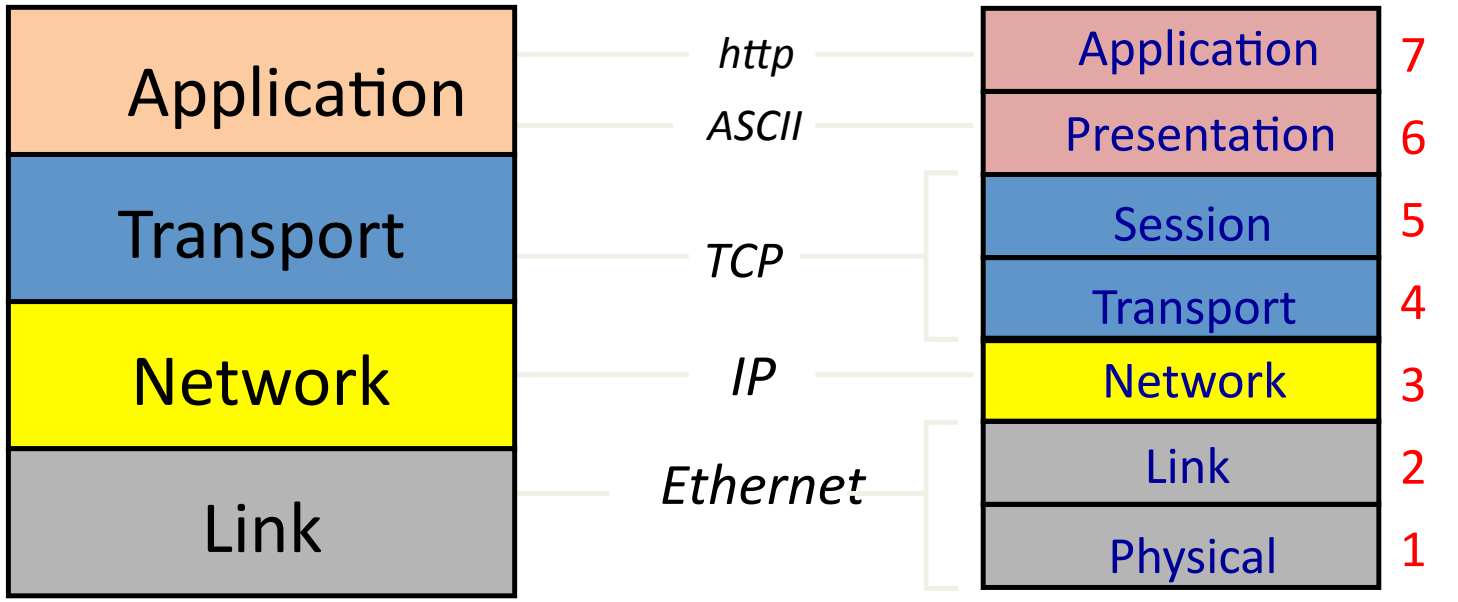
CS144
An Introduction to Computer Networks

