CSCI-1680
Transport Layer I

Nick DeMarinis

Based partly on lecture notes by Rodrigo Fonseca, David Mazières, Phil Levis, John Jannotti

Administrivia

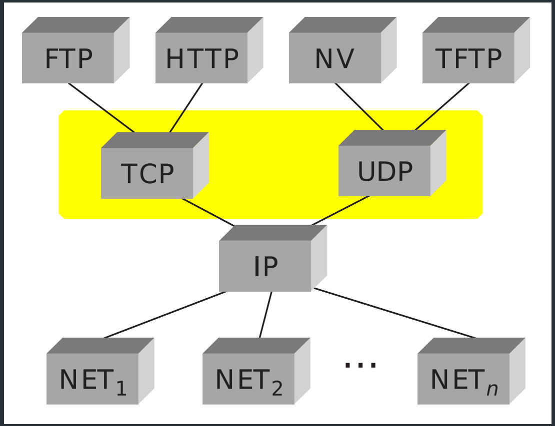
- Later today: Look for message about IP grading
  - Meeting slots first week after break (and during break)
- TCP: Draft of assignment out today
  - Read it over before break, start when we get back
- Summer/UTA hiring: Expect a message from me today/ tomorrow

Today

Light overview of the transport layer and TCP

- Why we need TCP
- What components are involved
- What you will do in the project

