

Lecture 5: Flow Control

CSE 123: Computer Networks
Aaron Schulman





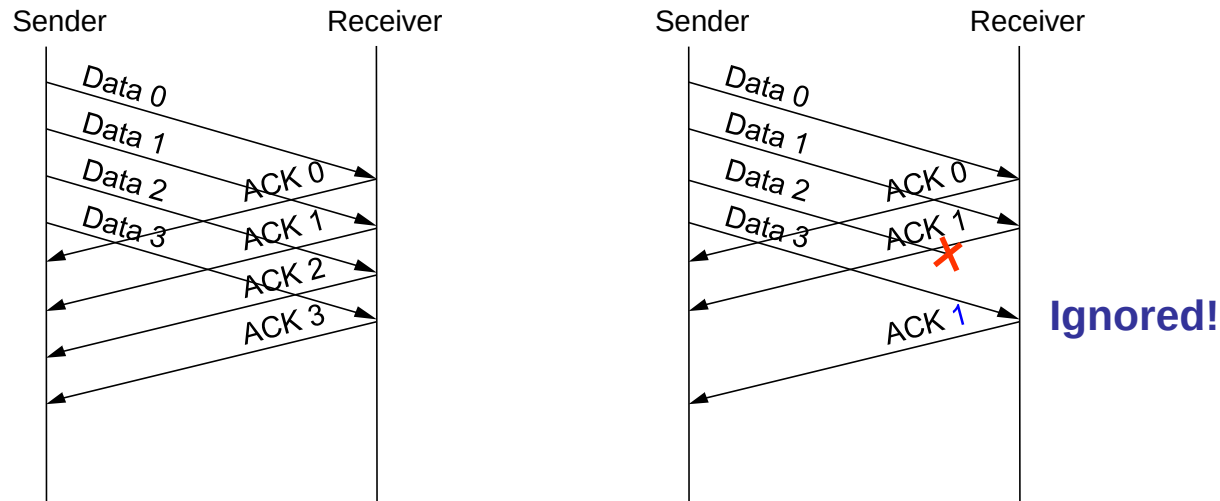
Review: Stop-and-Wait



- Lousy performance if $x_{\text{mit}} 1 \text{ pkt} \ll \text{prop. delay}$
 - ◆ How bad?
- Want to utilize all available bandwidth
 - ◆ Need to keep more data “in flight”
 - ◆ How much? Called the **bandwidth-delay product**
- Also limited by quality of timeout (how long?)



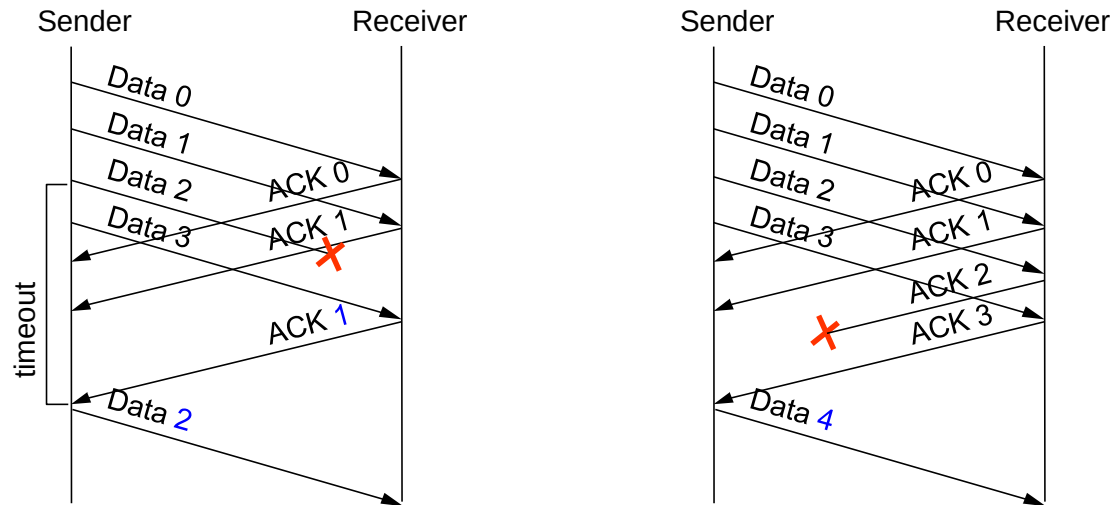
Review: Pipelined Transmission



- Keep multiple packets “in flight”
 - ◆ Allows sender to make efficient use of the link
 - ◆ Sequence numbers ensure receiver can distinguish frames
- Sender buffers outstanding un-acked packets
 - ◆ Receiver ACKs the highest *consecutive* frame received
 - » ACKs are **cumulative** (covers current frame and all previous)