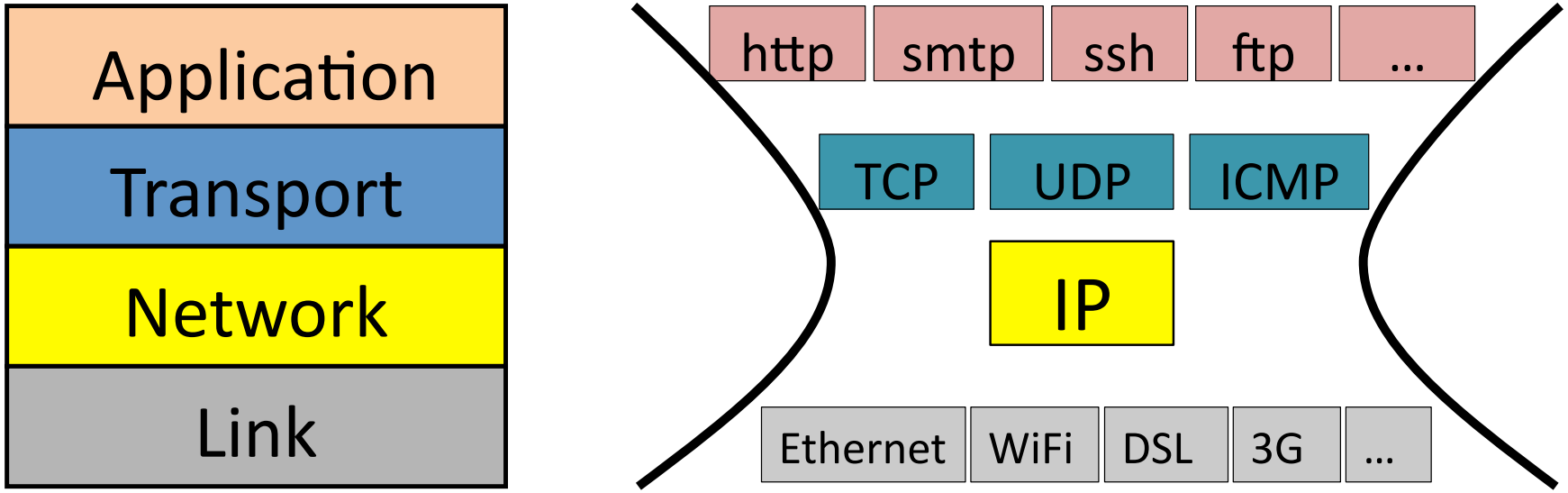
Unit 2: Transport

The 7-layer OSI Model

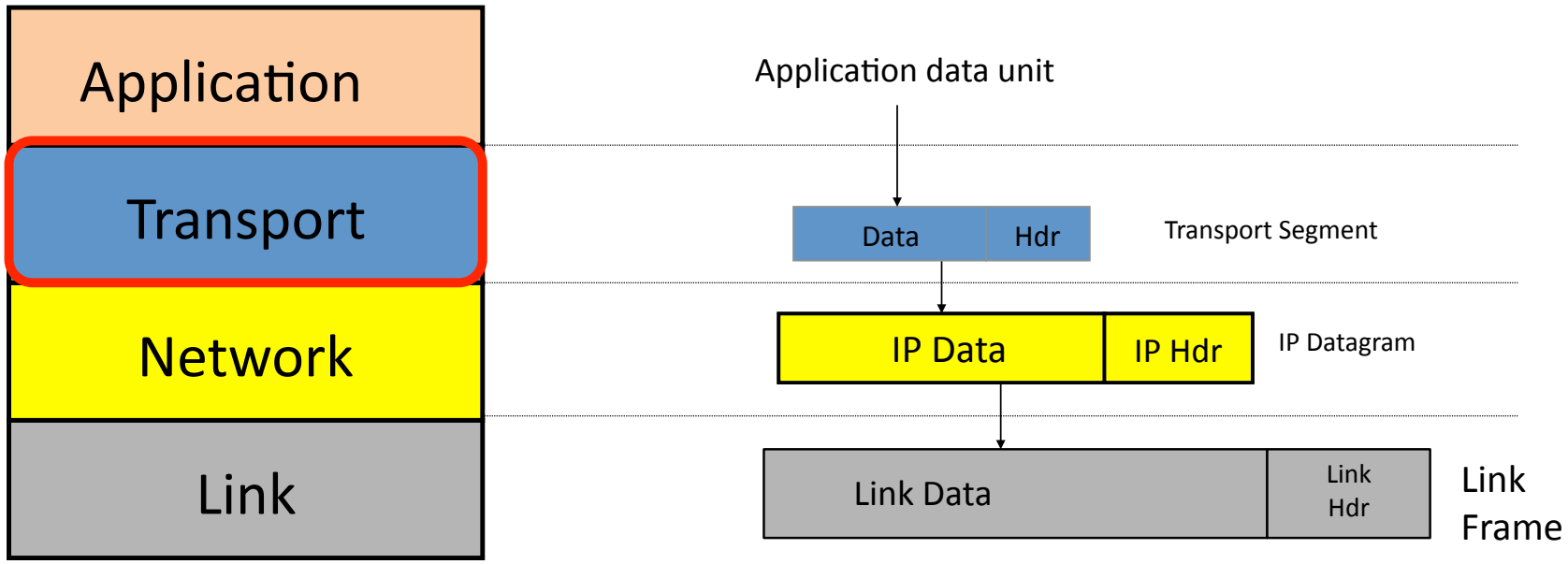


The 7-layer OSI Model

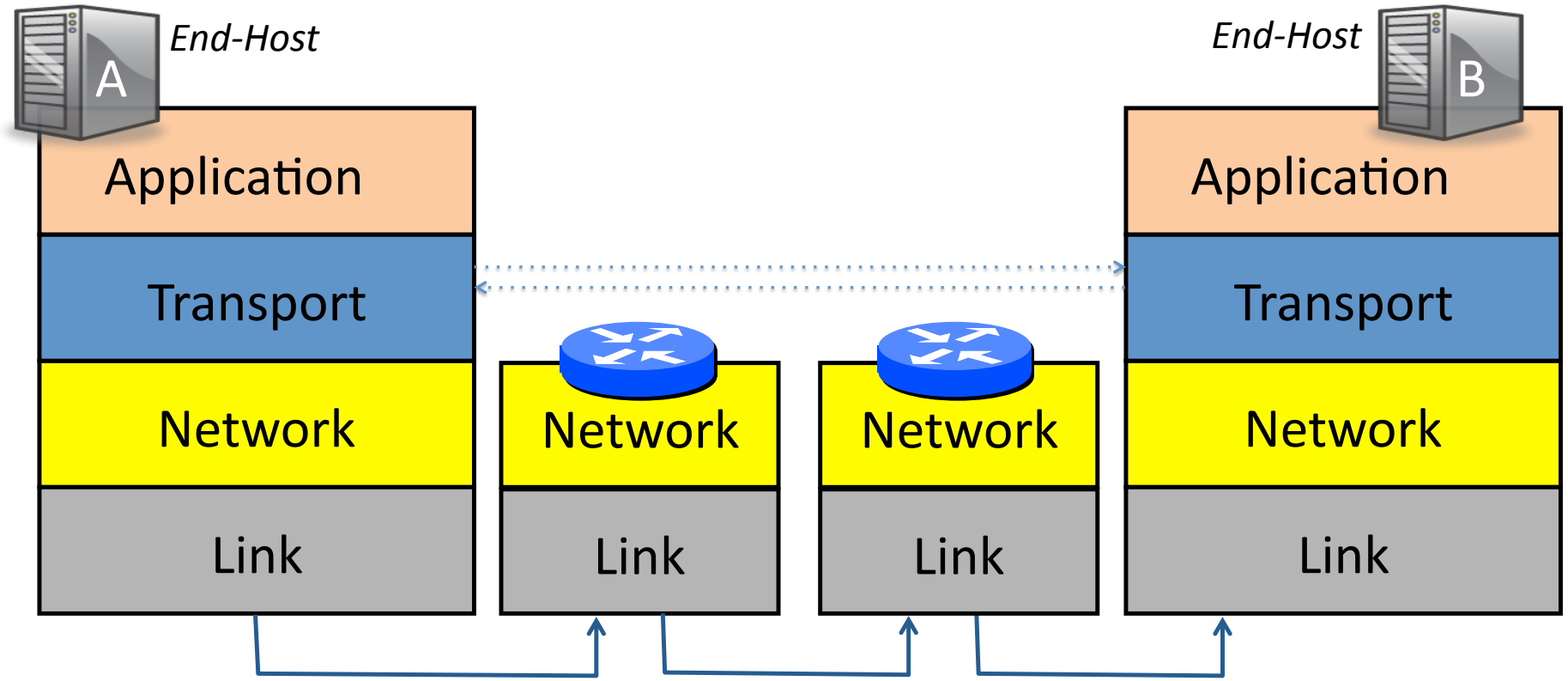
IP is the “thin waist”