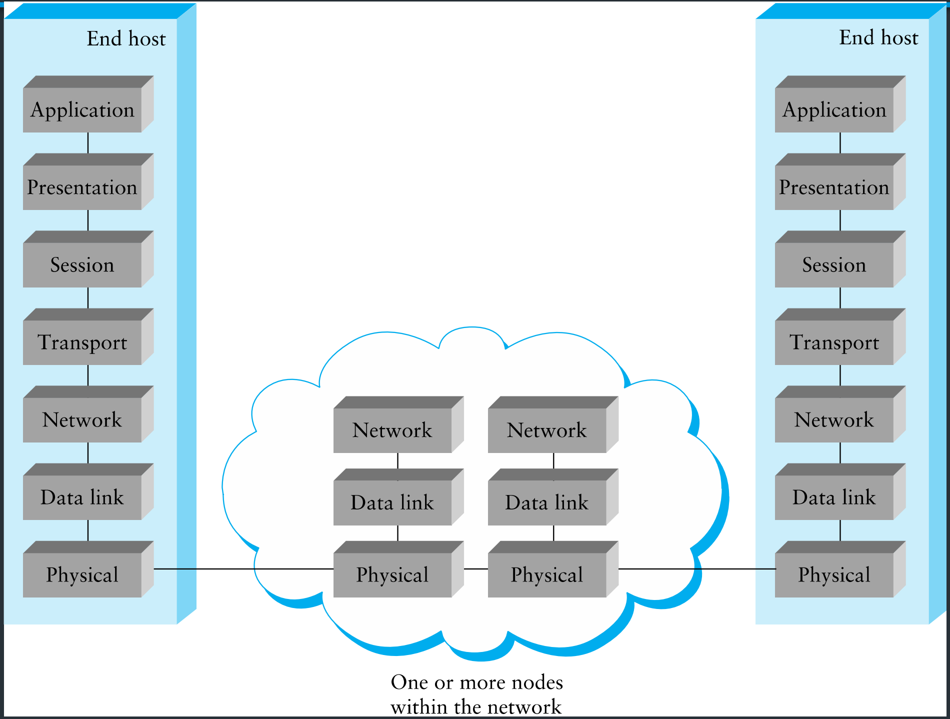
Transport Layer



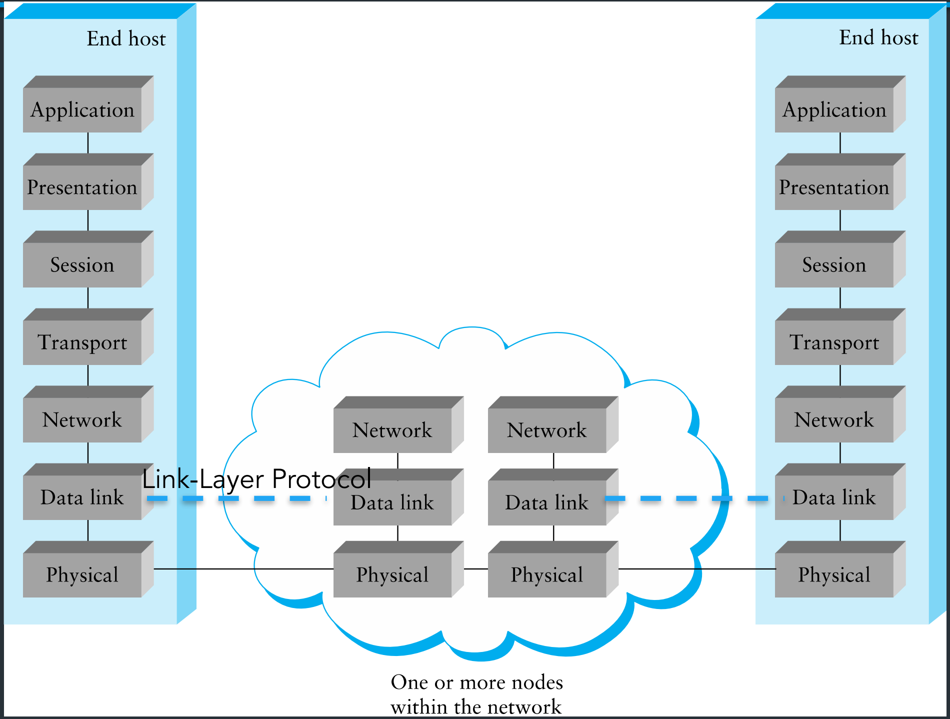
SCTP
QUIC

- Transport protocols sit on top of network layer
- Problem solved: communication among processes
  - Application-level multiplexing ("ports")
  - Error detection, reliability, etc.

