Lecture 14:
Transport Layer Protocols

CSE 123: Computer Networks

Stefan Savage

UCSDCSE

Announcements

- Homework #2 and Midterms to be returned at end of class
  - Midterm mean 51, median 52
- Project #2
  - Up now, due Dec 4th at noon
  - SRMP: Build a sliding window protocol with flow control
  - We provide framework for you to plug into

Overview

- Process naming/demultiplexing
- User Datagram Protocol (UDP)
- Transport Control Protocol (TCP)
  - Three-way handshake
  - Flow control

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

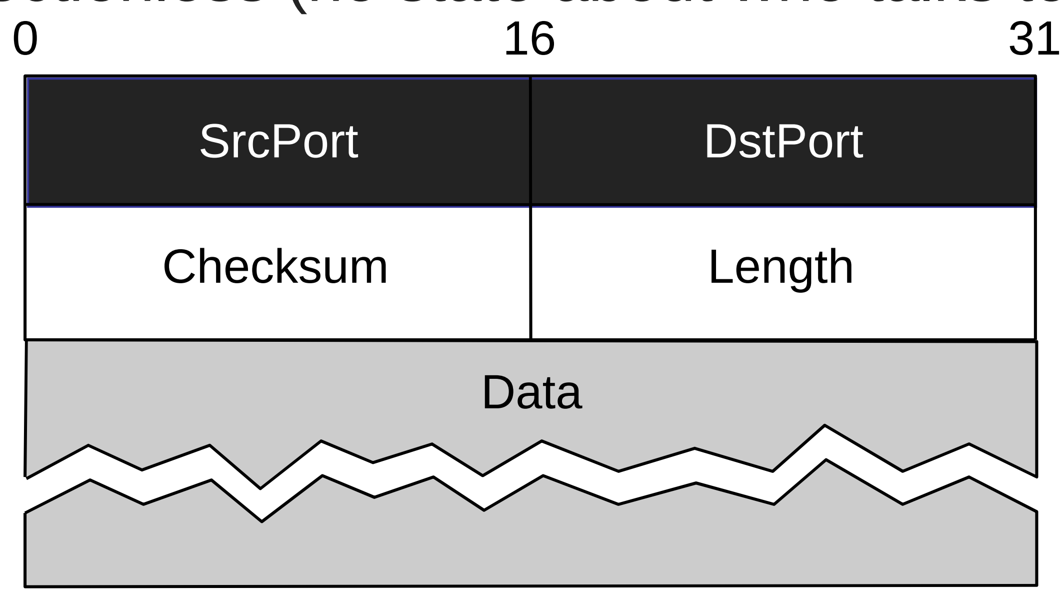
- We still have the problem of allocating port numbers
  - What port should a Web server use on host *X*?
  - To what port should you send to contact that Web server?

- 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

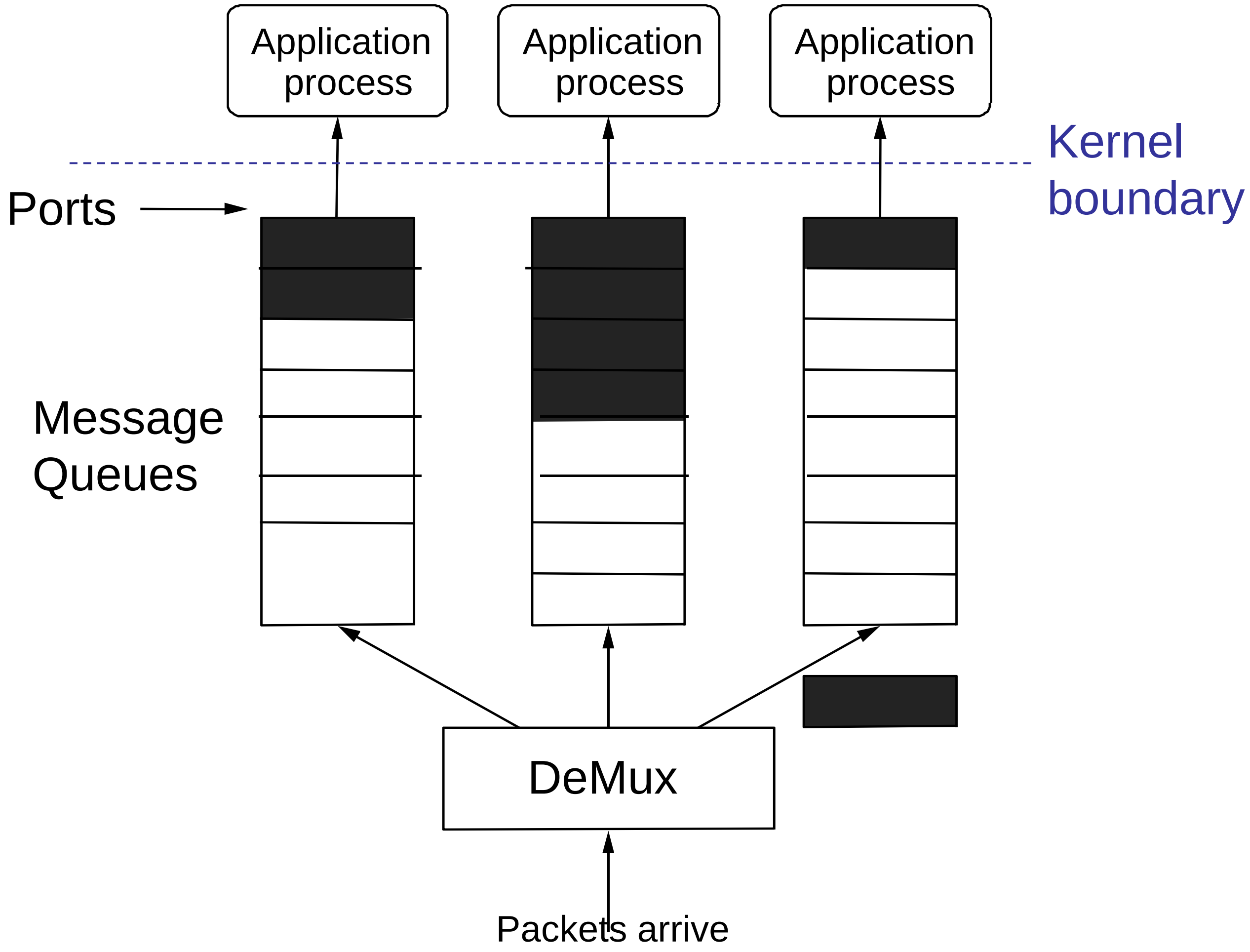
- Clients use OS-assigned temporary (ephemeral) ports
  - Above 1024, recycled by OS when client finished

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)

