

Lecture 19:

Transport Layer Protocols

CSE 123: Computer Networks
Stefan Savage

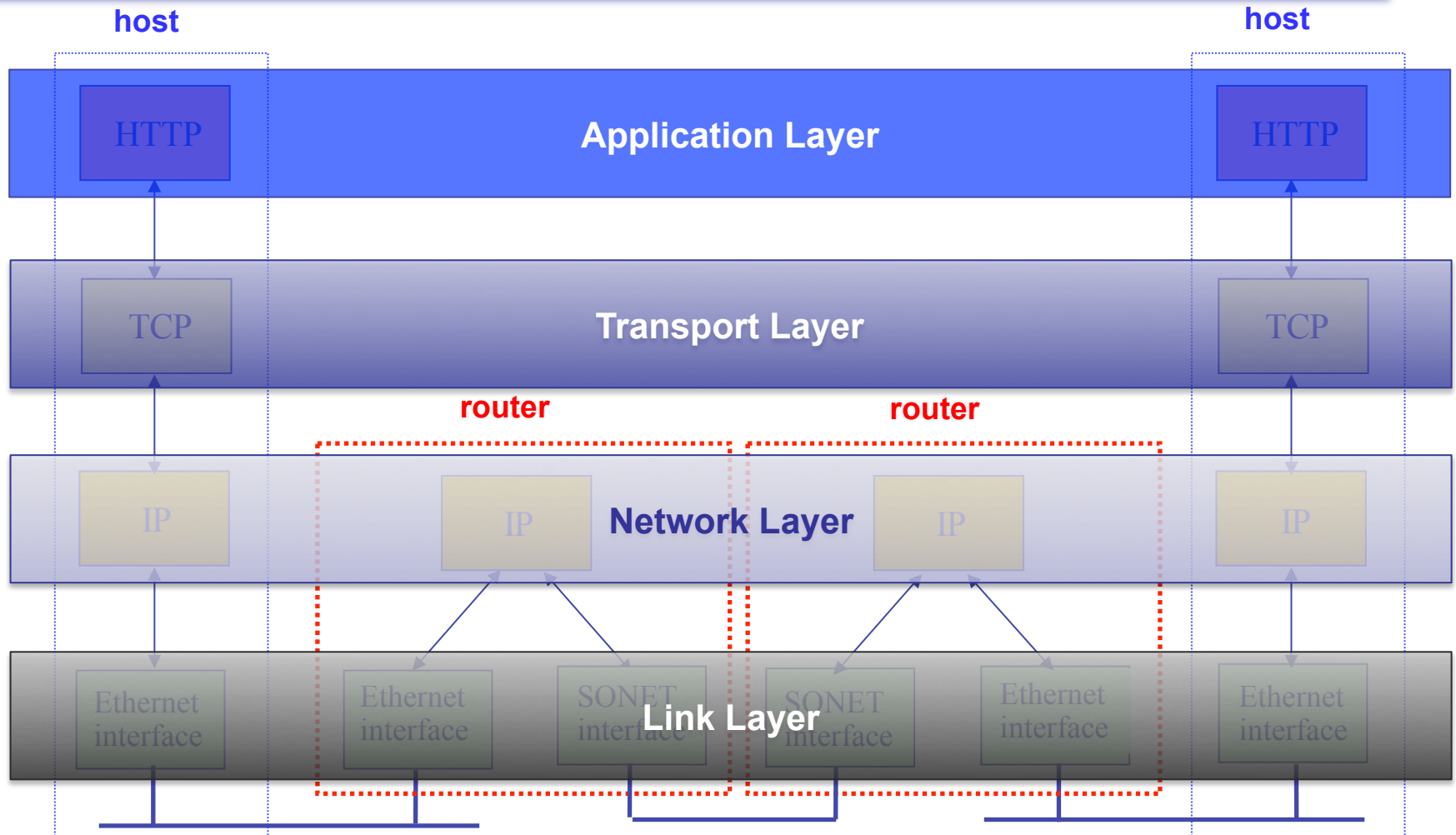




Overview

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

Transport Layer



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)