Transport Protocols



Peer transport layers communicate

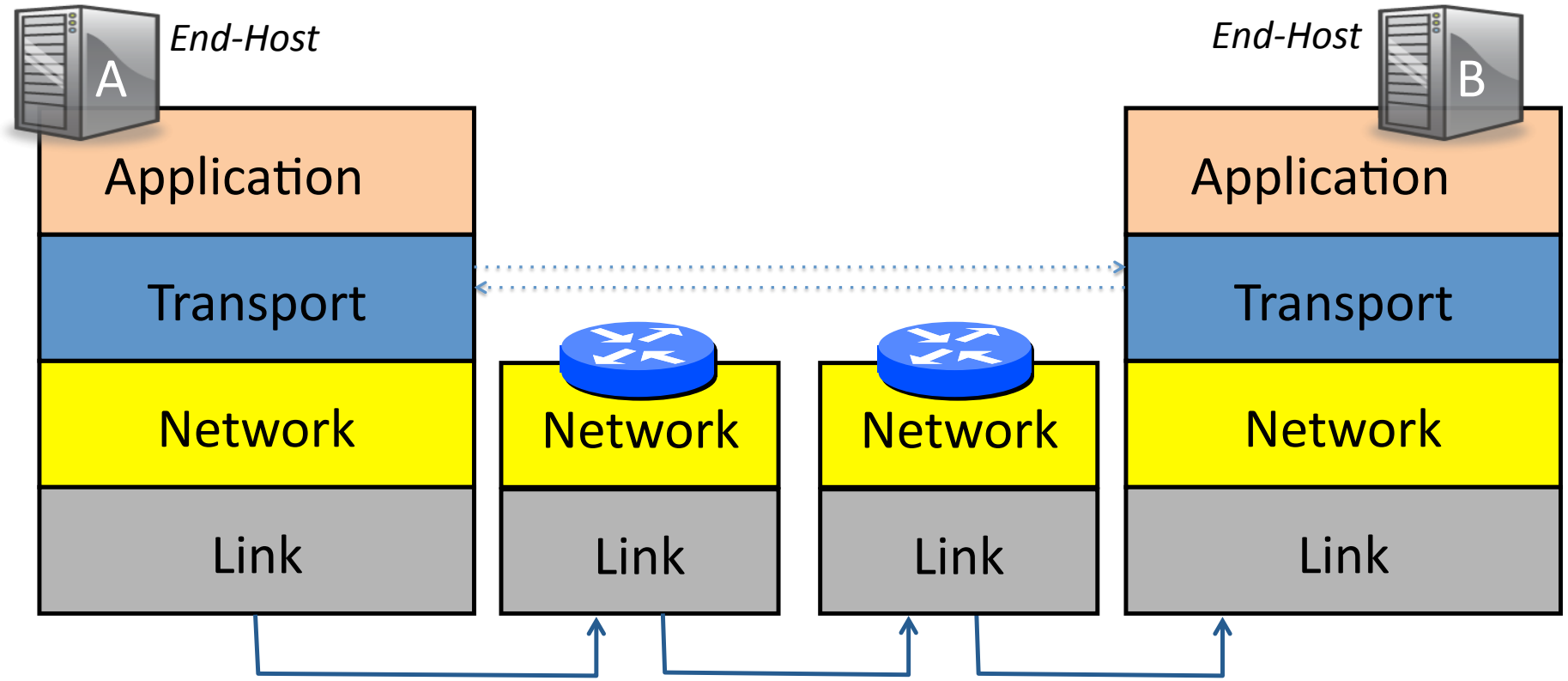


The End-To-End Principle

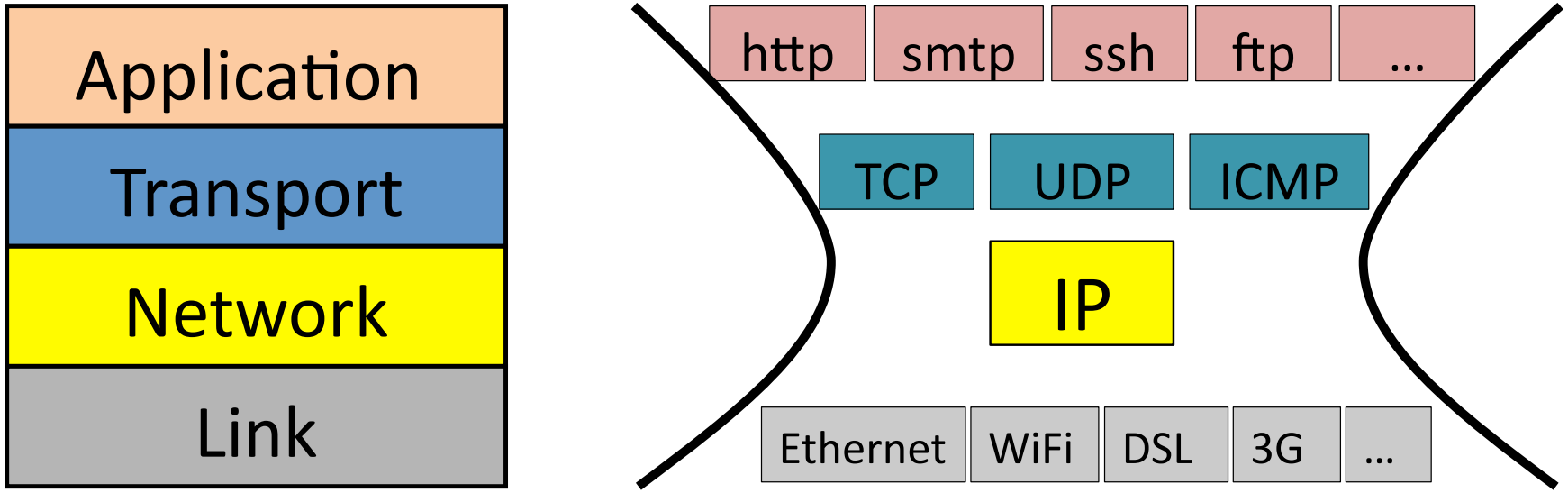
The function in question can completely and correctly be implemented only with the knowledge and help of the application standing at the end points of the communication system. Therefore, providing that questioned function as a feature of the communication system itself is not possible. (Sometimes an incomplete version of the function provided by the communication system may be useful as a performance enhancement.) We call this line of reasoning…"the end-to-end argument."

