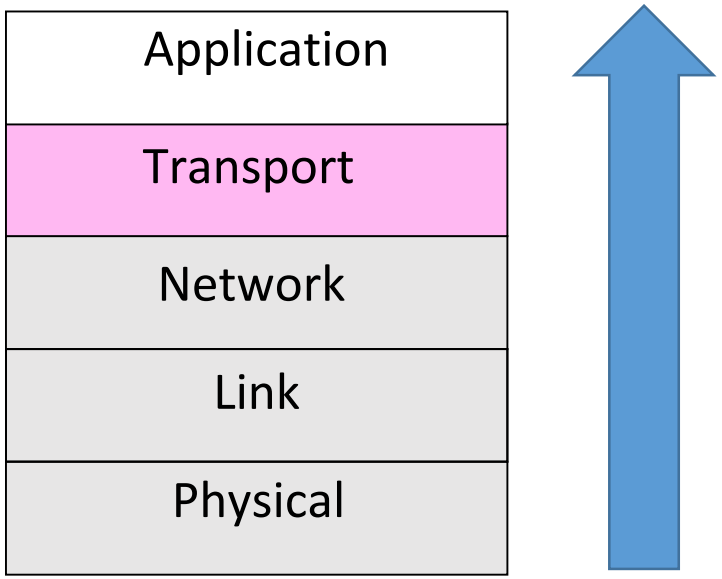
Transport Layer (TCP/UDP)

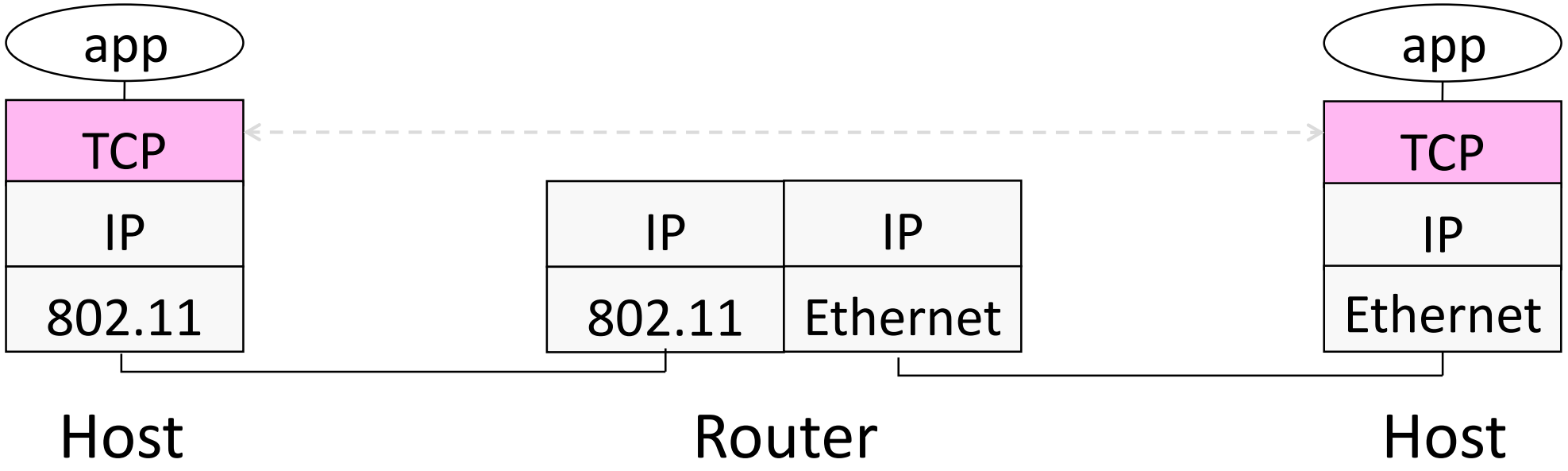
Where we are in the Course

- Moving on up to the Transport Layer!



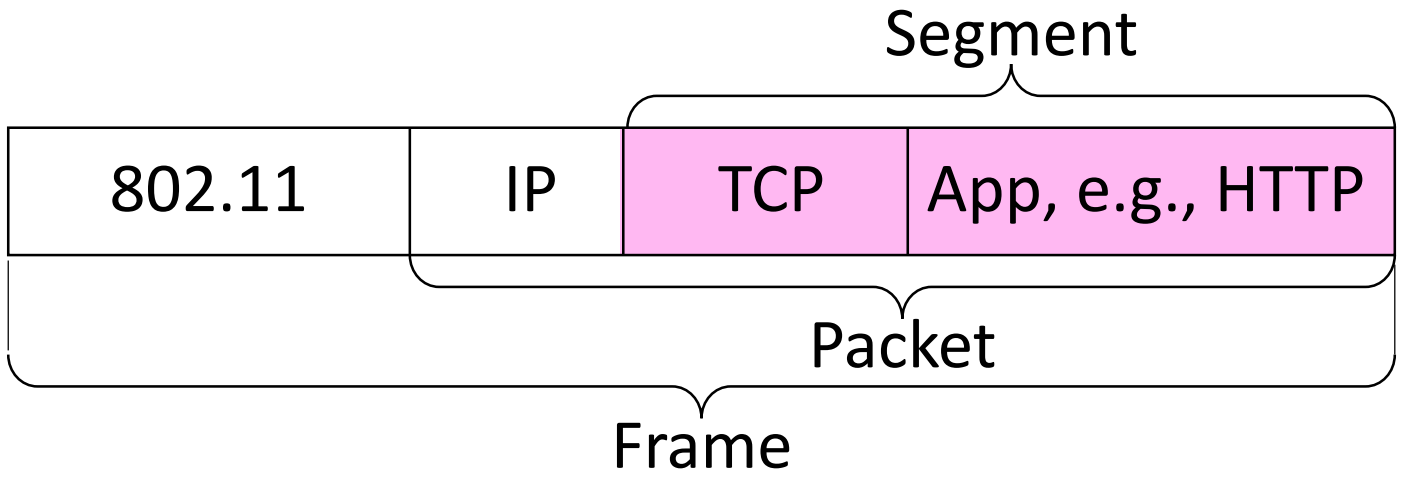
Recall

- Transport layer provides end-to-end connectivity across the network



Recall (2)

- Segments carry application data across the network
- Segments are carried within packets within frames



Transport Layer Services

- Provide different kinds of data delivery across the network to applications

	Unreliable	**Reliable**
Messages	Datagrams (UDP)	
Bytestream		Streams (TCP)

