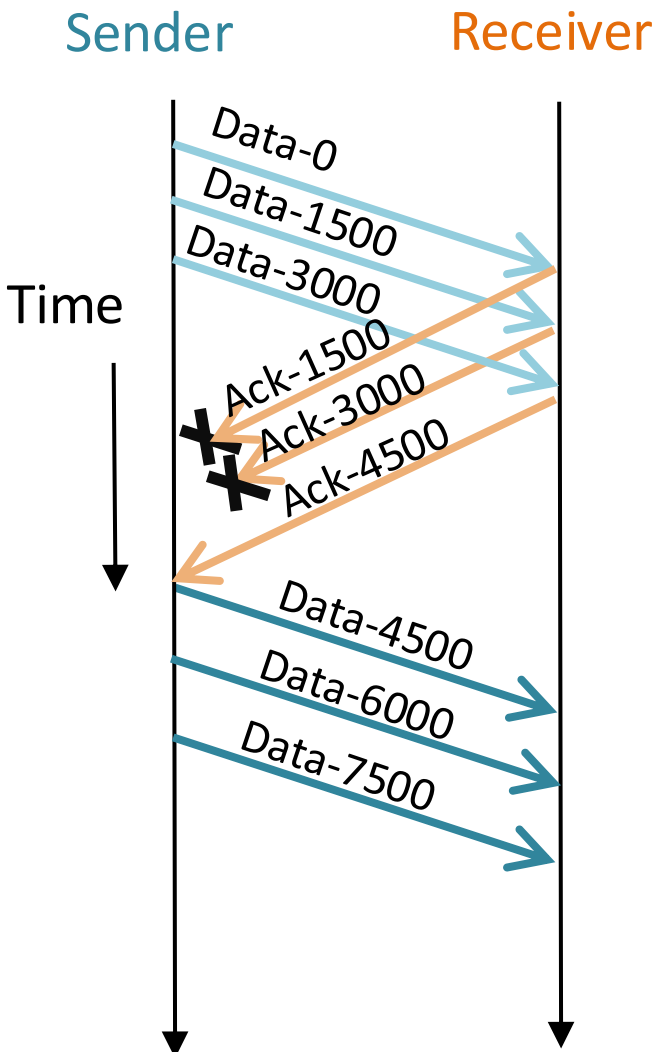
Recall: ARQ Protocol: Go-Back-N



- Retransmit from point of loss
  - Segments between loss event and retransmission are ignored
  - "Go-back-N" if a timeout event occurs

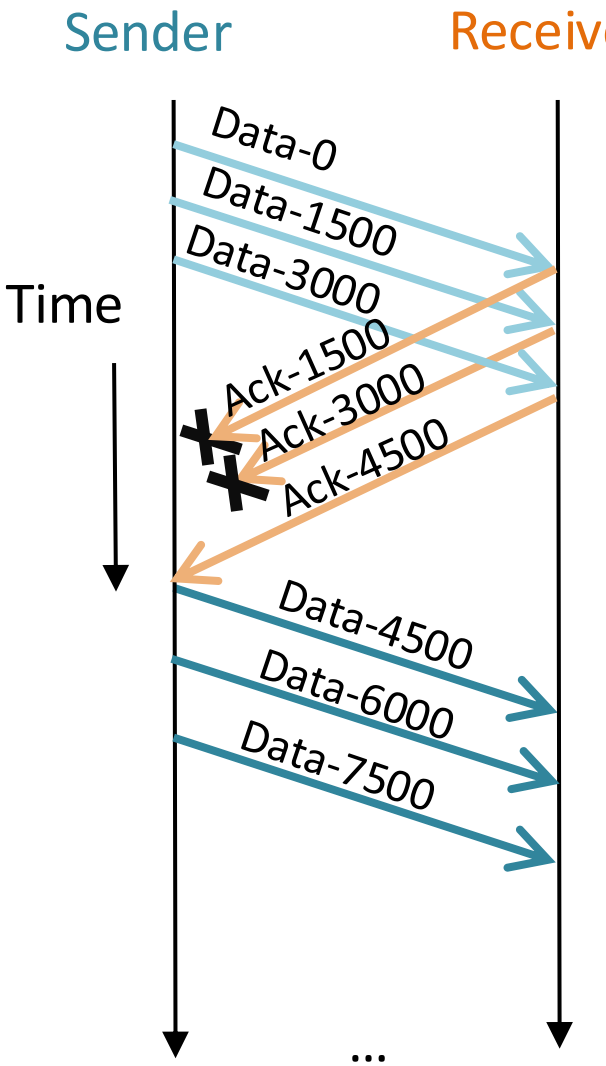
Cumulative Acknowledgements

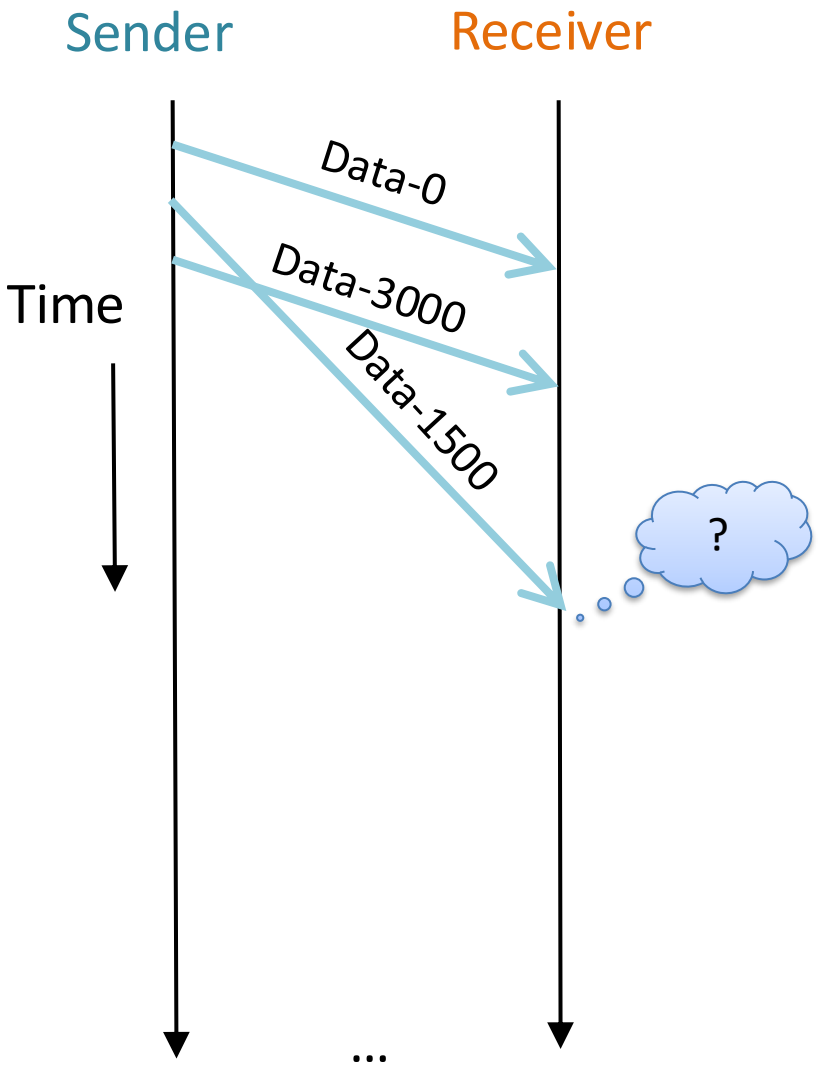
An ACK for sequence number N implies that all data prior to N has been received.