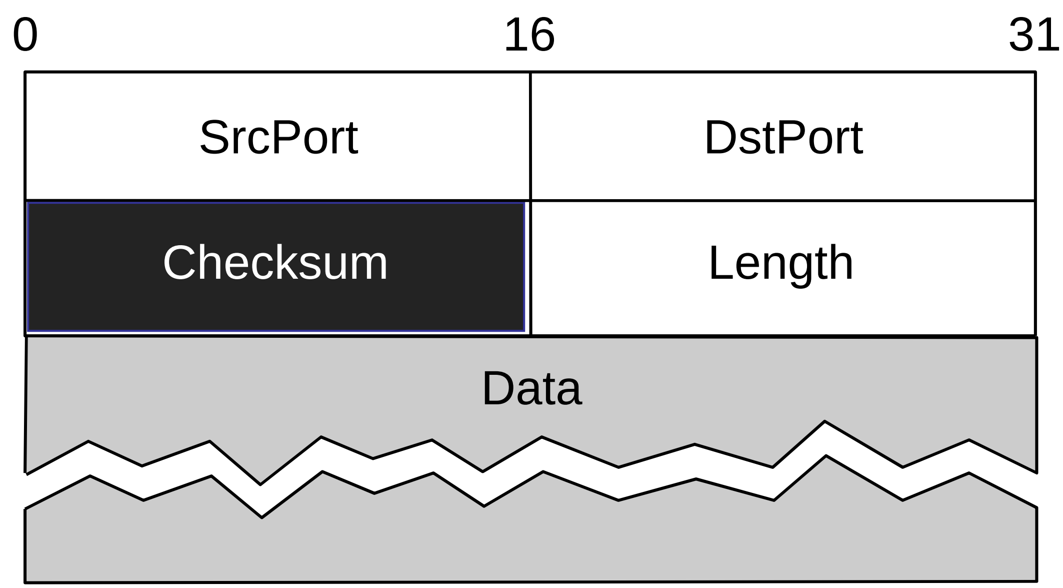


UDP Delivery



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



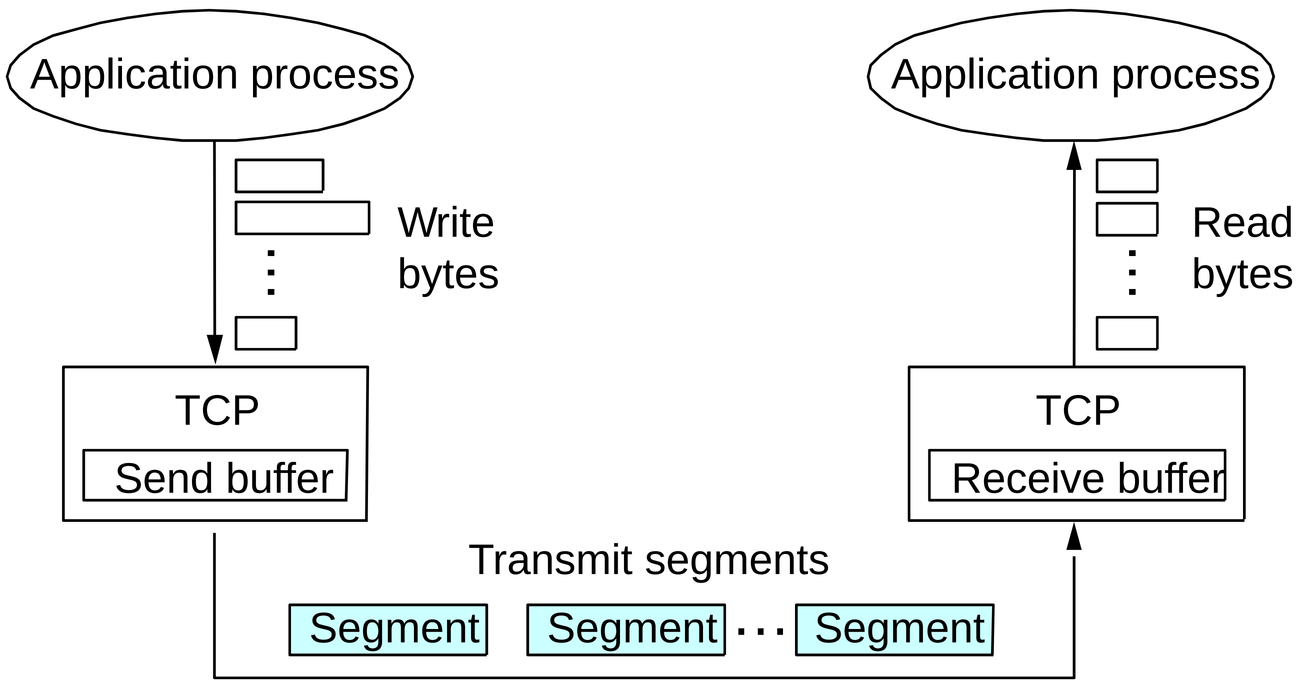
Applications for UDP

- Streaming media
- DNS (Domain Name Service)
- NTP (Network Time Protocol)
- Why is UDP appropriate for these?

Transmission Control Protocol

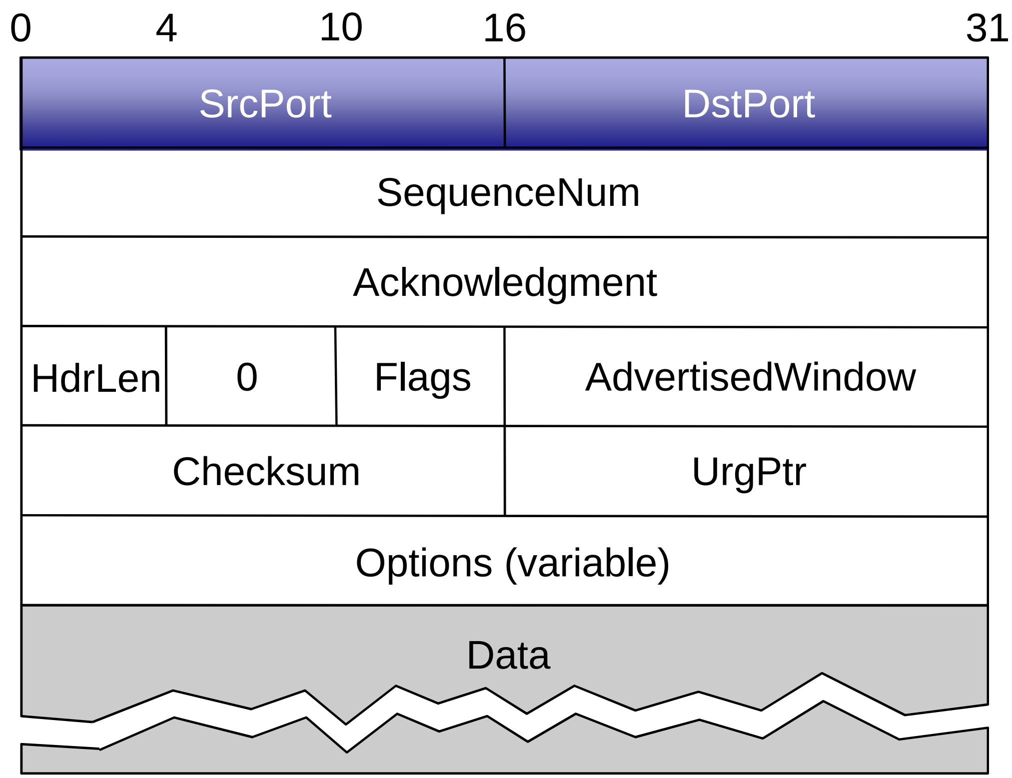
- Reliable bi-directional bytestream between processes
  - Uses a sliding window protocol for efficient transfer
- Connection-oriented
  - Conversation between two endpoints with beginning and end
- Flow control
  - Prevents sender from over-running receiver buffers
- Congestion control (next class)
  - Prevents sender from over-running network capacity