- Saltzer, Reed, and Clark,
End-to-end Arguments in System Design, 1984

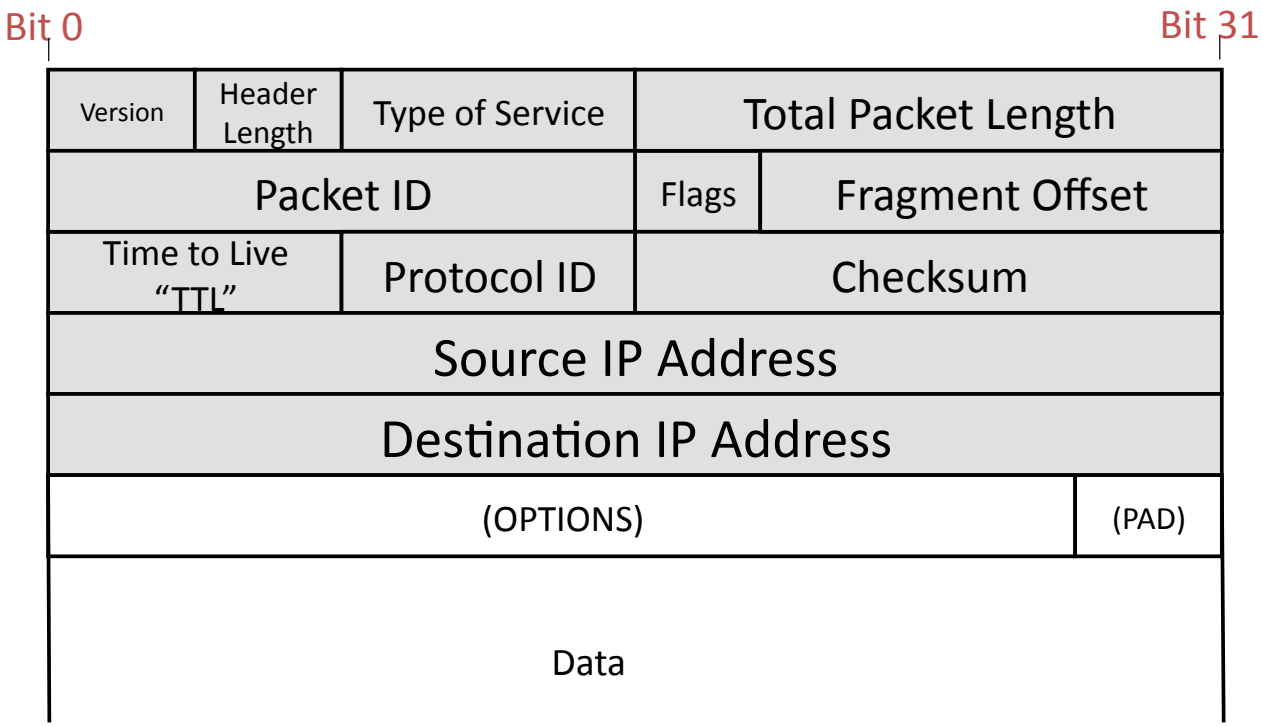
Peer transport layers communicate



IP is the “thin waist”



IPv4 Datagram



Vision vs. Reality

- Original Internet design embraced idea of many transport protocols
  - TCP, UDP, SCTP, STCP, RTP, DCCP…
- Today we have 3
  - App to app: TCP, UDP
  - Kernel to kernel: ICMP
- You'll learn why in Unit 5 (NATs)