Comparison of Internet Transports

- TCP is full-featured, UDP is a glorified packet

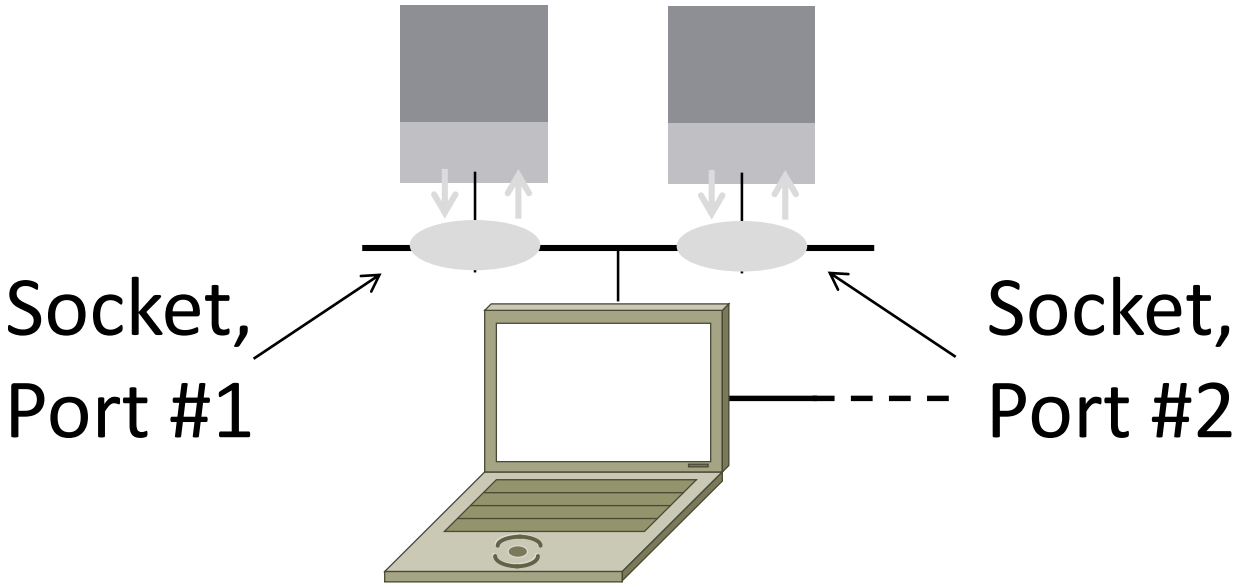
TCP (Streams)	UDP (Datagrams)
Connections	Datagrams
Bytes are delivered once, reliably, and in order	Messages may be lost, reordered, duplicated
Arbitrary length content	Limited message size
Flow control matches sender to receiver	Can send regardless of receiver state
Congestion control matches sender to network	Can send regardless of network state

Socket API

- Simple abstraction to use the network
  - The “network” API (really Transport service) used to write all Internet apps
  - Part of all major OSes and languages; originally Berkeley (Unix) ~1983
- Supports both Internet transport services (Streams and Datagrams)

Socket API (2)

- Sockets let apps attach to the local network at different ports



Socket API (3)

- Same API used for Streams and Datagrams

	Primitive	Meaning
	SOCKET	Create a new communication endpoint
	BIND	Associate a local address (port) with a socket
Only needed for Streams	LISTEN	Announce willingness to accept connections
Only needed for Streams	ACCEPT	Passively establish an incoming connection
Only needed for Streams	CONNECT	Actively attempt to establish a connection
To/From for Datagrams	SEND(TO)	Send some data over the socket
To/From for Datagrams	RECEIVE(FROM)	Receive some data over the socket
	CLOSE	Release the socket

Ports

- Application process is identified by the tuple IP address, protocol, and port
  - Ports are 16-bit integers representing local “mailboxes” that a process leases
- Servers often bind to “well-known ports”
  - <1024, require administrative privileges
- Clients often assigned “ephemeral” ports
  - Chosen by OS, used temporarily

Some Well-Known Ports

Port	Protocol	Use
20, 21	FTP	File transfer
22	SSH	Remote login, replacement for Telnet
25	SMTP	Email
80	HTTP	World Wide Web
110	POP-3	Remote email access
143	IMAP	Remote email access
443	HTTPS	Secure Web (HTTP over SSL/TLS)
543	RTSP	Media player control
631	IPP	Printer sharing

Topics

- Service models
  - Socket API and ports
  - Datagrams, Streams
- User Datagram Protocol (UDP)
- Connections (TCP)
- Sliding Window (TCP)
- Flow control (TCP)
- Retransmission timers (TCP)
- Congestion control (TCP)

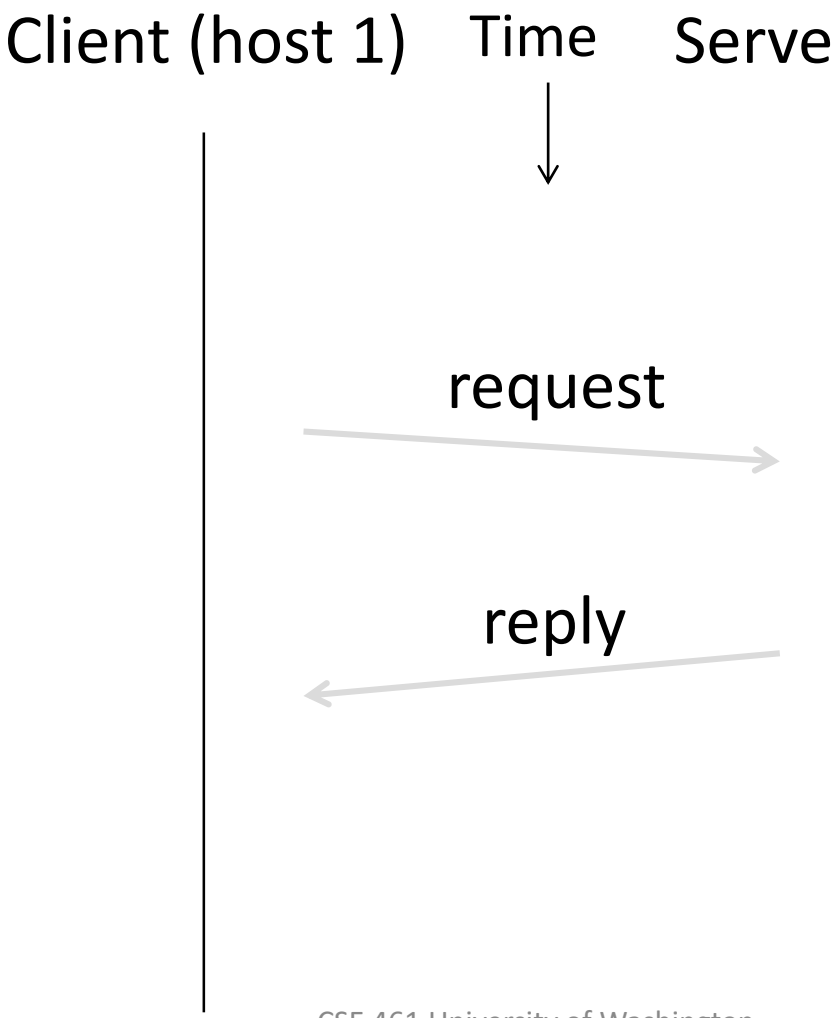
UDP

User Datagram Protocol (UDP)

