

Lecture 5: Flow Control

CSE 123: Computer Networks
Alex C. Snoeren



HW 1 due before class WEDNESDAY





Lecture 5 Overview

- Sliding Window
- Transport layer tasks
 - ◆ Process naming/demultiplexing
- User Datagram Protocol (UDP)
- Transport Control Protocol (TCP)
 - ◆ State transitions
 - ◆ Connection teardown

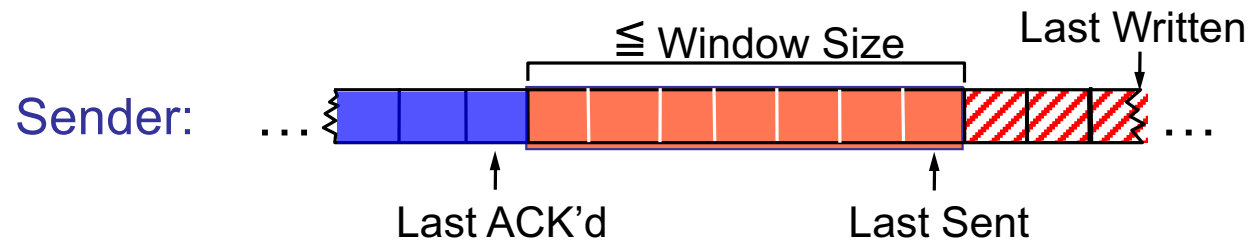
Sliding Window



- Single mechanism that supports:
 - ♦ Multiple outstanding packets
 - ♦ Reliable, in-order delivery
 - ♦ **Flow control**: Don't send more than the receiver can handle
- Sender and receiver each maintain “window” abstractions to track outstanding frames/bytes
 - ♦ At the core of all modern ARQ protocols including TCP
- Go-Back-N is a special case
 - ♦ Receive window size of one



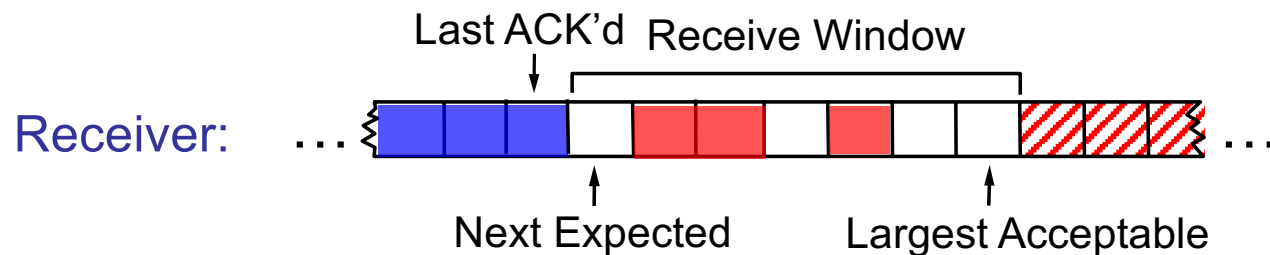
Sliding Window – Sender



- 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



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 sends an ACK for every frame/segment:
 - ♦ **Cumulative**, Selective (exempt missing frames), or Negative (only missing frames)



Deciding When to Retransmit

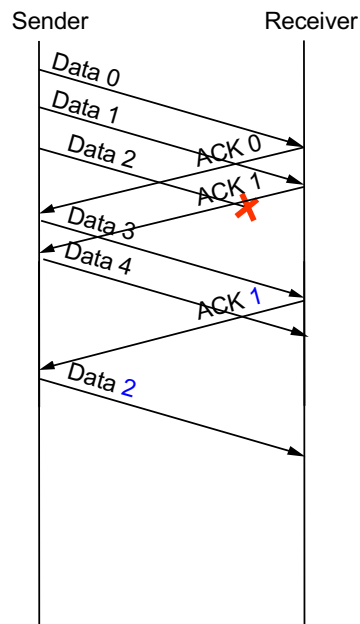
- How do you know when a packet has been lost?
 - ♦ Ultimately sender uses timers to decide when to retransmit
- But how long should the timer be?
 - ♦ Too long: inefficient (large delays, poor use of bandwidth)
 - ♦ Too short: may retransmit unnecessarily (causing extra traffic)
- Right timer is based on the **round-trip time (RTT)**
 - ♦ Which can vary greatly for reasons we'll 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 duplicate cumulative ACKs imply that a packet was lost
- Fast retransmit
 - ♦ When sender receives multiple duplicate acknowledgements resend missing packet

Fast retransmit



- Don't bother waiting
 - ◆ Receipt of duplicate acknowledgement (**dupACK**) indicates loss
 - ◆ Retransmit immediately
- Used in TCP (but not in P1a)
 - ◆ Need to be careful if frames can be reordered
 - ◆ TCP Reno identifies a loss if there are **three** duplicate ACKs in a row



Transport Layer Tasks

- Define and provide specific delivery semantics
 - ◆ To which end point?
 - ◆ When?
 - ◆ How?
 - ◆ If?
- Reliability (our focus so far)
- Multiplexing: sharing the link between multiple end points



Naming Processes/Services

- Process here is an abstract term for your Web browser (HTTP), Email servers (SMTP), hostname translation (DNS)
- How do we identify for remote communication?
 - ◆ Process id or memory address are OS-specific and transient
- So TCP and UDP use **ports**
 - ◆ 16-bit integers representing mailboxes that processes “rent”
 - ◆ Identify process uniquely as (IP address, protocol, port)