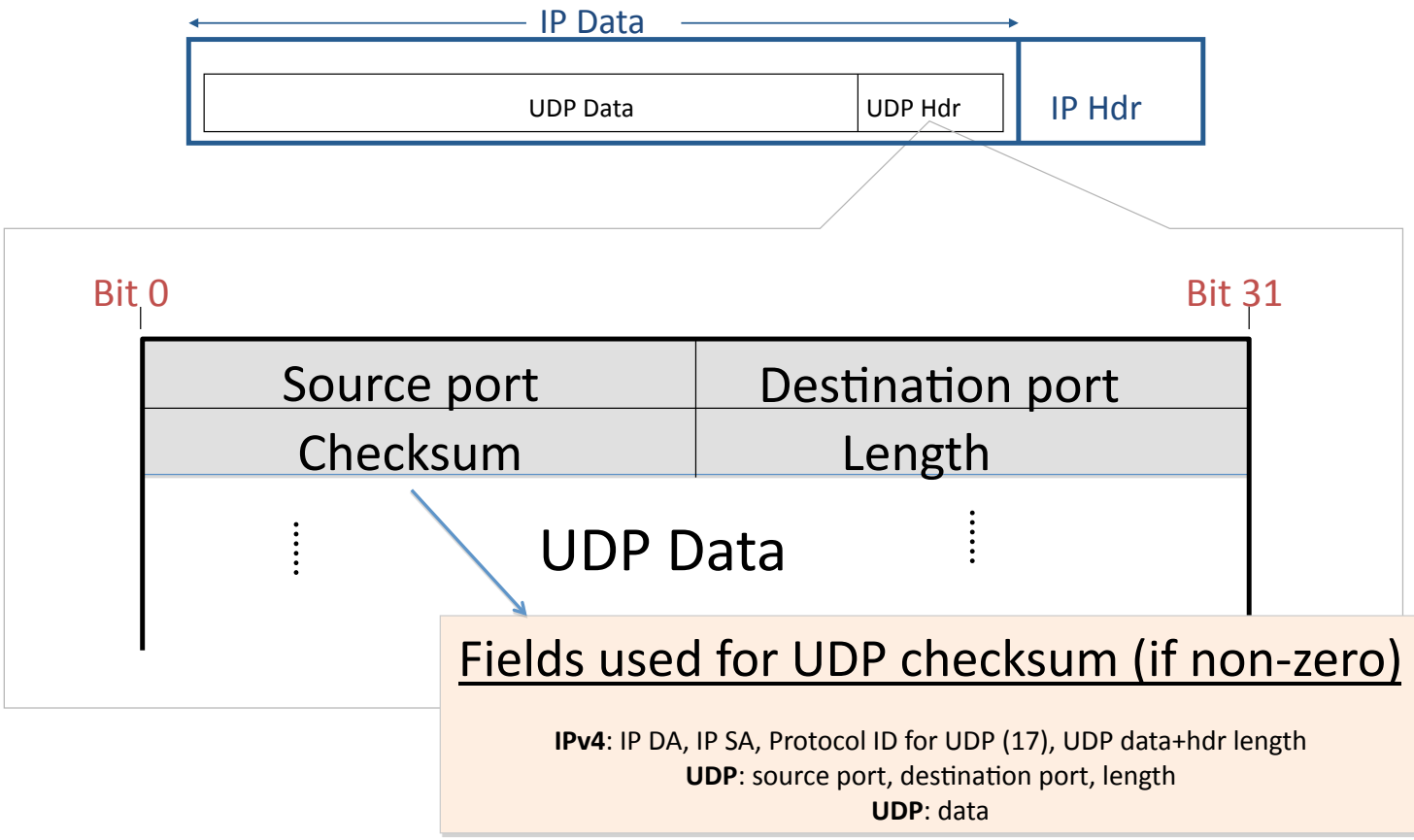
Rest of Lecture

- UDP
- TCP
- Stop and wait
- Sliding window
- Distributed systems: think about edge cases
  - Sequence number spaces
  - Connection establishment/teardown

User Datagram Protocol (UDP)

Property	Behavior
Connectionless Datagram Service	No connection established. Packets may show up in any order.
Self contained datagrams	
Unreliable delivery	1. No acknowledgments. 2. No mechanism to detect missing or mis-sequenced datagrams. 3. No flow control.

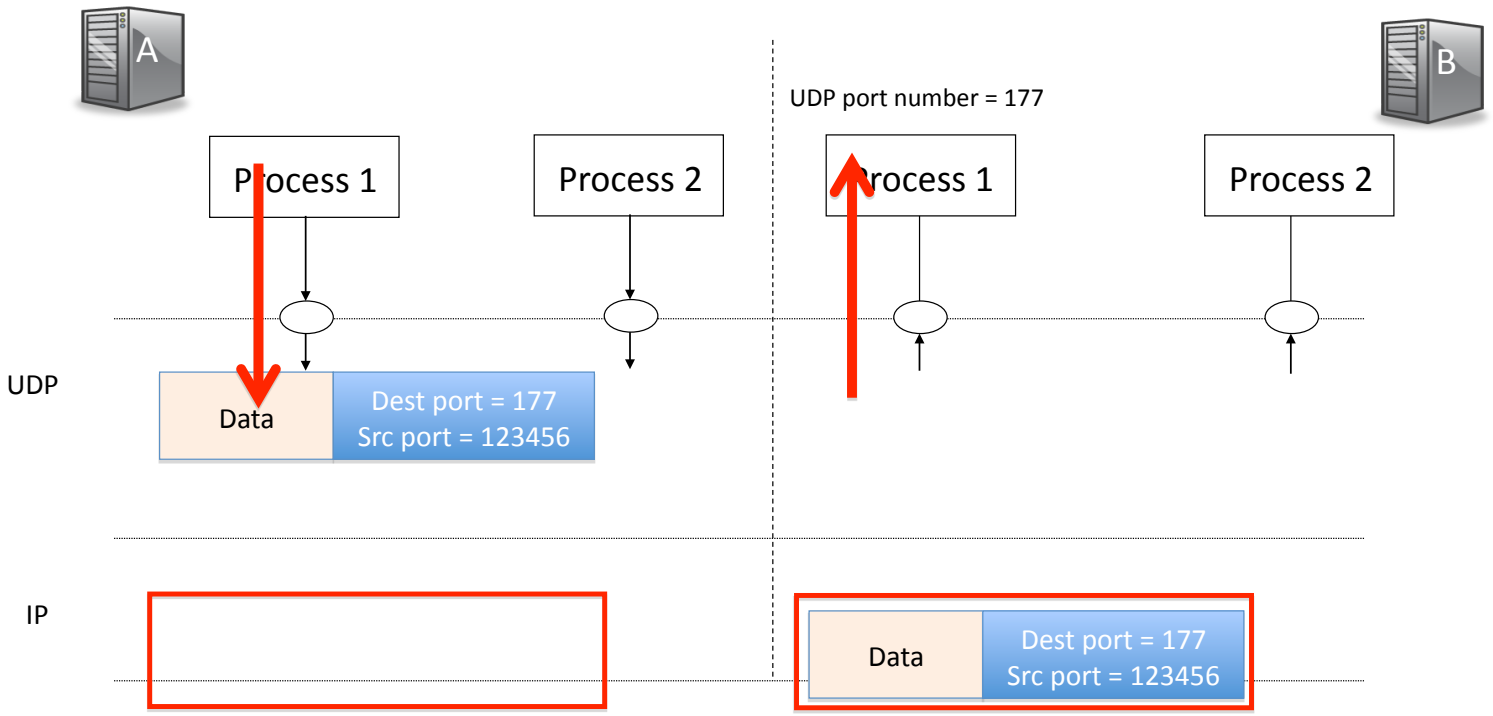
The UDP Datagram Format



Fields used for UDP checksum (if non-zero)

IPv4: IP DA, IP SA, Protocol ID for UDP (17), UDP data+hdr length
UDP: source port, destination port, length
UDP: data

UDP: Port Demultiplexing



Summary

UDP provides a simpler, datagram delivery service between application processes.

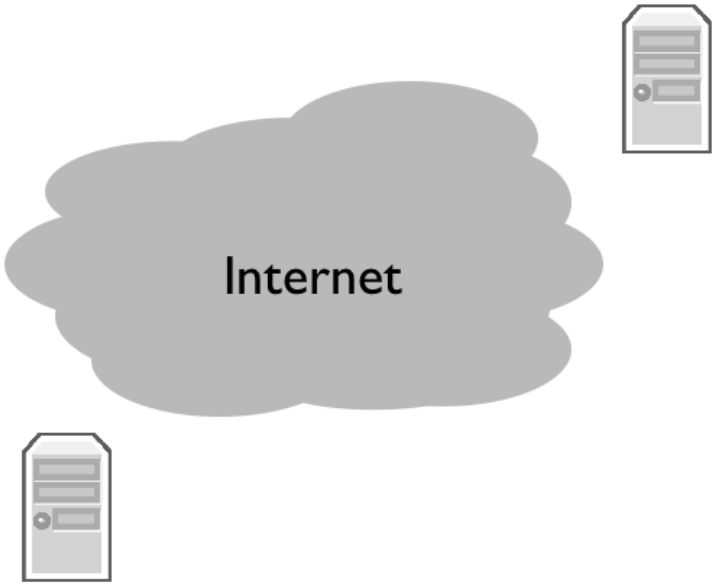
Used for DNS, DHCP, new transport protocols!

