

# Lecture 7:

# Transport Layer Protocols

## (cont)

---

CSE 123: Computer Networks

Aaron Schulman



# Overview

---



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



# Transport Layer Tasks

---

- Define and provide specific delivery semantics
  - ◆ To which end point (which application/process)?
  - ◆ When?
  - ◆ How?
  - ◆ If?
- Multiplexing different processes on the same links
- ... and Reliability (if the application needs it)



# 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

# Establishing well-known ports



Network Working Group  
Request for Comments #322  
NIC #9609

V. Cerf  
J. Postel  
UCLA-NMC  
26 March 72

## Well Known Socket Numbers

The Network Measurement Group would like to establish a network standard socket number for a Process Discard service (not all HOSTs need cooperate.) To do this, it is necessary to know which sockets at each HOST have already been allocated to standard network functions.

At all HOSTs which permit login for services, socket 1 is the Network Logger on which ICP may be performed.

We would like to catalog other sockets which are supposed to be well-known, so we would appreciate having a note or phone call from each HOST Technical Liaison describing the function and socket numbers of network service programs at each HOST. The catalog will be published and we would recommend that it be maintained at NIC.

Dr. Vint Cerf or Jon Postel  
3804 Boelter Hall  
UCLA Computer Science Department  
Los Angeles, California 90024

(213) 825-4864 Vint

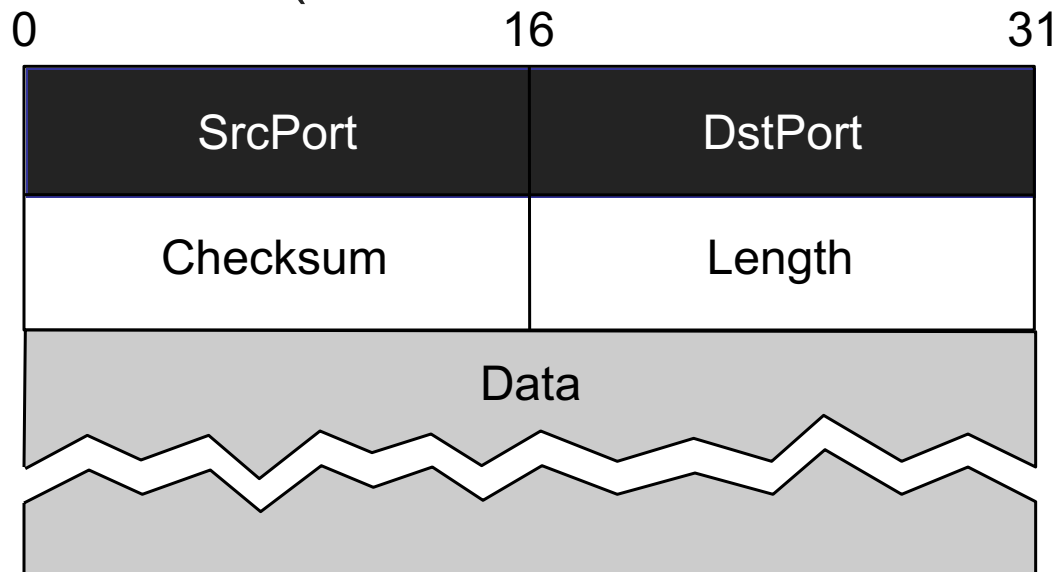
(213) 825-4733 Jon

(213) 825-2368 Secretary

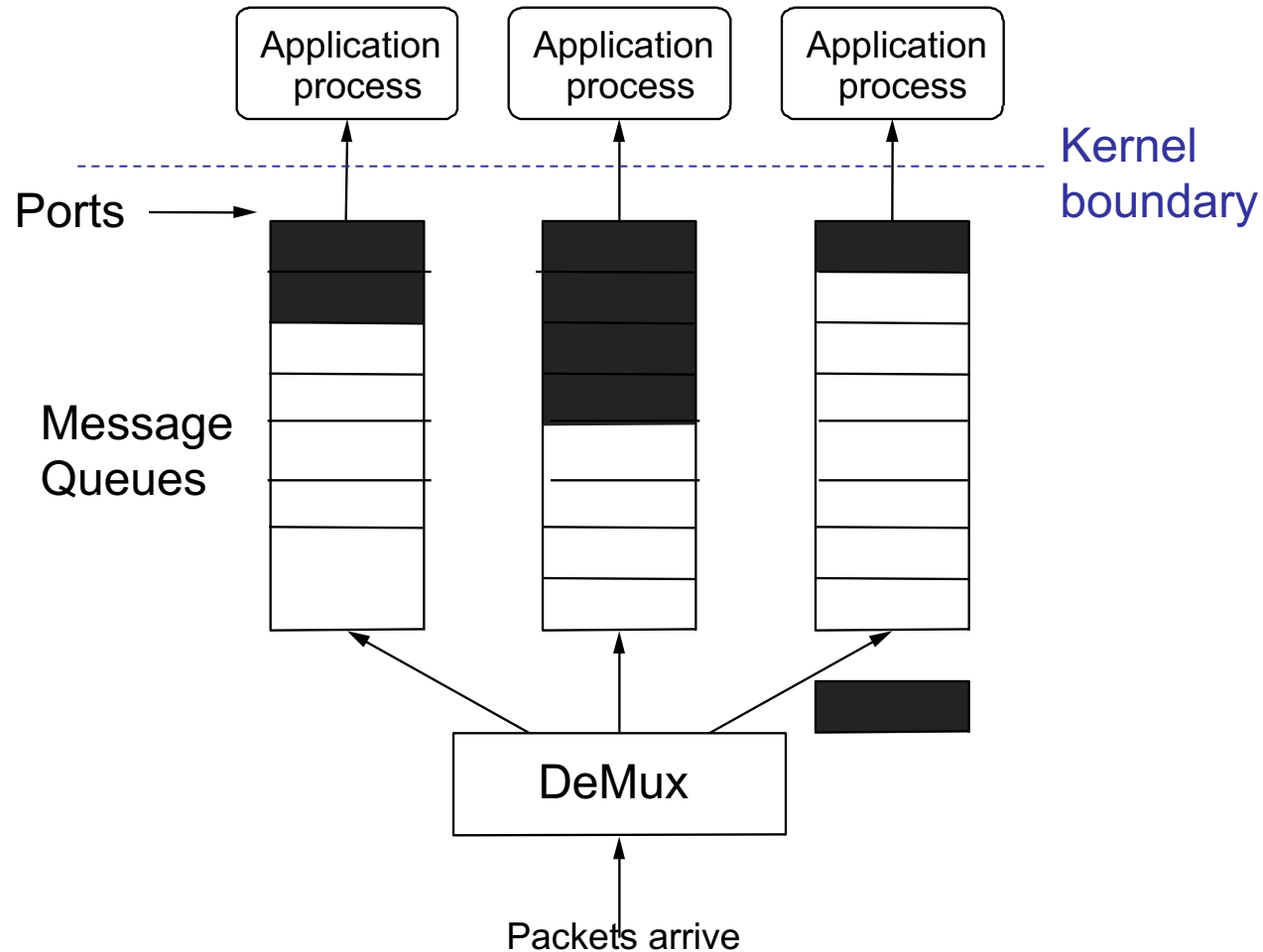
# User Datagram Protocol (UDP)