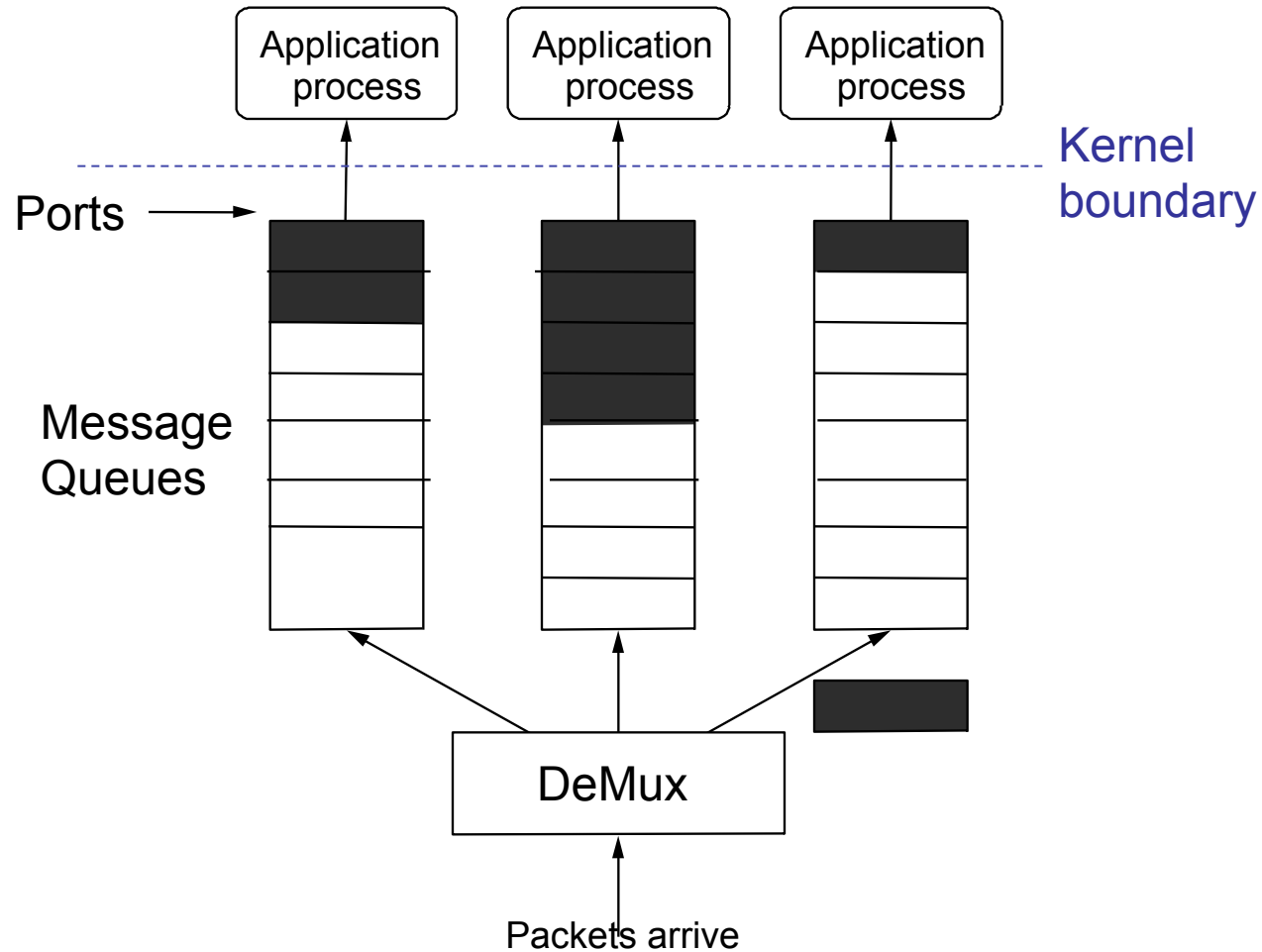
Picking Port Numbers

- 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



-
- The diagram illustrates the structure of a network packet. It is divided into several fields:
- SrcPort**: Source Port, located in the top-left section (bits 0 to 16).
 - DstPort**: Destination Port, located in the top-right section (bits 16 to 31).
 - Checksum**: Checksum field, located in the bottom-left section.
 - Length**: Length field, located in the bottom-right section.
 - Data**: The payload data, represented by a wavy line at the bottom.