TCP Delivery



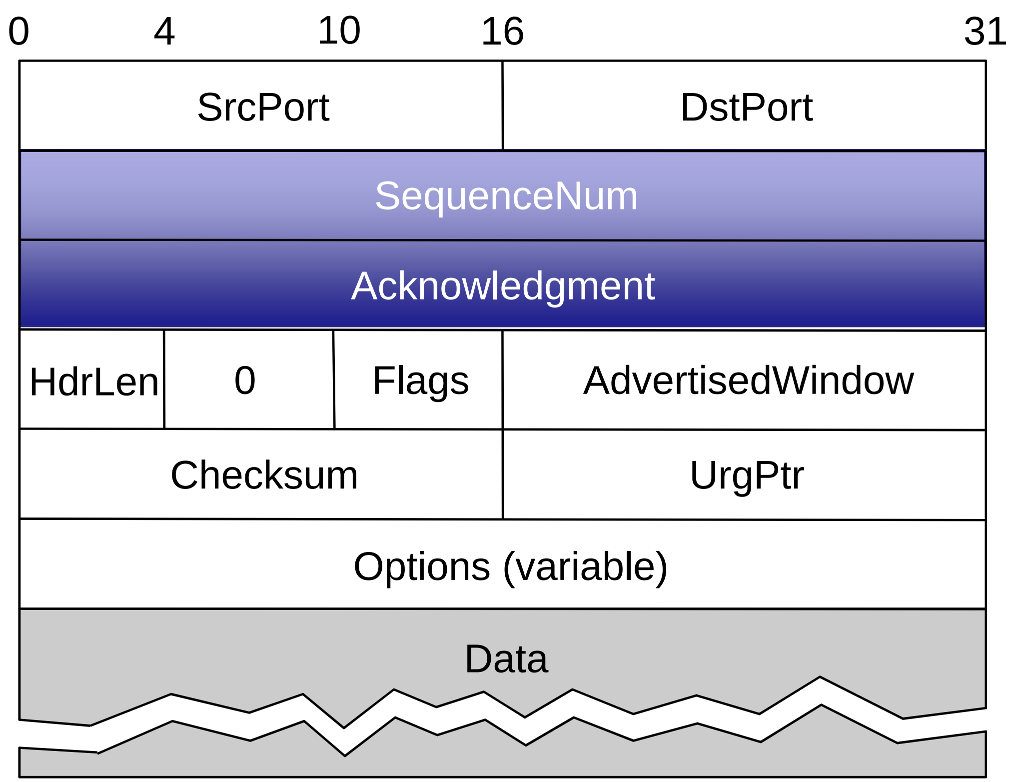
TCP Header Format

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



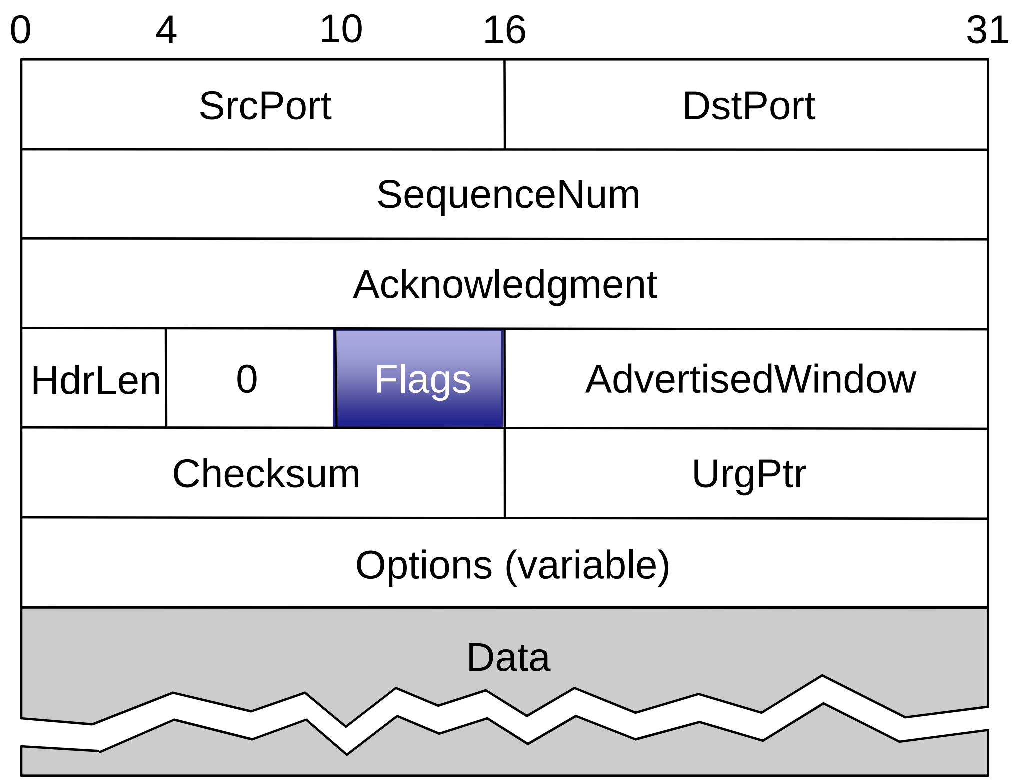
TCP Header Format

- 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

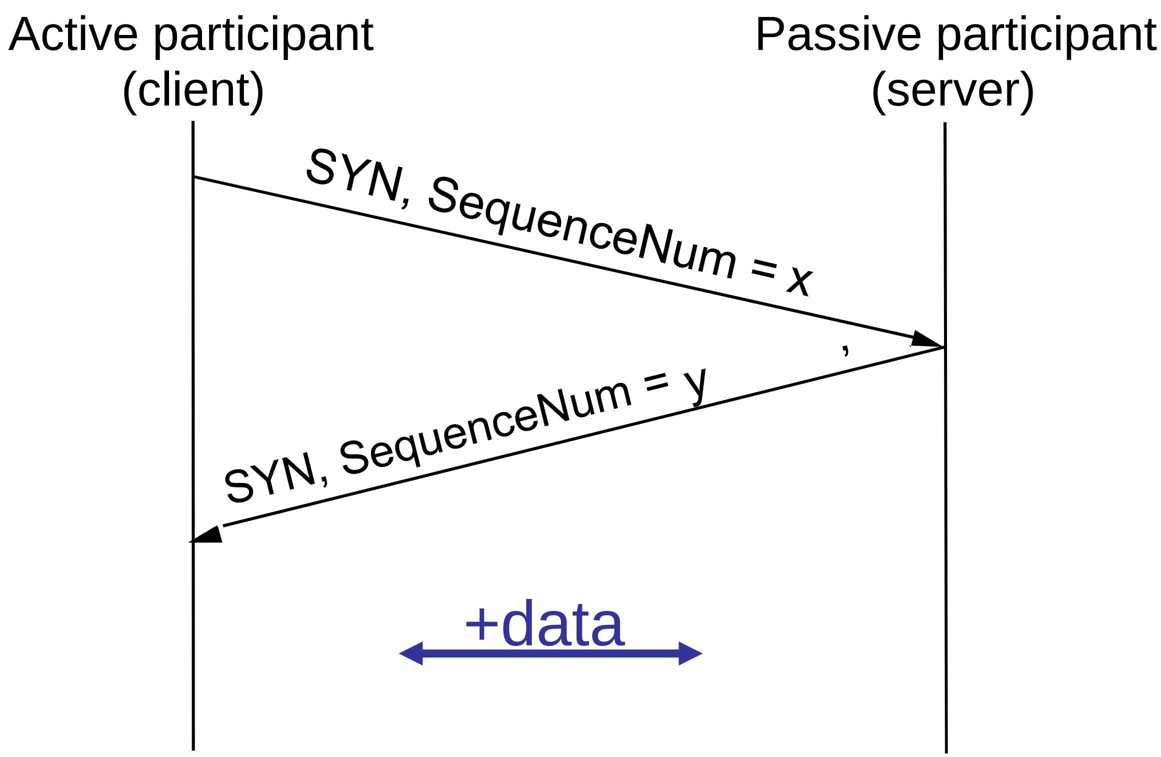


Connection Establishment

- 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 Maximum Segment Size

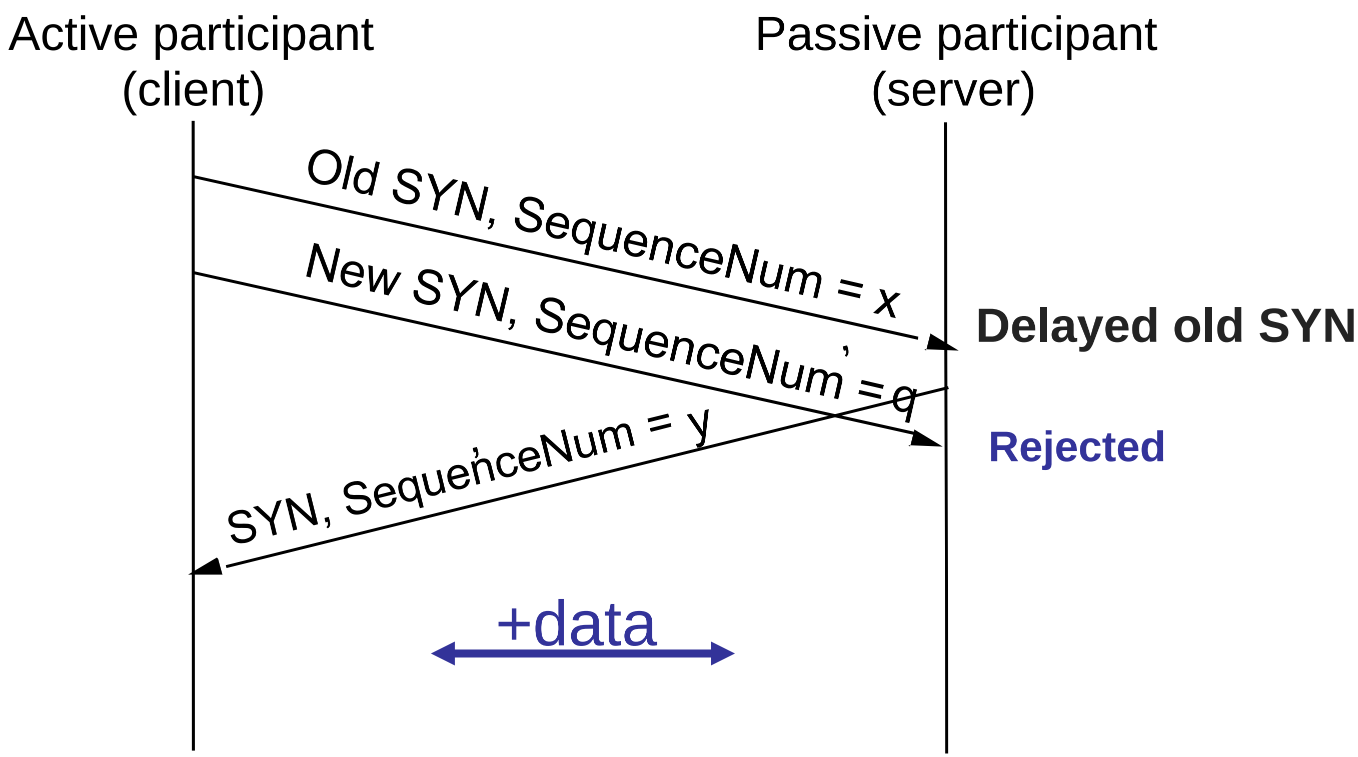
- Handshake protocols: setup state between two oblivious endpoints
  - Didn’t need it earlier because link had only two end points
  - Need to deal with delayed and reordered packets

Two-way handshake?



