Reliability and Flow control

TCP revisited

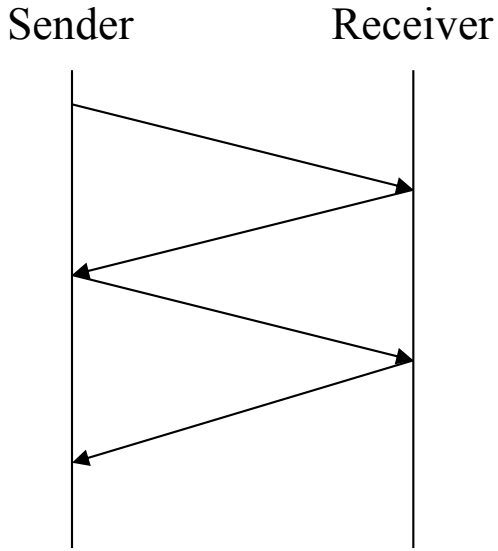
- TCP provides the following abstraction
  - In-order delivery
  - **Reliability**

- Reliability while improving performance requires
  - **Flow control**
  - **Congestion control**

Two common ways for reliability in networking

- Automatic Repeat Request (ARQ)
  - Usually used in all layers.
- Forward Error Correction (FEC)
  - Usually used in lower layers

Simple ARQ solution: Stop and Wait

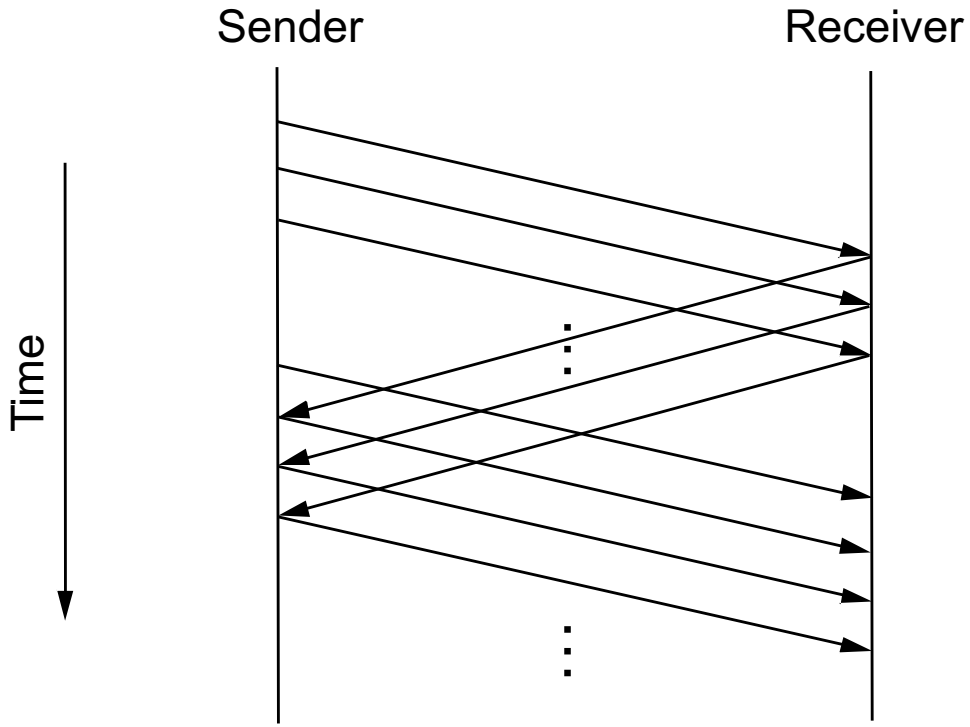


- Sender doesn’t send next packet until she’s sure receiver has last packet
- What are the problems with Stop and Wait?