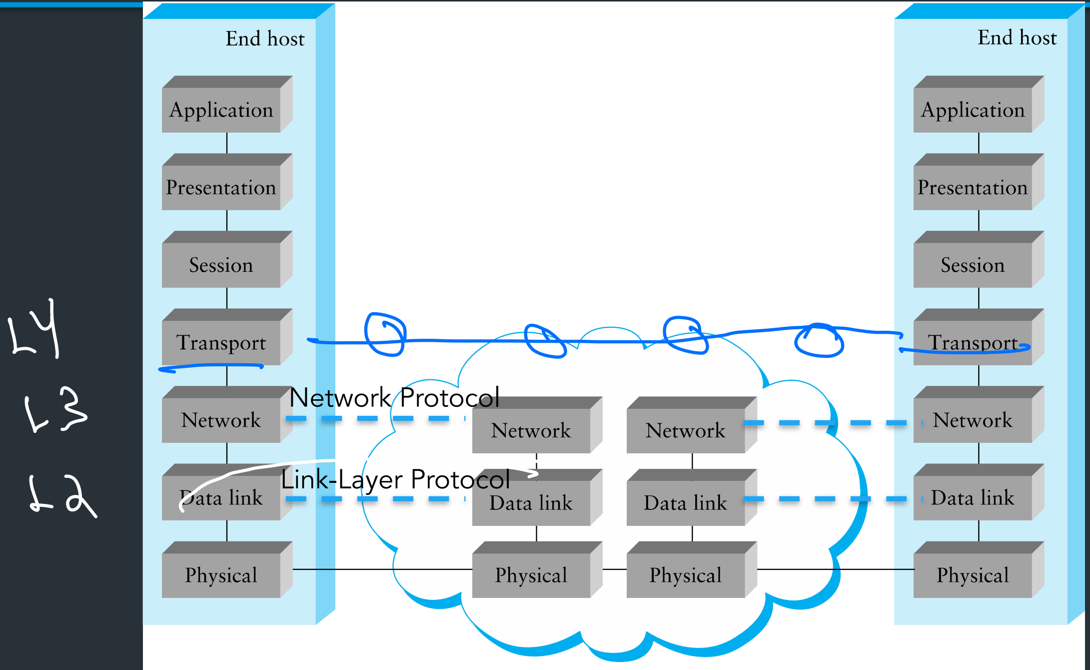
From Lec 2: OSI Reference Model



From Lec 2: OSI Reference Model

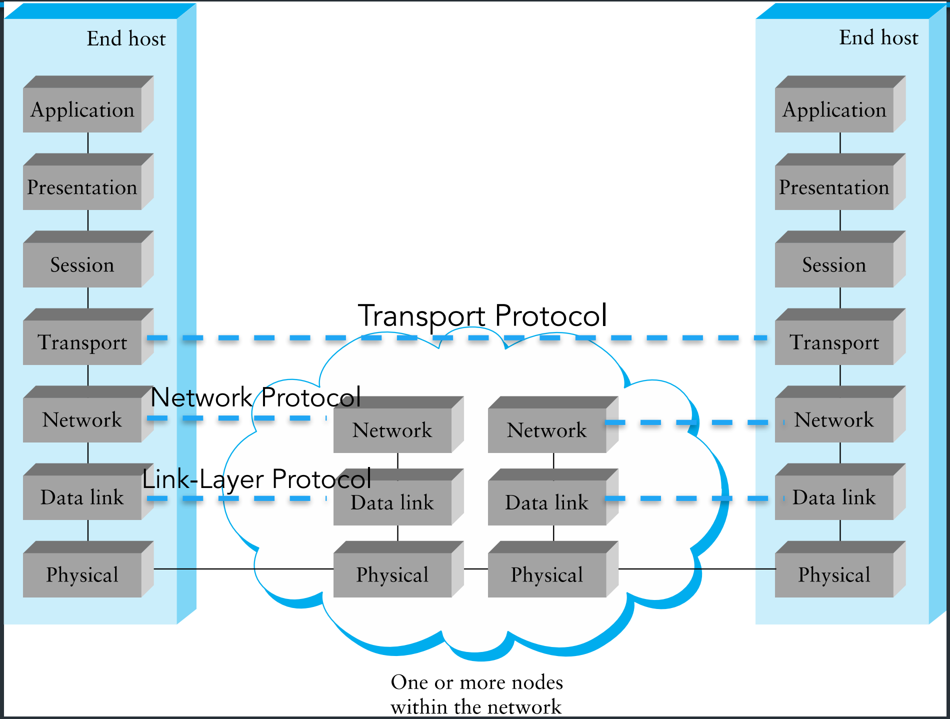


From Lec 2: OSI Reference Model



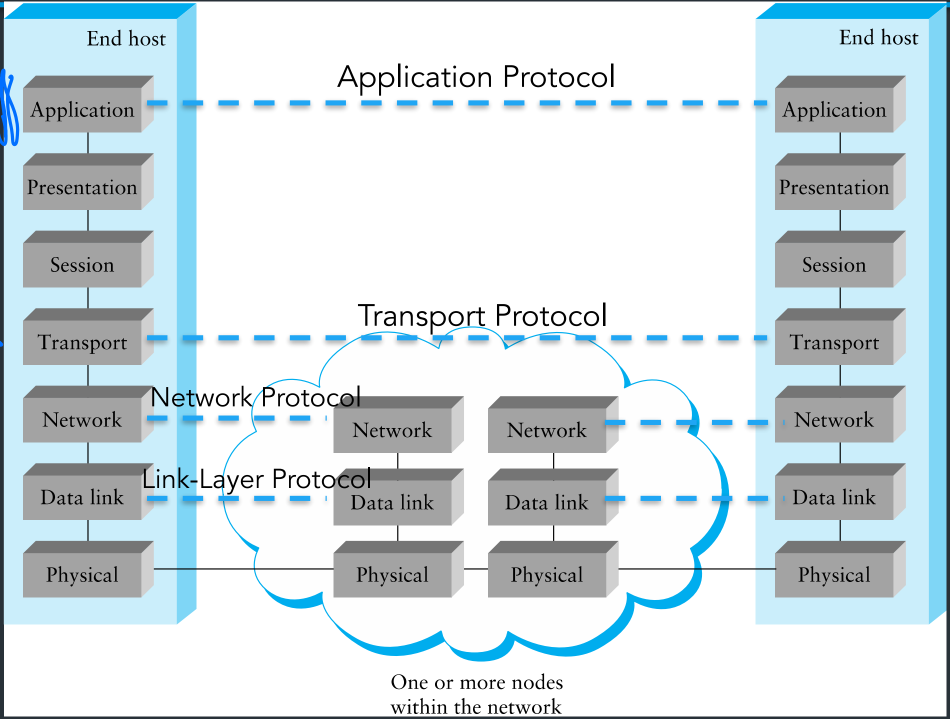
One or more nodes
within the network

From Lec 2: OSI Reference Model

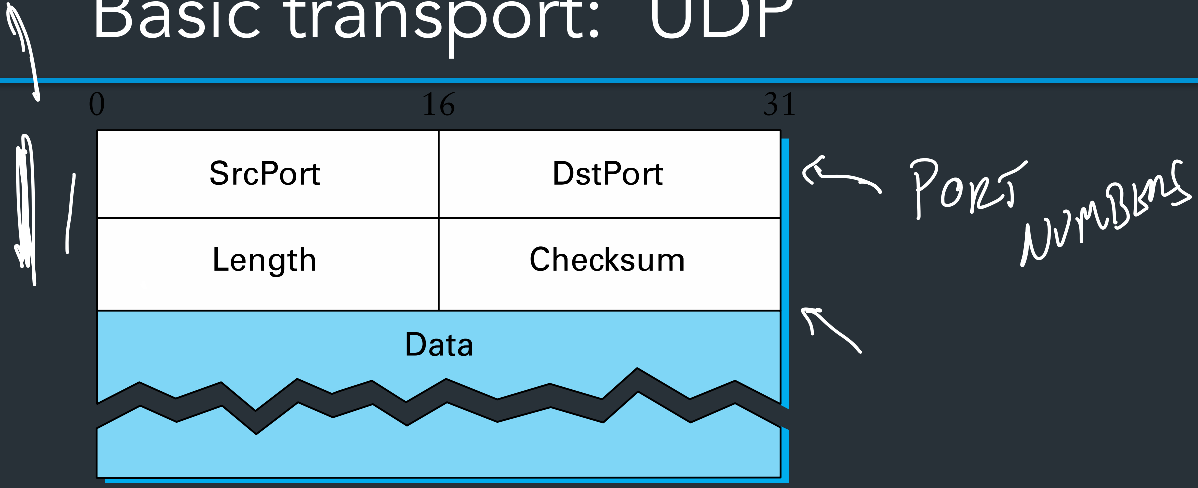


From Lec 2: OSI Reference Model

SOCKET



Basic transport: UDP



User Datagram Protocol

- Unreliable datagram service
- Adds multiplexing (via ports) and nothing else
- Checksum is pretty useless