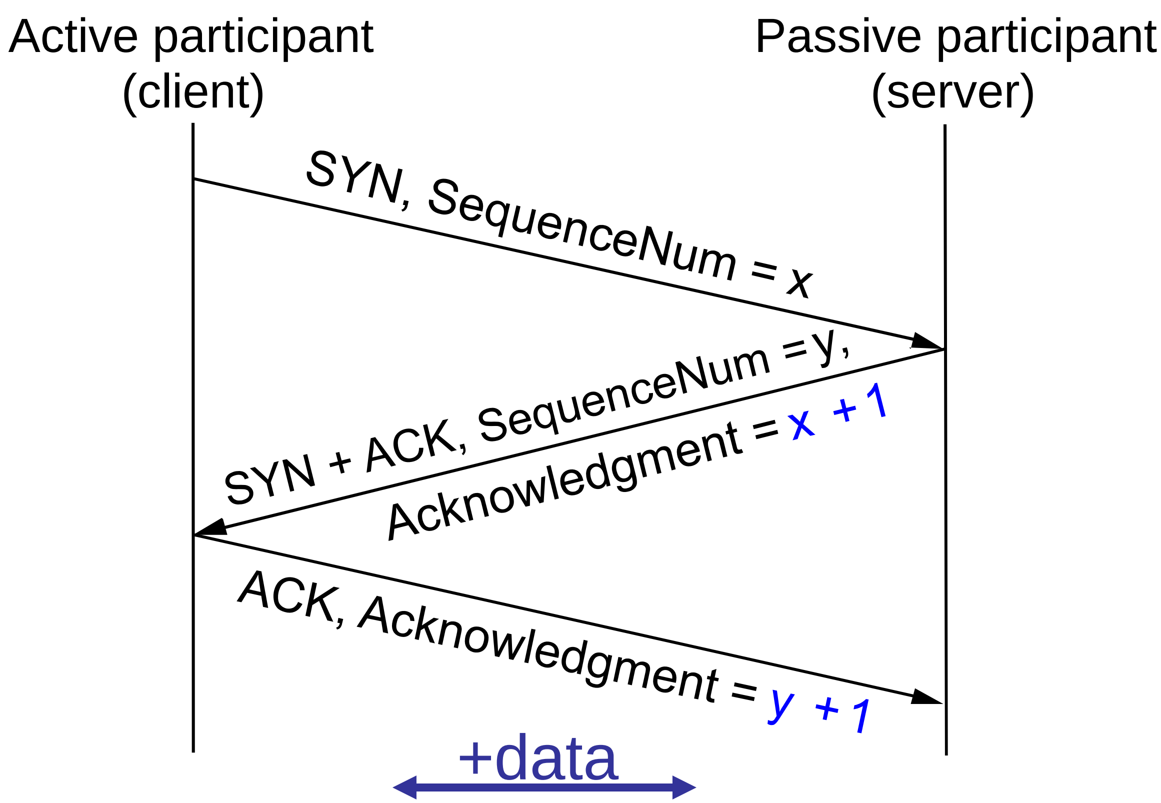
What's wrong here?

Two-way handshake?