Performance problem with stop and wait

- Problem: “keeping the pipe full”
  - If the bandwidth-delay product [BDP] is much larger than a packet size, the sender will be unable to keep the link busy
  - **Bandwidth-delay product** estimates the amount of data that can be pushed on the pipe

Sliding window



Sliding window

- TCP sends a window size of packets instead of stop-and-wait
- For window size *n*, sender may transmit *n* bytes without receiving an ACK
  - After each ACK, the window slides forward

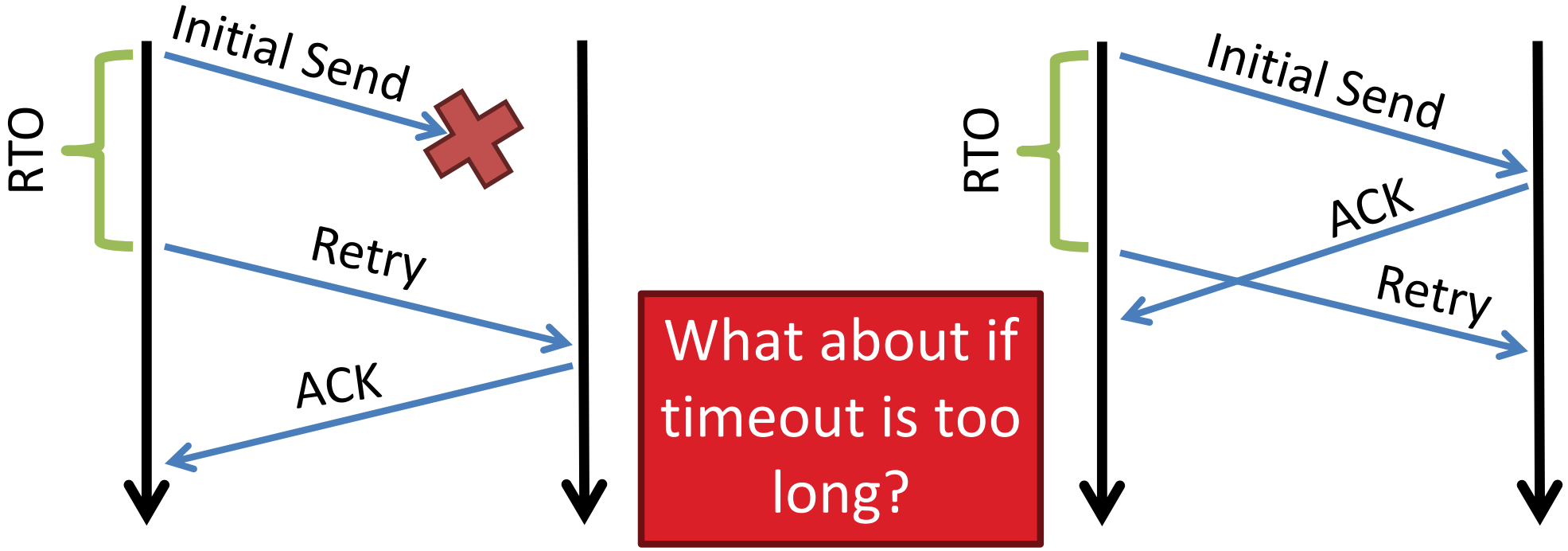
What happens when packets are lost?

Timeouts

- Lost segments detected by sender
  - Use timeout to detect missing ACKs
  - What should the timeout depend on?

Retransmission Time Outs (RTO)

