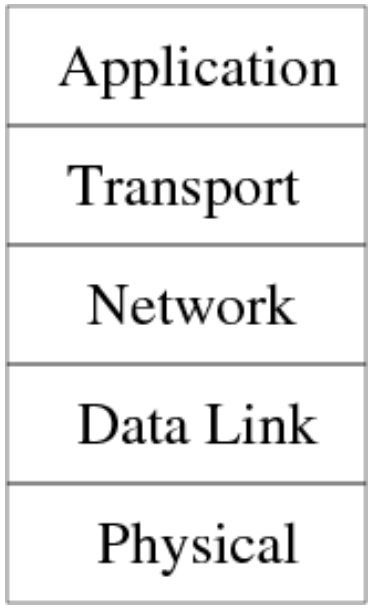
Layered Architecture



Transport Layer - Network Layer

Transport Layer - Network Layer

- **Transport Layer Protocols:** provide a logical communication between processes running on different hosts.
- **Network Layer Protocols:** provide a logical communication between hosts.

Transport Layer - Addressing

- **Transport Layer Protocols:** Addresses Application processes on Hosts.
- **Network Layer Protocols:** Addresses Interfaces (globally) using IP addresses
- **Data Link Layer:** Addresses Adapters (locally) using MAC addresses.

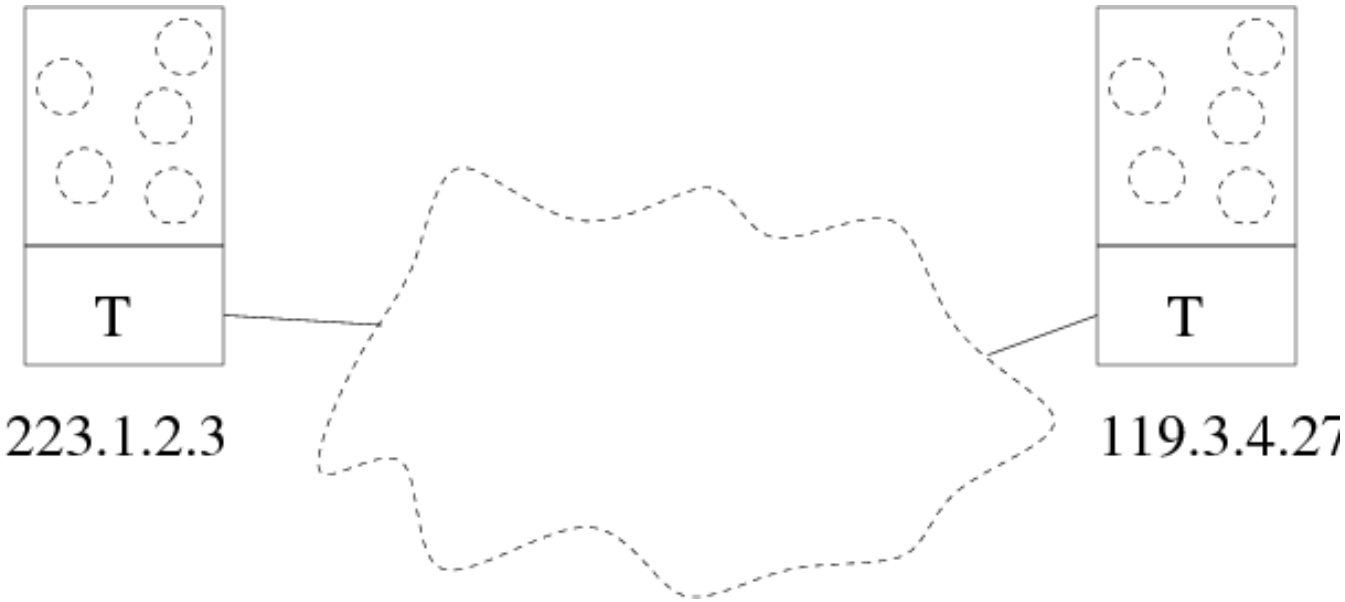
Internet Transport Layer Protocols

- UDP (User Datagram Protocol)
  - Application Multiplexing/Demultiplexing
  - Error Detection
- TCP (Transmission Control Protocol)
  - Application Multiplexing/Demultiplexing
  - Error Detection
  - Reliable Data Transfer
  - Congestion Control

