

Lecture 6:

Transport Layer Protocols

CSE 123: Computer Networks
Alex C. Snoeren



Project 1a due MONDAY



Overview



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

How was the homework?

- A. Trivial**
- B. Doable with provided materials**
- C. I had no idea how to answer some parts**
- D. Homework? There was a homework?**



Transport Layer Tasks

- Define and provide specific delivery semantics
 - ◆ To which end point?
 - ◆ When?
 - ◆ How?
 - ◆ If?
- Multiplexing
- Reliability



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)



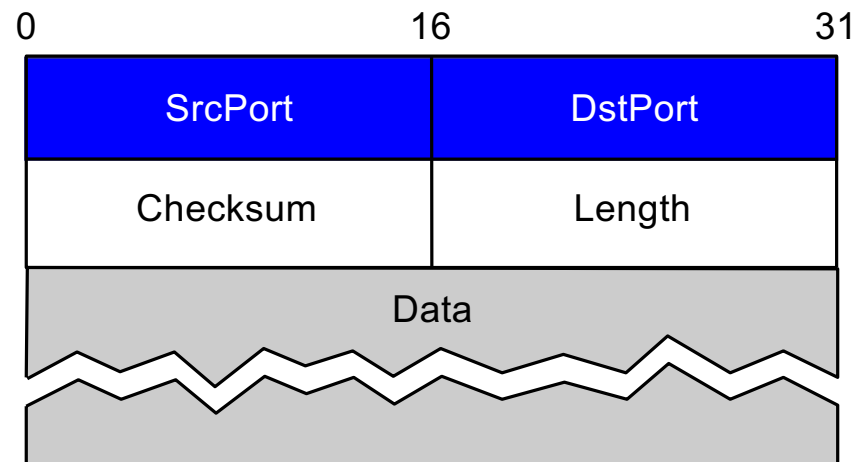
Picking Port Numbers

- 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

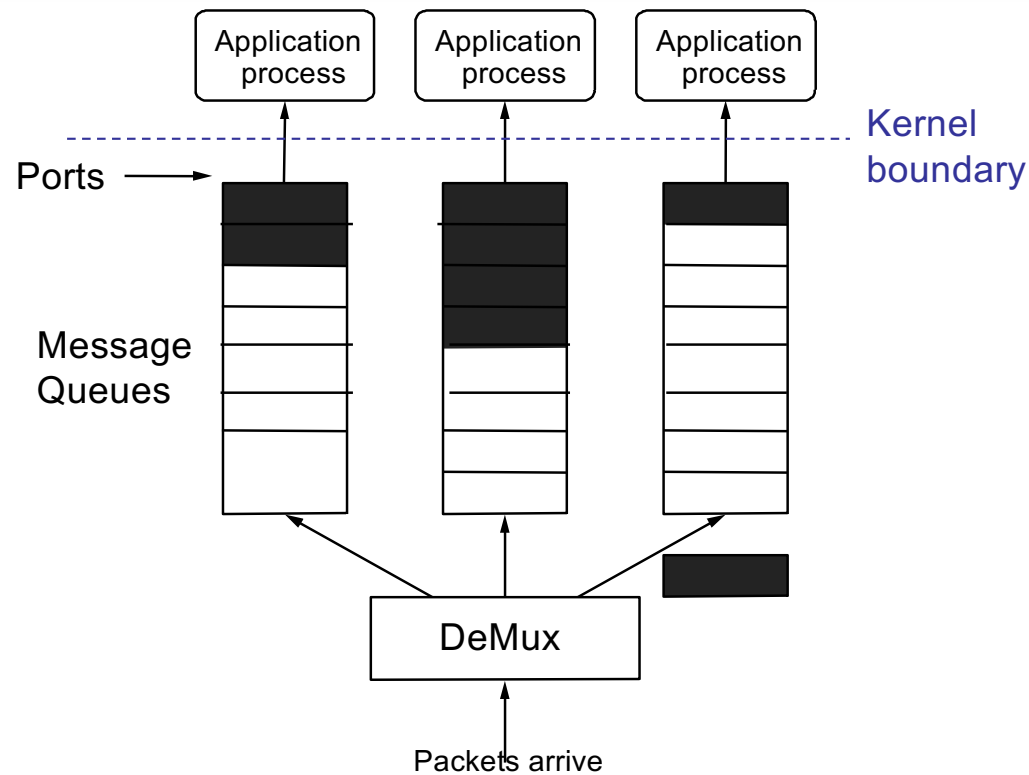


User Datagram Protocol (UDP)