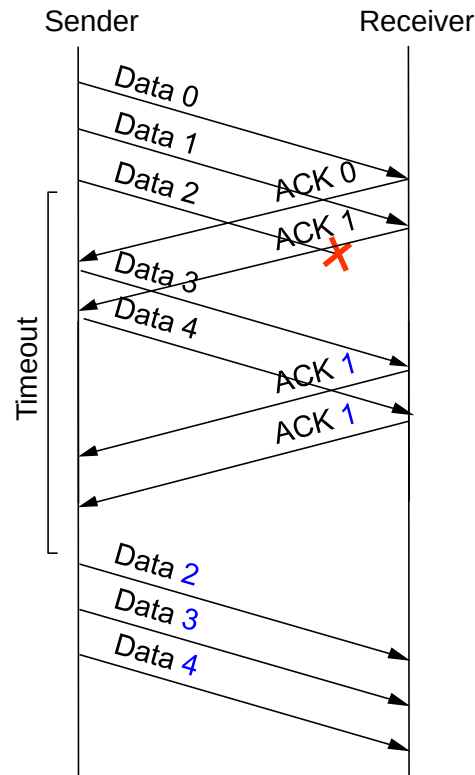
Go-Back-N



- Retransmit all packets from point of loss
 - ◆ Packets sent after loss event are ignored (i.e., sent again)
- Simple to implement (receiver very simple)
- Sender controls how much data is “in flight”



Send Window



**Go-Back-N Example
with window size 3**

- Bound on number of outstanding packets
 - ◆ Window “opens” upon receipt of *new* ACK
 - ◆ Window resets entirely upon a timeout
- Limits amount of waste
 - ◆ Still lots of duplicates
 - ◆ We can do better with **selective retransmission**



Sliding Window

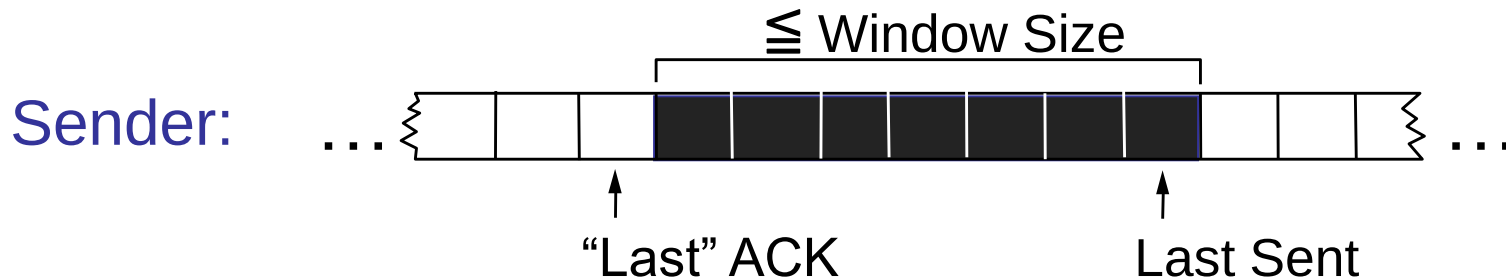
- Single mechanism that supports:
 - ◆ Multiple outstanding packets
 - ◆ Reliable delivery
 - ◆ In-order delivery
 - ◆ Flow control

- Sender and receiver each maintain “window” abstractions to track outstanding packets
 - ◆ At the core of all modern ARQ protocols