Goal

- Understand Application Multiplexing/Demultiplexing
- Understand TCP
- Know Issues in Congestion Control

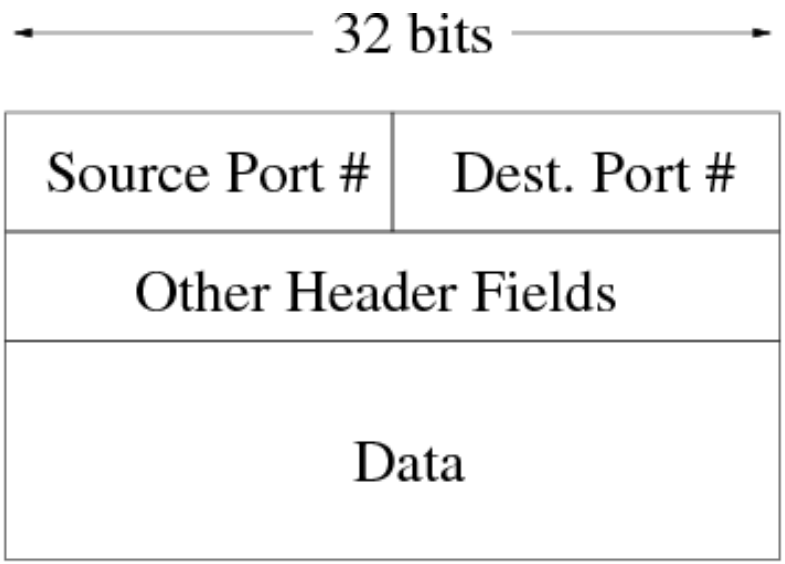
Application Multiplexing/Demultiplexing



- Many Application Processes at one Host
- Many Processes of the same Type

$->$ Need to Address Processes

Application Multiplexing/Demultiplexing



- Port Number: 16 bits (0-65,535)
- Well-Known Port numbers: 0-1023
  - HTTP: 80
  - FTP: 21

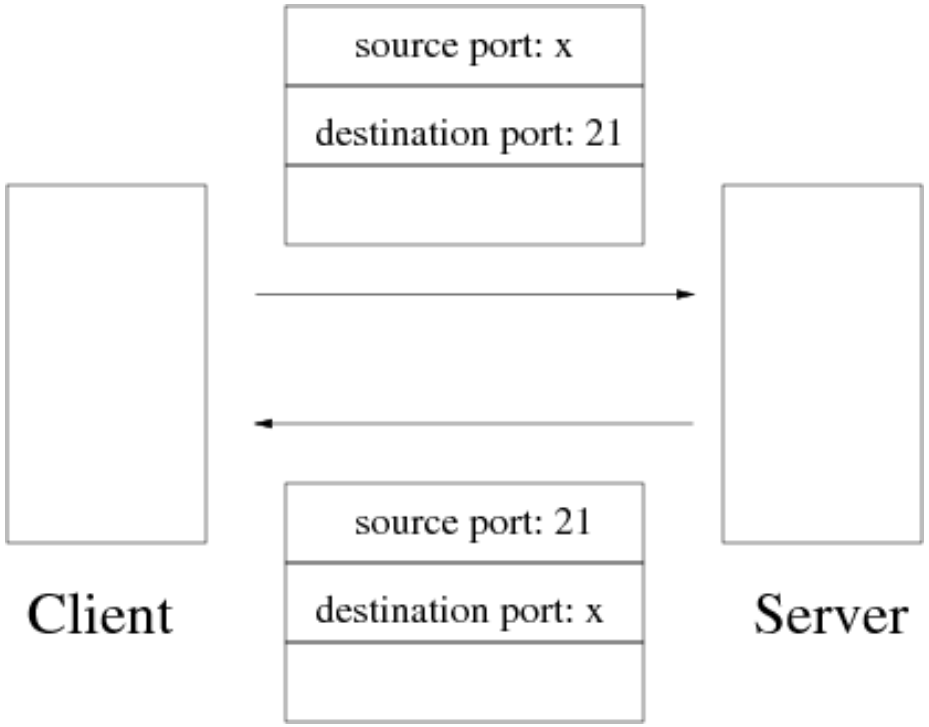
Application Multiplexing/Demultiplexing

Why two port numbers?