- 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)



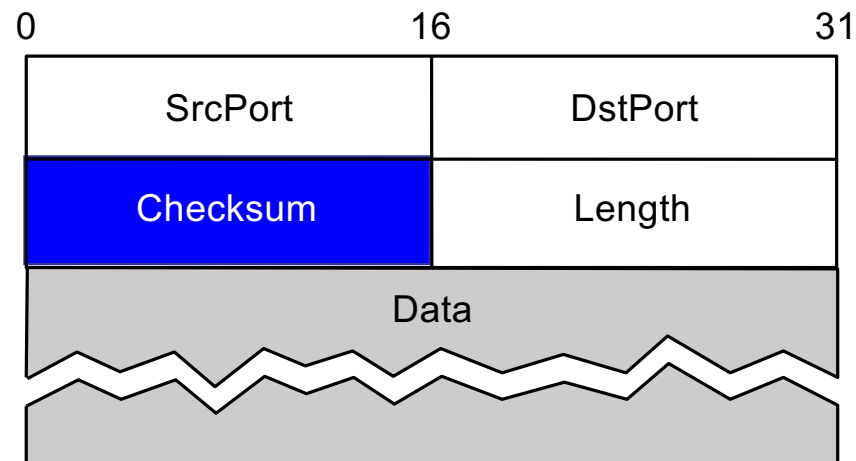
UDP Delivery



UDP Checksum



- UDP includes optional protection against errors
 - ◆ Checksum intended as an end-to-end check on delivery
 - ◆ So it covers data, UDP header, and **IP pseudoheader (history)**





Applications for UDP

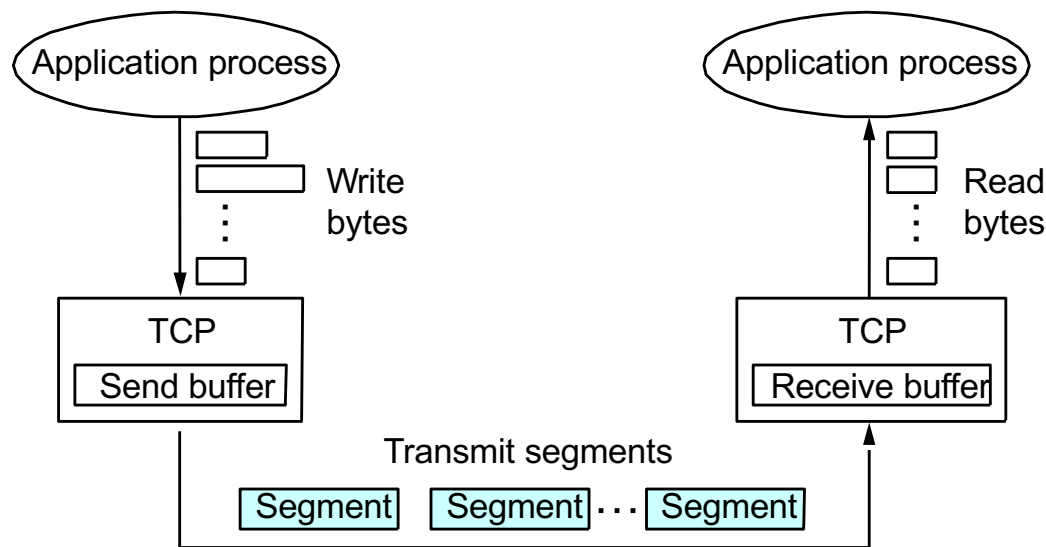
- Streaming media (e.g., live video)
- DNS (Domain Name Service)
- NTP (Network Time Protocol) (synchronizing clocks)
- FPS multi-player video games (e.g., Call of Duty)
- Why might UDP be appropriate for these?