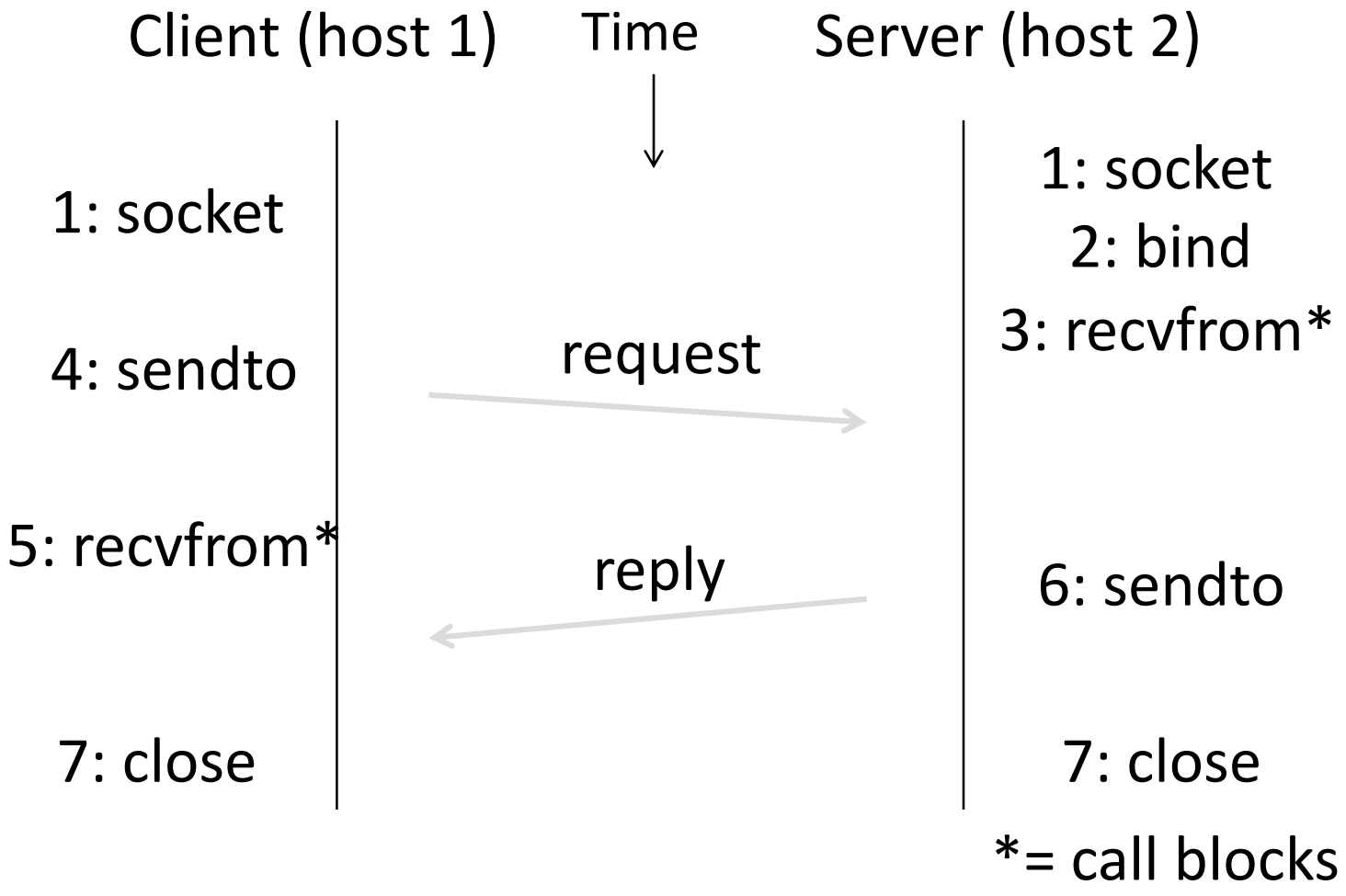
- Used by apps that don't want reliability or bytestreams
  - Voice-over-IP
  - DNS, RPC
  - DHCP

(If application wants reliability and messages then it has work to do!)