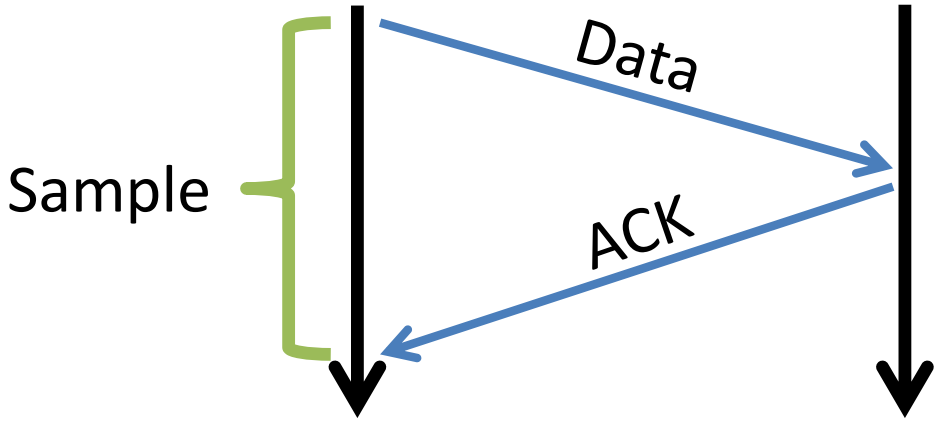
- Problem: time-out could be too short



TCP's timeout

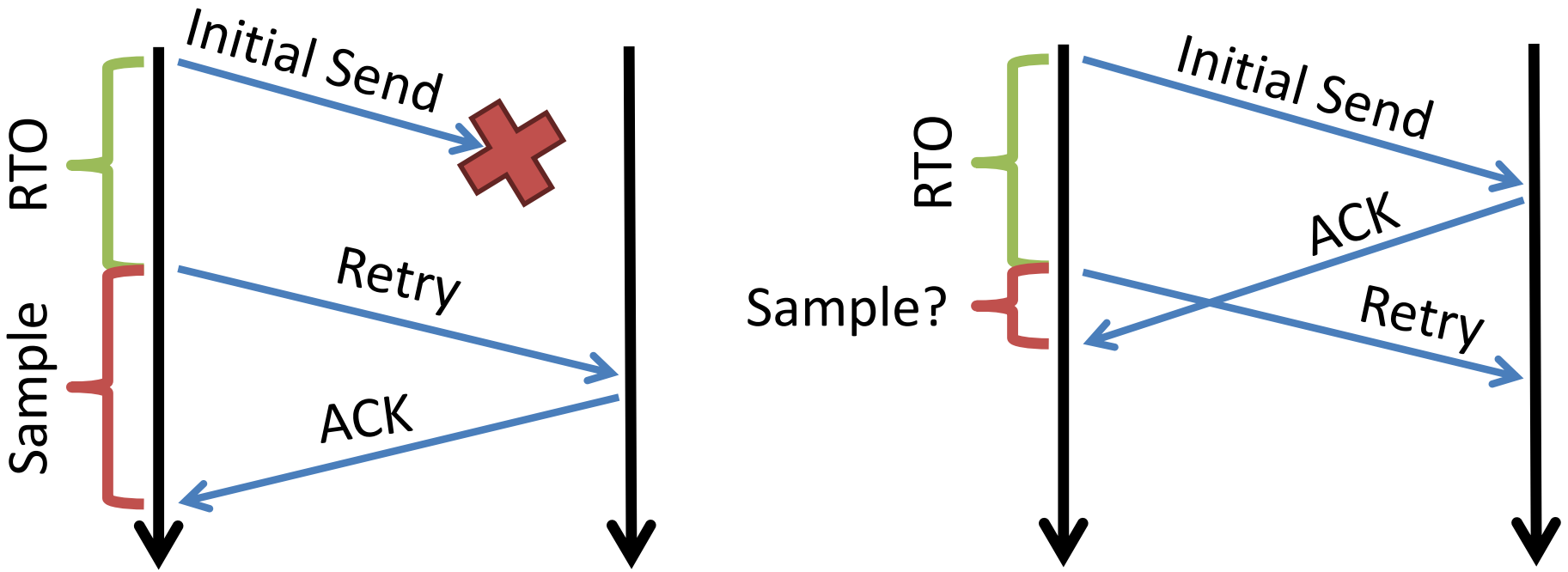
- One idea
  - RTO = 2 * RTT'
  - RTT' is the estimated RTT
  - TCP is conservative
- RFC gives the exact RTO measure
  - https://tools.ietf.org/html/rfc6298
- How should we estimate RTT?