Three-Way Handshake

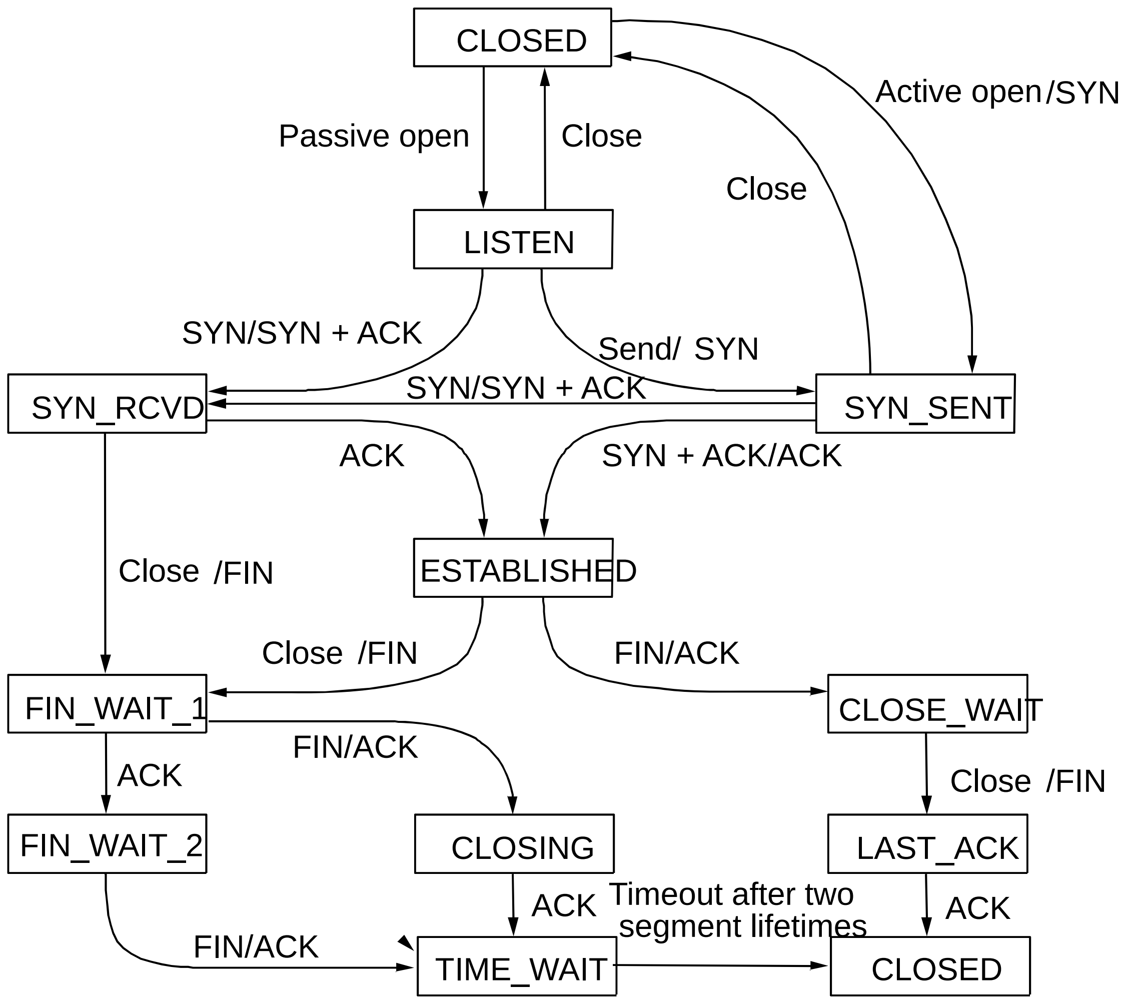
- Opens both directions for transfer



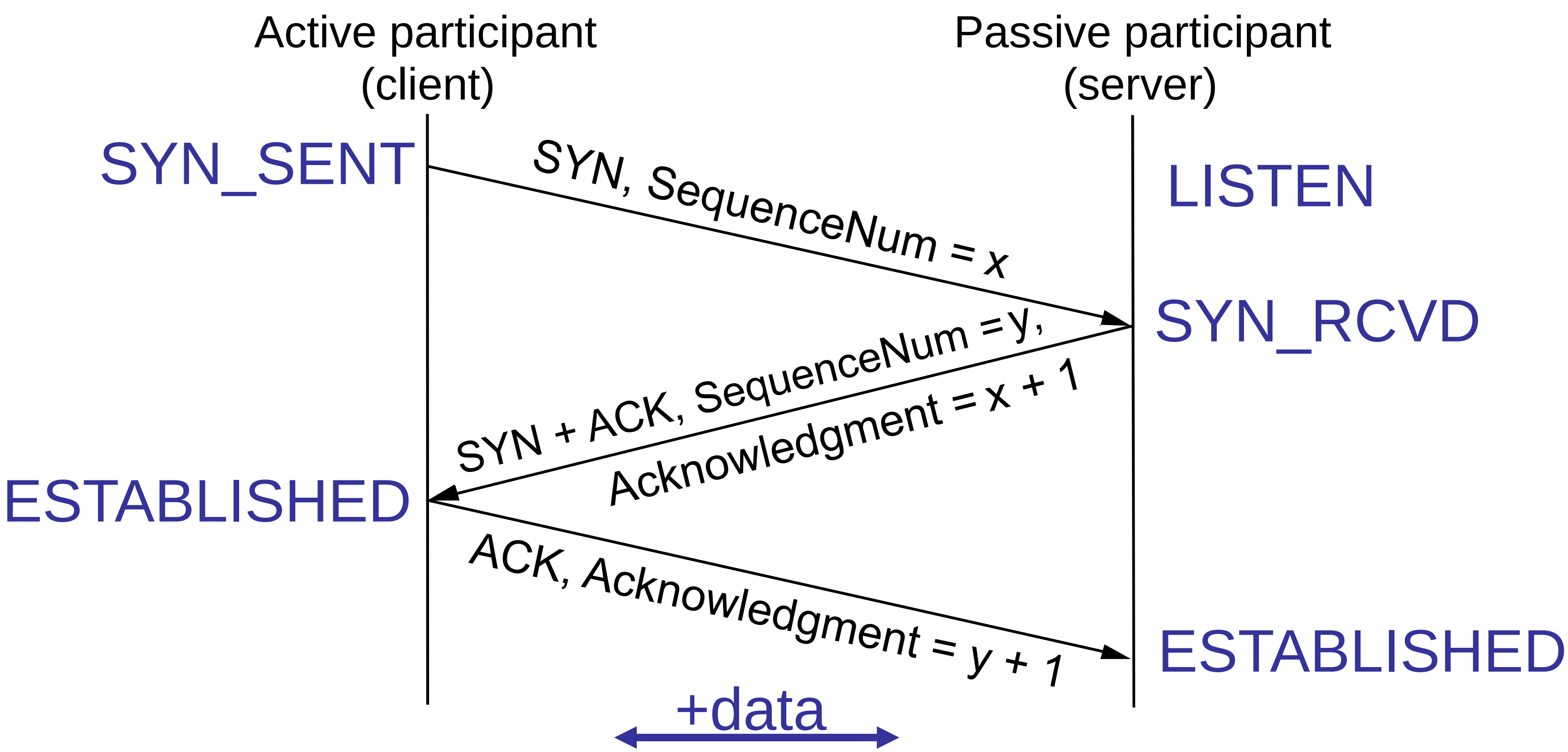
Some Comments

- We could abbreviate this setup, but it was chosen to be robust, especially against delayed duplicates
  - Three-way handshake from Tomlinson 1975
- Choice of changing initial sequence numbers (ISNs) minimizes the chance of hosts that crash getting confused by a previous incarnation of a connection
- How to choose ISNs?
  - Maximize period between reuse
  - Minimize ability to guess (why?)

TCP State Transitions



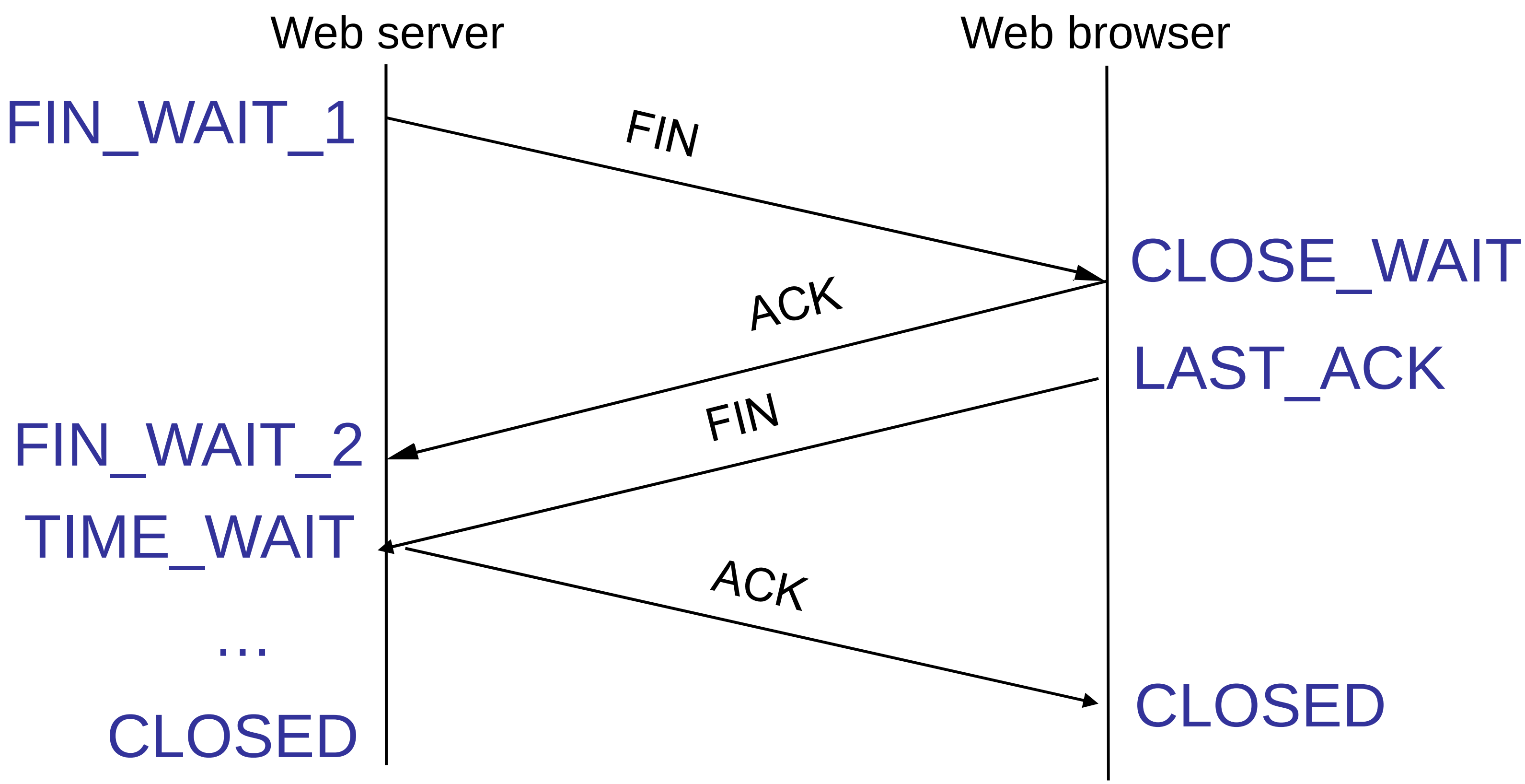
Again, with States



Connection Teardown

- Orderly release by sender and receiver when done
  - Delivers all pending data and “hangs up”
- Cleans up state in sender and receiver
- TCP provides a “symmetric” close
  - Both sides shutdown independently

TCP Connection Teardown



The TIME_WAIT State

- We wait 2*MSL (maximum segment lifetime of 60 seconds) before completing the close
  - Why?
- ACK might have been lost and so FIN will be resent
  - Could interfere with a subsequent connection
- Real life: Abortive close
  - Don't wait for 2*MSL, simply send Reset packet (RST)
  - Why?