- A host maybe be running two processes of the same type at the same time, and thus the port destination number of an application may not be enough to identify a specific process.

Application Multiplexing/Demultiplexing

How is second port number created?

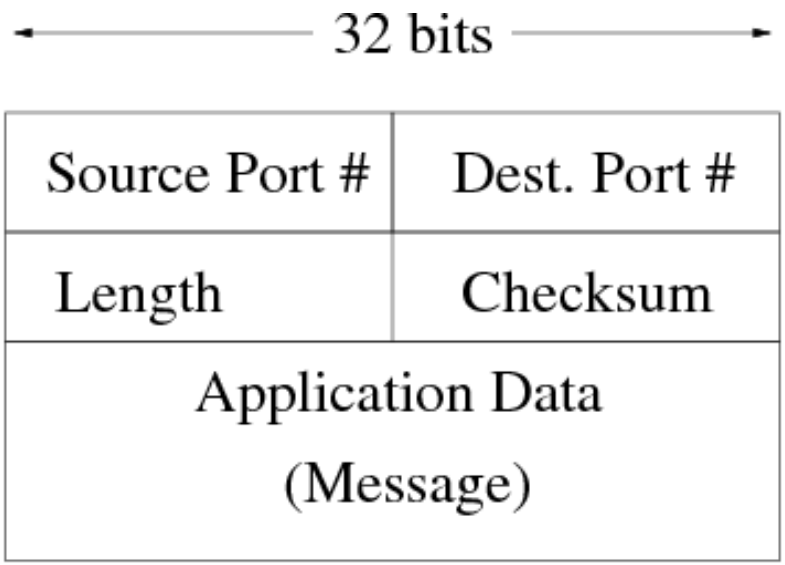


Application Multiplexing/Demultiplexing

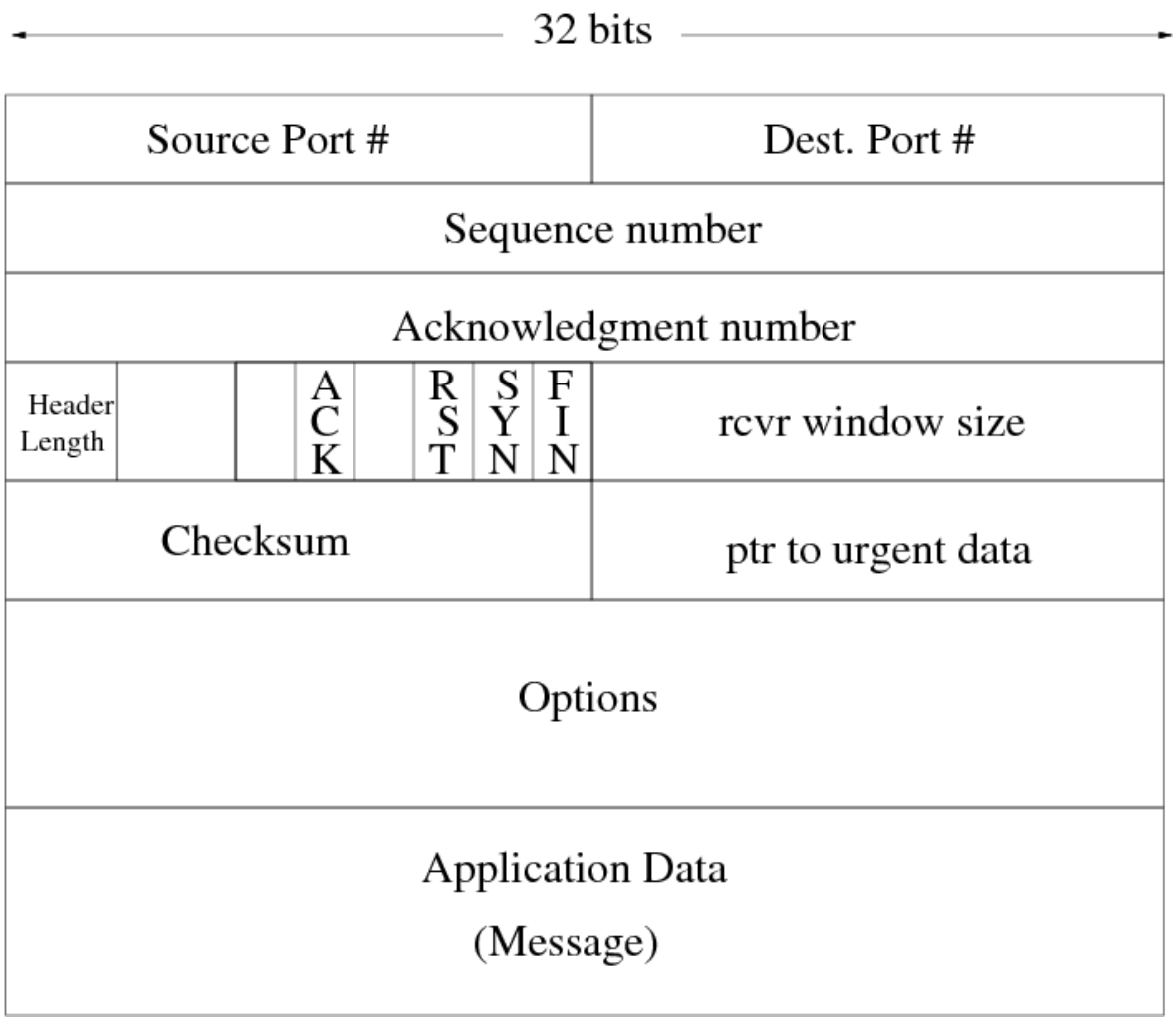
What happens when two clients use the same destination port number?