Cumulative Acknowledgements

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





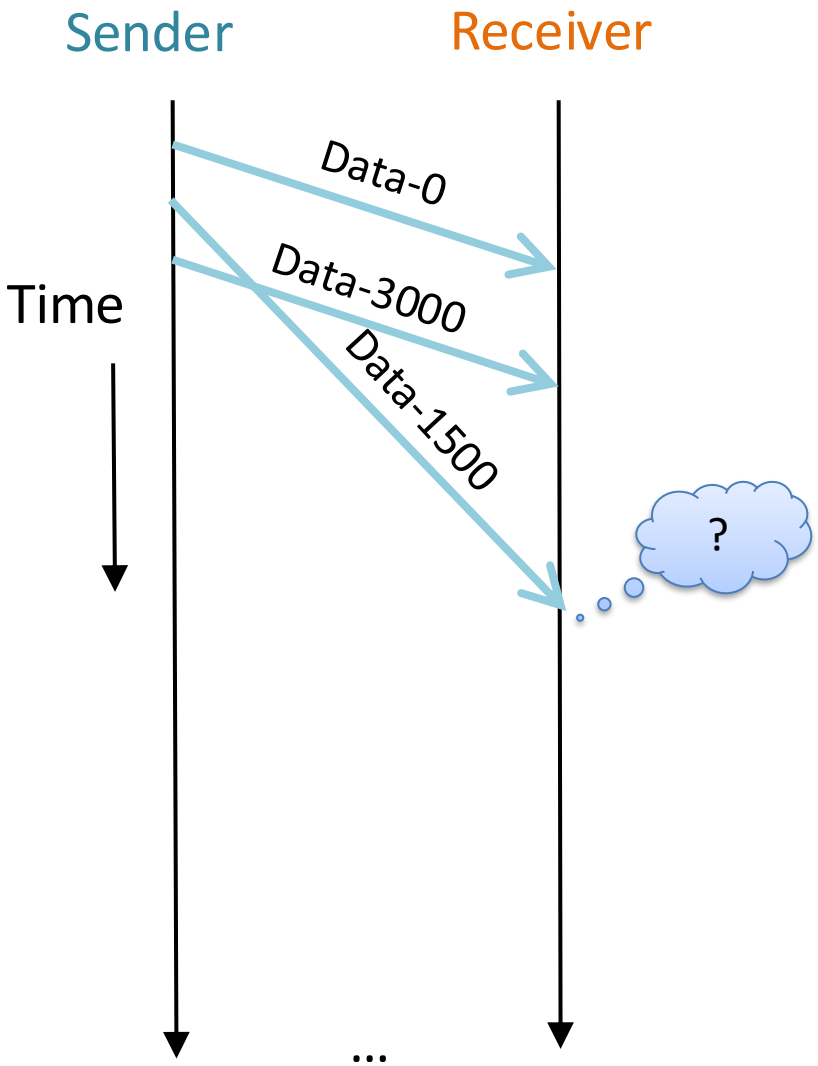
What should we do with an out-of-order segment at the receiver?

A. Drop it.

B. Save it and ACK it.

C. Save it, don't ACK it.

D. Something else (explain).



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

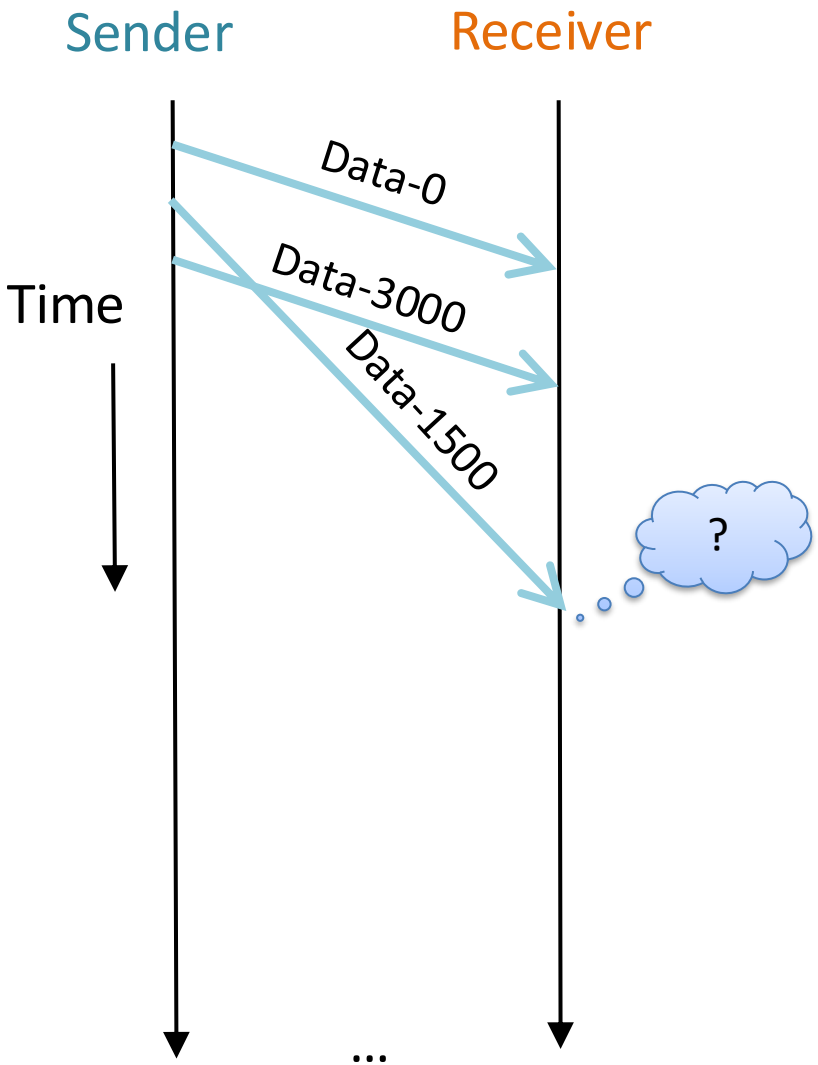
A. Drop it.

B. Save it and ACK it.

C. Save it, don't ACK it.

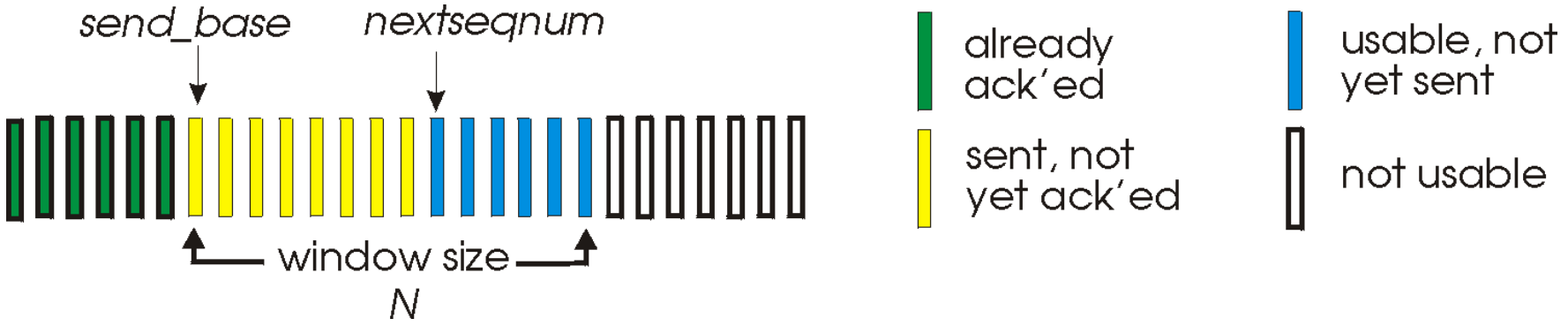
D. Something else (explain).