Next Problem: Reliability

We talked briefly about link-layer reliability:

Problem	Mechanism
Dropped Packets	Acknowledgments + Timeout
Duplicate Packets	Sequence Numbers
Packets out of order	Receiver Window
Maximizing throughput	Sliding Window (Pipelining)

Next Problem: Reliability

We talked briefly about link-layer reliability:

Problem	Mechanism
Dropped Packets	Acknowledgments + Timeout
Duplicate Packets	Sequence Numbers
Packets out of order	Receiver Window
Maximizing throughput	Sliding Window (Pipelining)

- **Single link: things were easy… ☺**

Transport Layer Reliability

- Extra difficulties
  - Multiple hosts
  - Multiple hops
  - Multiple potential paths

Transport Layer Reliability

- Extra difficulties
  - Multiple hosts
  - Multiple hops
  - Multiple potential paths
- What does this mean?
  - Multiple opportunities for failure
  - Hosts have different resources
  - Varying RTTs

– DIFFERENT LINK TYPES/SPEEDS

– DIFFERENT HOST CAPABILITIES

Extra Difficulties (cont.)

- Out of order packets
  - Not only because of drops/retransmissions
  - Can get very old packets (up to 120s), must not get confused

Extra Difficulties (cont.)

- Out of order packets
  - Not only because of drops/retransmissions
  - Can get very old packets (up to 120s), must not get confused
- Unknown resources at other end
  - Must be able to discover receiver buffer: flow control

TELL SENDER TO
STOP/LIMIT DATA

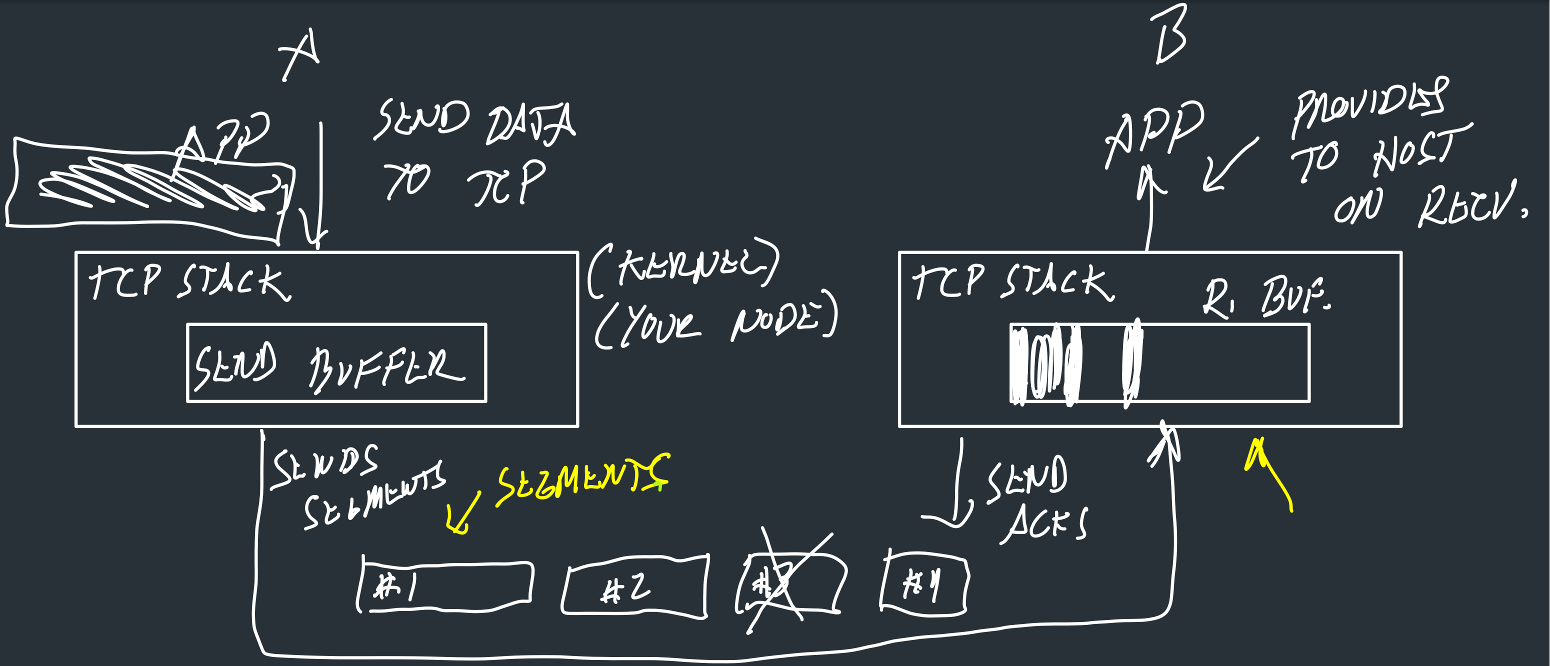
Extra Difficulties (cont.)