- 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)



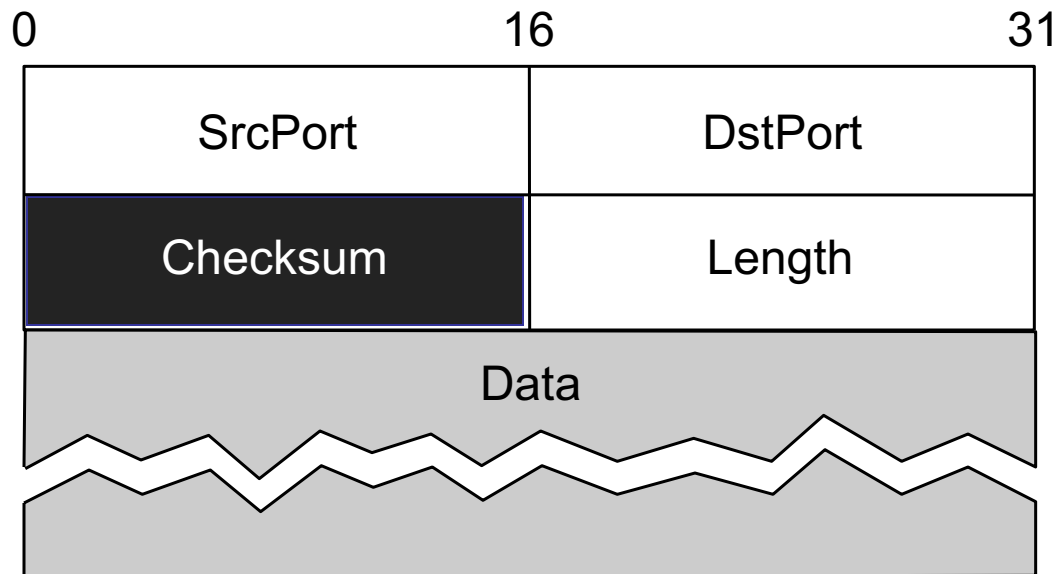
# UDP Delivery





# 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?

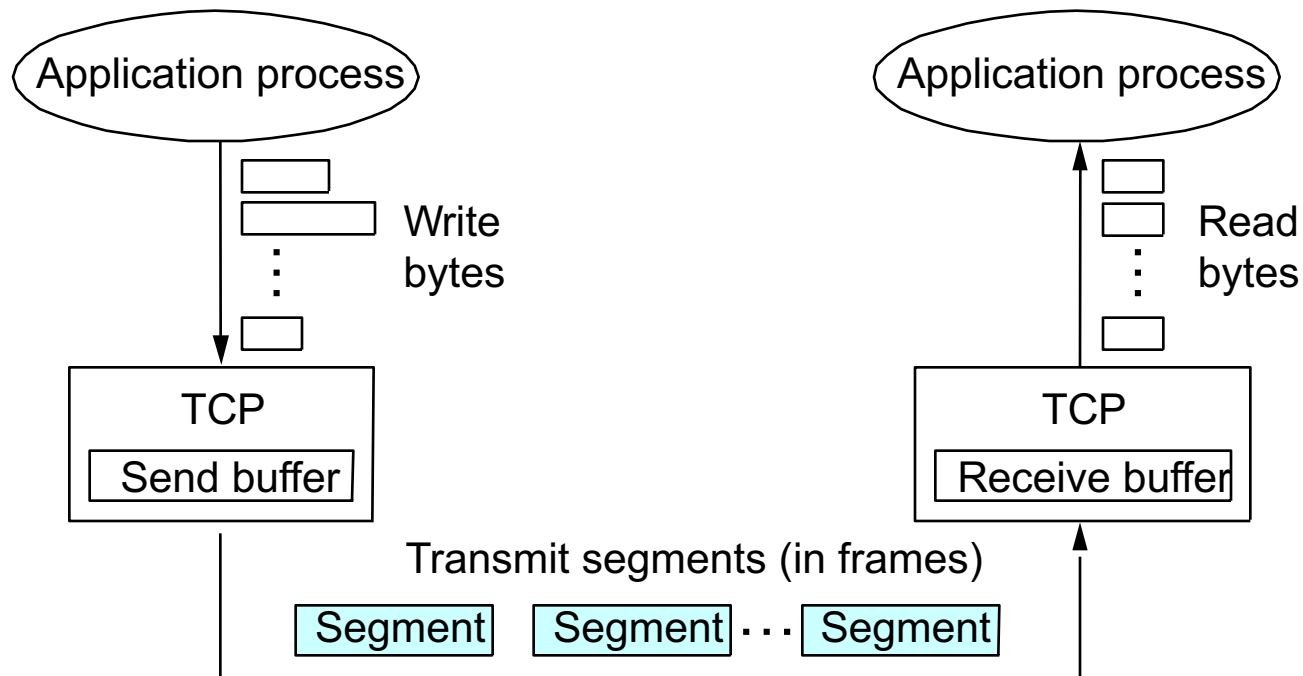
# Transmission Control Protocol

---



- Reliable bi-directional **bytestream** between processes
  - ◆ Uses a sliding window protocol!
- Connection-oriented
  - ◆ Conversation between two endpoints with a beginning and end
