

Lecture 4:

Reliable Transmission

CSE 123: Computer Networks

Aaron Schulman



Project 1a due one week from Friday





Lecture 4 Overview

- Handling errors
 - ◆ Automatic Repeat Request (ARQ)
 - ◆ Acknowledgements (ACKs) and timeouts
 - ◆ Stop-and-Wait
- Flow control
 - ◆ Sliding window

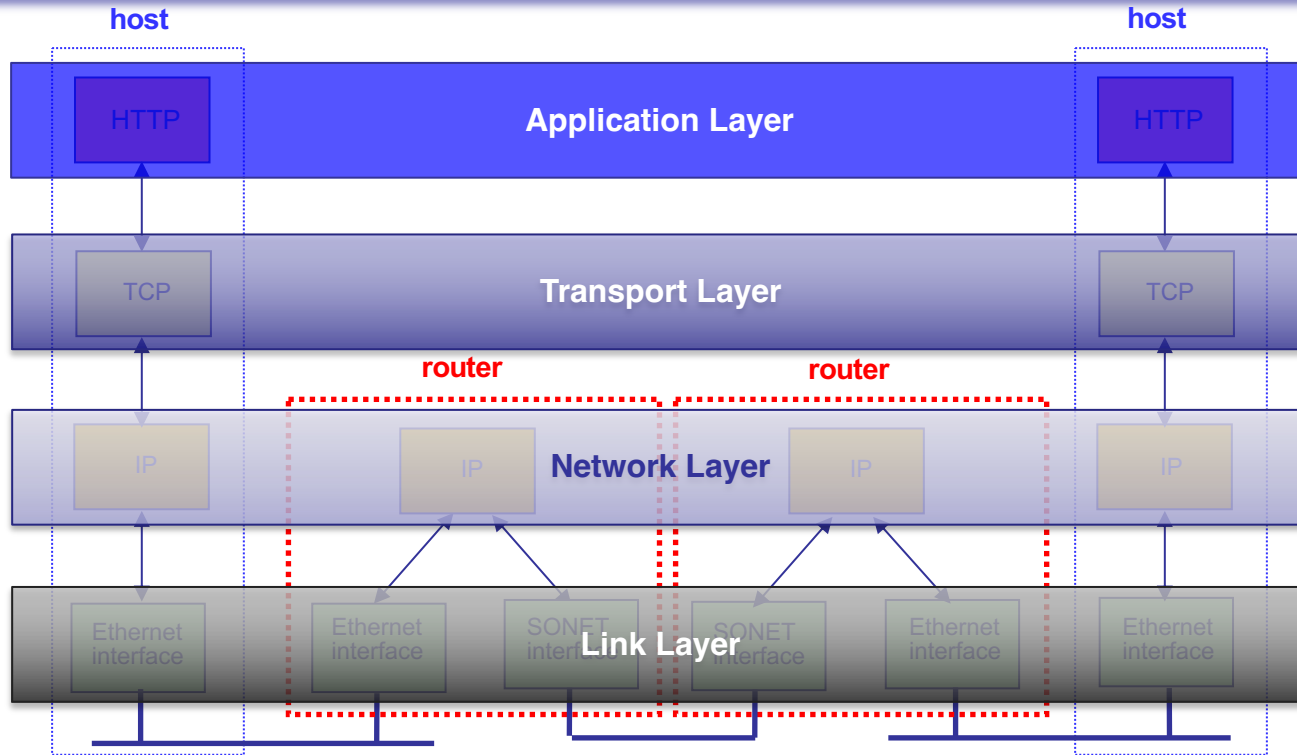


Picking up the Pieces

- Link layer is lossy
 - ◆ We deliberately throw away corrupt frames!
 - ◆ Infrequent bit errors still lead to occasional frame errors
 - » 10,000+ bits in each frame
- Things get even harrier if we consider multiple links
 - ◆ In a few lectures, we'll start sending frames on long trips
 - ◆ Each intermediate stop might lose, corrupt, *reorder*, etc.
 - ◆ Regardless of cause, we'll call loss events **drops**
- We want to provide reliable, in-order delivery
 - ◆ Can—and will—do this at multiple layers