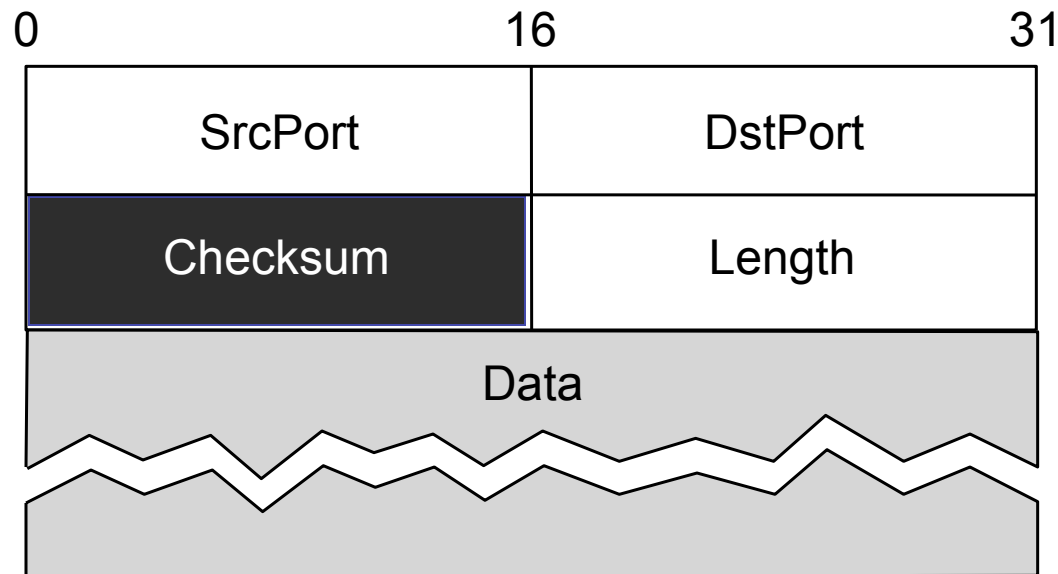
UDP Delivery





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

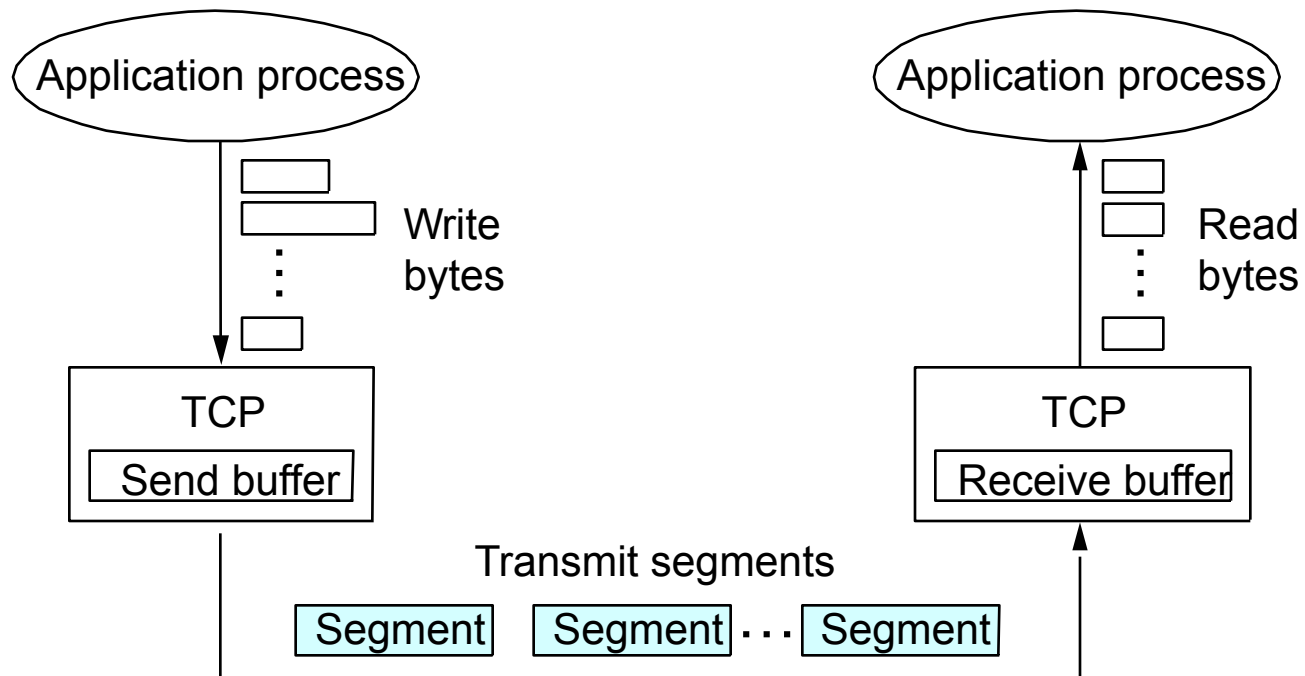
- Streaming media (e.g., live video)
- DNS (Domain Name Service)
- NTP (Network Time Protocol) (synchronizing clocks)
- FPS multi-player video games (e.g., Call of Duty)
- Why might UDP be appropriate for these?

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
 - ◆ Prevents sender from over-running receiver buffers
 - ◆ (tell sender how much buffer is left at receiver)
- Congestion control (next class)
 - ◆ Prevents sender from over-running network capacity