- Flow control (last lecture) [*receiver window*]
  - ◆ Prevents sender from over-running receiver buffers
  - ◆ (tell sender how much buffer is left at receiver)
- Congestion control (later in term) [*sender window*]
  - ◆ Prevents sender from over-running network capacity

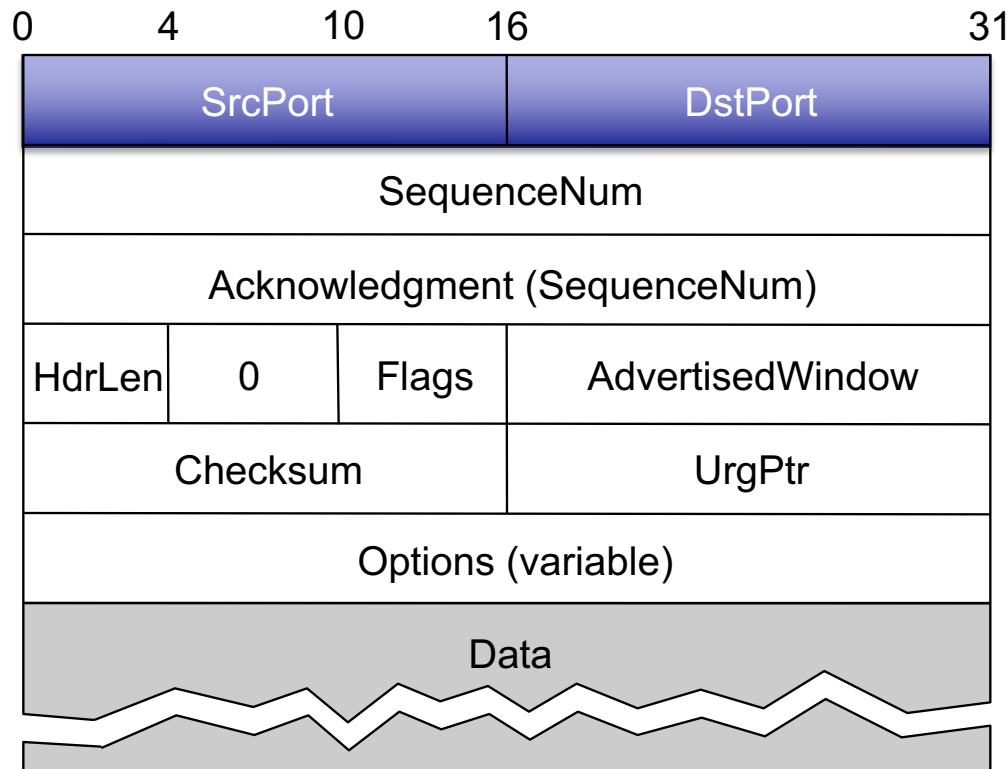
# TCP Delivery





# TCP Header Format

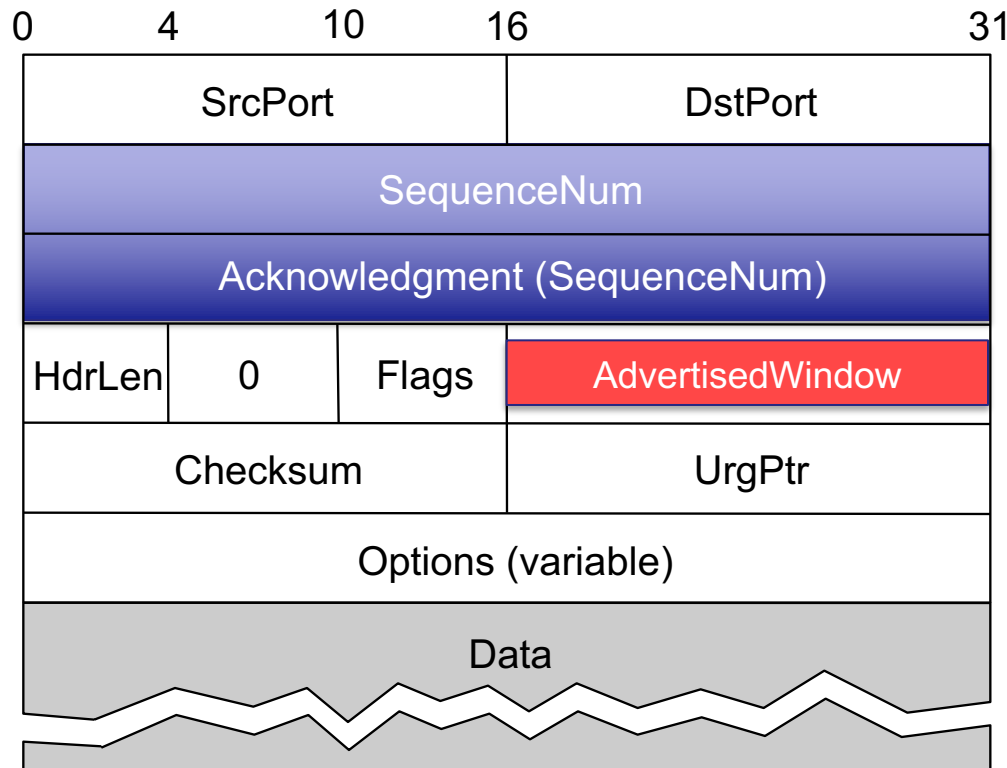
- Ports plus IP addresses identify a connection (4-tuple)