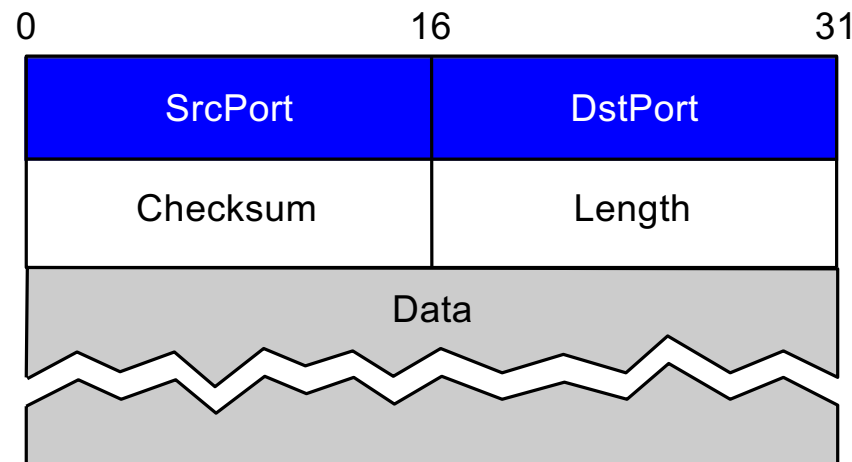
Picking Port Numbers

- We still have the problem of allocating port numbers
 - ♦ What port should a Web server use on host *X*?
 - ♦ To what port should you send to contact that Web server?
- Servers typically bind to **well-known** port numbers
 - ♦ e.g., HTTP 80, SMTP 25, DNS 53, ... look in /etc/services
 - ♦ Ports below 1024 traditionally reserved for well-known services
- Clients use OS-assigned temporary (**ephemeral**) ports
 - ♦ Above 1024, recycled by OS when client finished

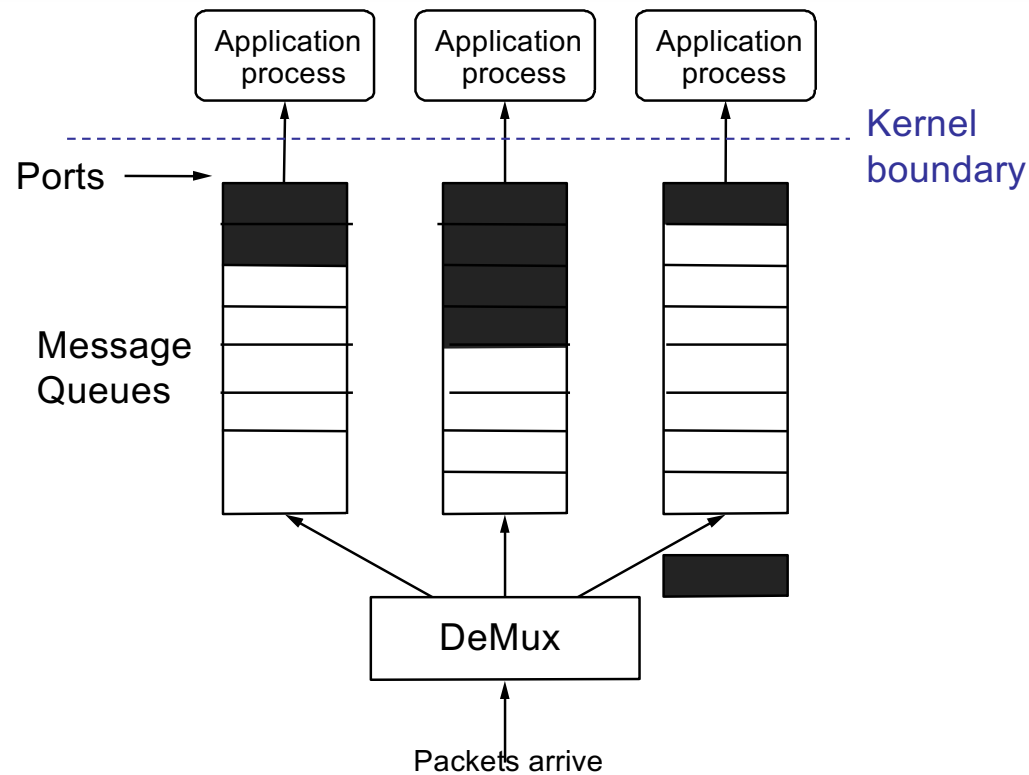


User Datagram Protocol (UDP)

- Provides *unreliable message delivery* between processes
 - ◆ Source port filled in by OS as message is sent
 - ◆ Destination port identifies UDP delivery queue at endpoint
- Connectionless (no state about who talks to whom)



UDP Delivery





Applications for UDP

- Streaming media (e.g., live video)
- DNS (Domain Name Service)
- NTP (Network Time Protocol) (synchronizing clocks)
- FPS multi-player video games (e.g., Call of Duty)
- Why might UDP be appropriate for these?



Transmission Control Protocol

- Reliable bi-directional **bytestream** between processes
 - ◆ Uses a sliding window protocol for efficient transfer
- Connection-oriented
 - ◆ Conversation between two endpoints with beginning and end
- Flow control (receive window)
 - ◆ Prevents sender from over-running receiver buffers
 - ◆ (tell sender how much buffer is left at receiver)
- Congestion control (Project 1b)
 - ◆ Prevents sender from over-running network capacity

For Next Time



- Read 12.2-3 in P&D
- HW 1 due at the beginning of class Wednesday
- Discussion this afternoon will cover HW and Projects
- (Keep) going on the project...