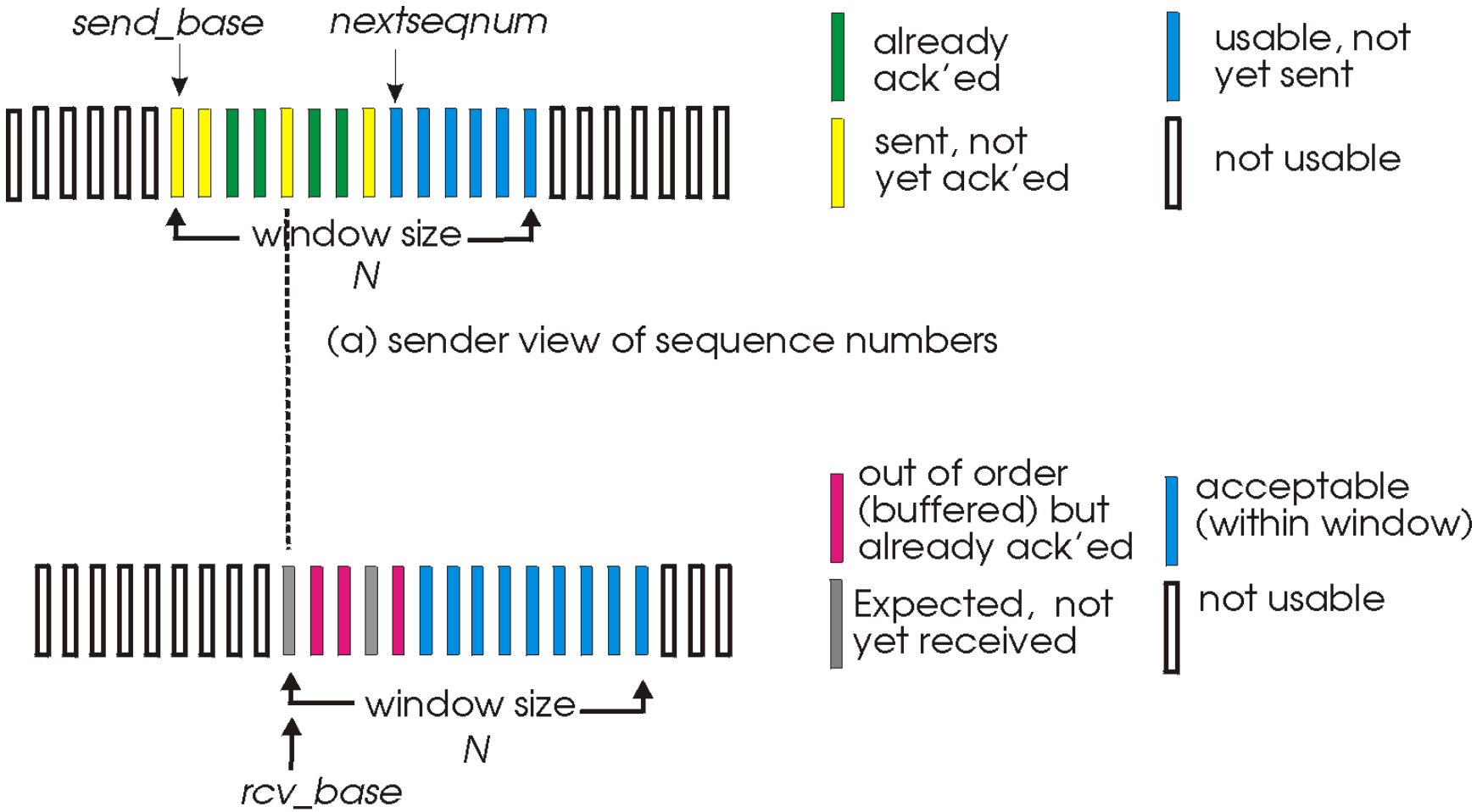
Go-back-N

- 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)

Selective Repeat



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

A. Go-back-N

B. Selective repeat

C. Something else (explain)

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

A. Go-back-N

B. Selective repeat

C. Something else (explain)



Sliding window

- How many bytes to pipeline?
- How big do we make that window?
  - Too small: link is under-utilized
  - Too large: congestion, packets dropped
  - Other concerns: fairness

Practical Reliability Questions

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

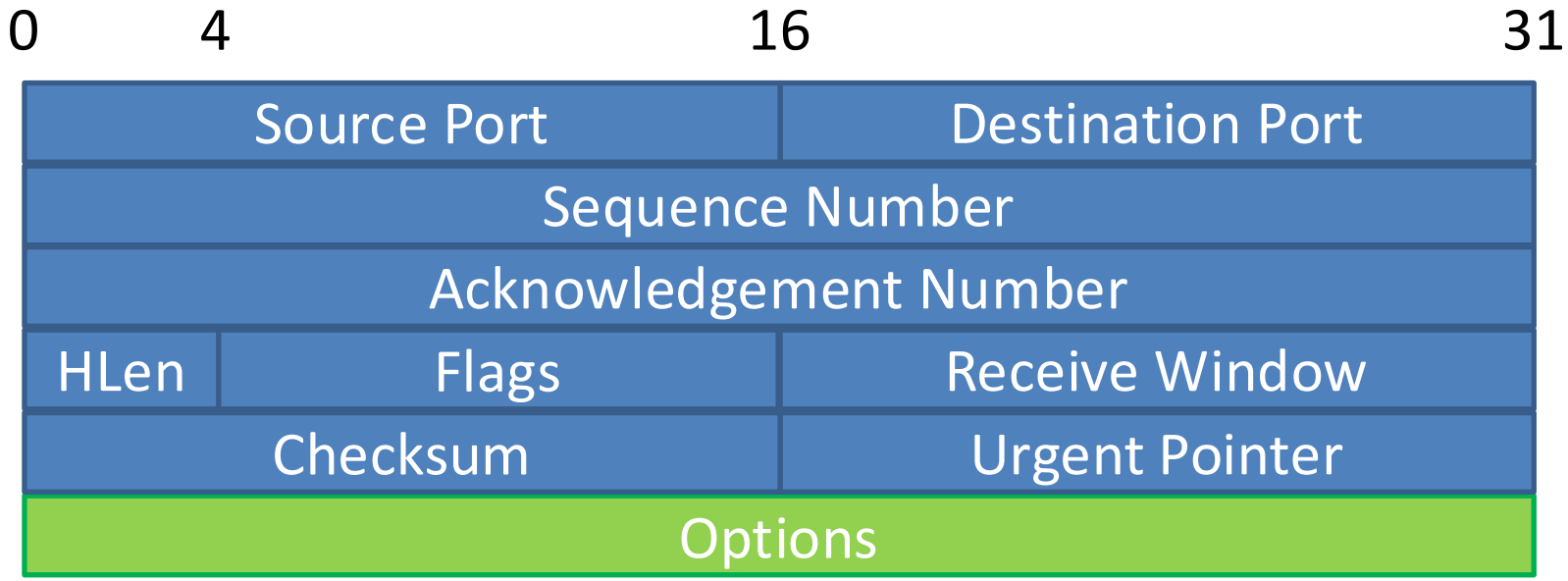
TCP Overview

- Point-to-point, full duplex
  - One pair of hosts
  - Messages in both directions
- Reliable, in-order byte stream
  - No discrete message
- 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

Transmission Control Protocol

Reliable, in-order, bi-directional byte streams

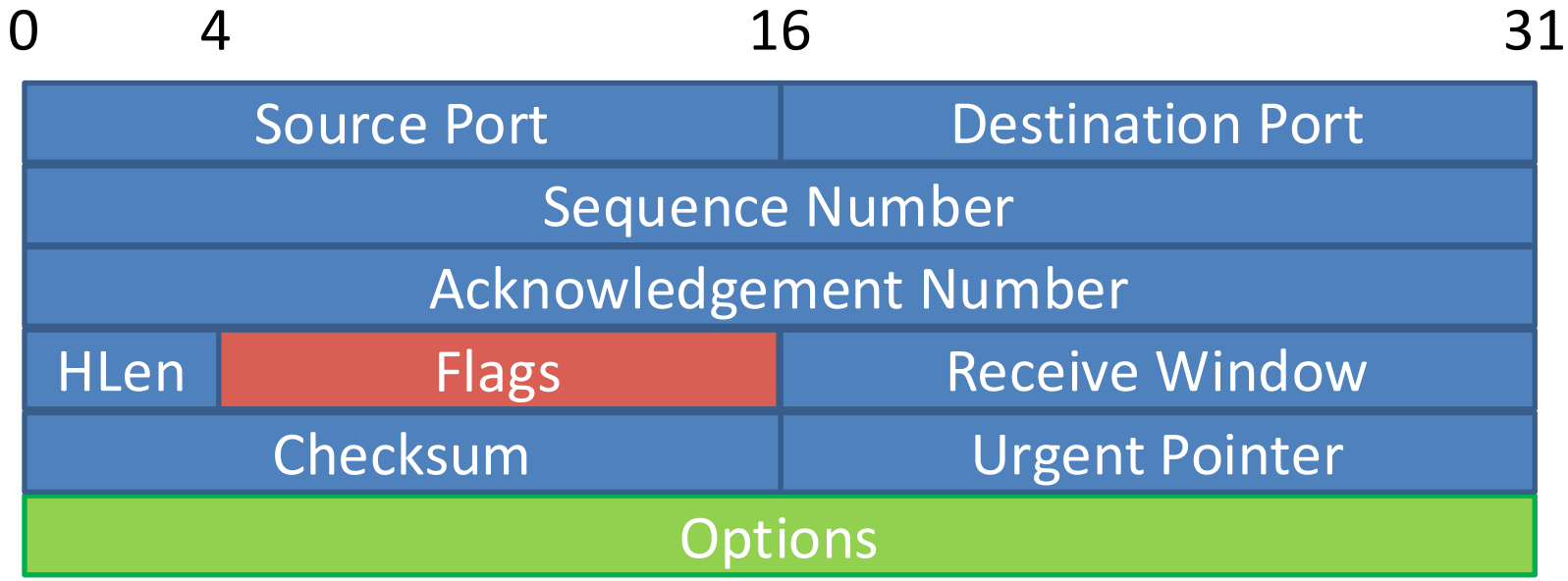
- Port numbers for demultiplexing
- Flow control
- Congestion control, approximate fairness