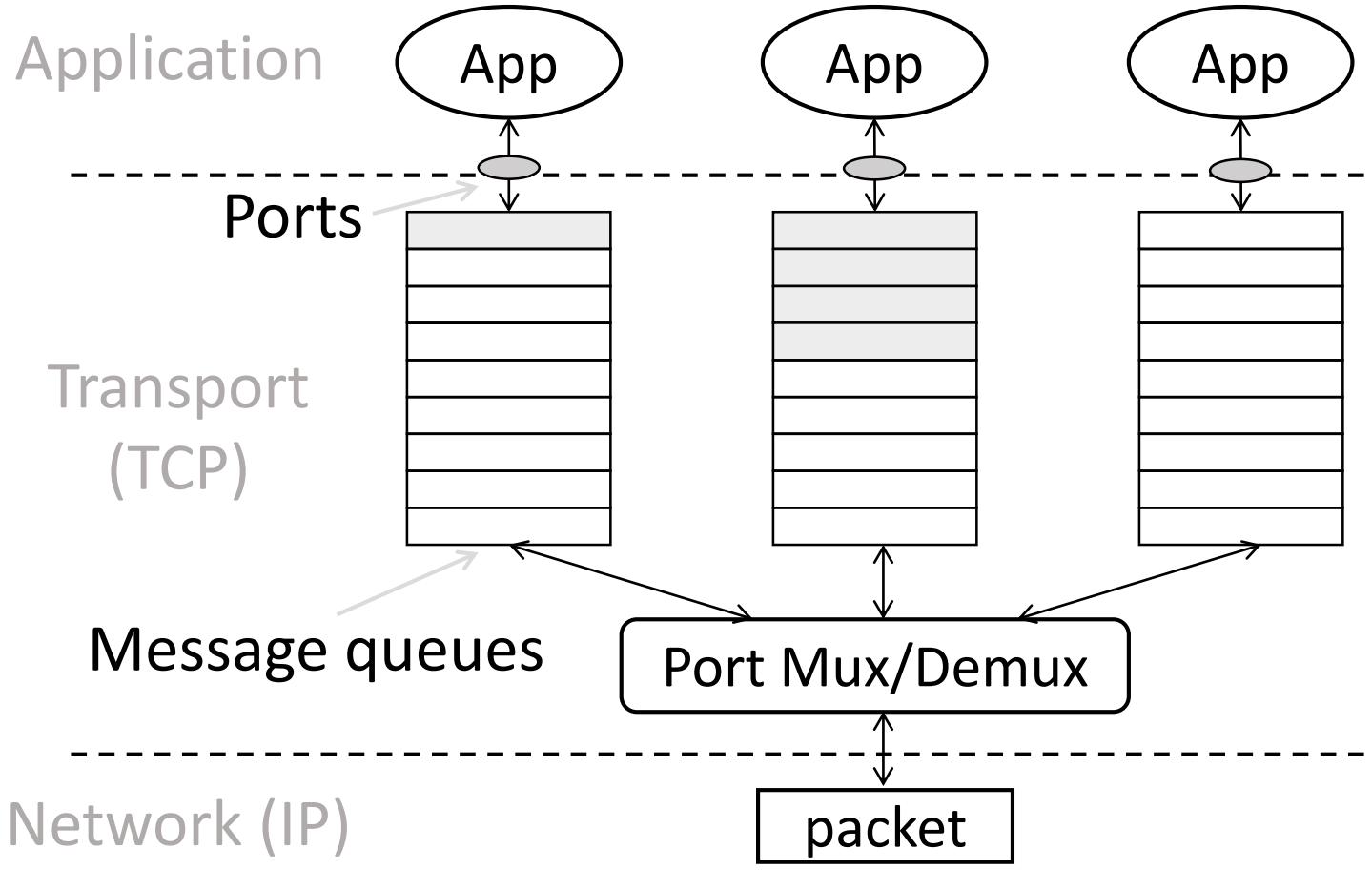
Datagram Sockets



Datagram Sockets (2)

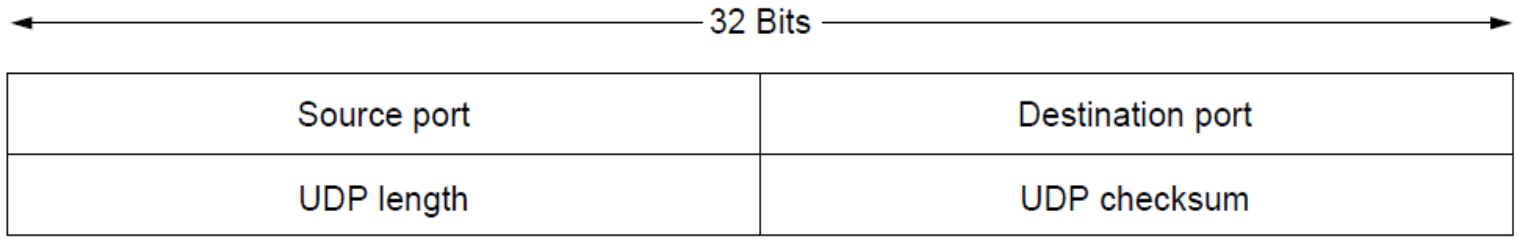


UDP Buffering



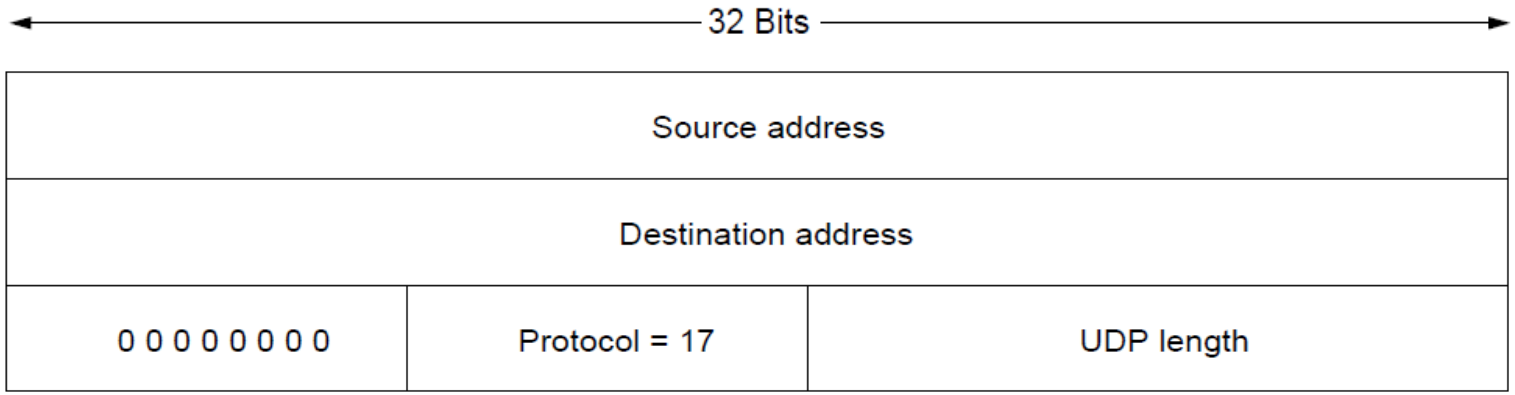
UDP Header

- Uses ports to identify sending and receiving application processes
- Datagram length up to 64K
- Checksum (16 bits) for reliability



UDP Header (2)

- Optional checksum covers UDP segment and IP pseudoheader
  - Checks key IP fields (addresses)
  - Value of zero means “no checksum”