- Because NATs support UDP…

The TCP Service Model

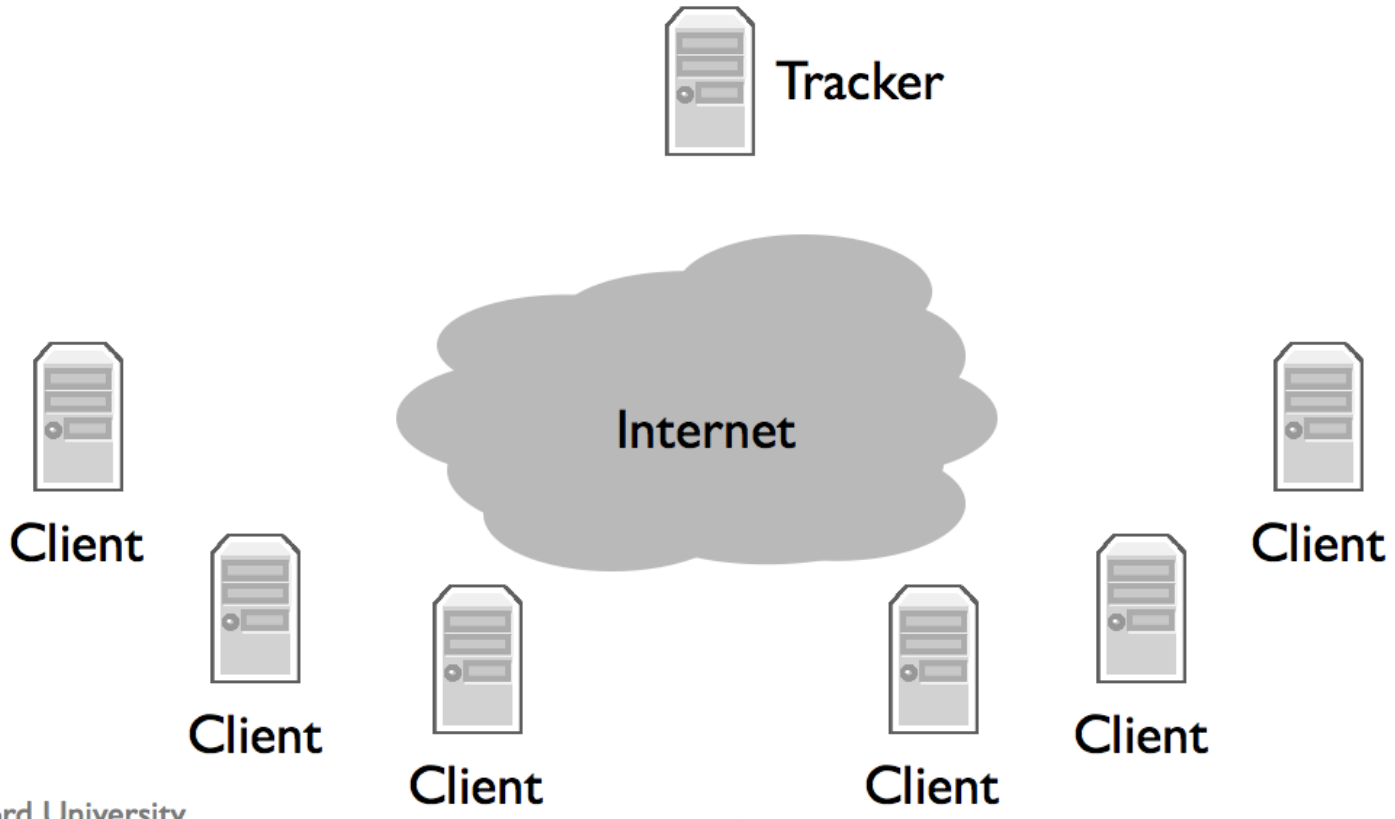
Property	Behavior
Stream of bytes	Reliable byte delivery service.
Reliable delivery	1. Acknowledgments indicate correct delivery. 2. Checksums detect corrupted data. 3. Sequence numbers detect missing data. 4. Flow-control prevents overrunning receiver.
In-sequence	Data delivered to application in sequence transmitted.
(Congestion Control	Controls network congestion.)

Network Applications



- Read and write data over network
- Dominant model: bidirectional, reliable byte stream connection
  - One side reads what the other writes
  - Operates in both directions
  - Reliable (unless connection breaks)