Transmission Control Protocol

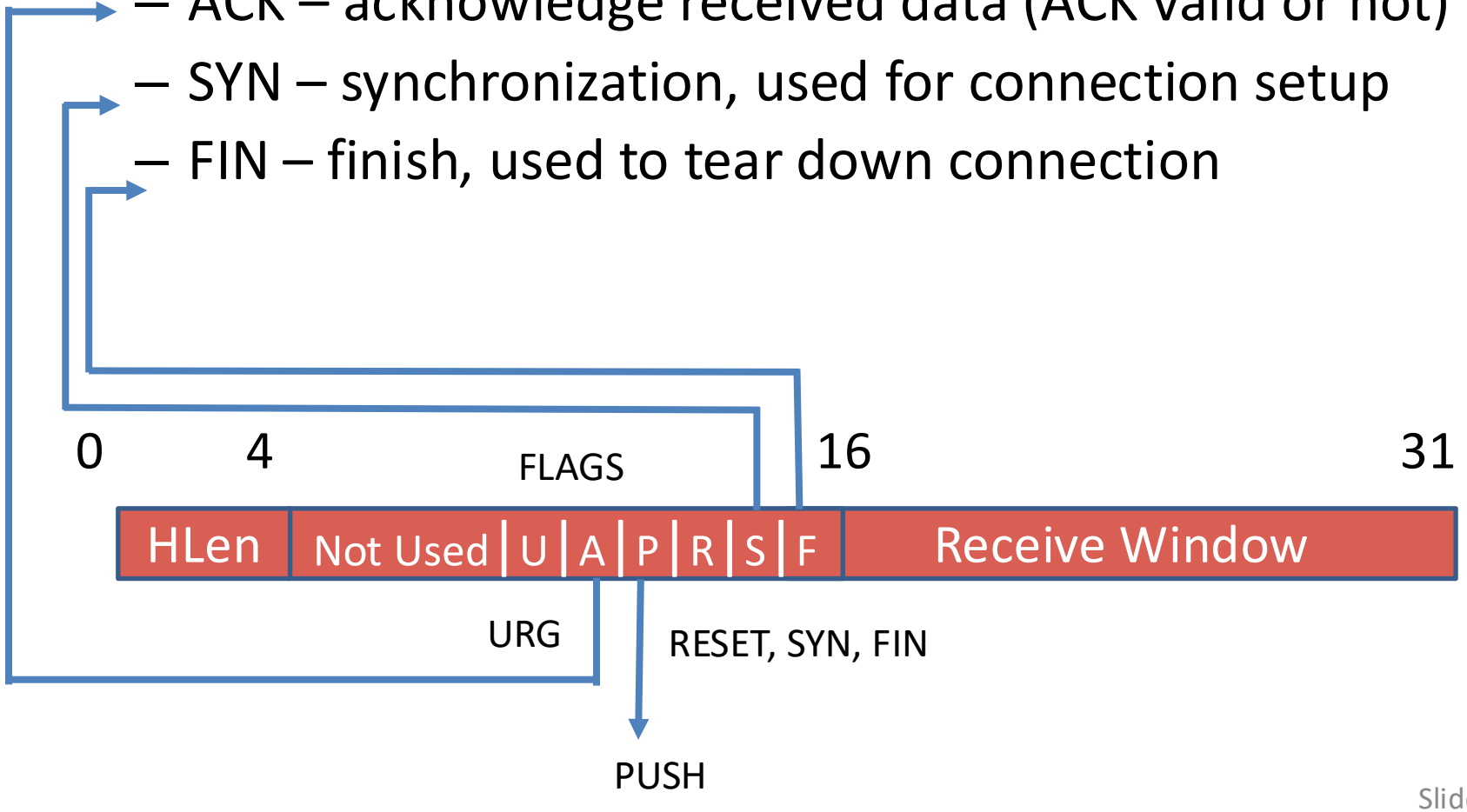
Reliable, in-order, bi-directional byte streams

- Port numbers for demultiplexing
- Flow control
- Congestion control, approximate fairness



Transmission Control Protocol

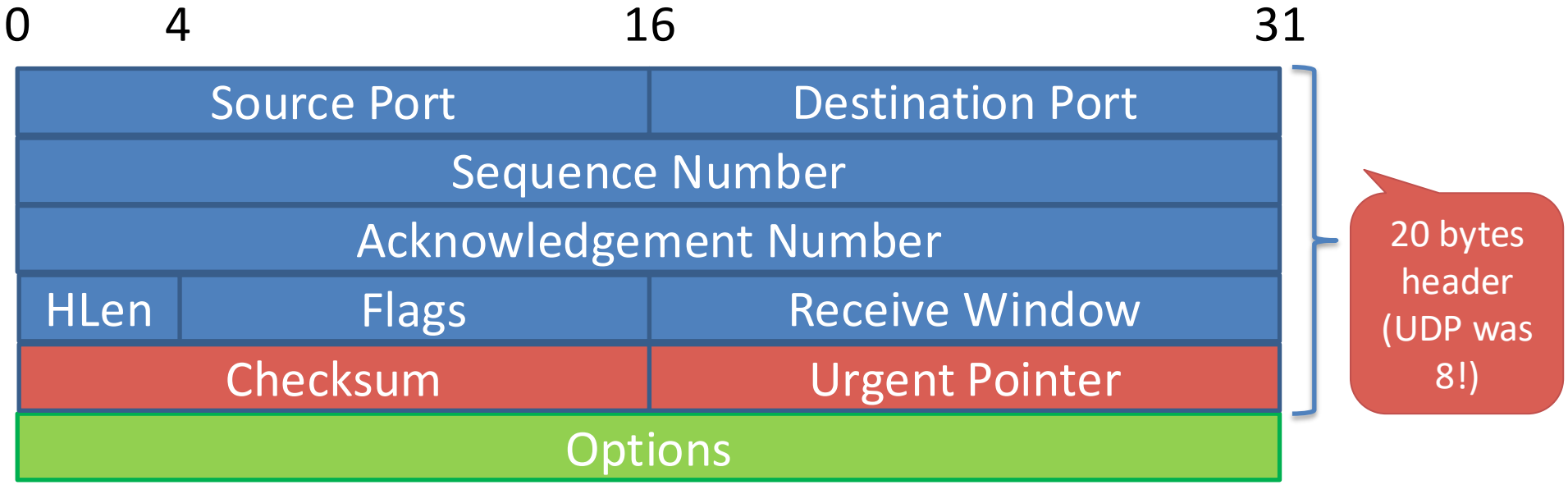
- Important TCP flags (1 bit each)
  - ACK – acknowledge received data (ACK valid or not)
  - SYN – synchronization, used for connection setup
  - FIN – finish, used to tear down connection



Transmission Control Protocol

Reliable, in-order, bi-directional byte streams

- Checksum: similar to TCP
- Urgent Pointer: Goes along with URG (U) flag in flags field
- Options: extensibility to TCP/not required