TCP

TCP

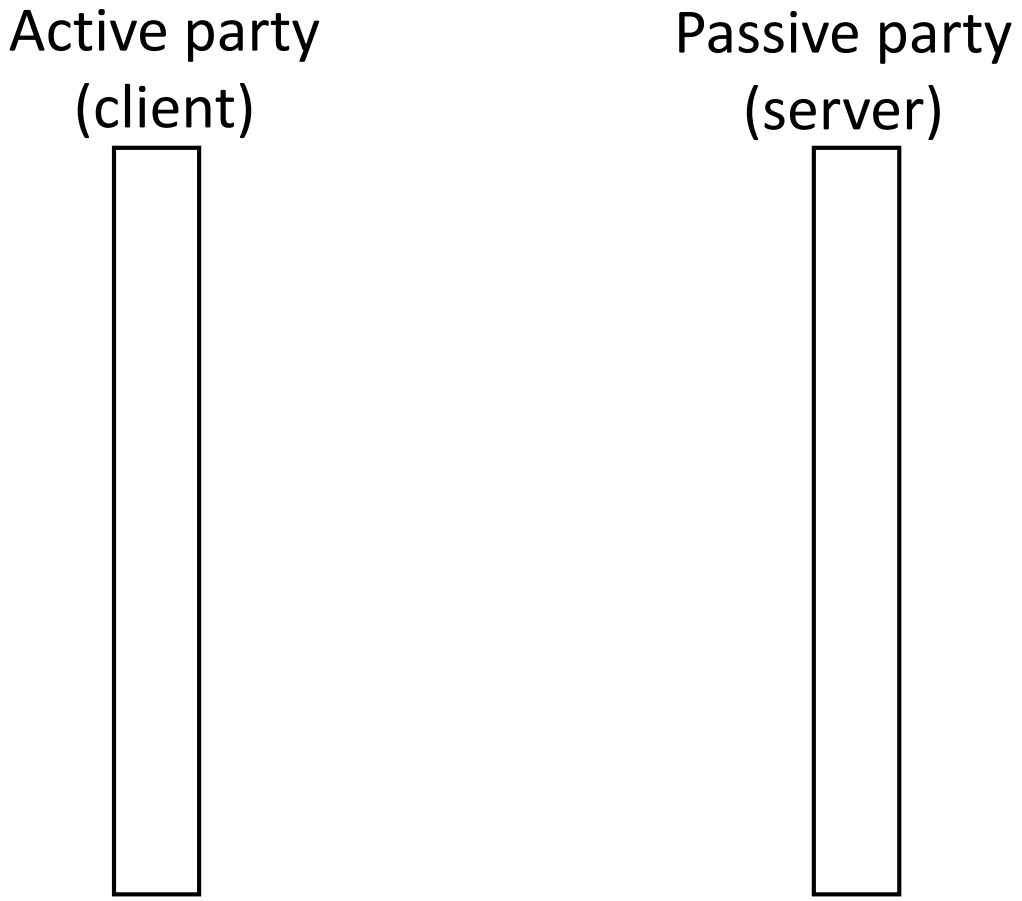
- TCP Consists of 3 primary phases:
  - Connection Establishment (Setup)
  - Sliding Windows/Flow Control
  - Connection Release (Teardown)

Connection Establishment

- Both sender and receiver must be ready before we start the transfer of data
  - Need to agree on a set of parameters
  - e.g., the Maximum Segment Size (MSS)
- This is signaling
  - It sets up state at the endpoints
  - Like “dialing” for a telephone call

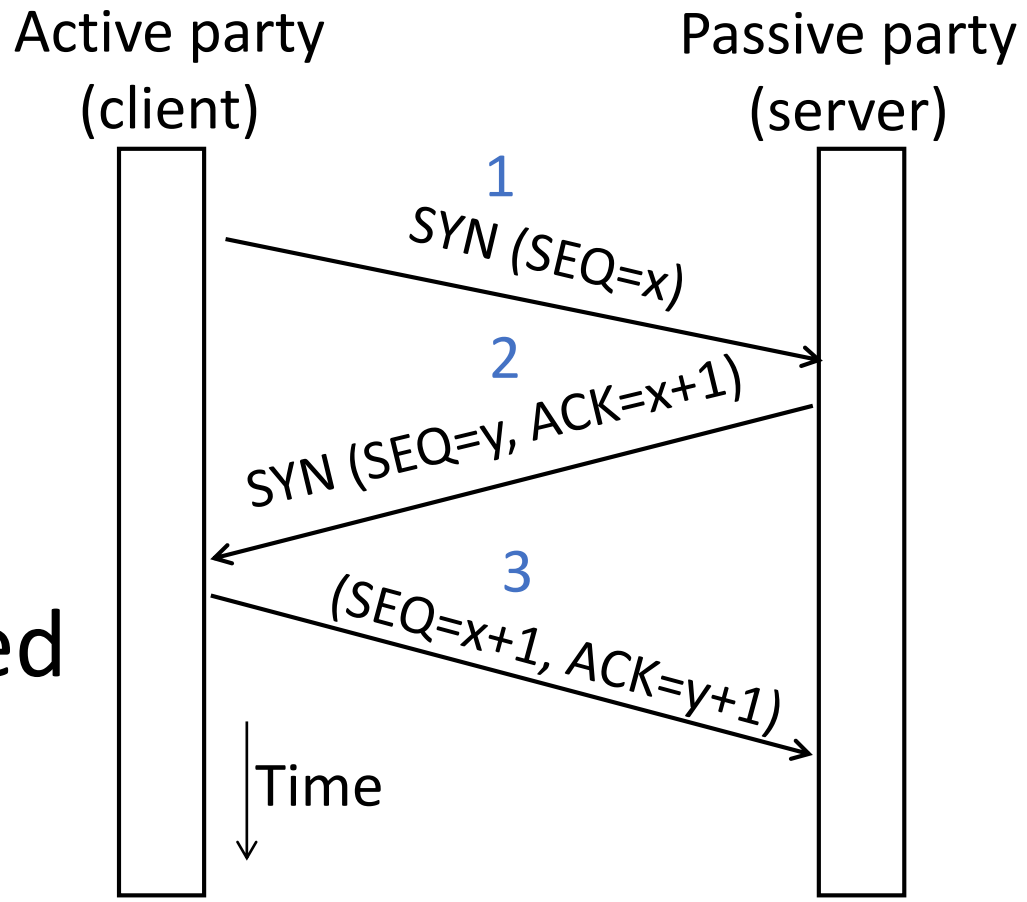
Three-Way Handshake

- Used in TCP; opens connection for data in both directions
- Each side probes the other with a fresh Initial Sequence Number (ISN)
  - Sends on a SYNchronize segment
  - Echo on an ACKnowledge segment
- Chosen to be robust even against delayed duplicates