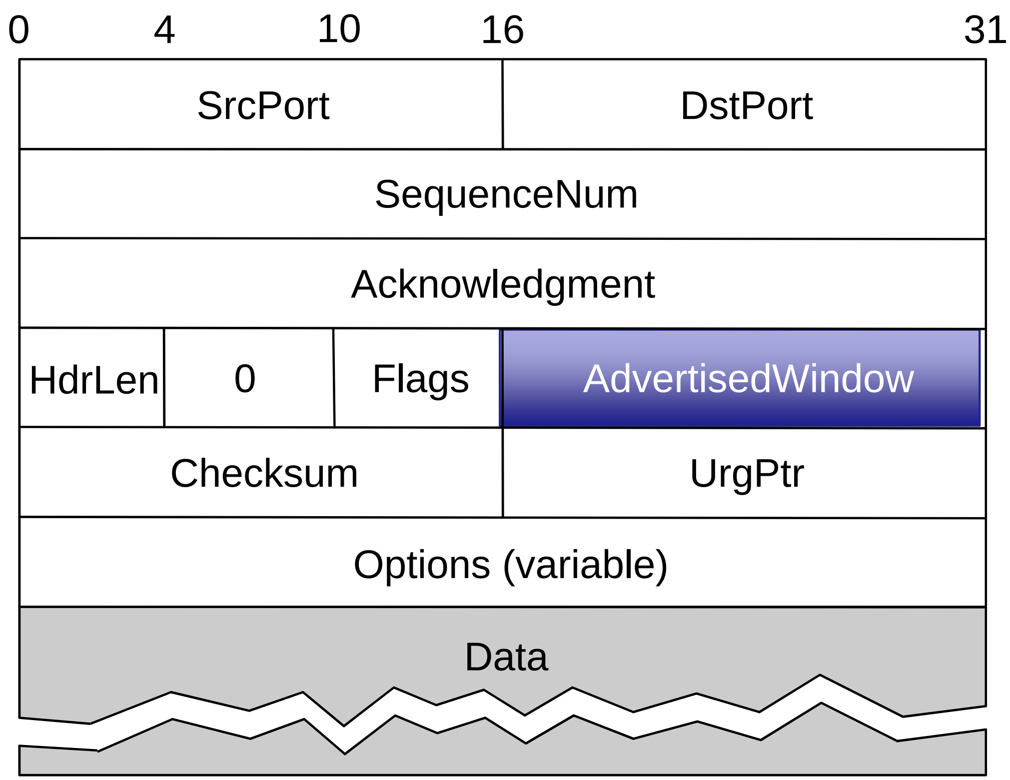
Flow Control

- Sender must transmit data no faster than it can be consumed by the receiver
  - Receiver might be a slow machine
  - App might consume data slowly

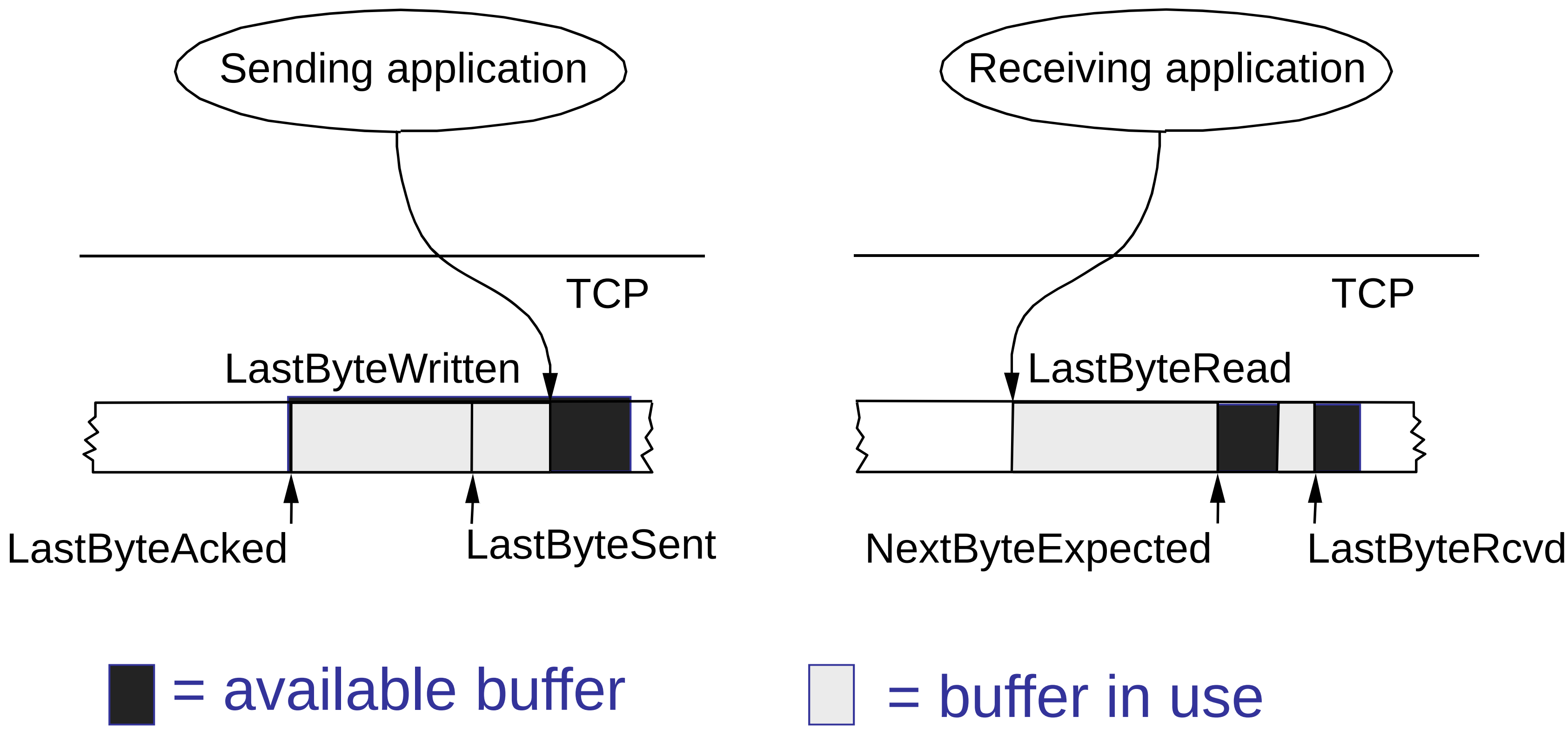
- TCP adjusts the size of the sliding window
  - This is the purpose of the Advertised Window field