- A triplet (source IP address, source port number, destination port number) is used to identify an application process.

UDP Segment Structure



TCP Segment Structure



TCP Segment Structure

Sequence number: TCP identifies the sequence number for each byte (rather than each segment). Why?
rcv window-size: Buffer space available at the receiver.
Flags:

- **ACK:** Acknowledgment number is valid
- **RST:** Reset connection
- **SYN:** Connection setup request
- **FIN:** Connection release

Options:

- Maximum Segment Size (MSS)
- Timestamping

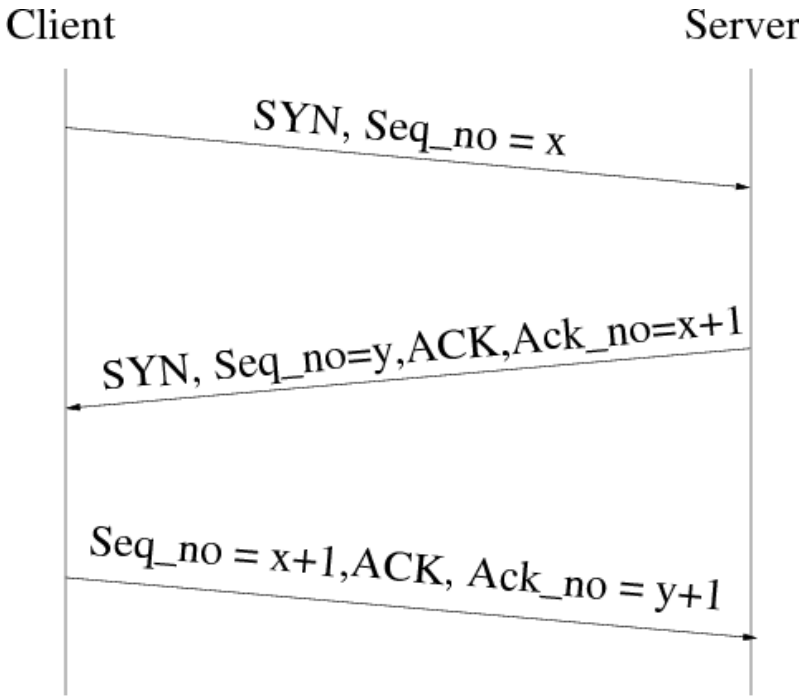
Preventing Duplicate Sequence Number

Given that a packet "dies" after T seconds, there is a way to safely establish a connection and avoid packets from different sessions getting confused.
Need two mechanisms

(1) Using a clock to determine sequence numbers.