BitTorrent

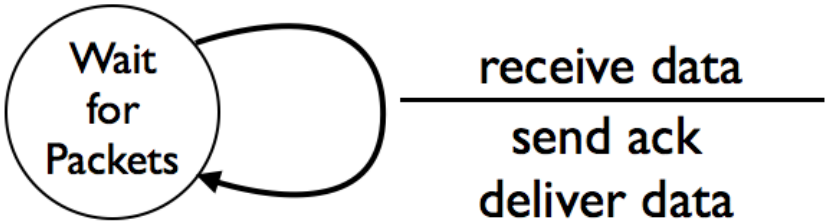

Tracker
Internet
Client
Client
Client
Client
Client
Client

Stop and Wait

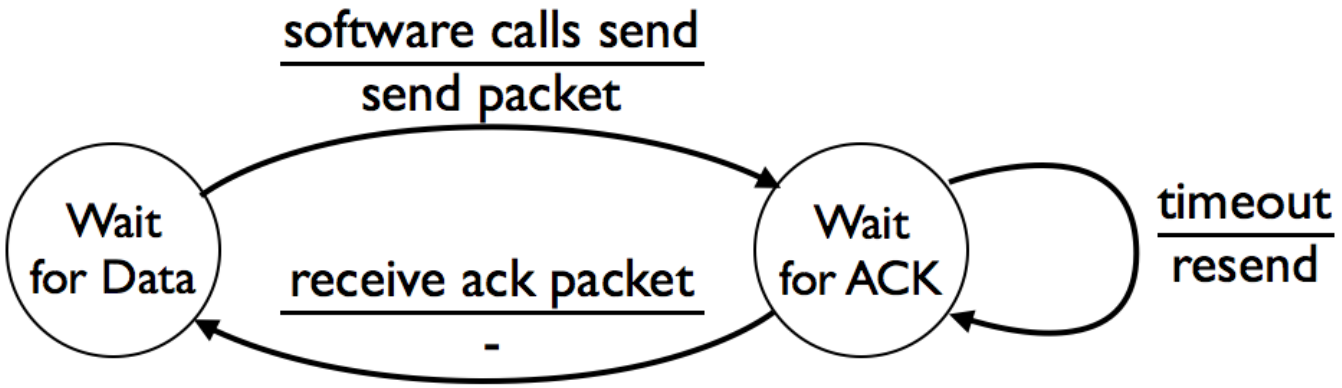
- At most one packet in flight at any time
- Sender sends one packet
- Receiver sends acknowledgment packet when it receives data
- On receiving acknowledgment, sender sends new data
- On timeout, sender resends current data

Stop and Wait FSM

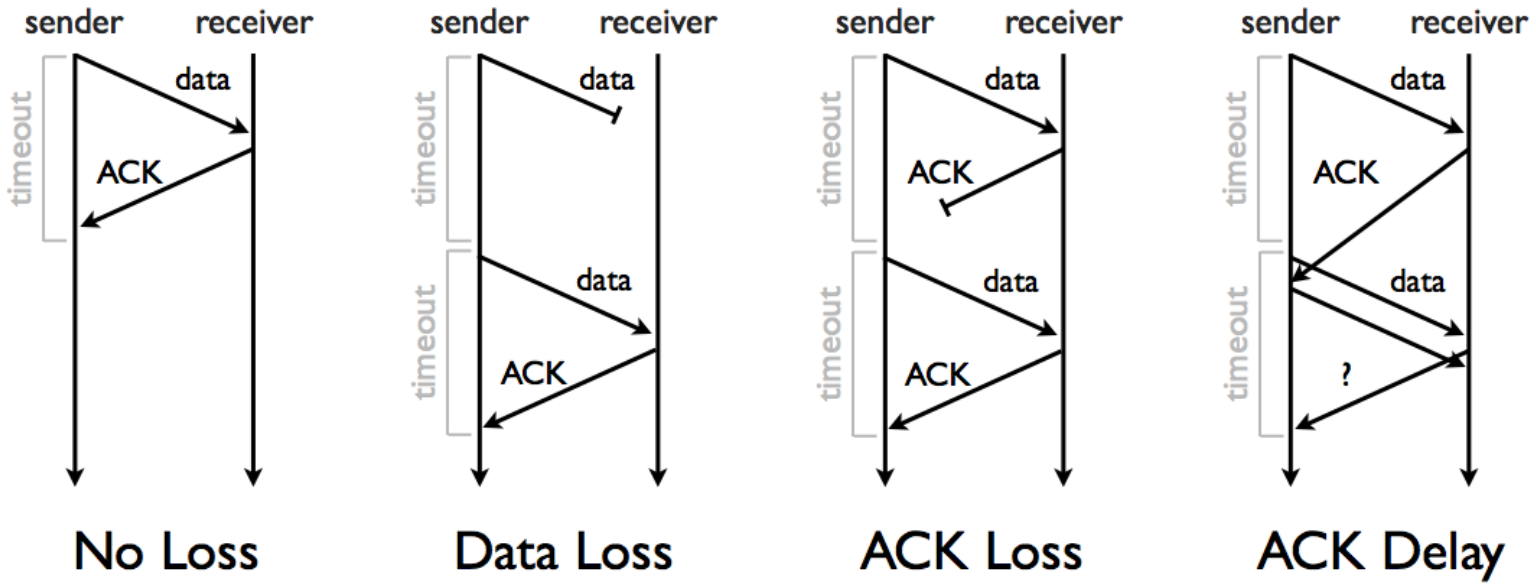
Receiver FSM



Sender FSM

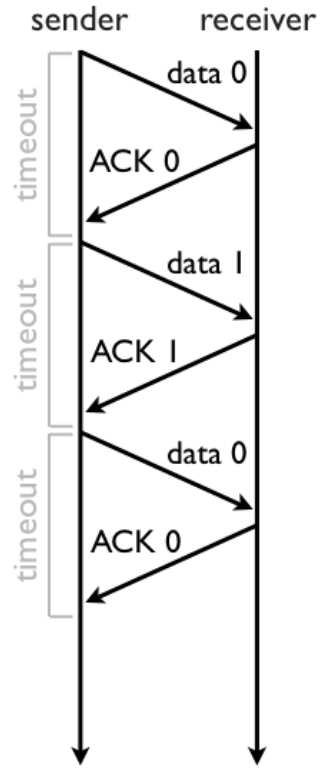


Example Executions



Duplicates

- Use 1-bit counter in data and acknowledgments
  - Receiver can tell if new data or duplicate
- Some simplifying assumptions
  - Network does not duplicate packets
  - Packets not delayed multiple timeouts



Stop and Wait Problem

Bottleneck is 10Mbps
RTT is 50ms

Sliding Window Sender

- Every segment has a sequence number (SeqNo)
- Maintain 3 variables
  - Send window size (SWS)
  - Last acknowledgment received (LAR)
  - Last segment sent (LSS)
- Maintain invariant: (LSS - LAR) ≤ SWS
- Advance LAR on new acknowledgment
- Buffer up to SWS segments

Sliding Window Receiver

- Maintain 3 variables
  - Receive window size (RWS)
  - Last acceptable segment (LAS)
  - Last segment received (LSR)
- Maintain invariant: (LAS - LSR) ≤ RWS
- If received packet is < LAS, send acknowledgment
  - Send *cumulative* acks: if received 1, 2, 3, 5, acknowledge 3
  - NOTE: TCP acks are next *expected* data (e.g., ack 4 in above example)