A — B

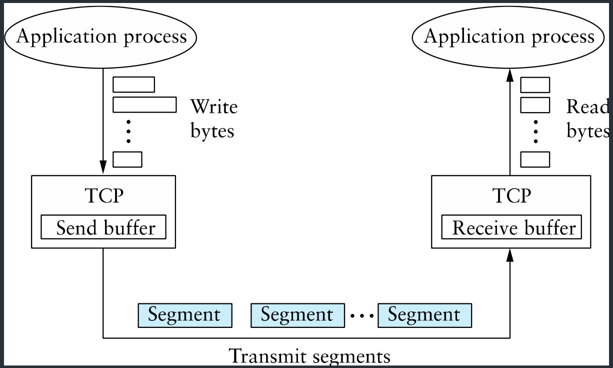
- Out of order packets
  - Not only because of drops/retransmissions
  - Can get very old packets (up to 120s), must not get confused
- Unknown resources at other end
  - Must be able to discover receiver buffer: flow control
- Unknown resources in the network
  - Should not overload the network
  - But should use as much as safely possible to maximize throughput

"FAIRNESS" IN USE OF SCARCE BW

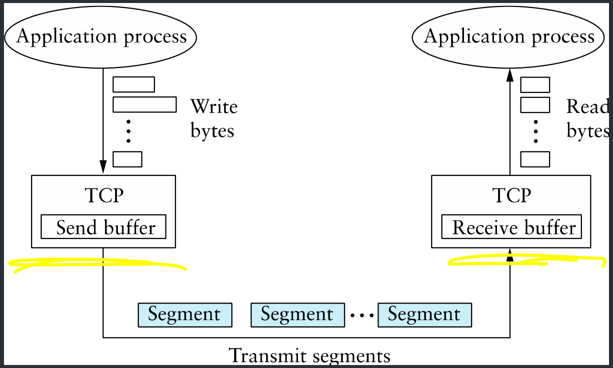
TCP – Transmission Control Protocol



TCP – Transmission Control Protocol

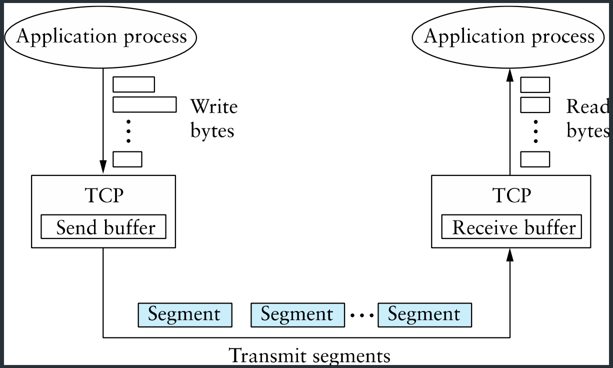


TCP – Transmission Control Protocol



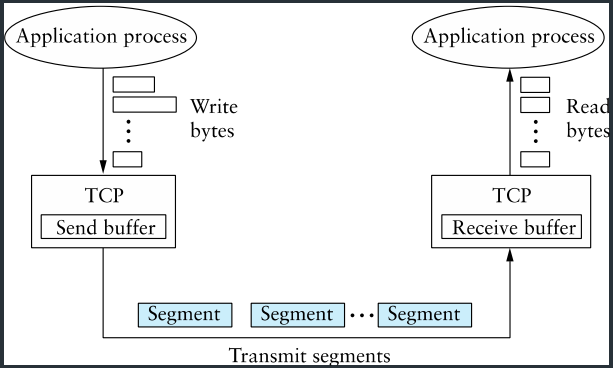
- Service model: "reliable, connection oriented, full duplex ordered byte stream"

TCP – Transmission Control Protocol



- Service model: "reliable, connection oriented, full duplex ordered byte stream"
- Flow control: If one end stops reading, writes at other eventually stop/fail

TCP – Transmission Control Protocol



- Service model: "reliable, connection oriented, full duplex ordered byte stream"
- Flow control: If one end stops reading, writes at other eventually stop/fail
- Congestion control: Keeps sender from overloading the network

TCP

- Specification
  - RFC 793 (1981), RFC 1222 (1989, some corrections), RFC 5681 (2009, congestion control), …
- Was born coupled with IP, later factored out
- End-to-end protocol
  - Minimal assumptions on the network
  - All mechanisms run on the end points
- What if you had link-layer reliability instead?

HOPS IN THE MIDDLE JUST FORWARD PACKETS – NOT AWARE OF TCP.

Why not provide X on the network layer?

X = Reliability, security, message ordering…

- Cost
  - These functionalities are not free: don’t burden those who don’t need them

Why not provide X on the network layer?