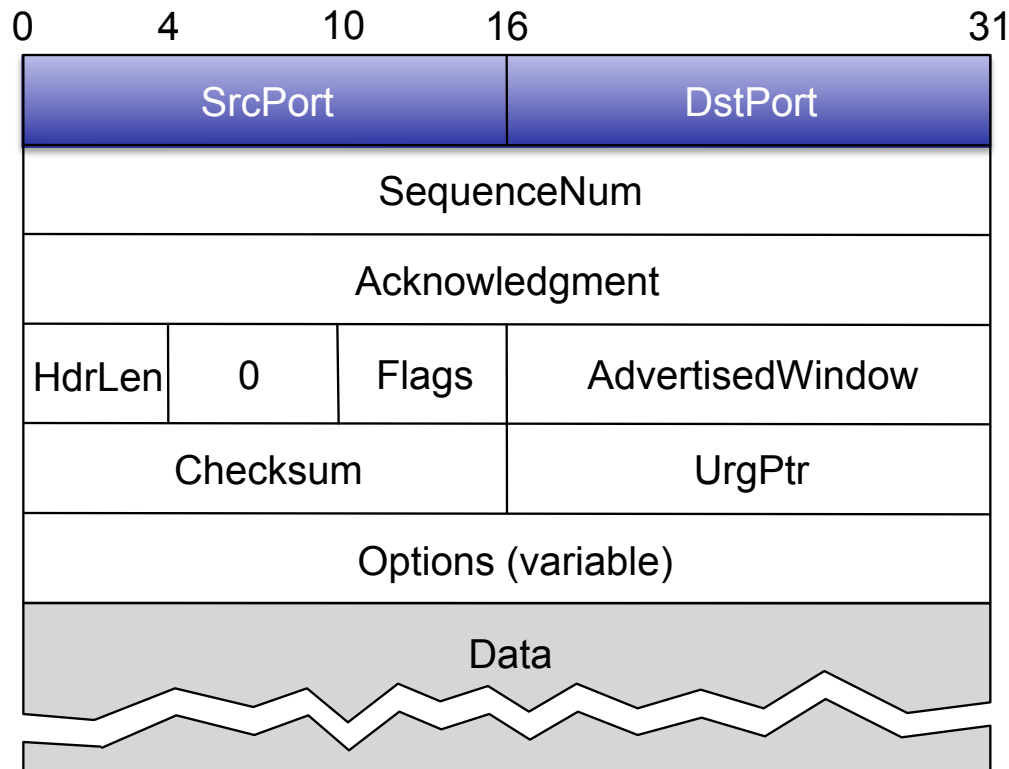
TCP Delivery





TCP Header Format

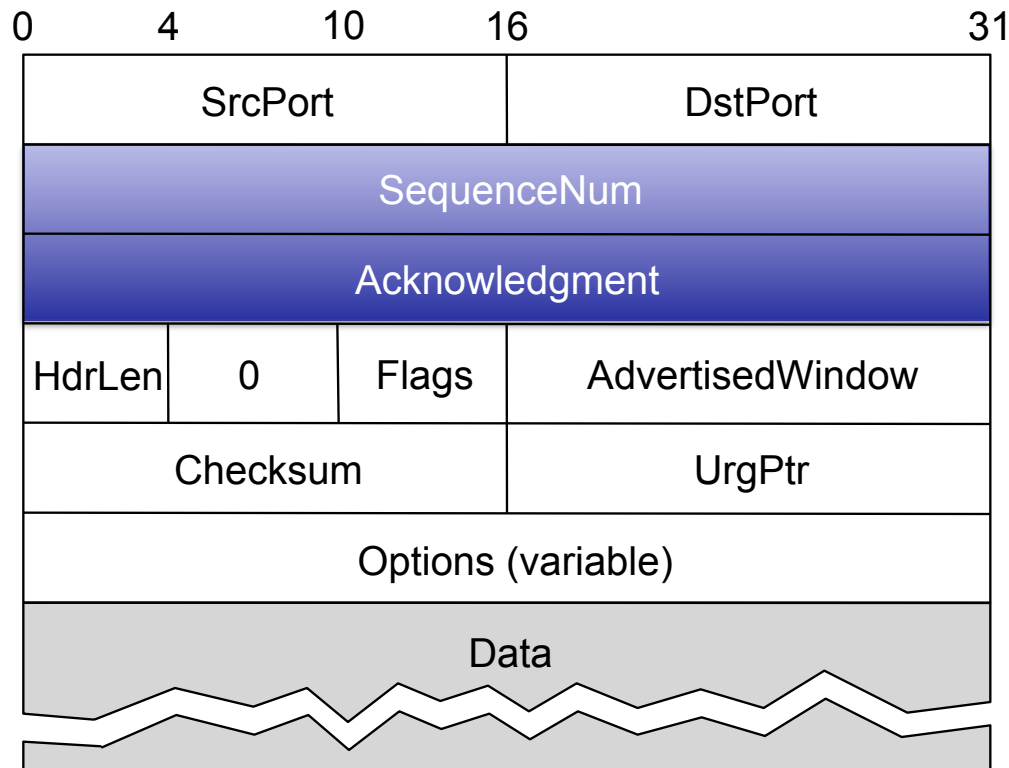
- Ports plus IP addresses identify a connection (4-tuple)





TCP Header Format

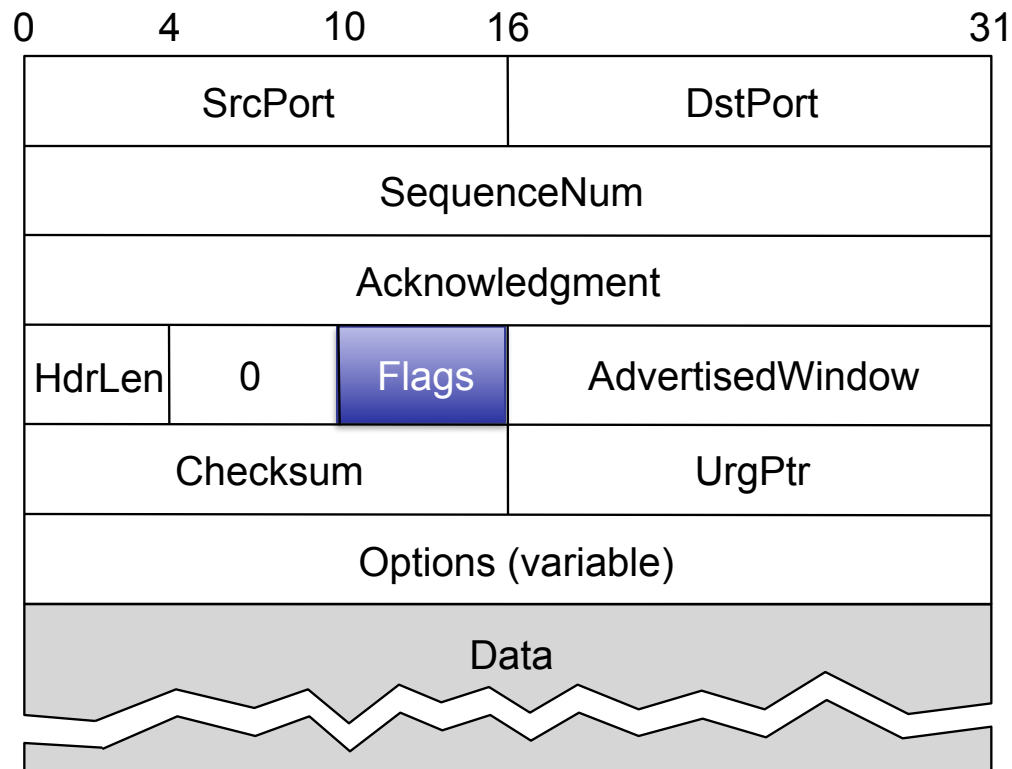
- Sequence, Ack numbers used for the sliding window
 - ◆ How big a window? Flow control/congestion control determine