# TCP Header Format

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

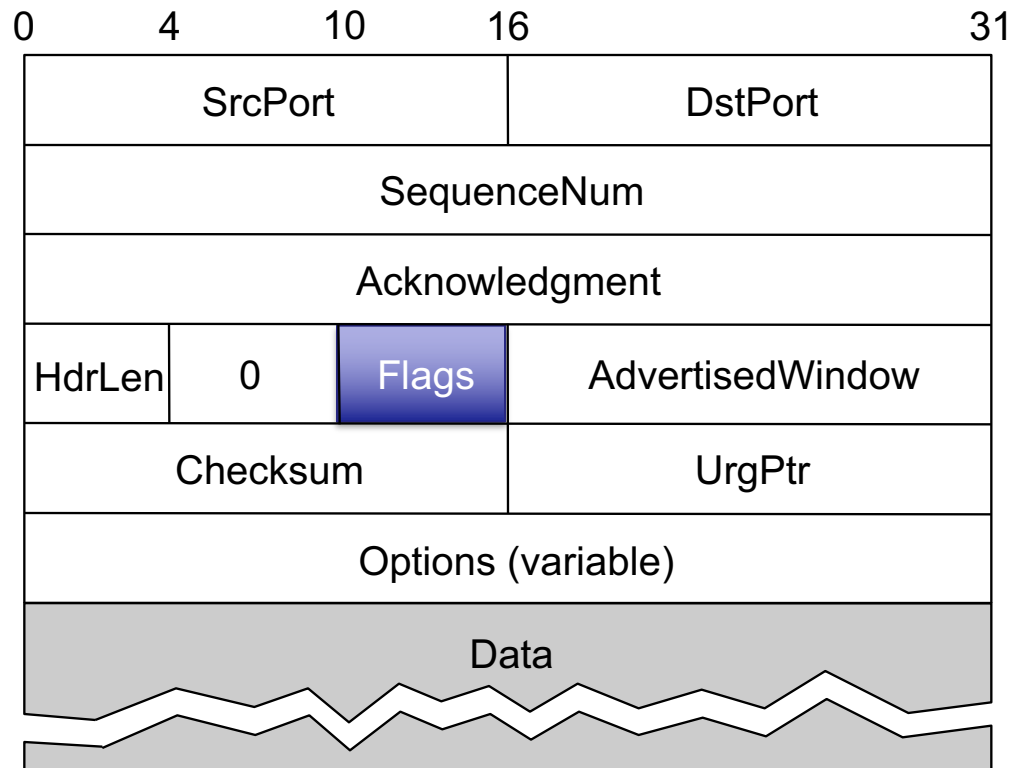


**Flow control –**  
Receiver tells the  
transmitter how big its  
remaining window is



# TCP Header Format

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