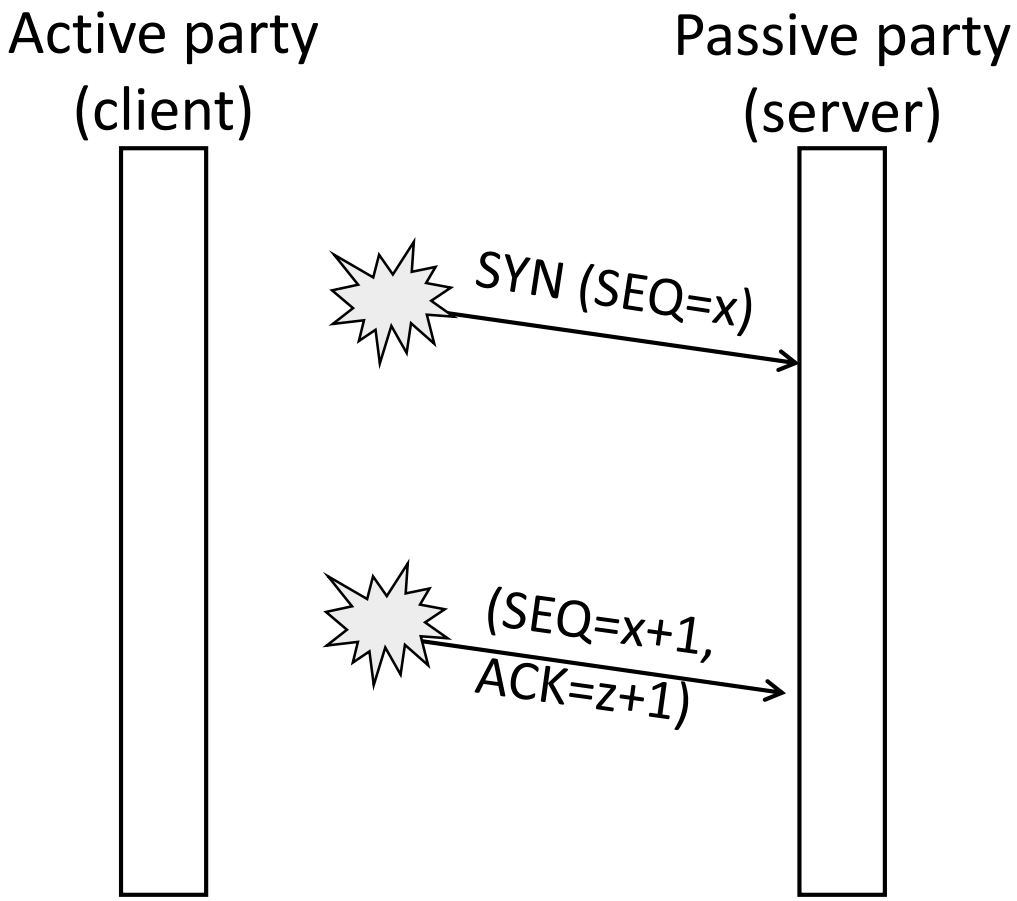
Three-Way Handshake (2)

- Three steps:
  - Client sends SYN(x)
  - Server replies with SYN(y)ACK(x+1)
  - Client replies with ACK(y+1)
  - SYNs are retransmitted if lost
- Sequence and ack numbers carried on further segments



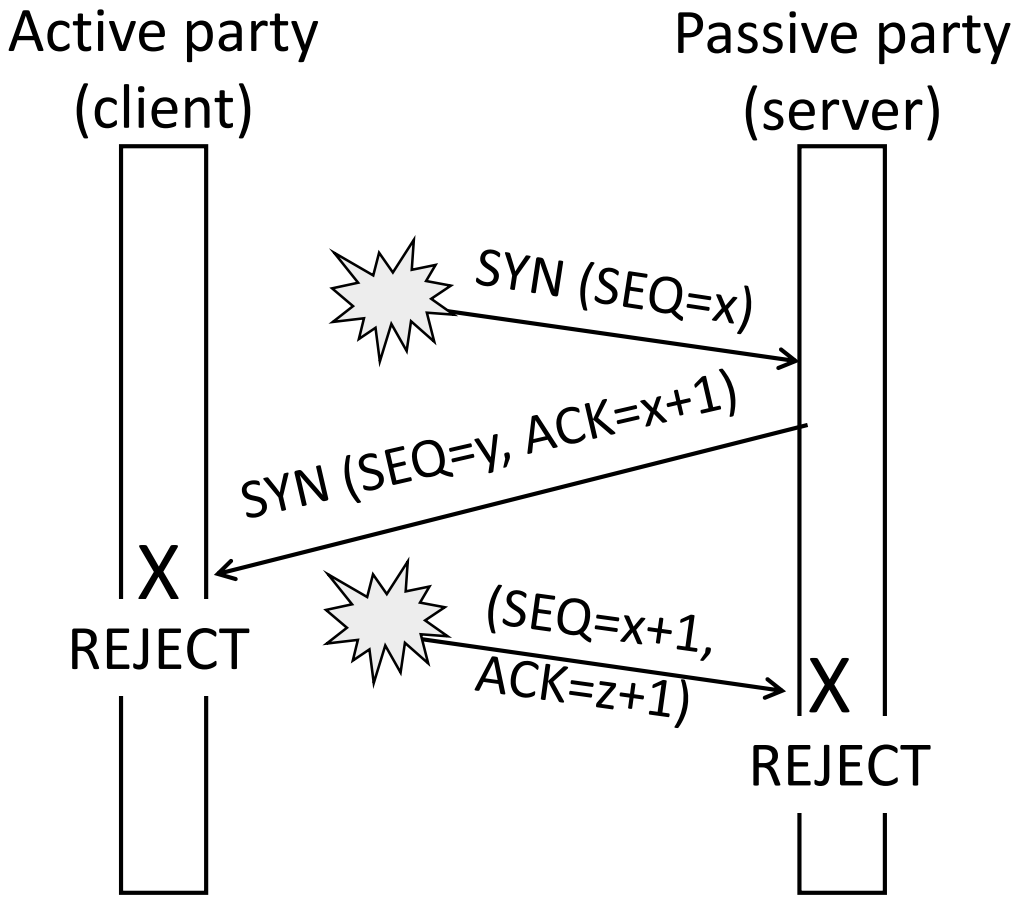
Three-Way Handshake (3)

- Suppose delayed, duplicate copies of the SYN and ACK arrive at the server!
  - Improbable, but anyhow …



Three-Way Handshake (4)