Moving up the Stack



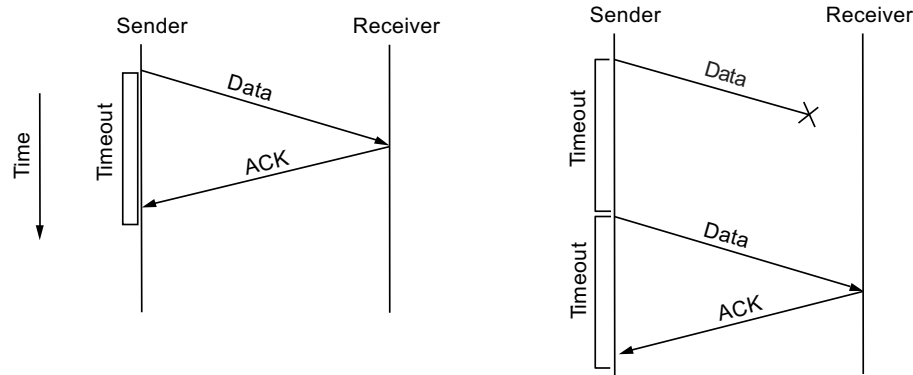


Reliable Transmission

- The data networking version of the same problem
 - ◆ How do we reliably send a message when packets can be lost/corrupted in the network?
- Two options
 - ◆ **Detect a loss/corruption and retransmit**
 - ◆ Send data redundantly to tolerate loss/corruption



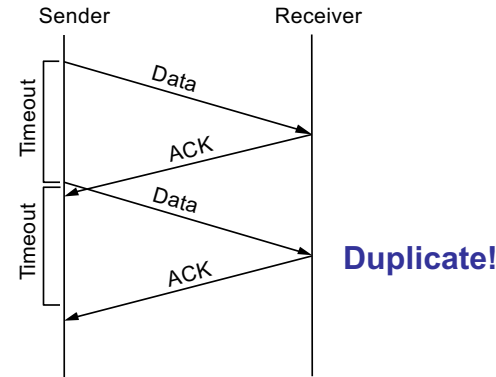
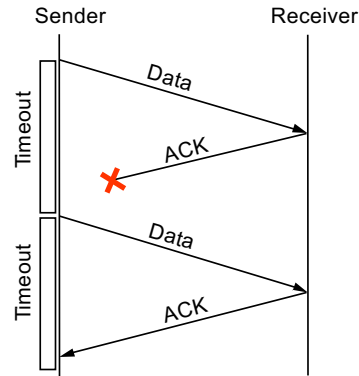
Simple Idea: ARQ



- Receiver sends **acknowledgments** (ACKs)
 - ◆ Sender “times out” and retransmits if it doesn’t receive them
- Basic approach is generically referred to as **Automatic Repeat Request** (ARQ)



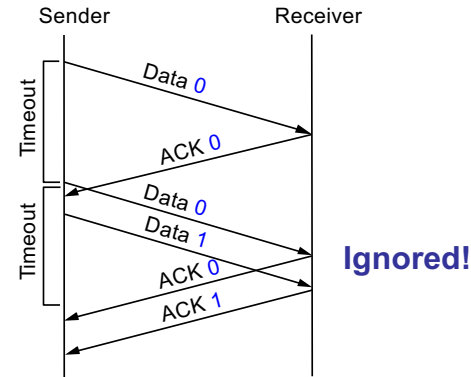
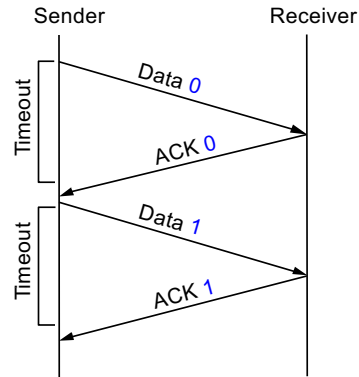
Not So Fast...



- Loss can occur on ACK channel as well
 - ◆ Sender cannot distinguish data loss from ACK loss
 - ◆ Sender will retransmit the data frame
- ACK loss—or early timeout—results in duplication
 - ◆ The receiver thinks the retransmission is new data



Sequence Numbers



- Sequence numbers solve this problem
 - ◆ Receiver can simply ignore duplicate data
 - ◆ But must still send an ACK! (Why?)
- Simplest ARQ: **Stop-and-wait**
 - ◆ Only one outstanding frame at a time