- Go-Back-N is a special case
 - ◆ Receive window size of one

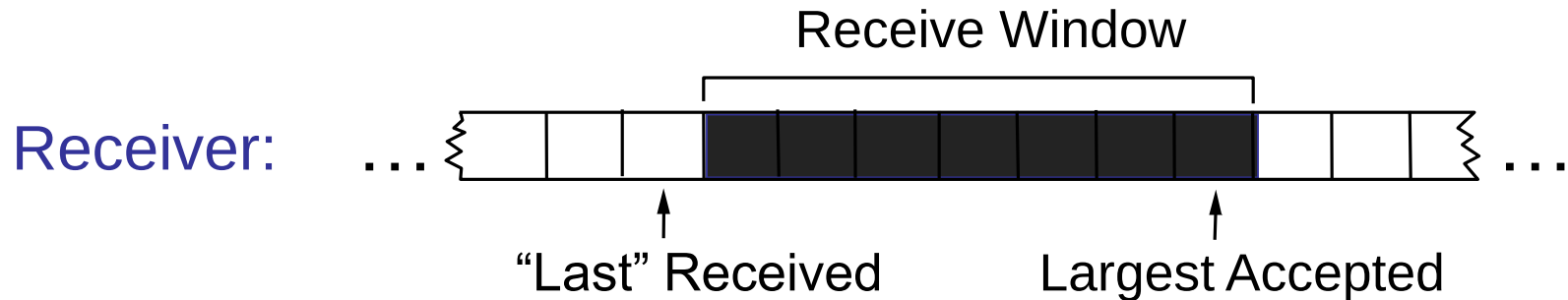


Sliding Window – Sender



- Window bounds outstanding unACKed data
 - ◆ Implies need for buffering at sender
- “Last” ACK applies to **in-order** data
- What to do on a timeout?
 - ◆ Go-Back-N: resend all unacknowledged data on timeout
 - ◆ Selective Repeat: timer per packet, resend as needed

Sliding Window – Receiver



- Receiver buffers too:
 - ◆ data may arrive out-of-order
 - ◆ or faster than can be consumed
 - » [Flow control](#): tell sender how much buffer left at receiver
- Receiver ACK choices:
 - ◆ Cumulative, Selective (exempt missing frames), Negative

Deciding When to Retransmit



- How do you know when a packet has been lost?
 - ◆ Ultimately sender uses timers to decide when to retransmit

- But how long should the timer be?
 - ◆ Too long: inefficient (large delays, poor use of bandwidth)
 - ◆ Too short: may retransmit unnecessarily (causing extra traffic)

- Right timer is based on the **round-trip time** (RTT)
 - ◆ Which can vary greatly for reasons well see later

Can we shortcut the timeout?

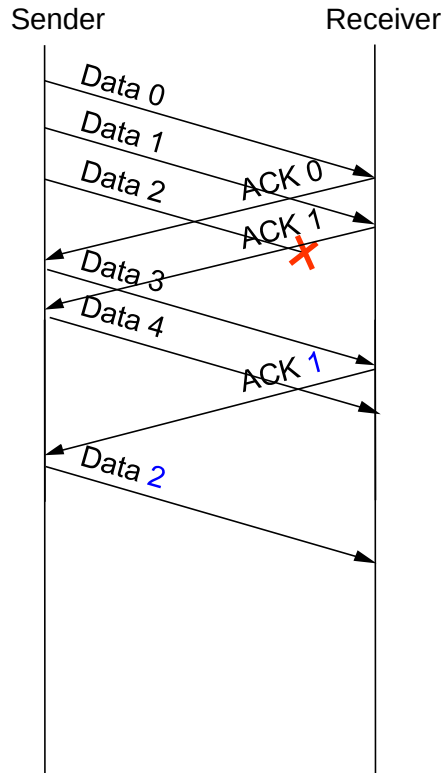


- Timeout is long in practice
 - ◆ Lots of variation in RTT and timeout must be conservative

- If packets are usually *in order* then *out-of-order* ACKs imply that a packet was lost
 - ◆ Negative ACK
 - » Receiver requests missing packet
 - ◆ Fast retransmit
 - » When sender receives multiple duplicate acknowledgements resends missing packet



Fast retransmit



- Don't bother waiting
 - ◆ Receipt of duplicate acknowledgement (**dupACK**) indicates loss
 - ◆ Retransmit immediately
- Used in TCP
 - ◆ Need to be careful if frames can be reordered
 - ◆ Today's TCP identifies a loss if there are **three** duplicate ACKs in a row



For Next Time

- ▣ Read 5-5.1 in P&D
- ▣ Homework (finally) out this evening and due in 1 week
- ▣ (Keep) going on the project...