TCP Header Format

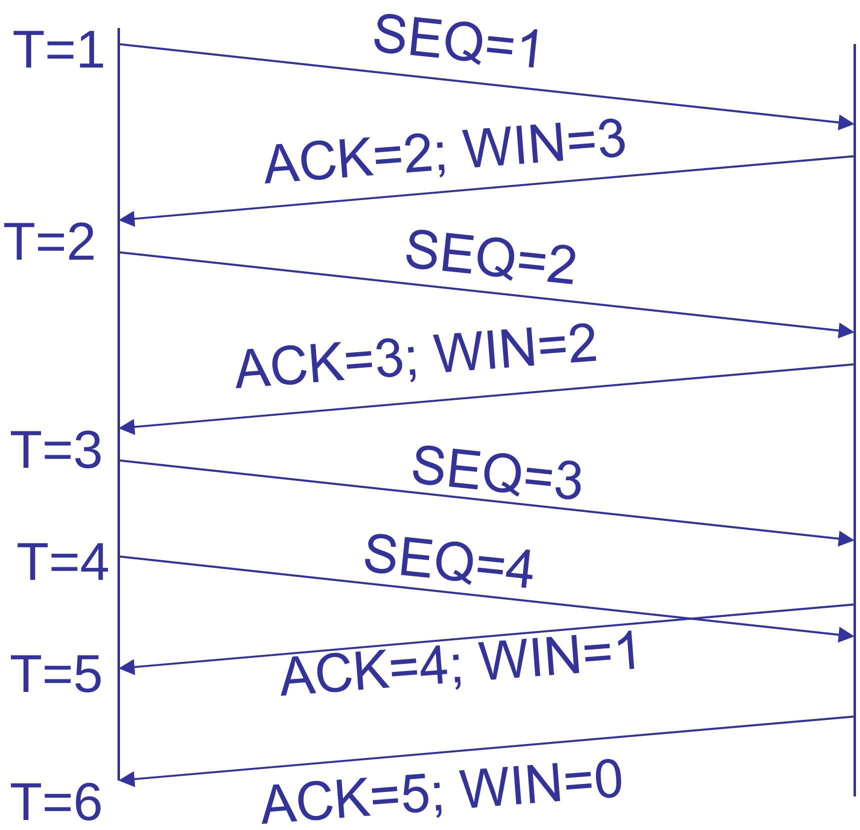
- Advertised window is used for flow control



Sender and Receiver Buffering



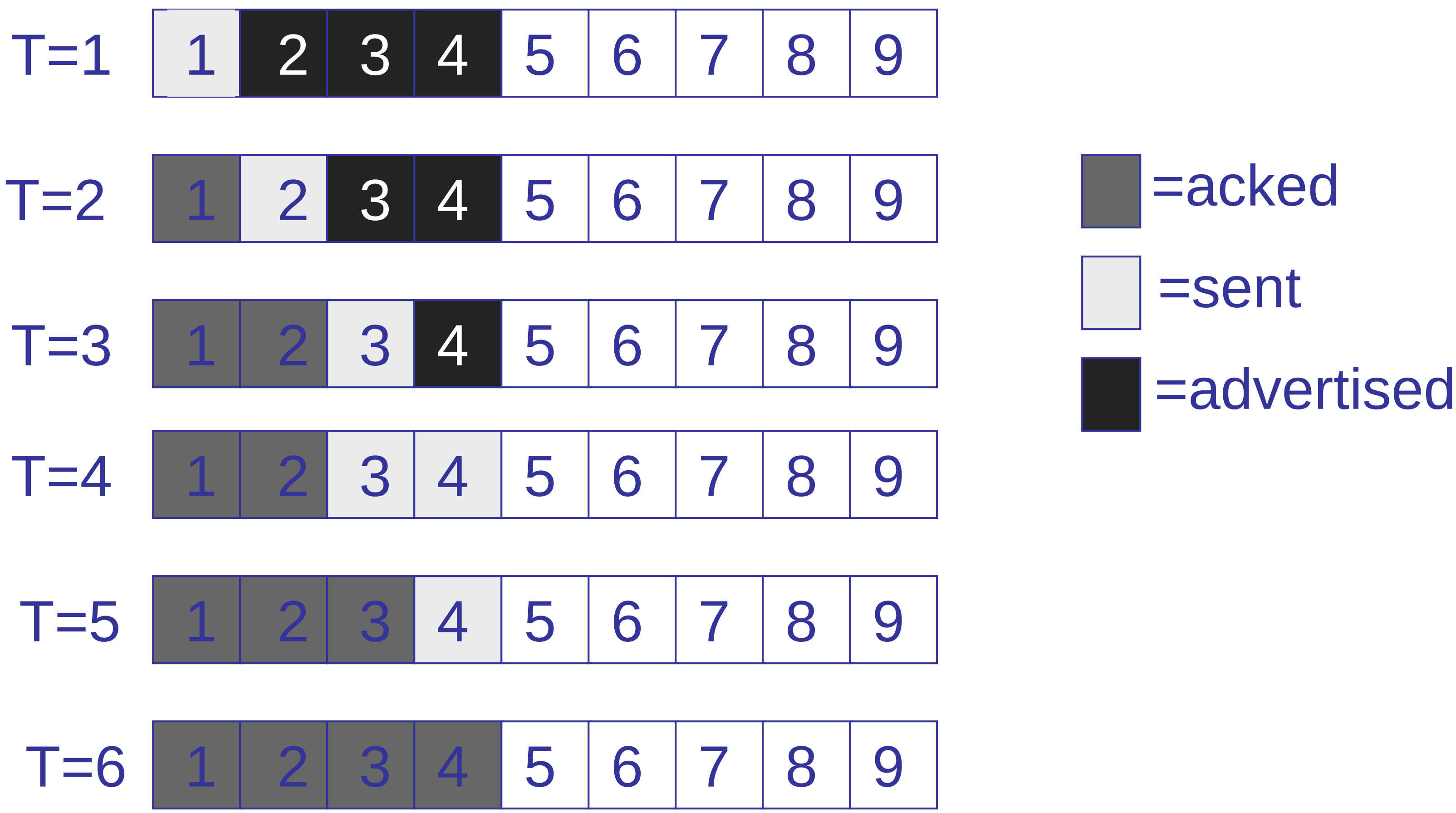
Window-Size Example



Stall due to flow control here

Receiver has buffer of size 4 and application doesn't read

Example – Buffer at Sender



Lots of Icky Details

- Window probes
- Silly Window Syndrome
- Nagle’s algorithm
- PAWS
- Etc…

- Steven’s books “TCP/IP Illustrated (vol 1,2)” is a great source of information on this

TCP applications

- HTTP/WWW
- FTP
- SMTP, POP, IMAP (E-mail)

- Why is TCP well suited to these applications?

Summary

- Transport layer provides demultiplexing
- Different protocols provide various services
  - UDP provides unreliable datagram delivery
  - TCP delivers reliable, in-order bytestreams
- Connection setup/teardown
- Flow control
  - Adjust sliding window to manage receiver buffer

For next time…

- Read Ch 6.3-4 in P&D
- Enjoy Thanksgiving