X = Reliability, security, message ordering…

- Cost
  - These functionalities are not free: don't burden those who don't need them
- Conflicting
  - Timeliness and in-order delivery, for example

Why not provide X on the network layer?

X = Reliability, security, message ordering…

- Cost
  - These functionalities are not free: don't burden those who don't need them
- Conflicting
  - Timeliness and in-order delivery, for example
- Insufficient
  - Example: reliability

End-to-end argument

- Functions placed at lower levels of a system may be redundant or of little value
  - They may **need** to be performed at a higher layer anyway
- But they may be justified for performance reasons → WIFI

End-to-end argument

- Functions placed at lower levels of a system may be redundant or of little value
  - They may **need** to be performed at a higher layer anyway
- But they may be justified for performance reasons
  - Or just because they provide most of what is needed
  - Example: retransmissions
- Takeaway: weigh the costs and benefits at each layer

TCP Header

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Source Port          |       Destination Port        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Sequence Number                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Acknowledgment Number                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Data |           |U|A|P|R|S|F|                               |
| Offset| Reserved  |R|C|S|S|Y|I|            Window             |
|       |           |G|K|H|T|N|N|                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Checksum            |         Urgent Pointer        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             data                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

WHERE THIS PACKET IS IN STREAM

LAST SEQUENCE RECEIVED

HOW MUCH SPACE THE RECEIVER HAS FOR DATA

FLAGS

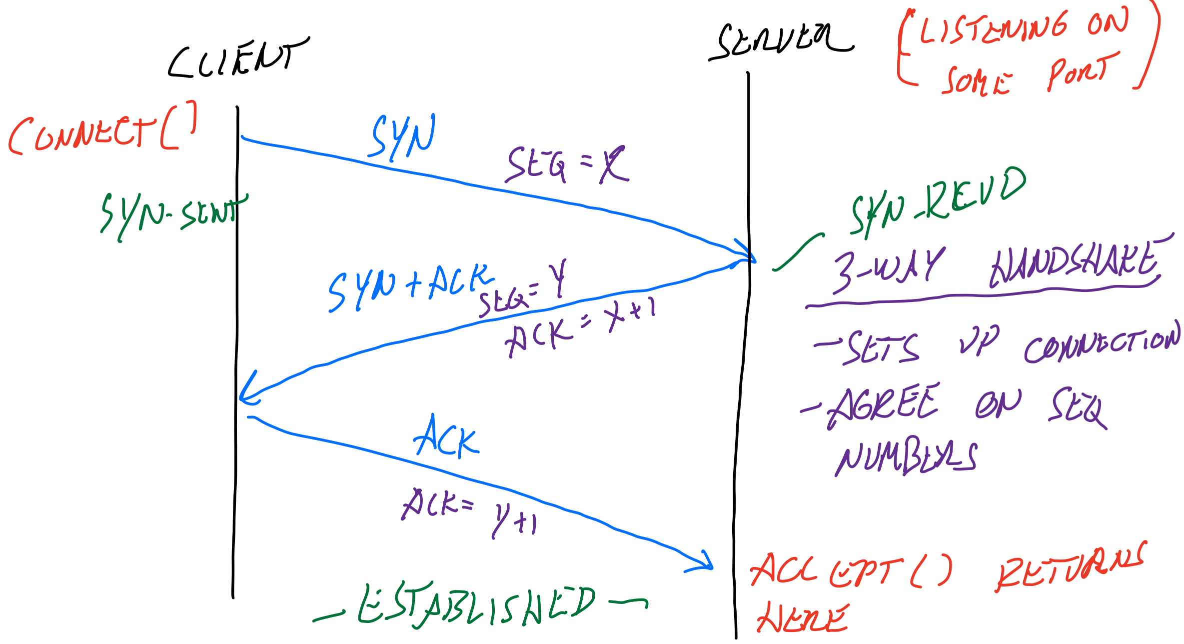
Header Fields

- Ports: multiplexing
- Sequence number
  - Correspond to bytes, not packets!
- Acknowledgment Number
  - Next expected sequence number
- Window: willing to receive
  - Lets receiver limit SWS (even to 0) for flow control
- Data Offset: # of 4 byte (header + option bytes)
- Flags, Checksum, Urgent Pointer