# Connection Establishment

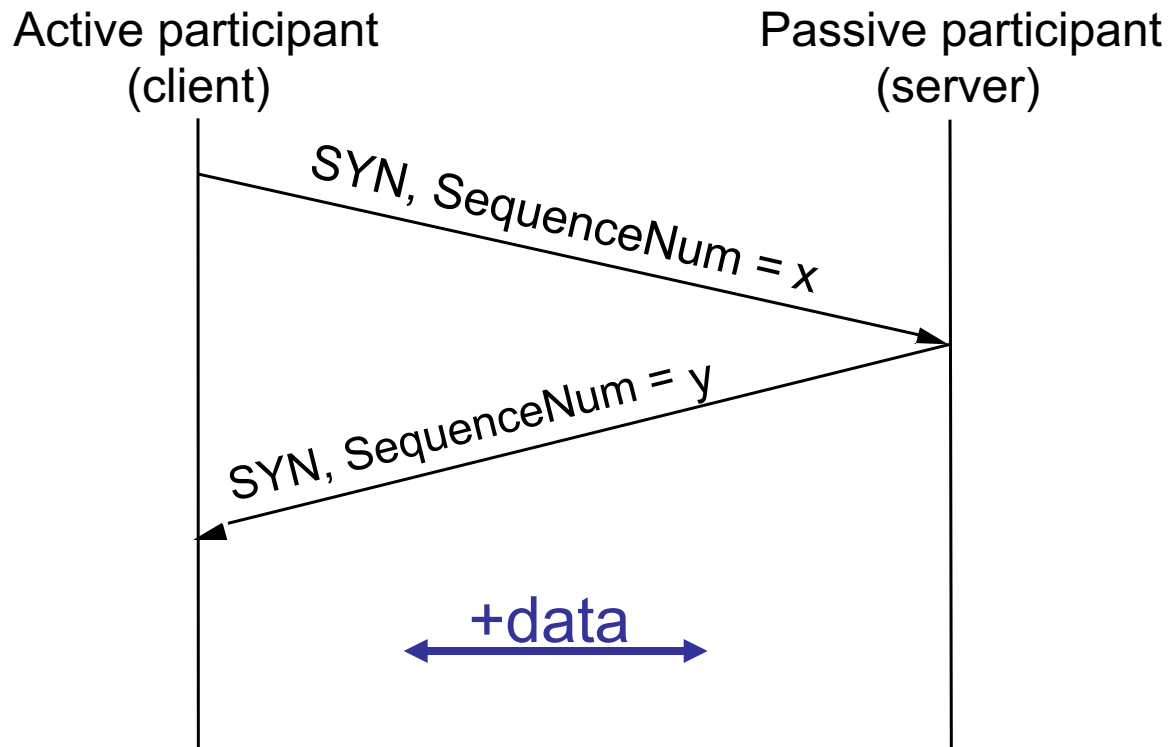
---



- 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
  - ◆ Need to deal with **delayed** and **reordered** packets



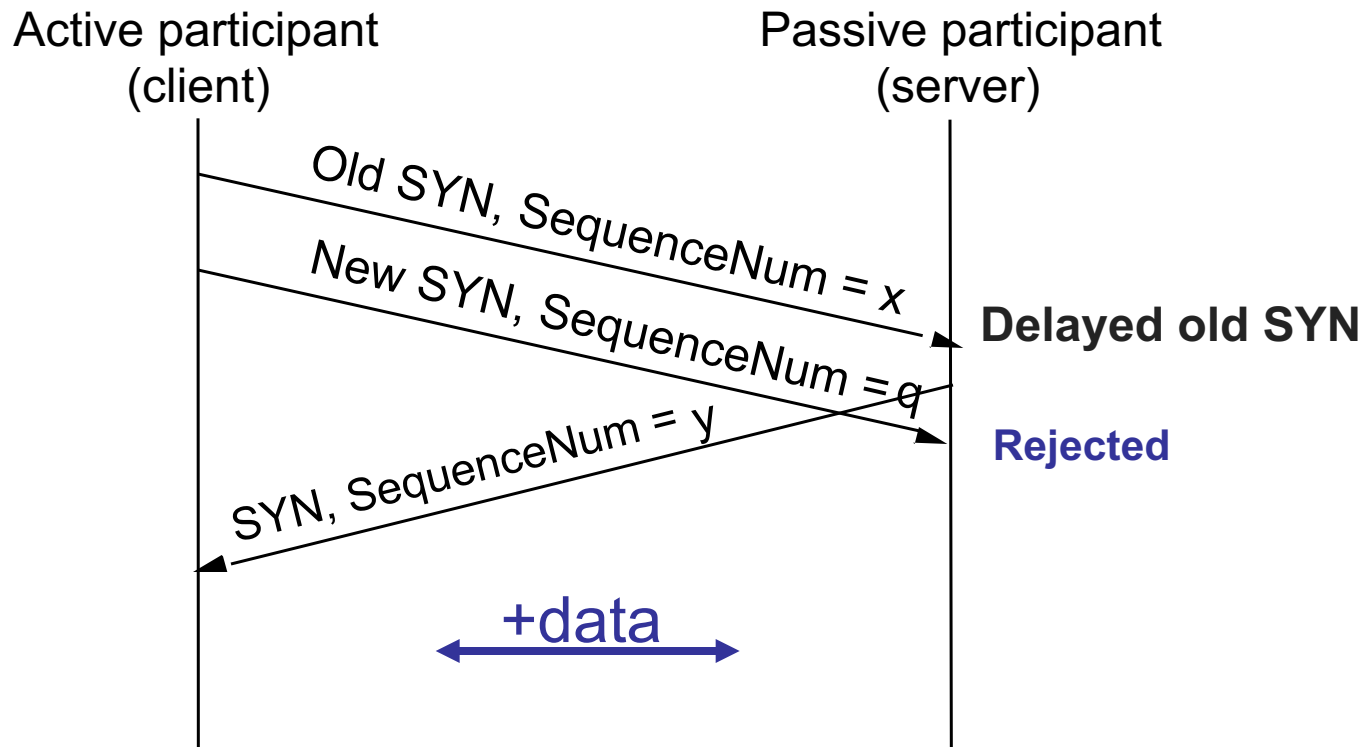
# Two-way handshake?