Practical Reliability Questions

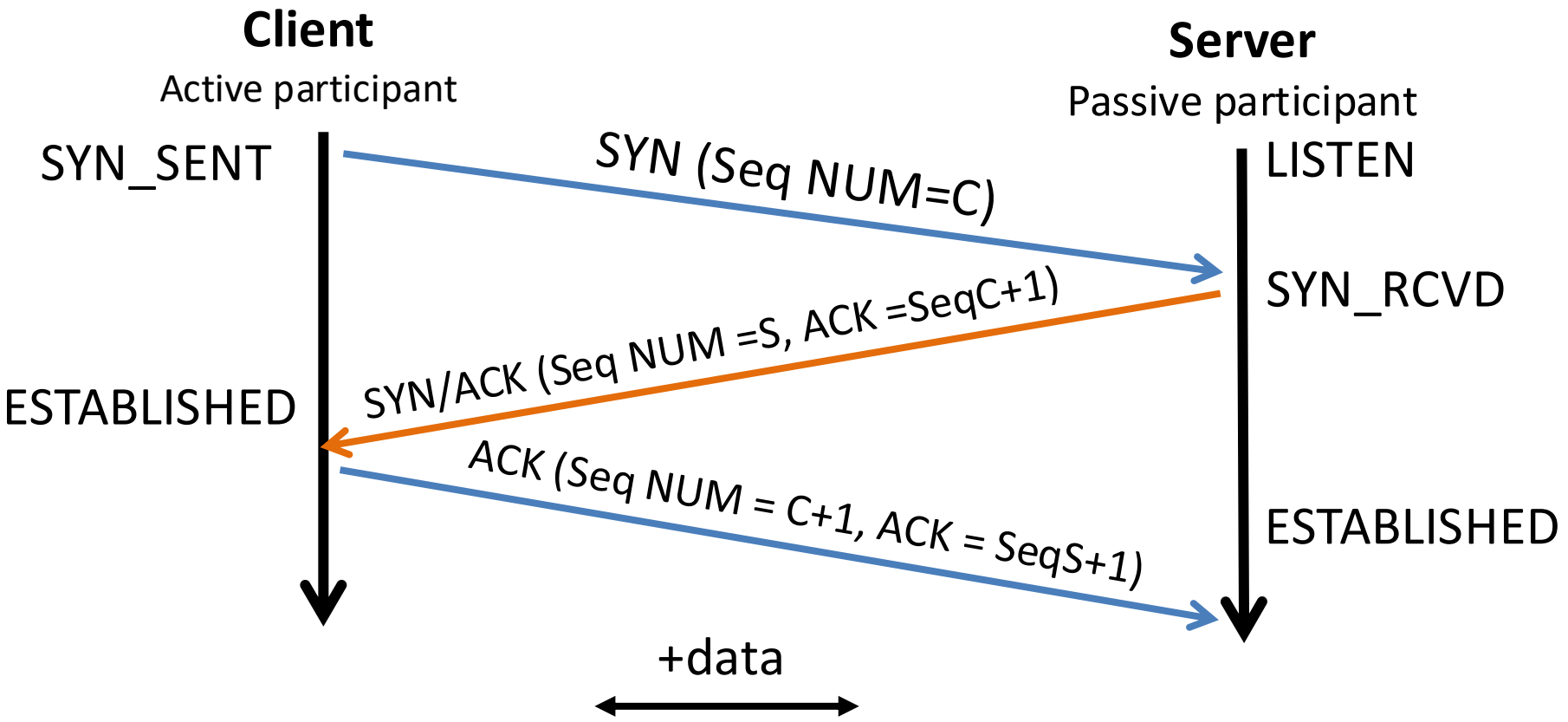
- What does connection establishment look like?
- How should we choose timeout values?
- How do the sender and receiver keep track of outstanding pipelined segments?
- How do we choose sequence numbers?
- How many segments should be pipelined?

Connections

- In TCP, hosts must establish a connection prior to communicating.

- Exchange initial protocol state.
  - sequence #s to use.
  - maximum segment size (MSS)
  - Initial window sizes, etc. (several parameters)

Three Way Handshake

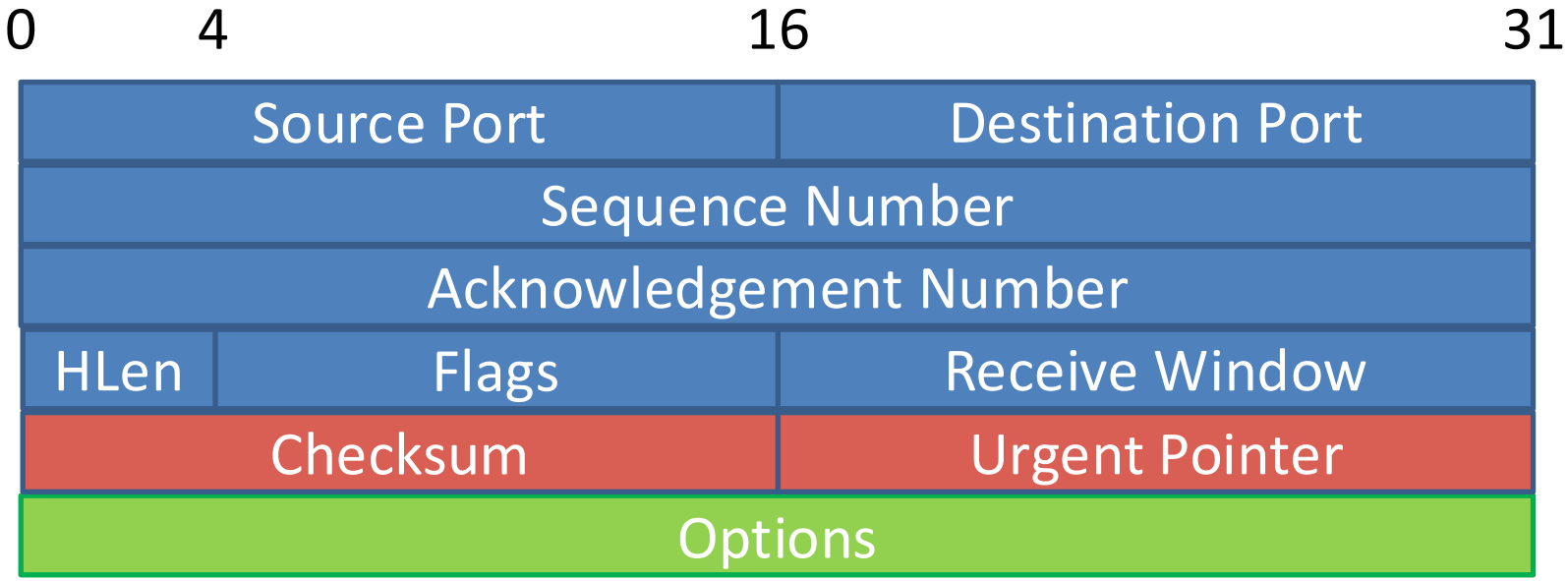


- Each side:
  - Notifies the other of starting sequence number
  - ACKs the other side's starting sequence number