TCP Header Format

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

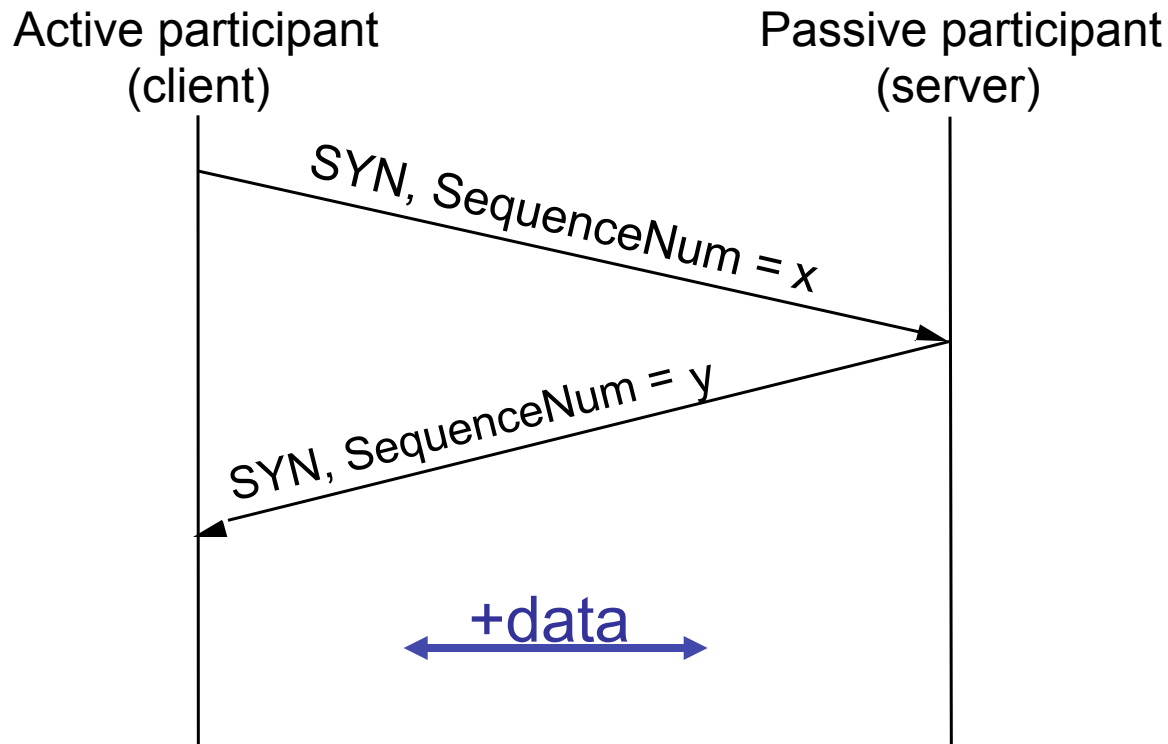


Connection Establishment



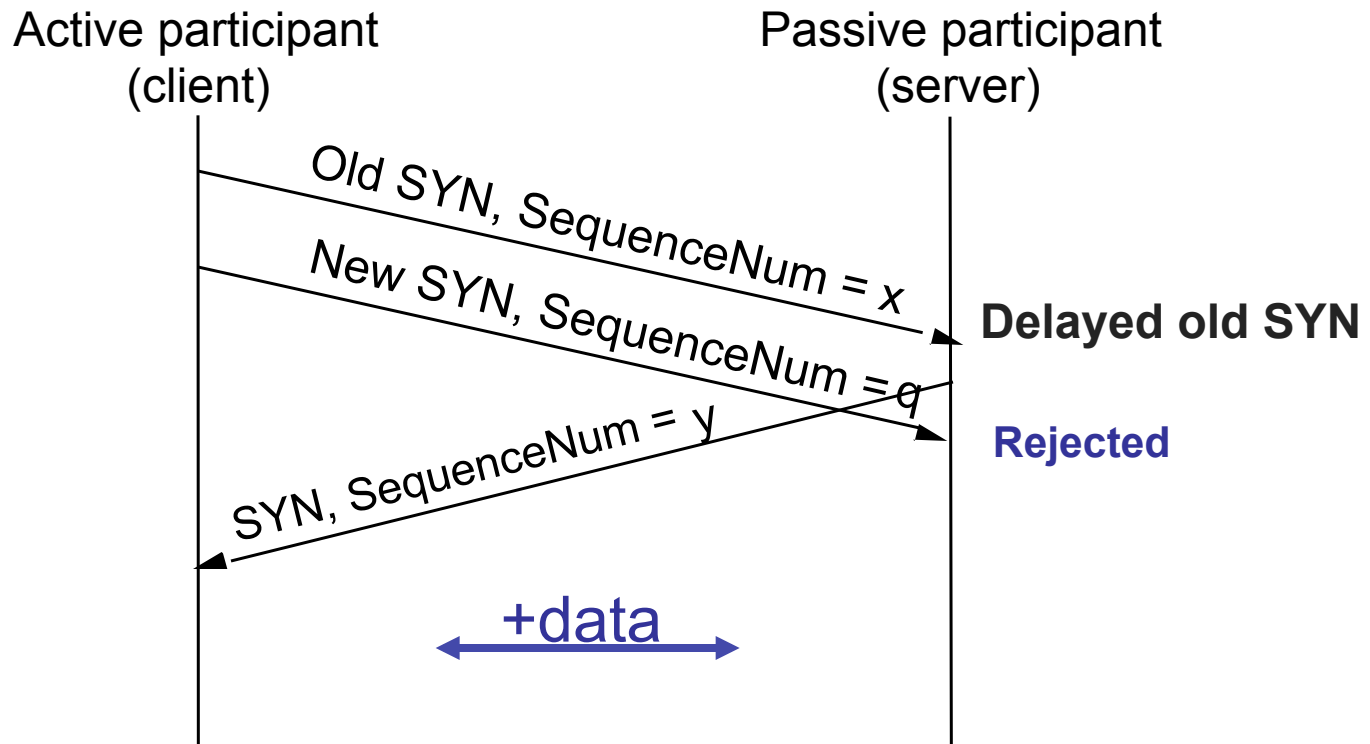
- Both sender and receiver must be ready before we start to transfer the data
 - ◆ Sender and receiver need to agree on a set of parameters