(2) Connection setup through "three way handshake"

TCP Connection Setup: Three Way Handshake



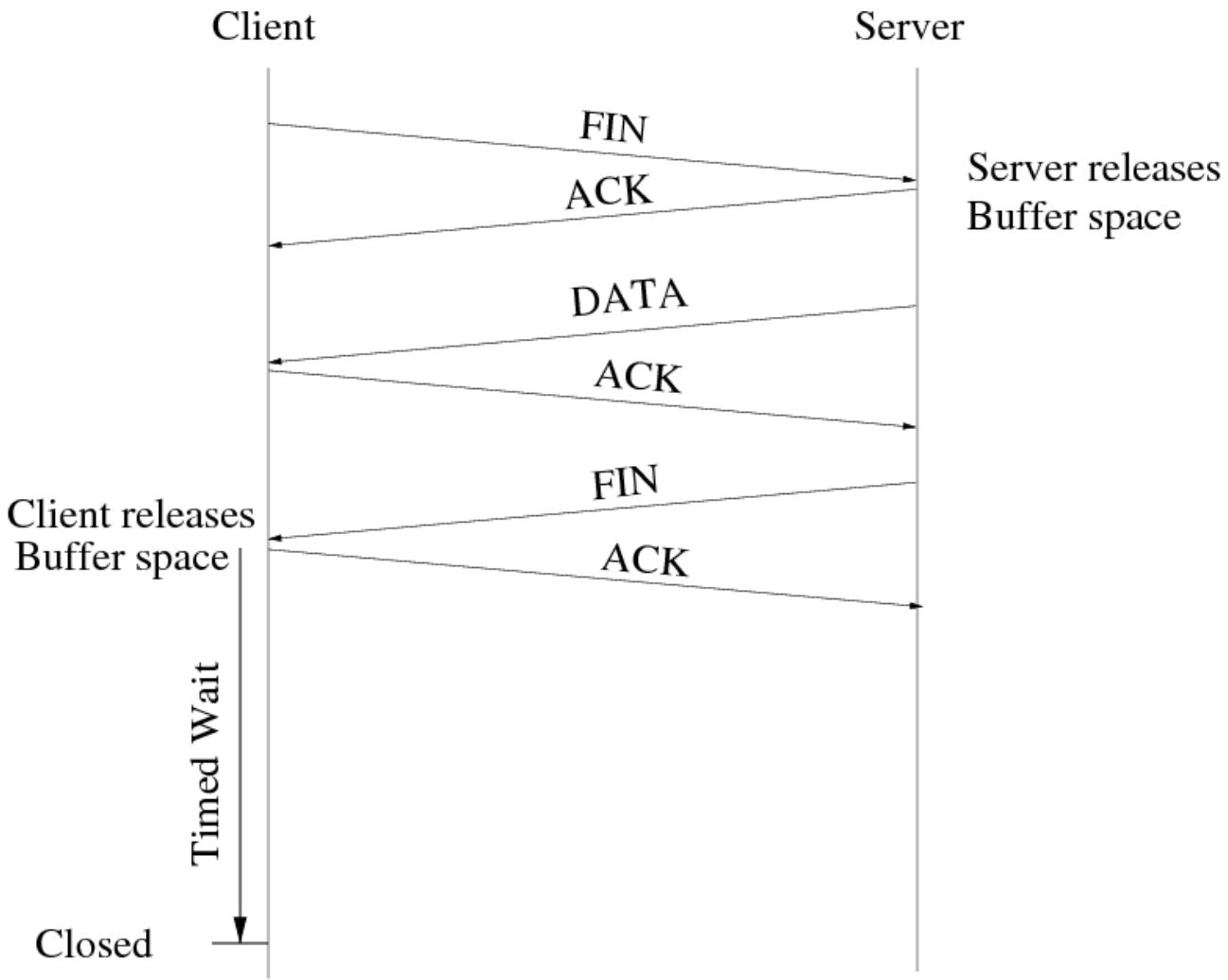
TCP Connection Setup: Three Way Handshake

(1) Client sends a connection-request segment, and registers its initial sequence number.

(2) Server allocates buffer space, send connection-granted segment, and registers its own initial sequence number.

(3) Client allocates buffer space, and acknowledges ‘connection-granted.

Closing a TCP Connection



TCP: Timeout

EstimatedRTT =

Timeout =

Deviation =