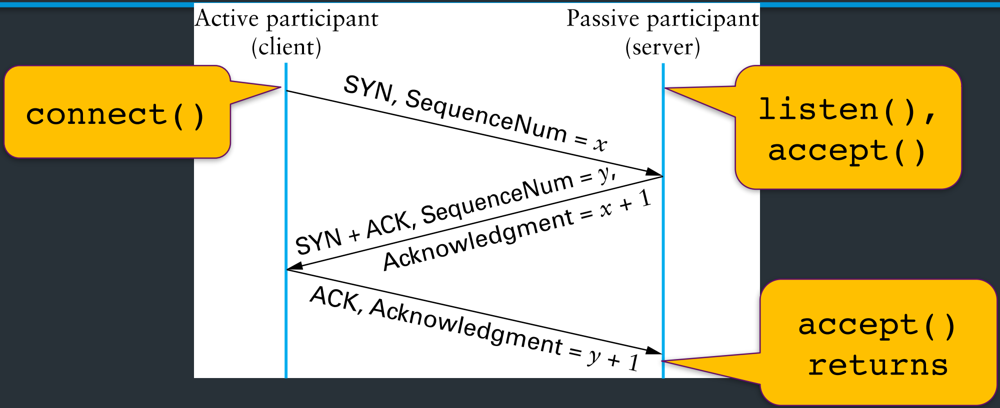
Header Flags

- URG: whether there is urgent data
- ACK: ack no. valid (all but first segment)
- PSH: push data to the application immediately
- RST: reset connection
- SYN: synchronize, establishes connection
- FIN: close connection

ESTABLISHING A TCP CONNECTION



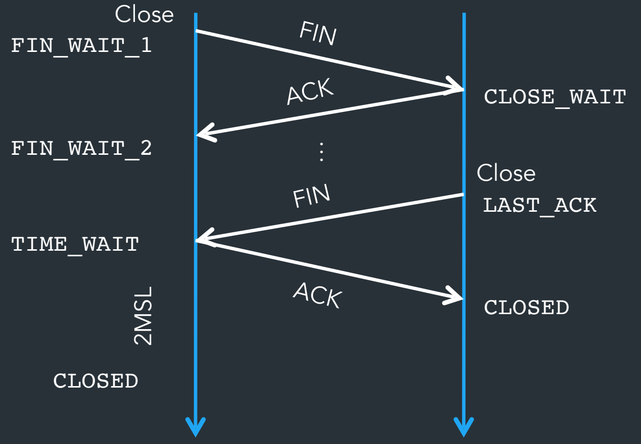
Establishing a Connection



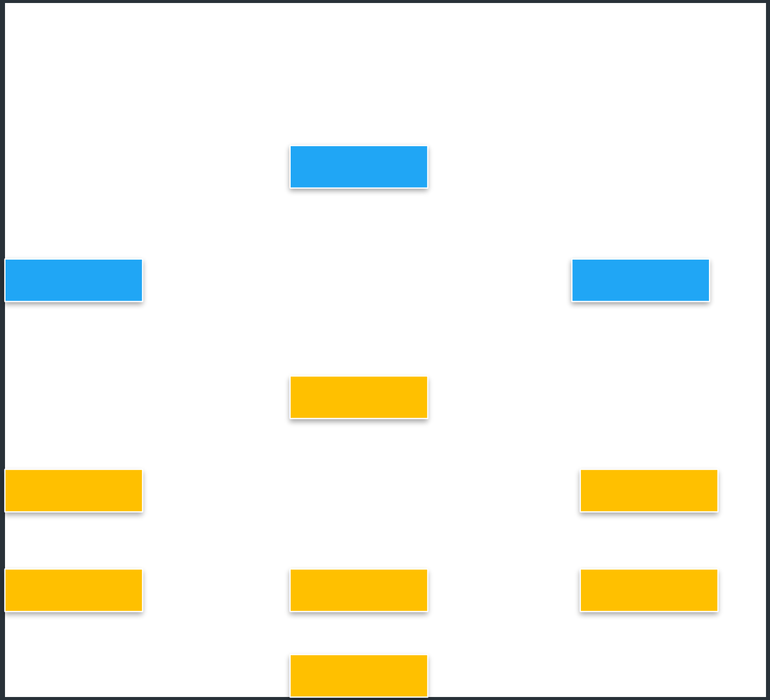
- Three-way handshake
  - Two sides agree on respective initial sequence nums
- If no one is listening on port: server sends RST
- If server is overloaded: ignore SYN
- If no SYN-ACK: retry, timeout

Connection Termination

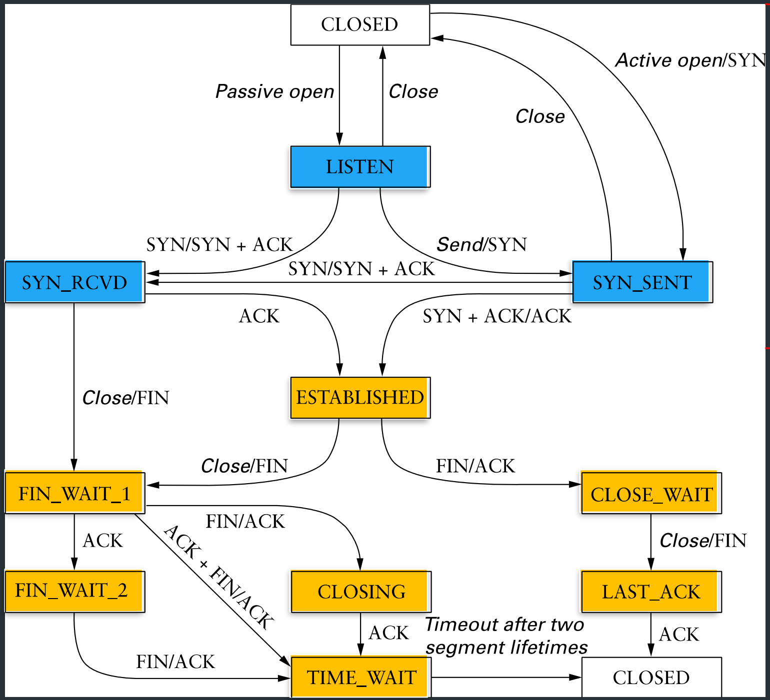
- FIN bit says no more data to send
  - Caused by close or shutdown
  - Both sides must send FIN to close a connection
- Typical close