Round Trip Time Estimation



- RTT = (1-α) (RTT) + α(new_sample)
  - Recommended α: 0.125

RTT Sample Ambiguity



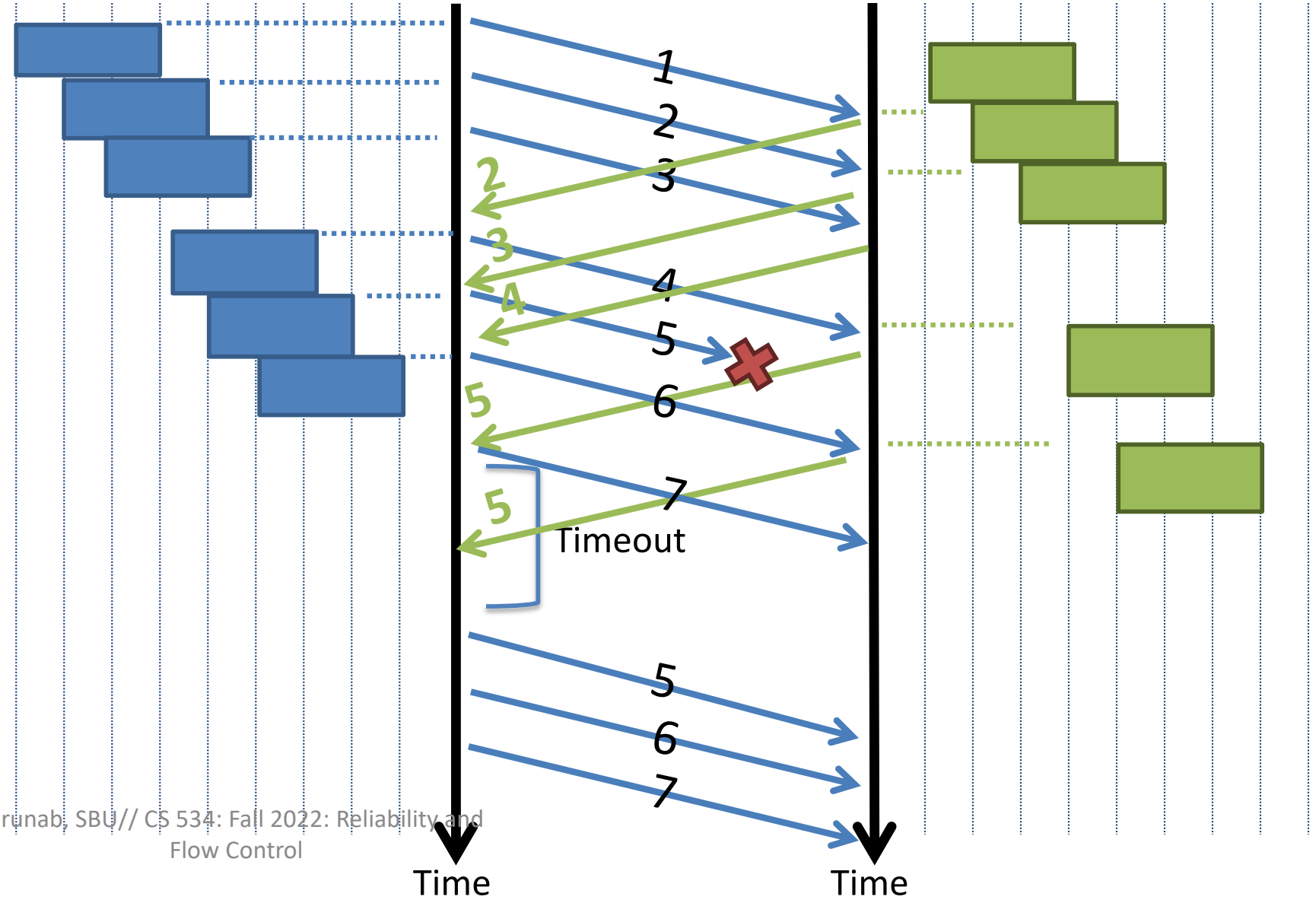
- Karn's algorithm: ignore samples for retransmitted segments

When should the Receiver ACK?

1. ACK every packet
2. Use *cumulative ACK*, where an ACK for sequence *n* implies ACKS for all $k < n$

Later we discuss triple duplicate acks, NACKs, and SACKs

Example: Cumulative acks
Sliding Window = 3



Sliding window problem

- Determine how many packets can be sent into the system
- Send too many
  - Can overwhelm the receiver
  - Can overwhelm the network
- Solution:
  - Flow control (not to overwhelm the receiver)
  - Congestion control (not to overwhelm the network)

Bandwidth delay product (BDP)

- It is theoretically the maximum number of packets you can send before receiving an acknowledgment
- Since acknowledgement takes roughly RTT time to get, the number of packets you can send in one RTT

  BDP = bandwidth x RTT
- In general, you cannot send BDP packets in the network because there are other bottlenecks
  - Receiver bottleneck and Network bottleneck

Receiver bottleneck and network bottleneck

- Receiver centric: Receiver does not have a big enough buffer to store un-ordered packets
  - Solution: Flow control
- Network centric: The routers do not have capacity to process the packets, causing queue buildup and packet drops
  - Solution: Congestion control

Flow Control

- Problem: how many packets should a sender transmit?
  - Too many packets may overwhelm the receiver
- Solution
  - Receiver tells the sender how big their buffer is
  - This is called the advertised window

Flow Control: Sender Side