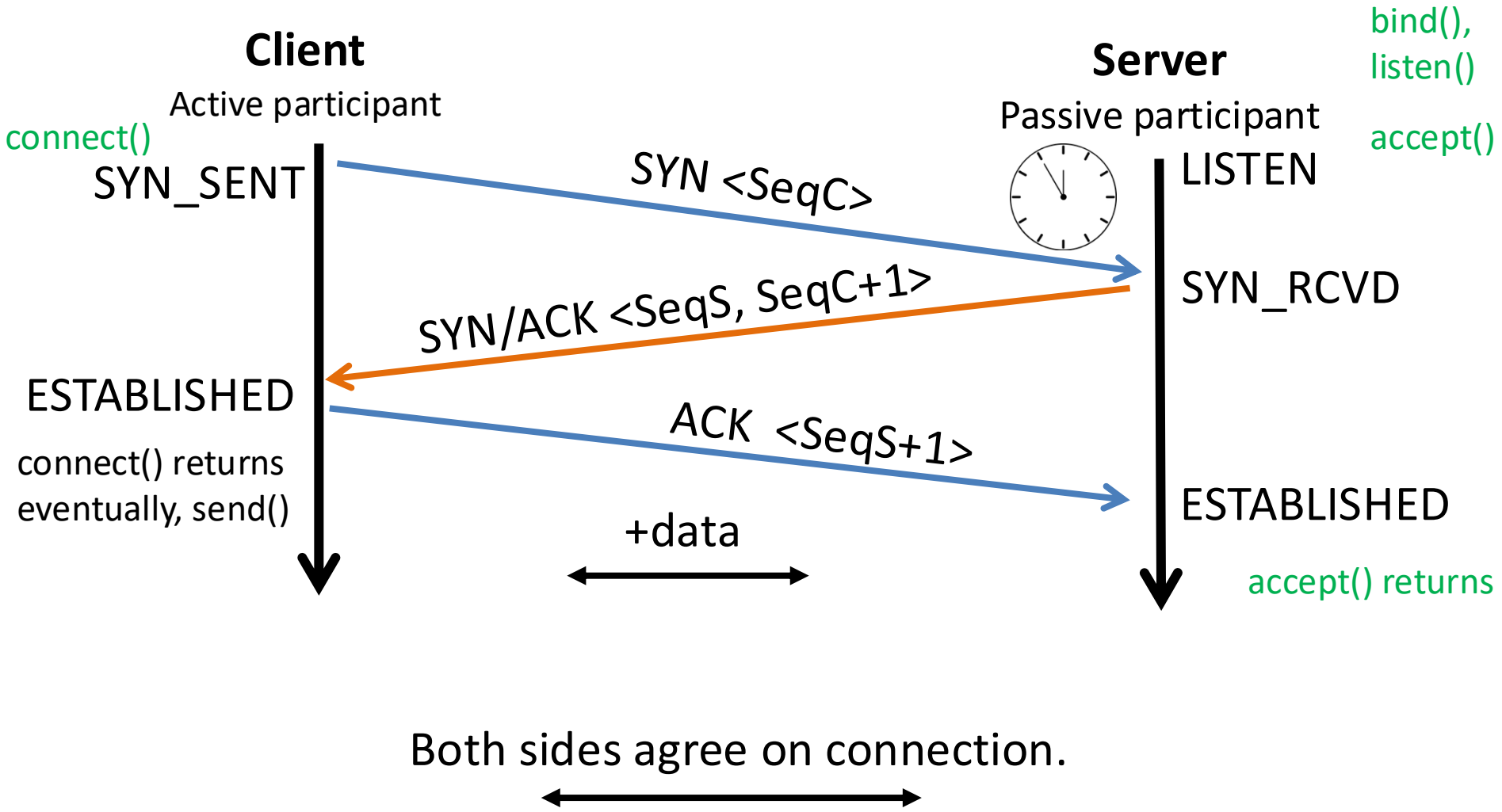
Transmission Control Protocol

Reliable, in-order, bi-directional byte streams

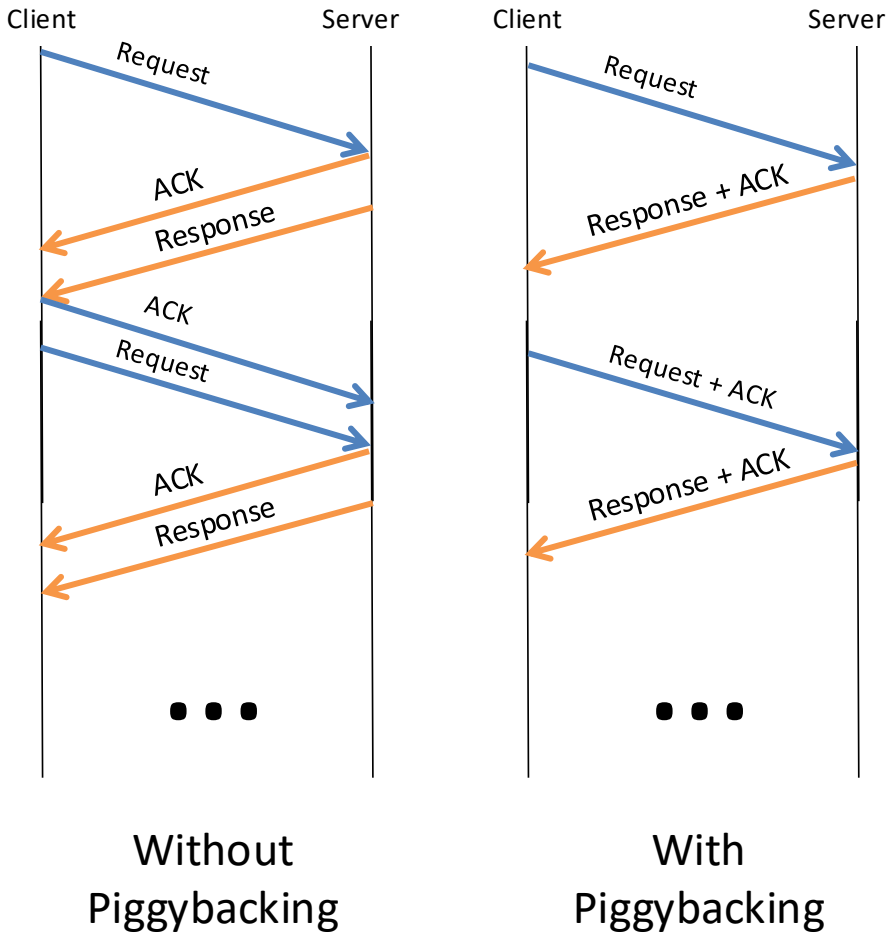
- Checksum: similar to TCP
- Urgent Pointer: Goes along with URG (U) flag in flags field
- Options: extensibility to TCP/not required



Three Way Handshake



Piggybacking



Initiator/Receiver

- Assumed distinct “sender” and “receiver” roles
- In reality, usually both sides of a connection send some data
- request/response is a common pattern