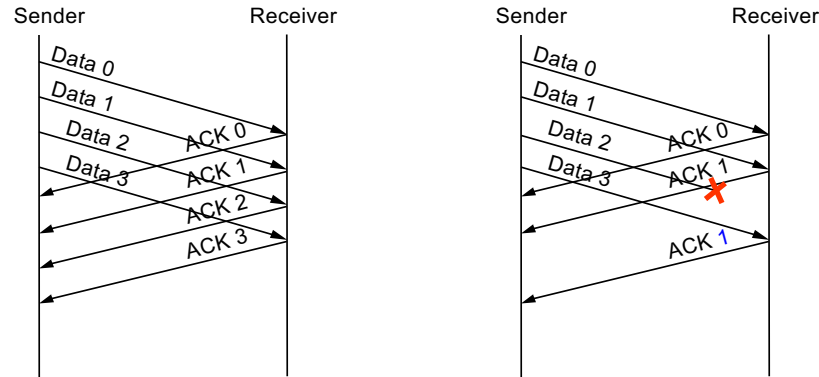
Stop-and-Wait Performance



- Lousy performance if time to transmit 1 packet (**serialization delay**) $\ll$ time to get to the receiver (**propagation delay**)
 - ◆ How bad? Depends on **round trip time (RTT)**
- 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?)



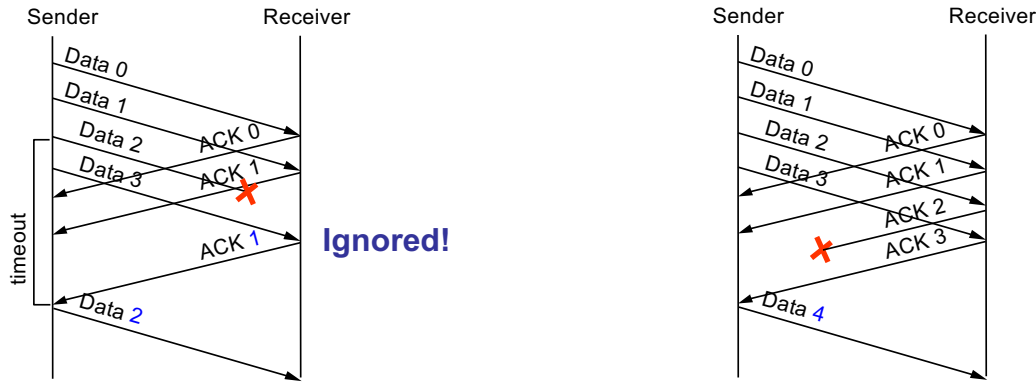
Pipelined Transmission



- 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)

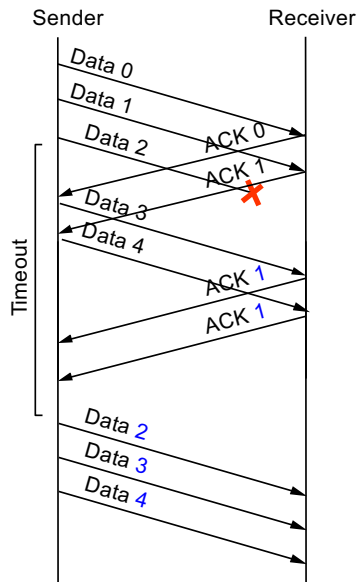


Go-Back-N



- Retransmit all packets from point of loss
 - ◆ Packets sent after loss event are ignored (i.e., sent again)
- Simple to implement (receiver doesn't need to buffer)
- 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 in Go-Back-N
 - ◆ We can do better with **selective retransmission**

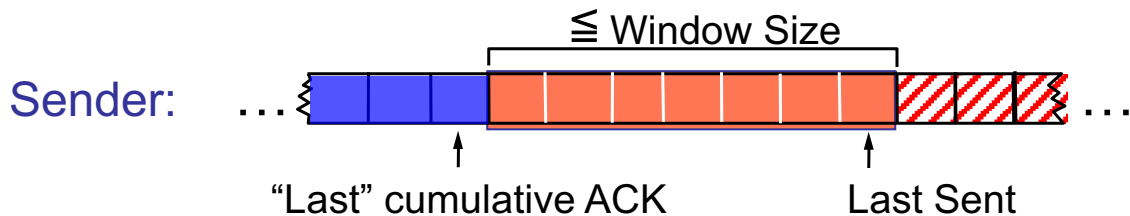


Sliding Window

- Single mechanism that supports:
 - ◆ Multiple outstanding packets
 - ◆ Reliable delivery
 - ◆ In-order delivery
 - ◆ **Flow control**: Don't send more than the receiver can handle
- 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 Retransmit: timer per packet, resend as needed



For Next Time

- Project 1a due one week from Friday: 01/26
- HW1 out today
- Read 5-5.1 in P&D