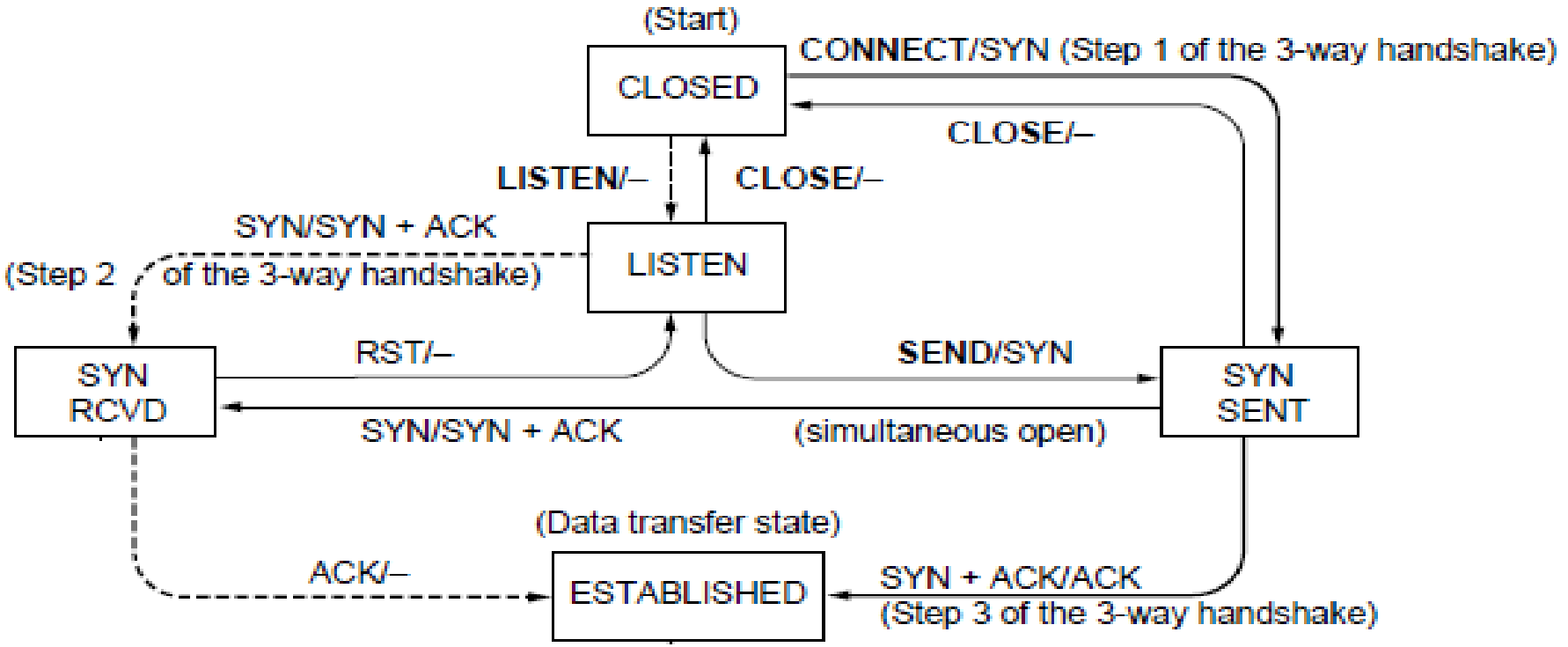
- Suppose delayed, duplicate copies of the SYN and ACK arrive at the server!
  - Improbable, but anyhow …
- Connection will be cleanly rejected on both sides ☺



TCP Connection State Machine

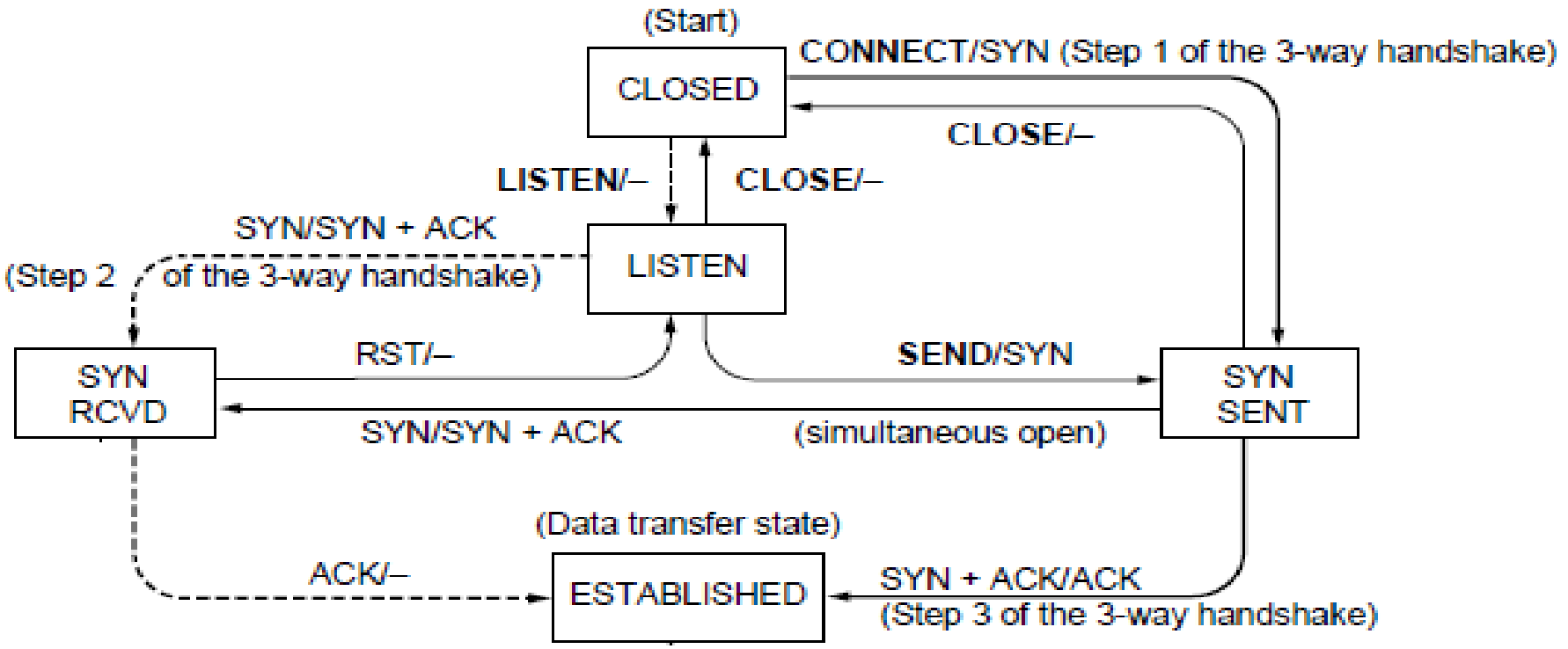
- Captures the states ([]) and transitions (->)
  - A/B means event A triggers the transition, with action B