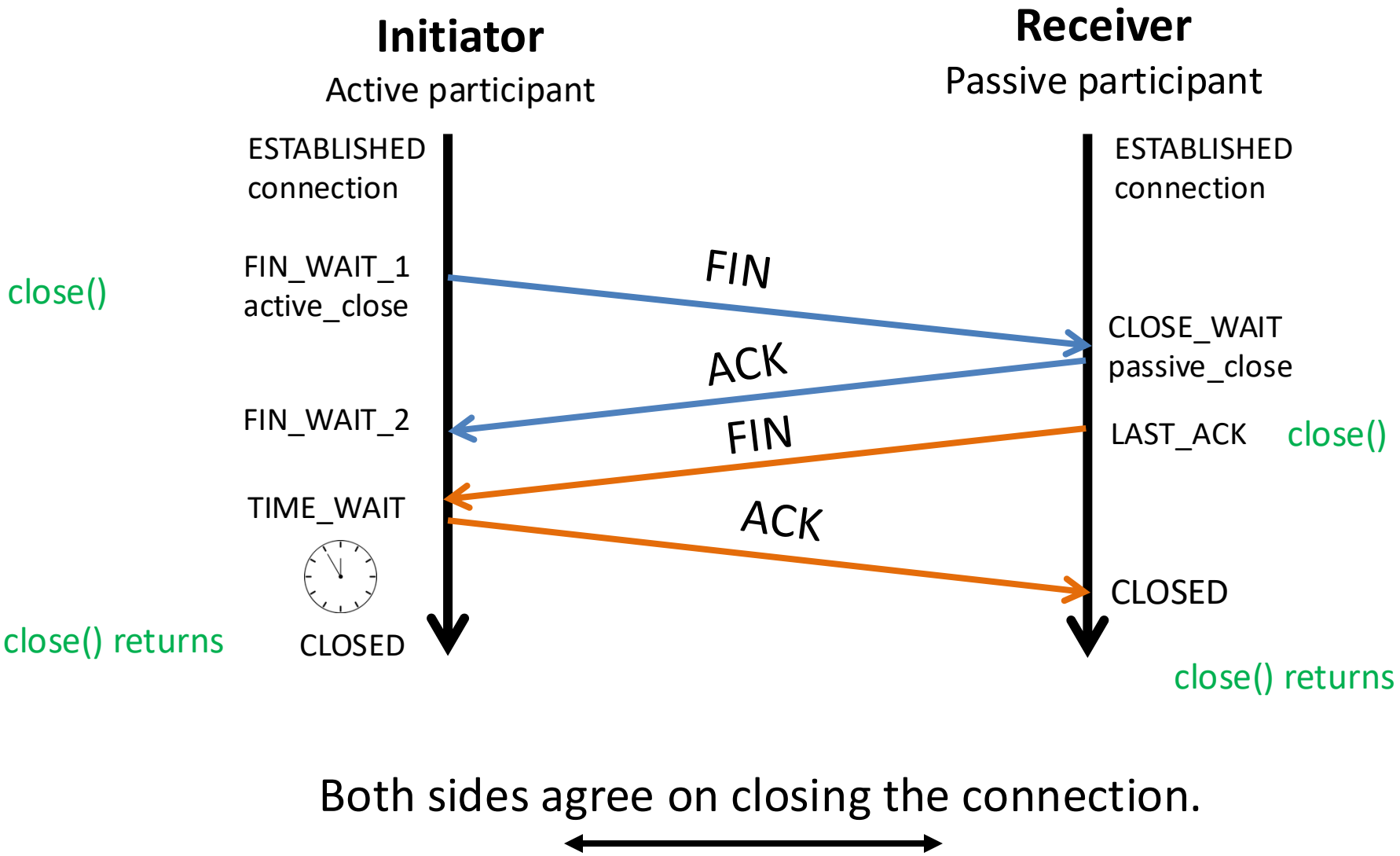
Initiator
Active participant

Receiver
Passive participant

Connection Teardown

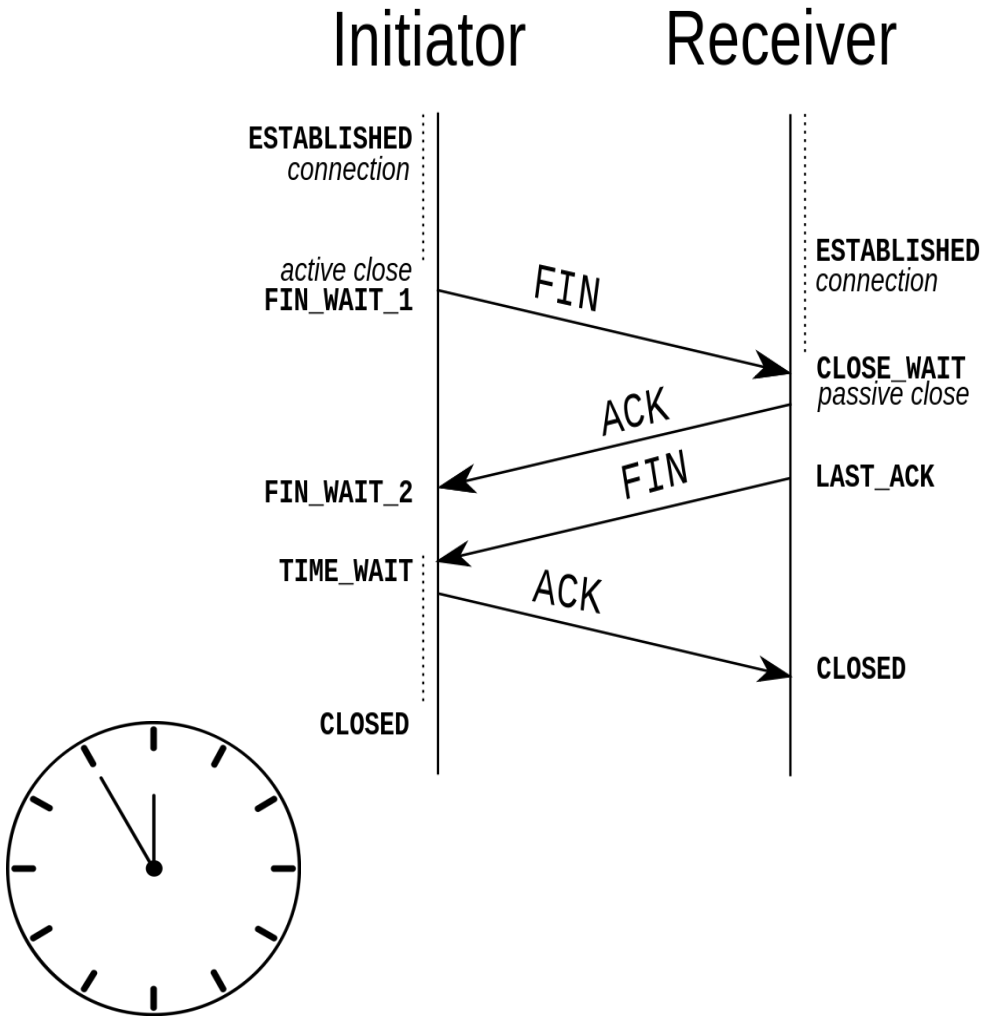
- Orderly release by sender and receiver when done
  - Delivers all pending data and “hangs up”
- Cleans up state in sender and receiver
- Each side may terminate independently

TCP Connection Teardown



Both sides agree on closing the connection.

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

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

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

Practical Reliability Questions

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

Timeouts

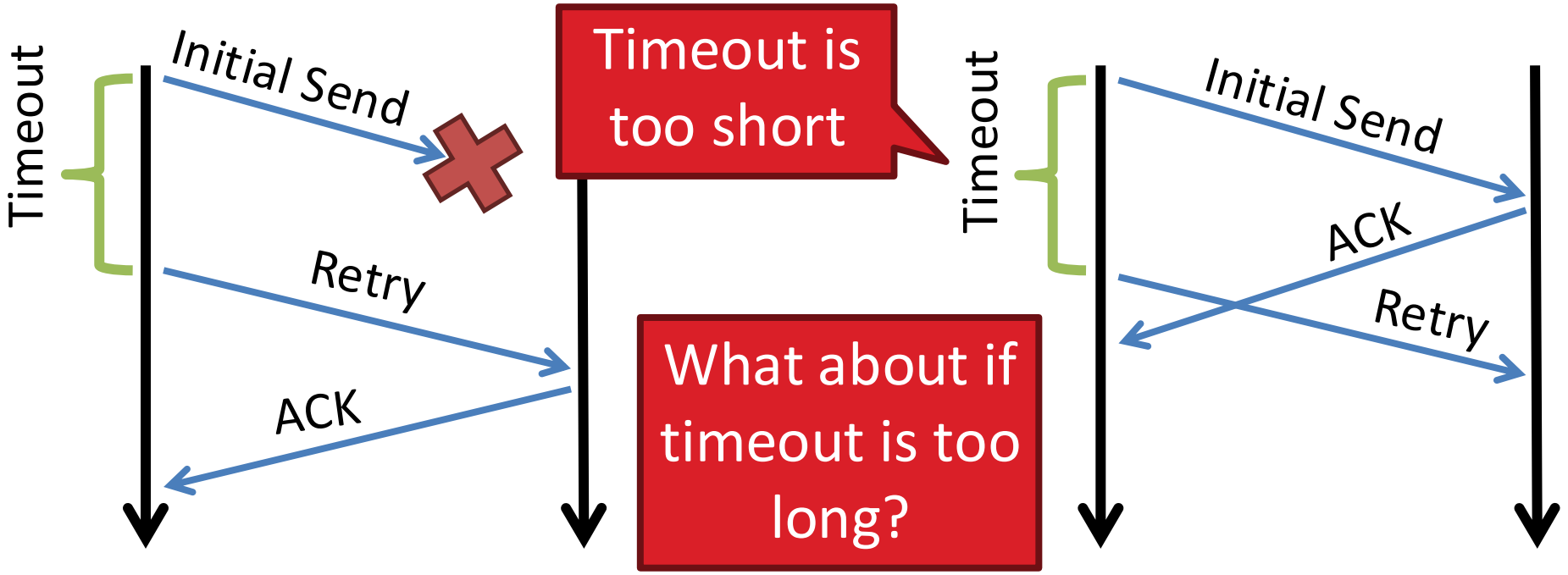
- How long should we wait before timing out and retransmitting a segment?

- Too short: needless retransmissions
- Too long: slow reaction to losses

- Should be (a little bit) longer than the RTT

Retransmission Timeouts

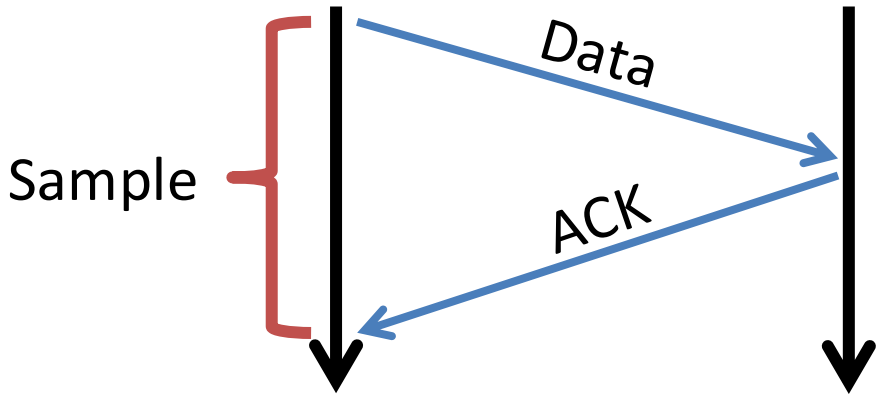
- Problem: time-out is linked to round trip time



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

Round Trip Time Estimation:
Exponentially Weighted Moving Average (EWMA)



EstimatedRTT = (1 – a) * EstimatedRTT + a * SampleRTT

- a is usually 1/8.