**What's wrong here?**



# Two-way handshake?

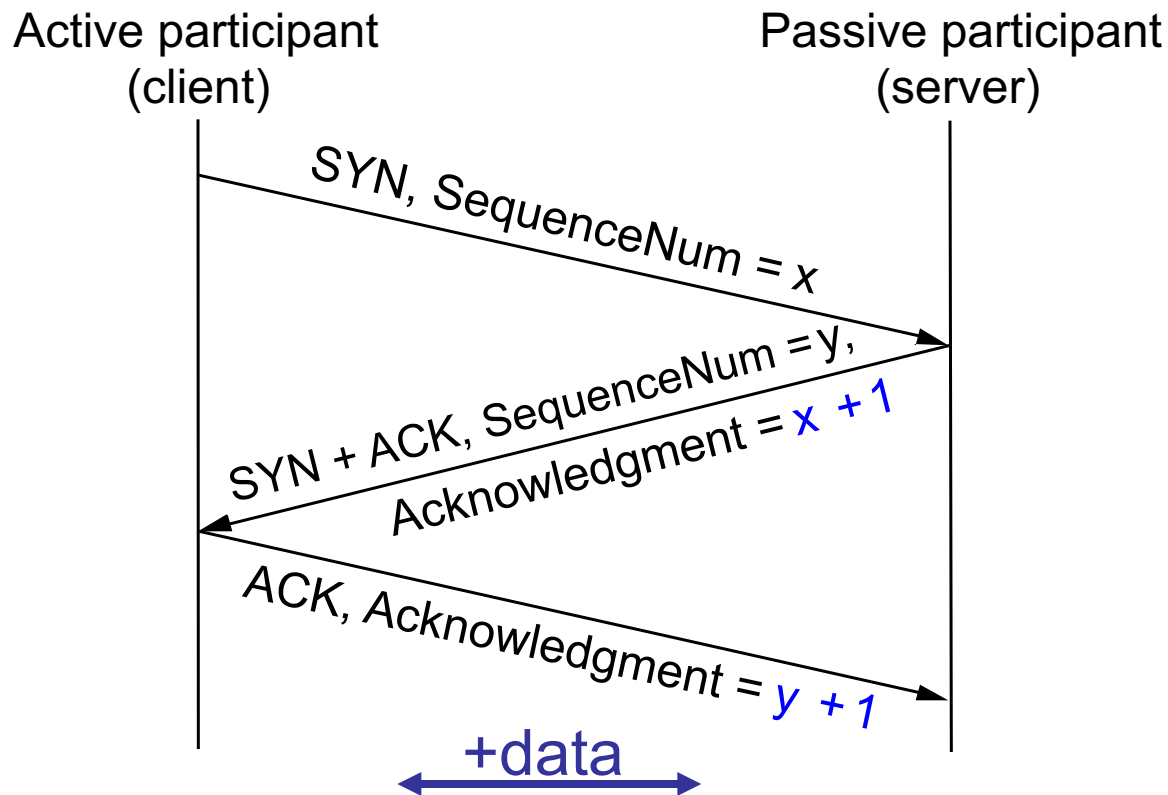


Client and server do not agree they are starting the same connection



# Three-Way Handshake

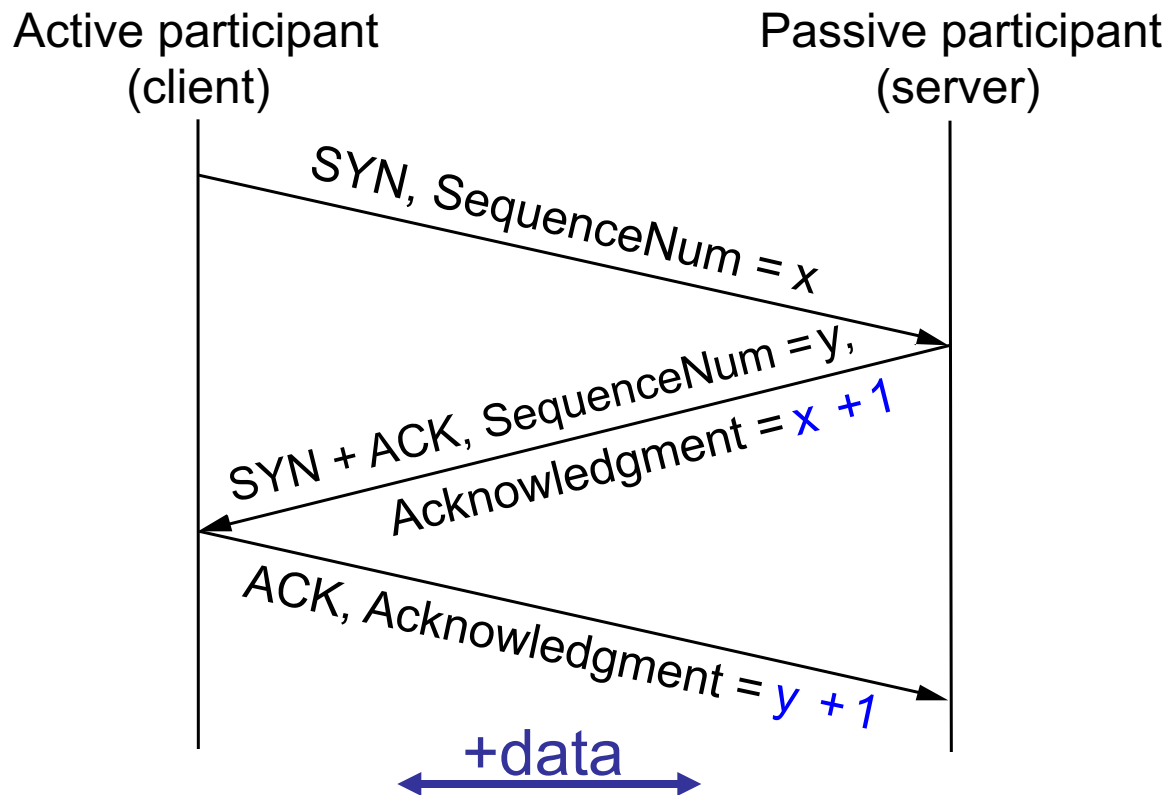
- Opens both directions for transfer





# Three-Way Handshake

- Opens both directions for transfer (two Seqnum spaces)