Summary of TCP States

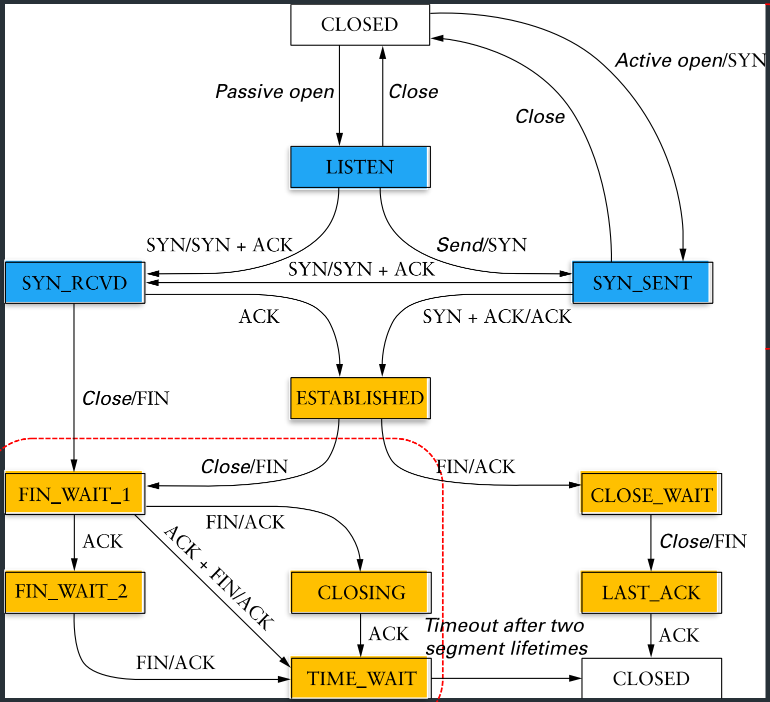


Summary of TCP States



Connection Establishment

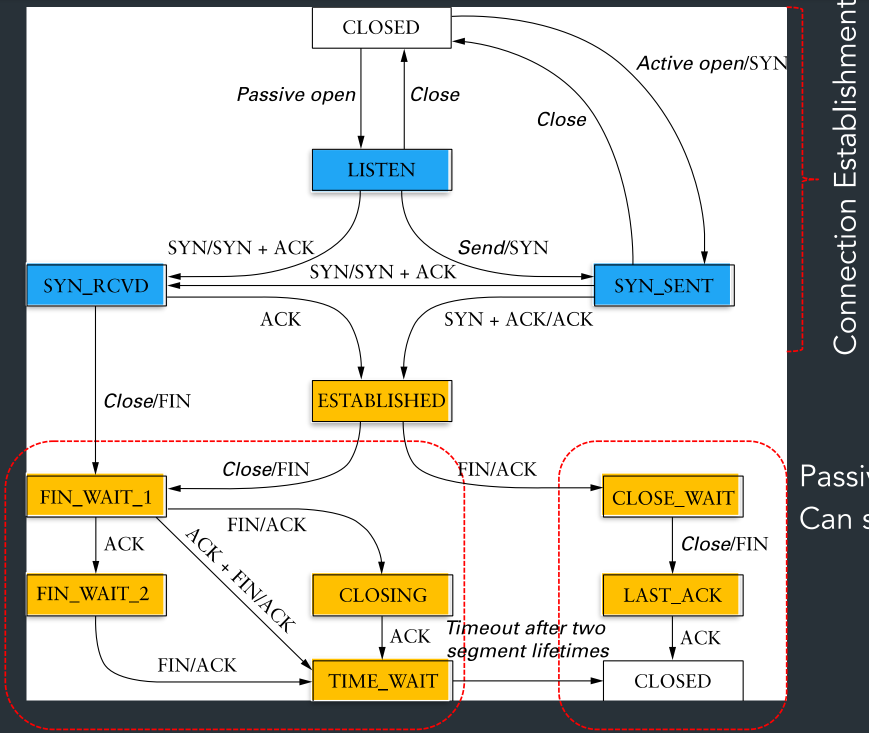
Summary of TCP States



Connection Establishment

Active close:
Can still receive

Summary of TCP States



Active close:
Can still receive

Passive close:
Can still send!

Next class

- Sending data over TCP