 - ◆ Most important: sequence number space in each direction
 - ◆ Lots of other parameters: e.g., the Maximum Segment Size
- Handshake protocols: setup state between 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?

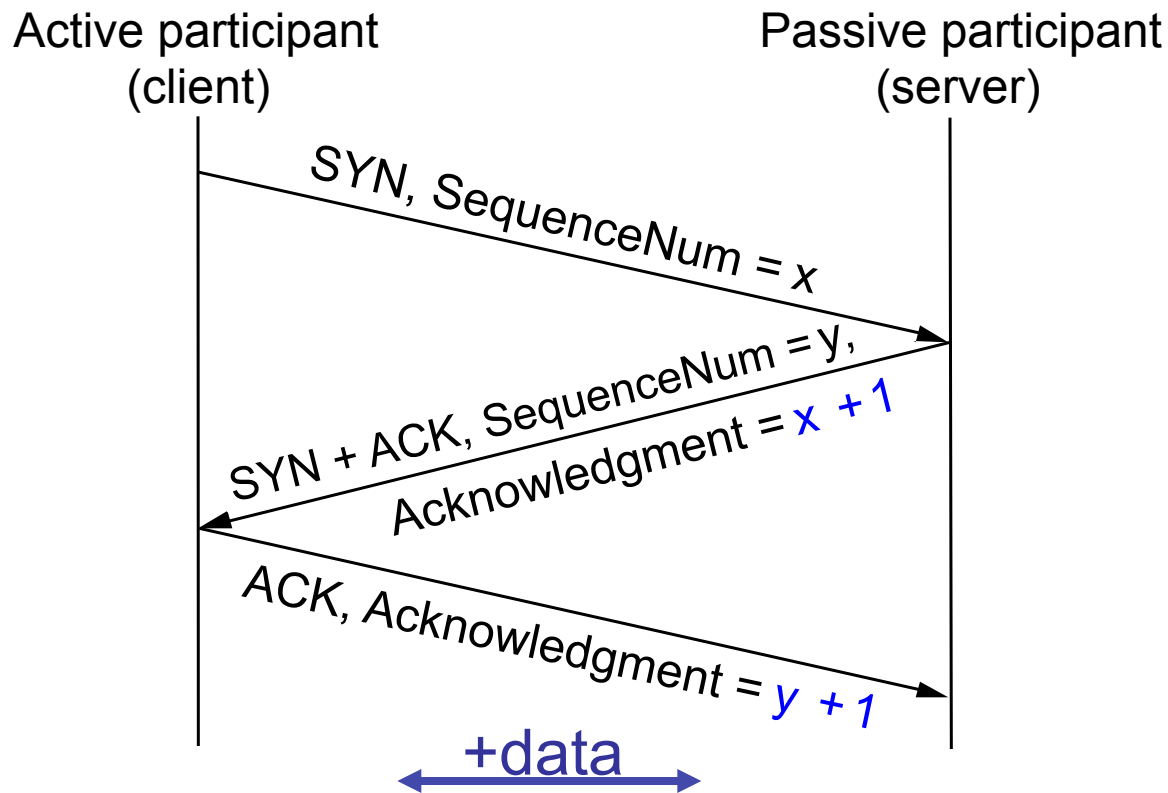
Two-way handshake?