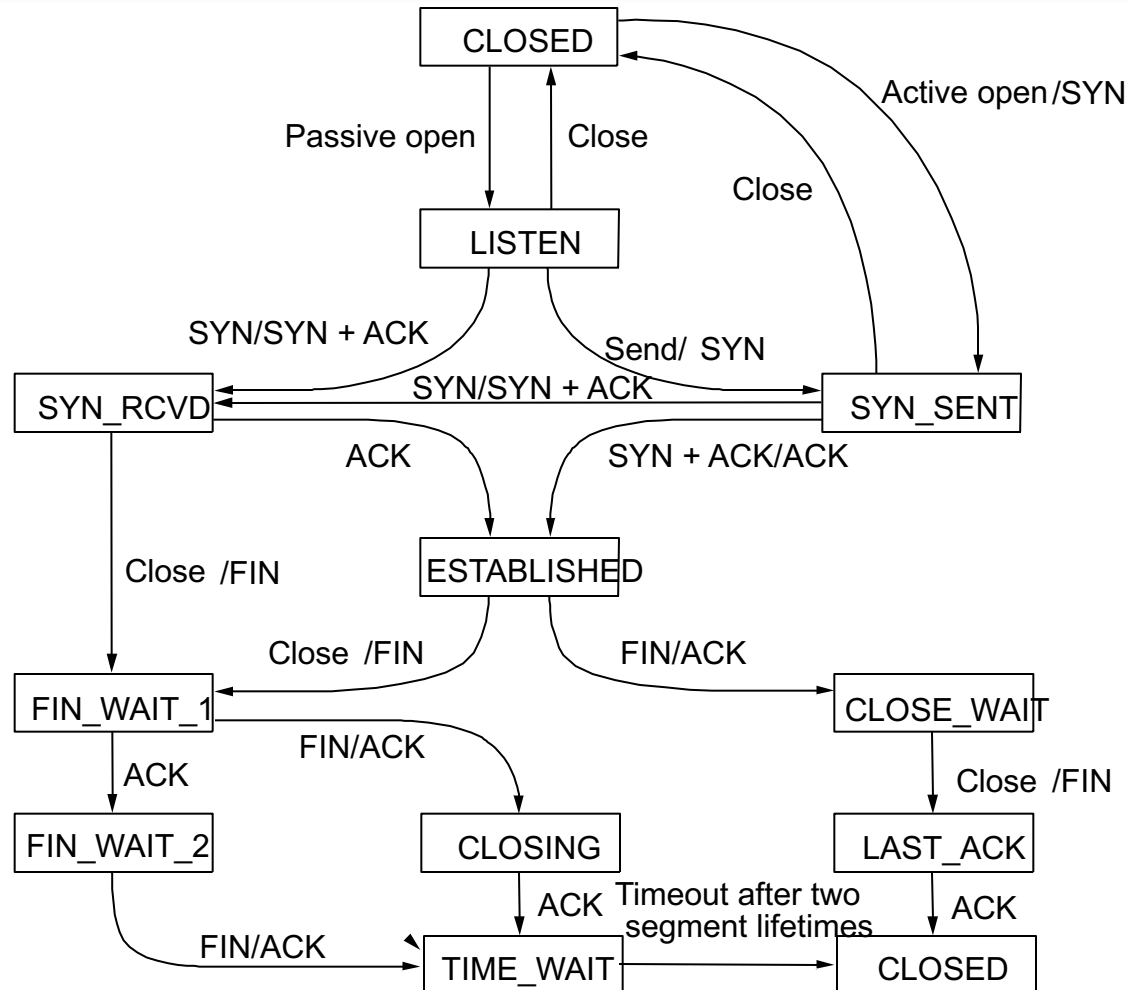
# Some Comments

---

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

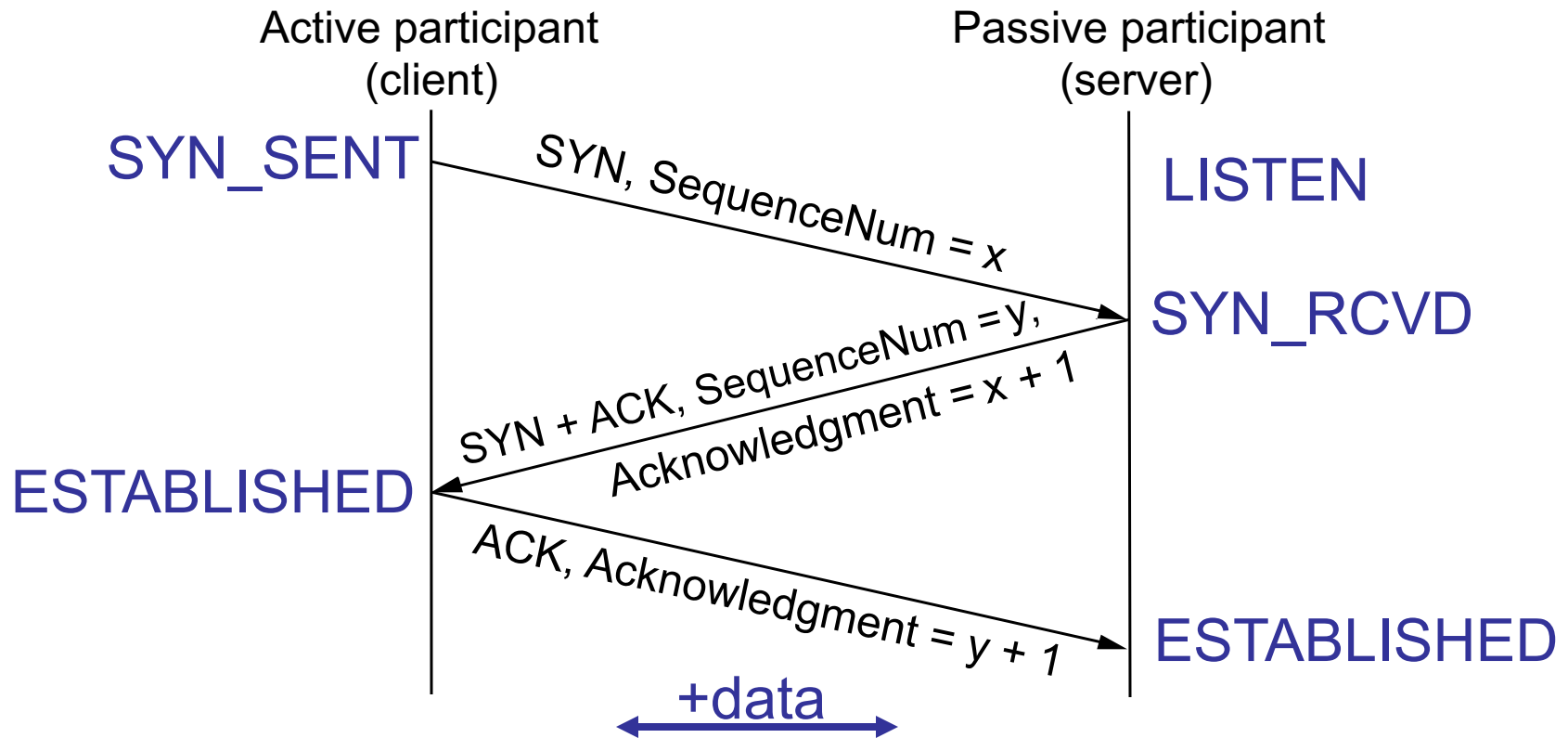


# TCP State Transitions





# Again, with States



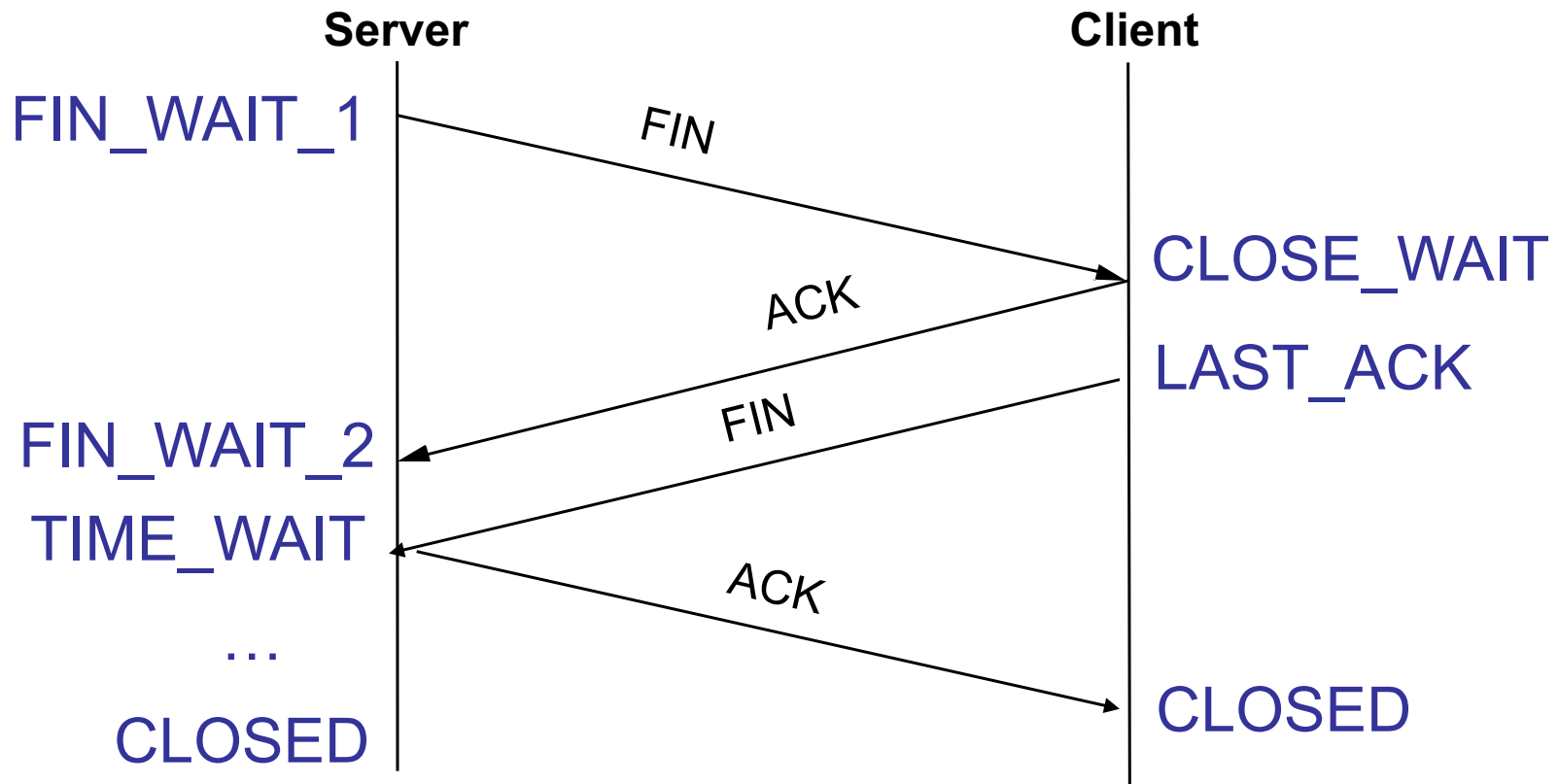


# Connection Teardown

---

- 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, why? You don’t know when the other side will be done!

# TCP Connection Teardown





# The TIME\_WAIT State

---

- We wait  $2*MSL$  (maximum segment lifetime) before completing the close
- Maximum segment lifetime is usually 60 seconds (longest period of time packet can be stuck in network)
  - ◆ 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?



# 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



# For next time...

---

- Connecting links together
- Read Ch in P&D 3-3.3