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 (later in term)
 - ◆ Prevents sender from over-running network capacity

TCP Delivery



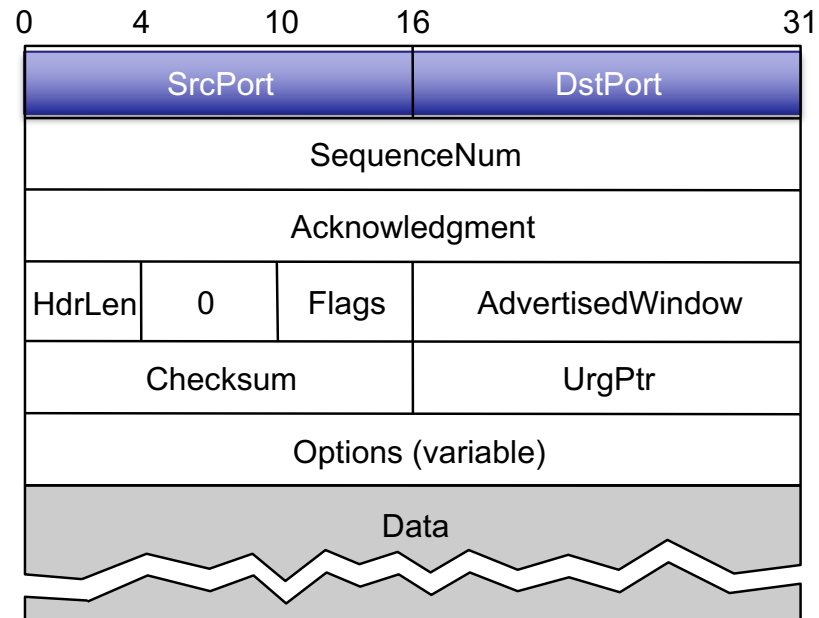
What corresponds to the receive window size?

- A. Segment size**
- B. Bandwidth*delay product**
- C. Send buffer space**
- D. Receive buffer space**



TCP Header Format

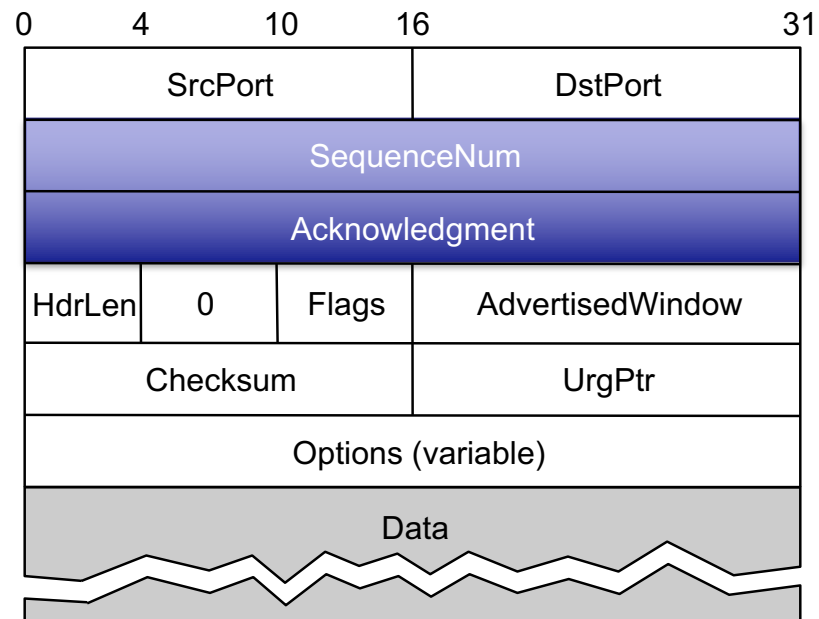
- Ports plus IP addresses identify a connection (4-tuple)