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?)



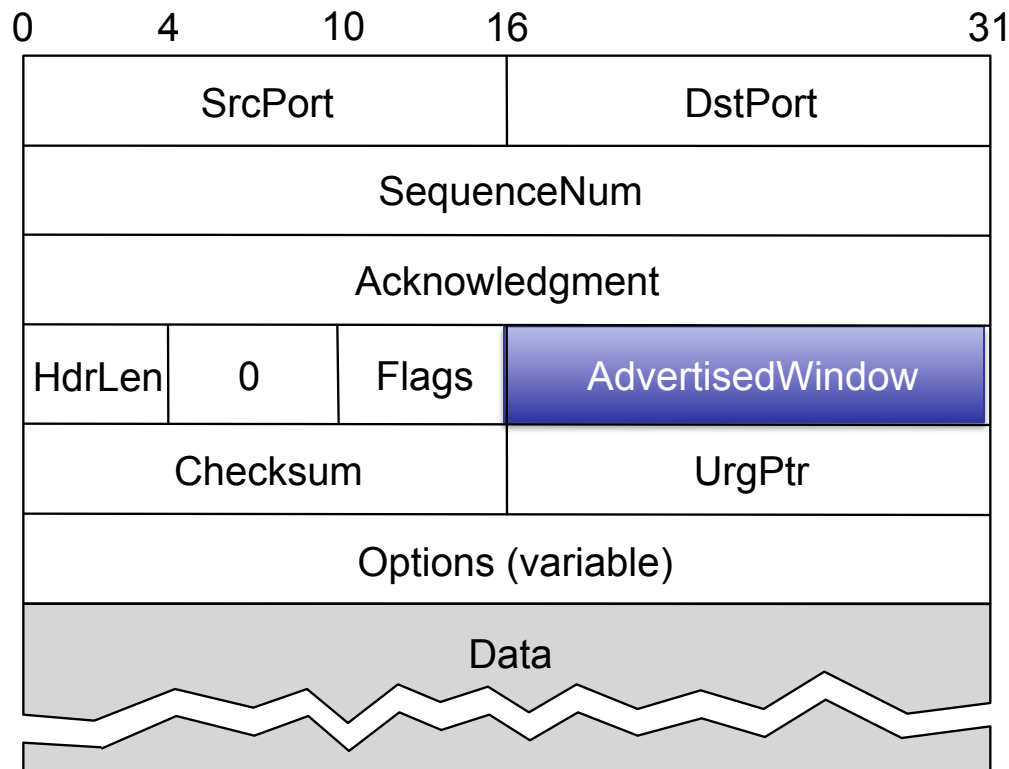
Flow Control

- 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

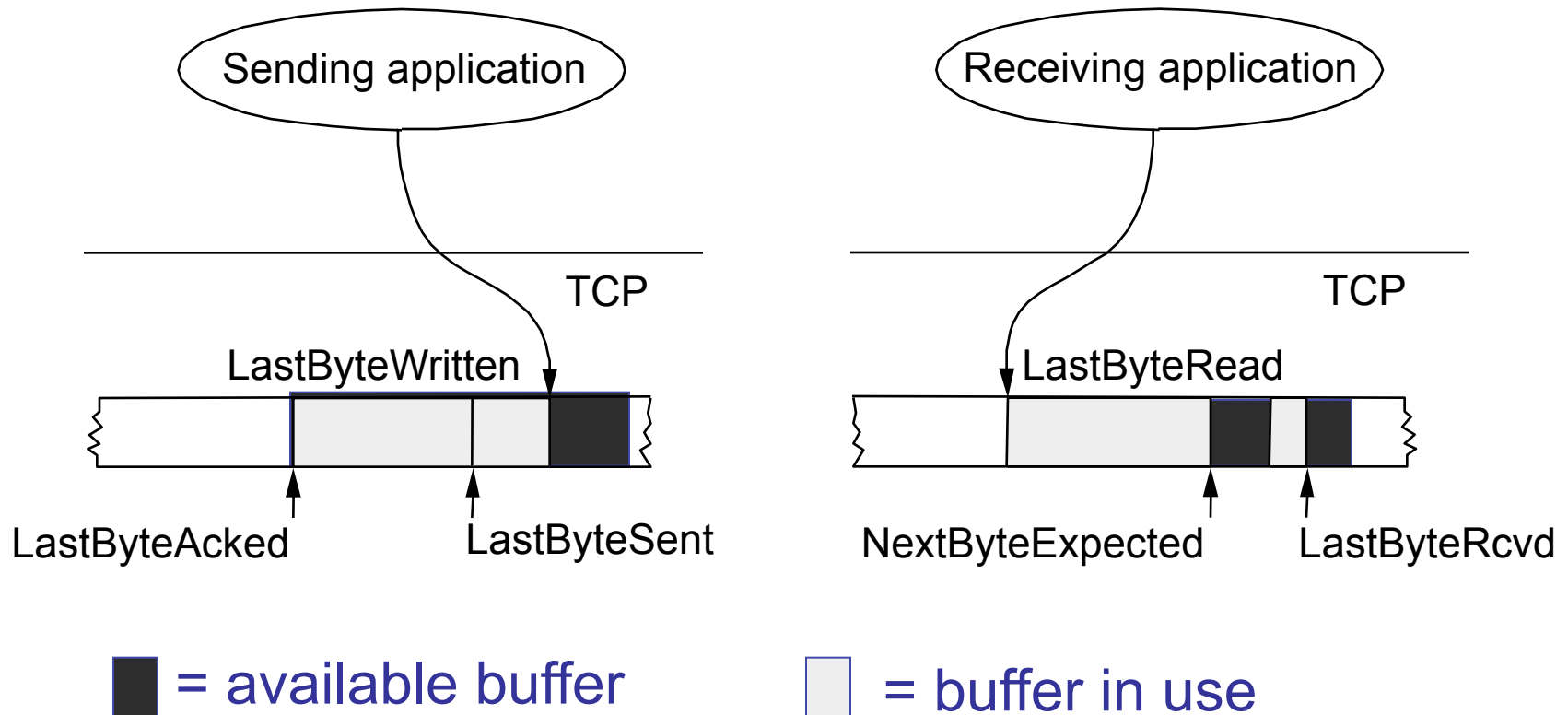


TCP Header Format

- Advertised window is used for flow control

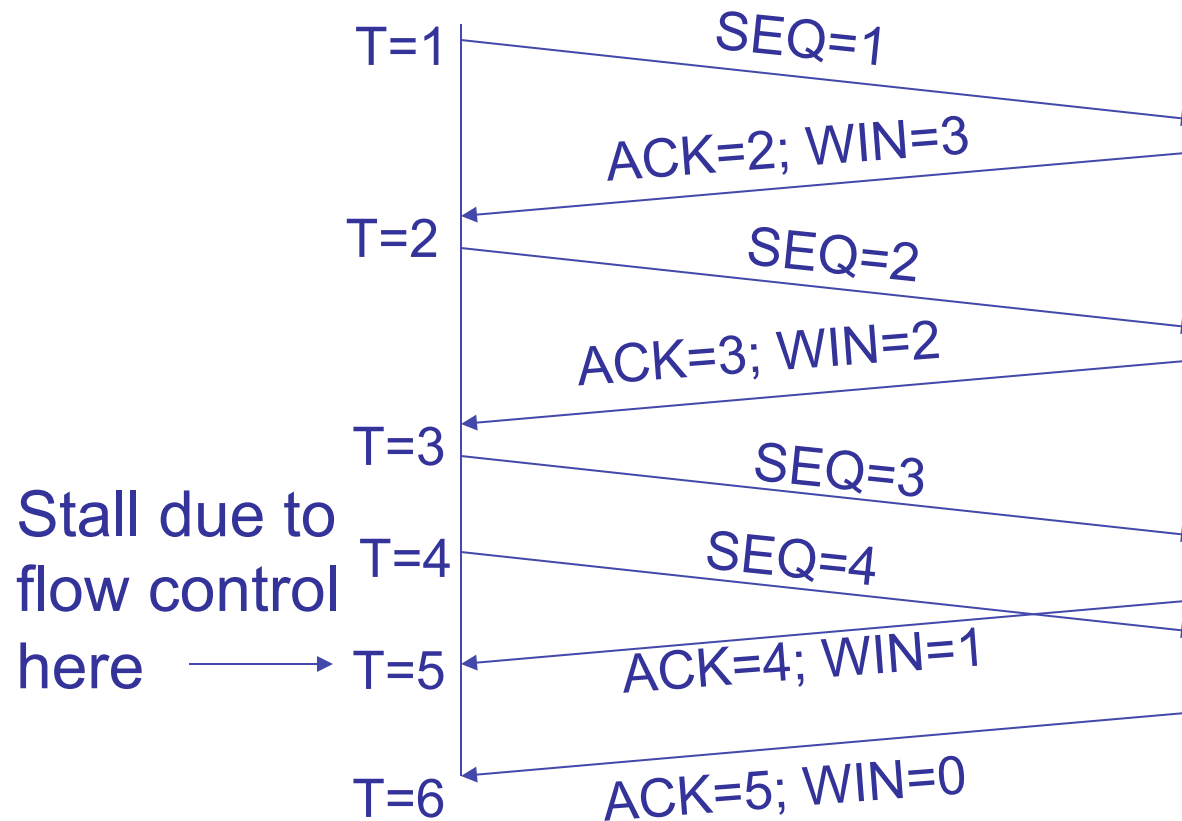


Sender and Receiver Buffering





Window-Size Example



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

Example – Buffer at Sender



T=1

1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---

T=2

1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---

T=3

1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---

T=4

1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---

T=5

1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---

T=6

1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---

 =acked
 =sent
 =advertised



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...

- Congestion control
- Read Ch 6.3-4 in P&D
- Guest lecture: Geoff Voelker