Both parties run instances of this state machine



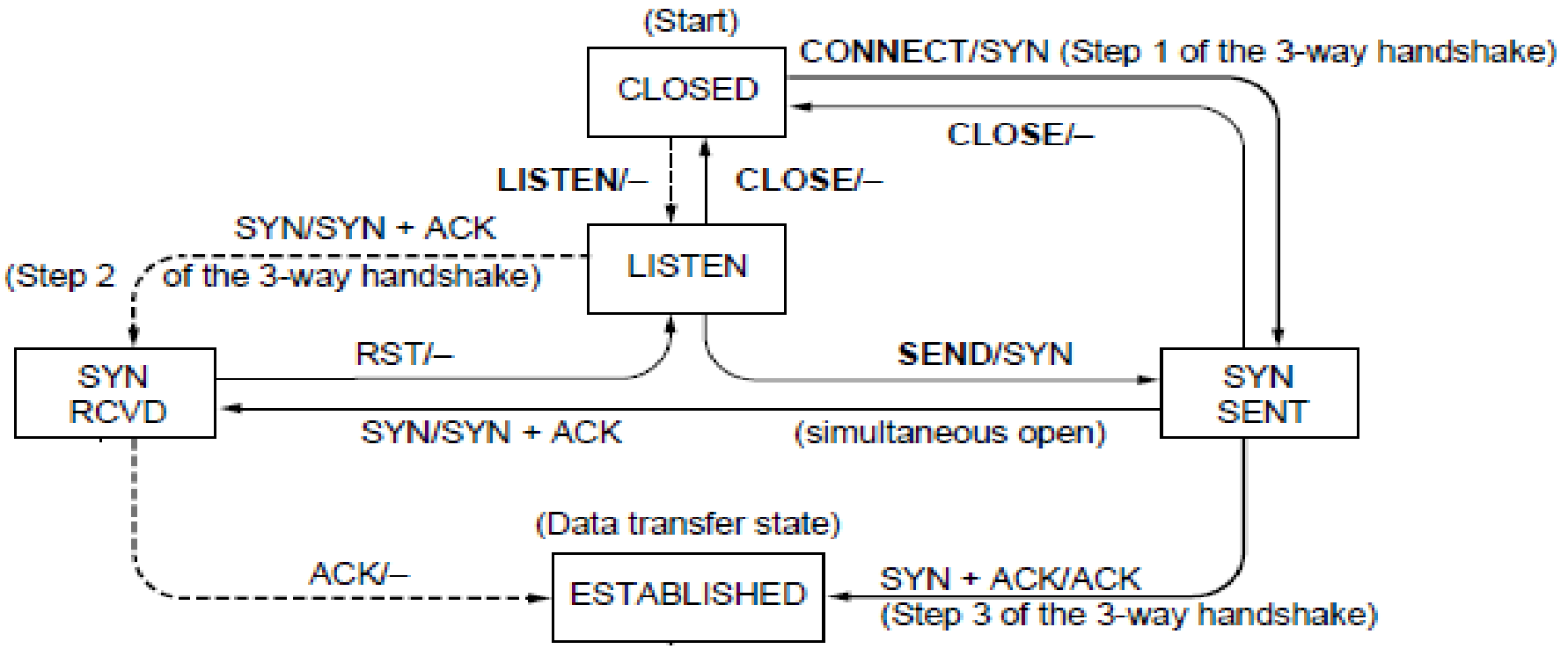
TCP Connections (2)

- Follow the path of the client:



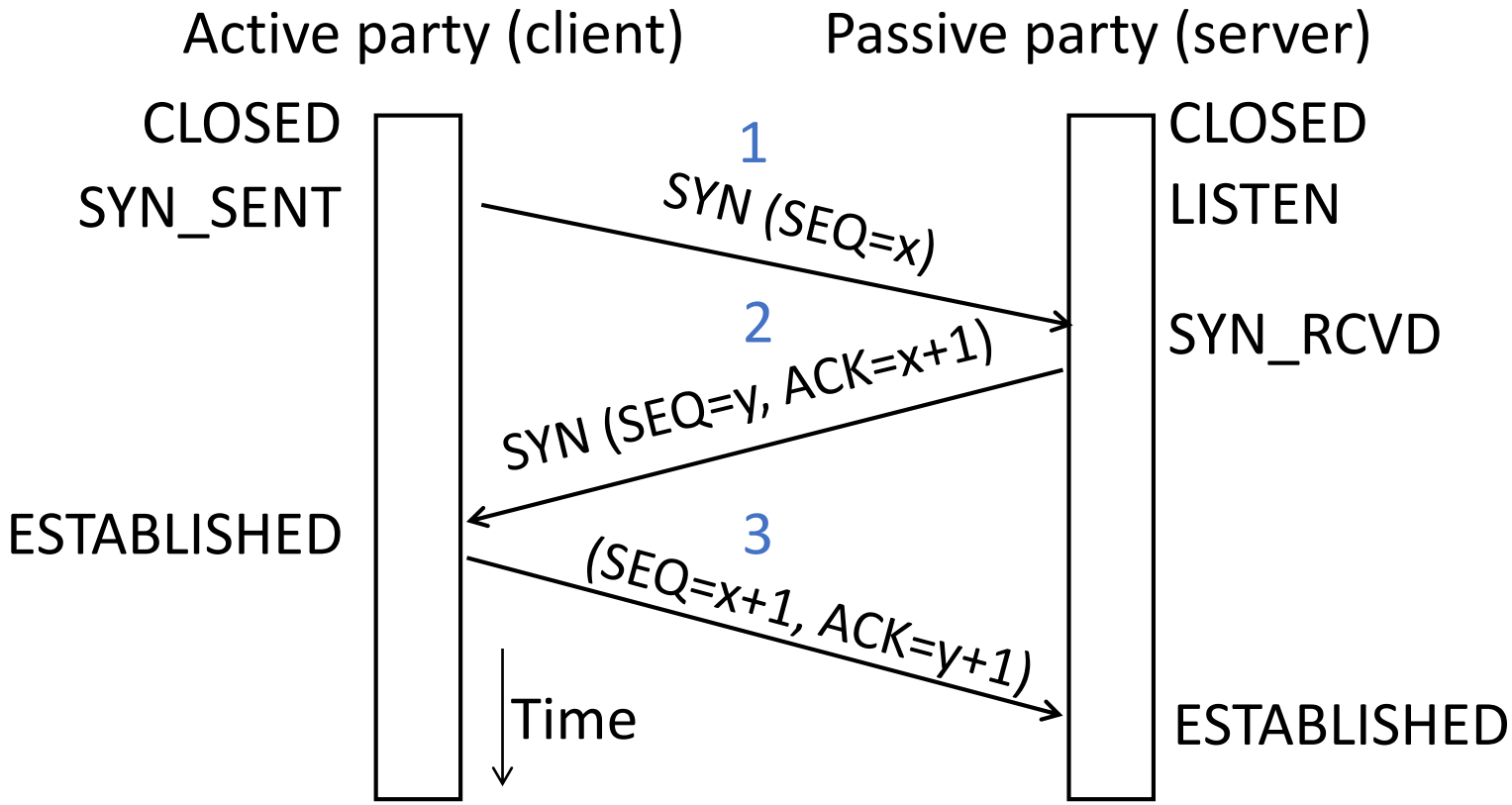
TCP Connections (3)

- And the path of the server:



TCP Connections (4)

- Again, with states …



TCP Connections (5)

- Finite state machines are a useful tool to specify and check the handling of all cases that may occur
- TCP allows for simultaneous open
  - i.e., both sides open instead of the client-server pattern
  - Try at home to confirm it works ☺