TCP Header Format

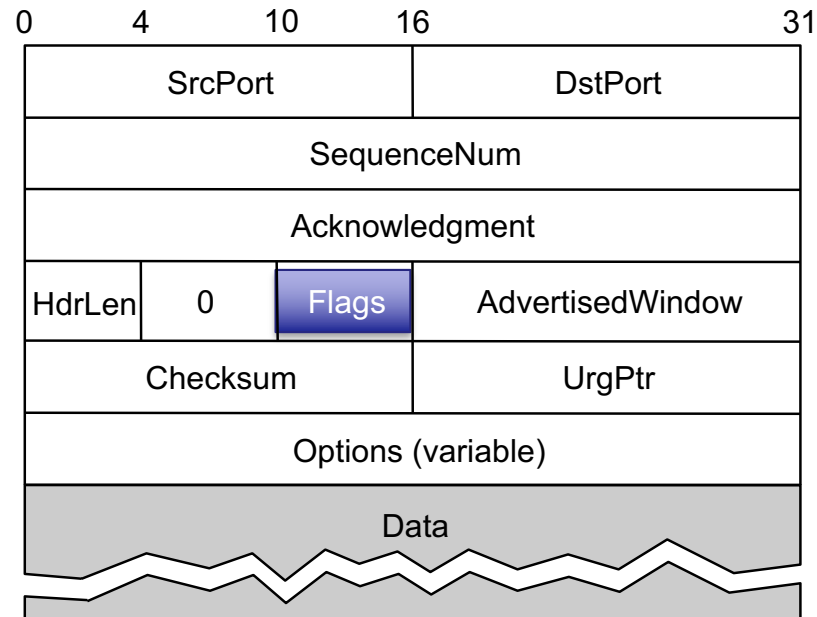
- Sequence, Ack numbers used for the sliding window
 - ◆ How big a window? Flow control/congestion control determine