In words current estimate is a blend of:

- 7/8 of the previous estimate
- 1/8 of the new sample.

DevRTT = (1 – B) * DevRTT + B * | SampleRTT – EstimatedRTT |

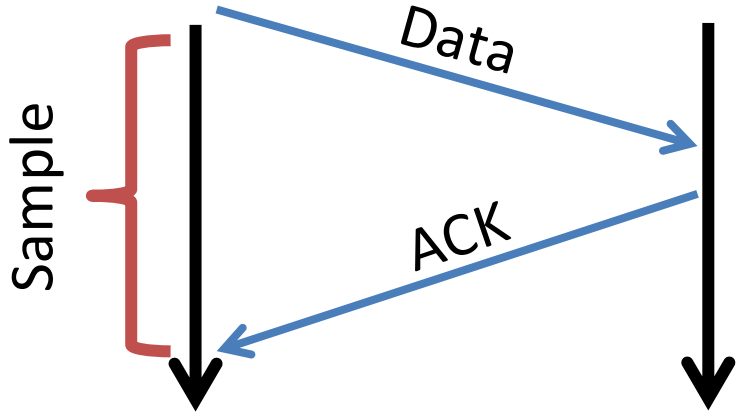
- B is usually 1/4

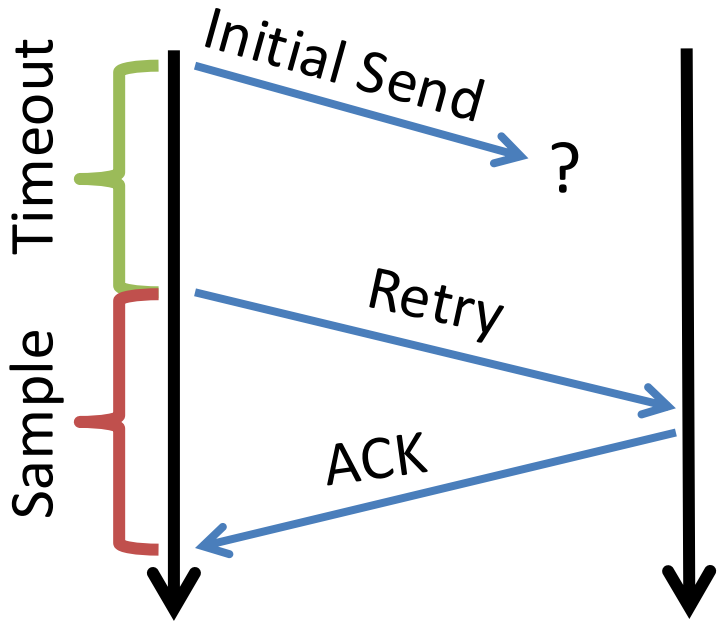
Estimating RTT

- 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

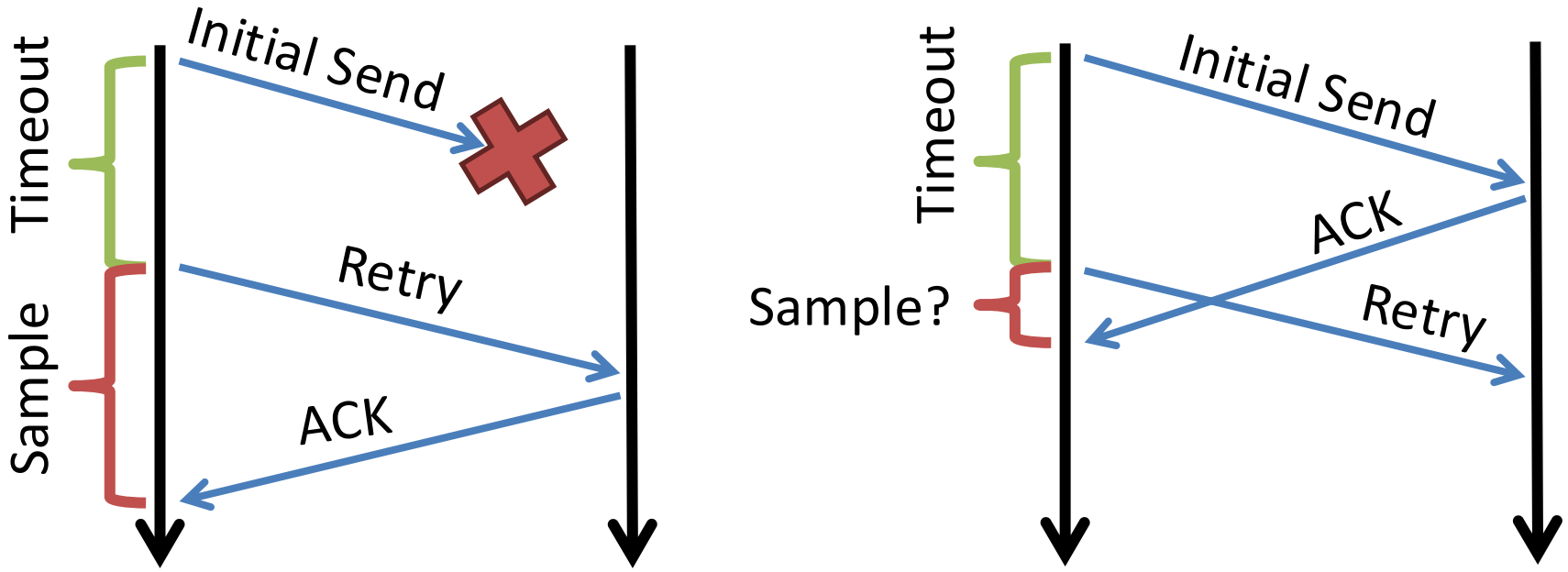
- Exponentially Weighted Moving Average (EWMA)

Round Trip Time Estimation:
Exponentially Weighted Moving Average (EWMA)





RTT Sample Ambiguity



Ignore samples for retransmitted segments

TCP Timeout Value

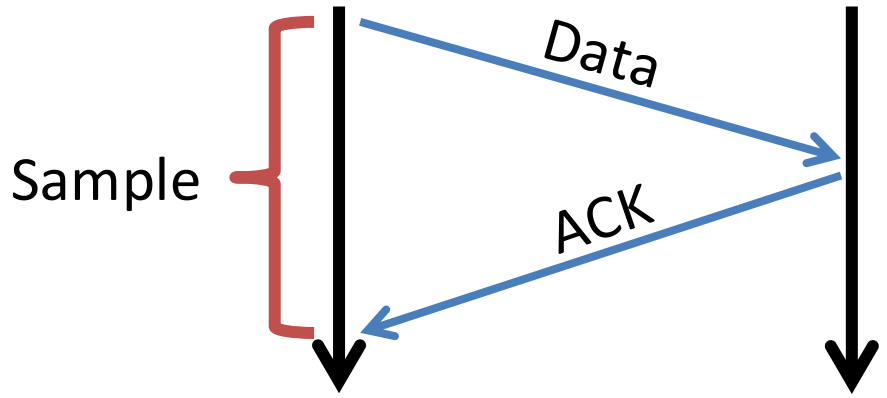
`TimeoutInterval = EstimatedRTT + 4*DevRTT`



estimated RTT

“safety margin”

Round Trip Time Estimation:
Exponentially Weighted Moving Average (EWMA)



EstimatedRTT = (1 – a) * EstimatedRTT + a * SampleRTT

- a is usually 1/8.

In words current estimate is a blend of:

- 7/8 of the previous estimate
- 1/8 of the new sample.

DevRTT = (1 – B) * DevRTT + B * | SampleRTT – EstimatedRTT |

- B is usually 1/4

Example RTT Estimation

- Suppose EstimateRTT = 64, Dev = 8
- Latest sample: 120

New estimate = 7/8 * 64 + 1/8 * 120 = 56 + 15= 71
New dev = 3/4 * 8 + 1/4 * | 120 - 71 | = 6 + 12 = 18

- Another sample: 400

New estimate = 7/8 * 71 + 1/8 * 400 = 62 + 50 = 112
New dev = 3/4 * 18 + 1/4 * | 400 - 112 | = 13 + 72 = 85

Example RTT Estimation (Smoothing)