TCP Header Format

- Flags may be ACK, SYN, FIN, URG, PSH, RST

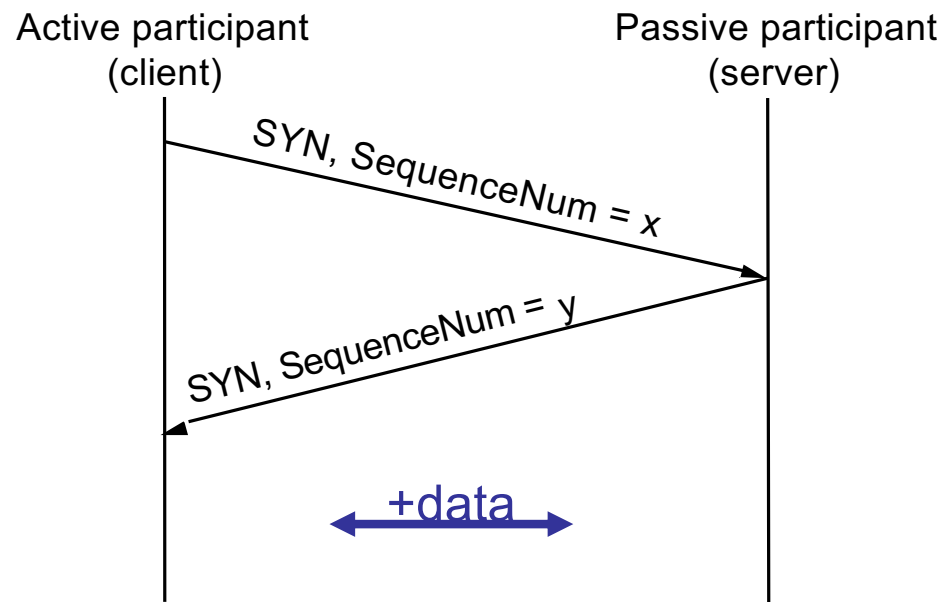


Connection Establishment



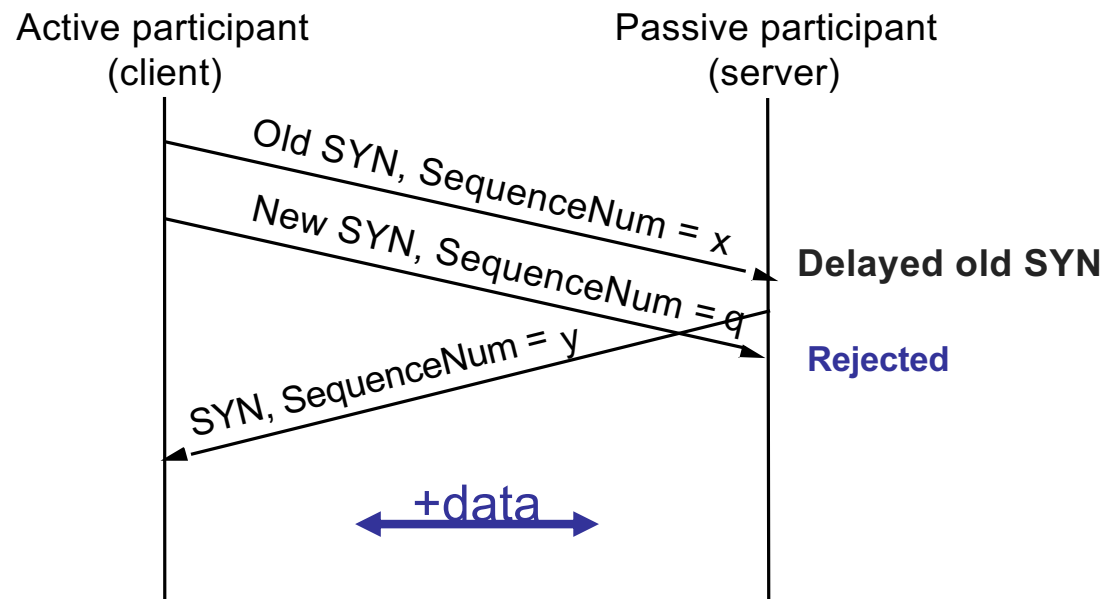
- Both sender and receiver must be ready before we start to transfer the data
 - ◆ Sender and receiver need to agree on a set of parameters
 - ◆ Most important: sequence number space in each direction
 - ◆ Lots of other parameters: e.g., the window size
- Handshake protocols: setup state between two oblivious endpoints
 - ◆ Need to deal with **delayed** and **reordered** packets

Two-way handshake?



What's wrong here?

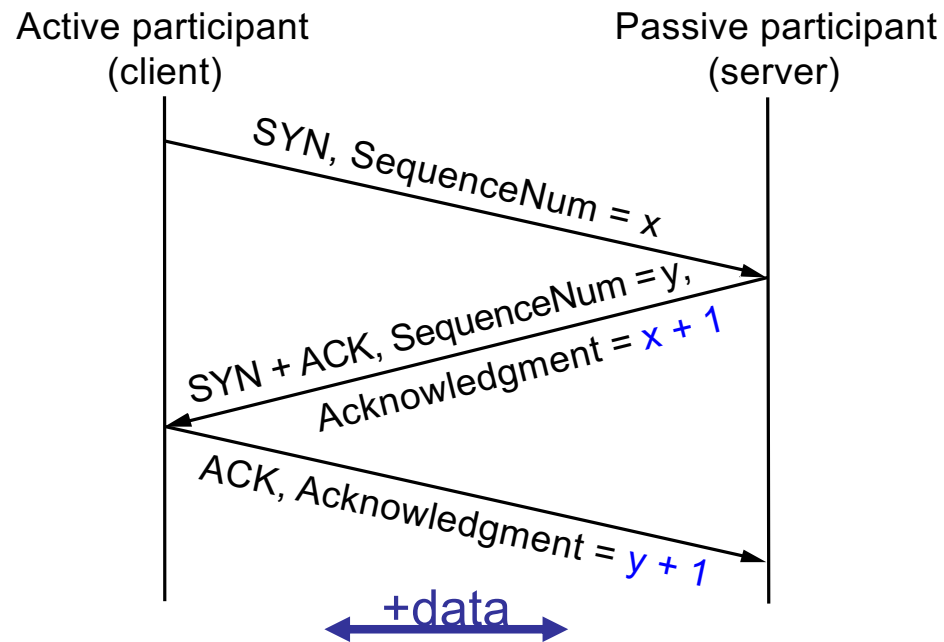
Two-way handshake?





Three-Way Handshake

- Opens both directions for transfer



For next time...



- Begin to discuss actual *networks*!
- Read Ch 3-3.1 in P&D
- NO CLASS MONDAY (Enjoy MLK Day)
- Project 1b introduced in Discussion TUESDAY