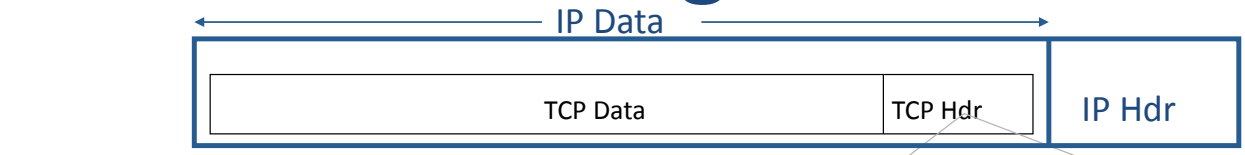
RWS, SWS, and Sequence Space

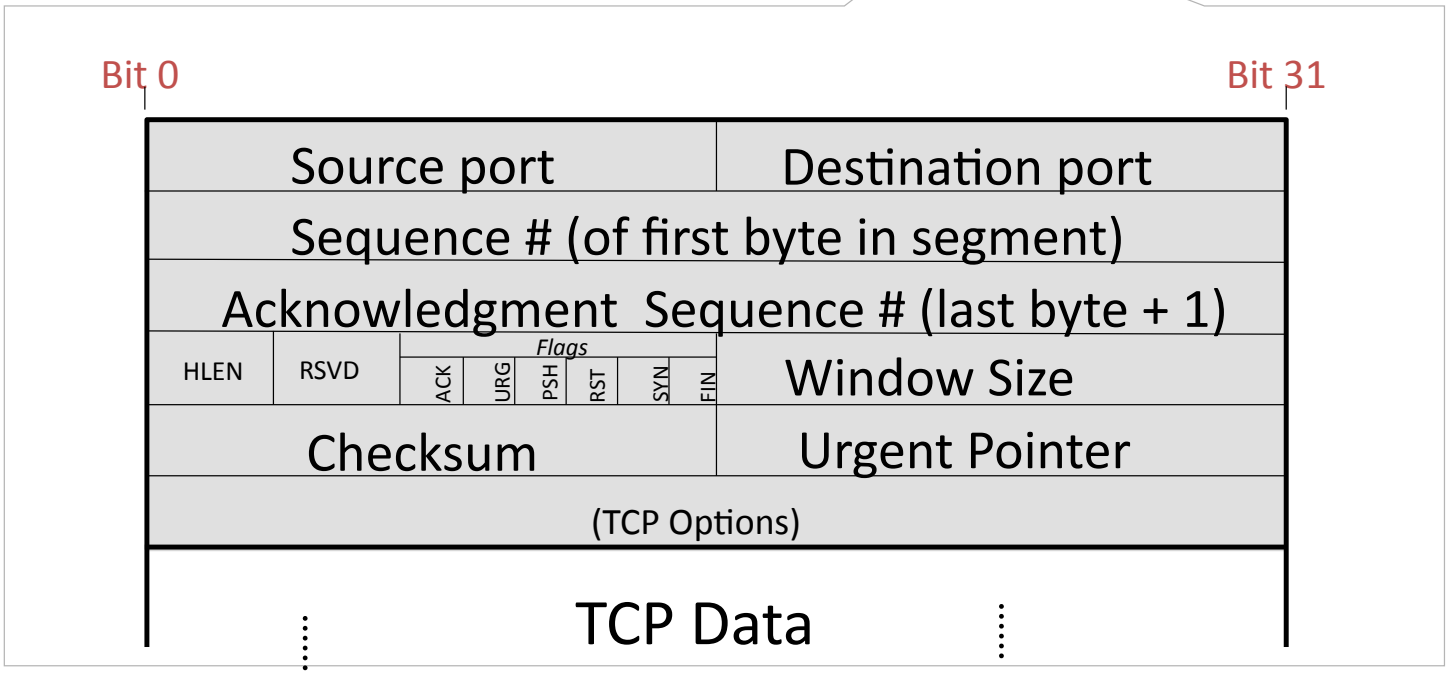
- RWS $\geq$ 1, SWS $\geq$ 1, RWS $\leq$ SWS
- Assuming packets not more than 2 RTTs:
  - If RWS = 1, "go back N" protocol, need SWS+1 sequence numbers
  - If RWS = SWS, need 2SWS sequence numbers
- Generally need RWS+SWS sequence numbers per 2 RTTs of delay
- Question: assume RWS=SWS; what can go wrong if there are fewer than 2SWS sequence numbers?

TCP and sequence numbers

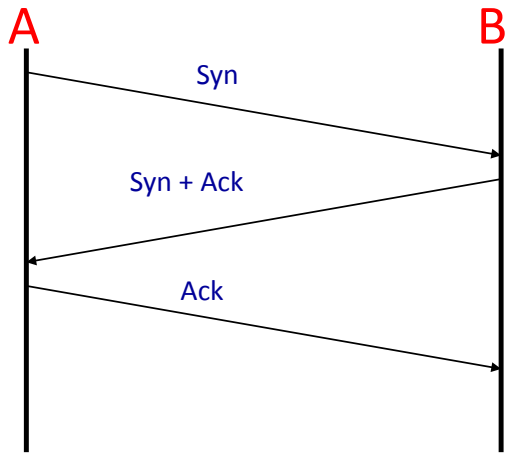
- Each side of a TCP connection has
  - a send window of sequence numbers
  - a receive window of sequence numbers
- To start a connection, each peer needs to tell other side what window it should receive/expect

The TCP Segment Format

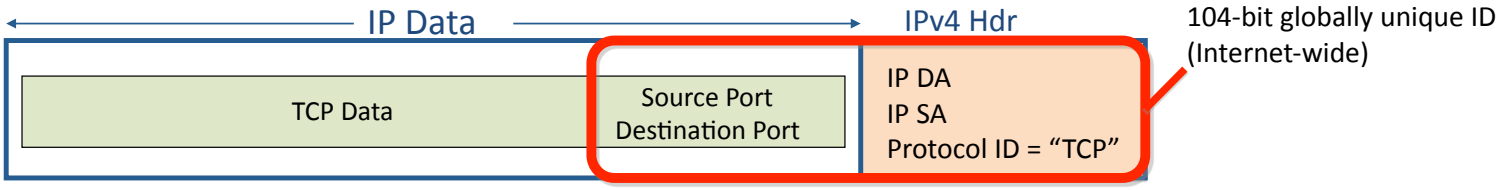




Connection setup
3-way handshake

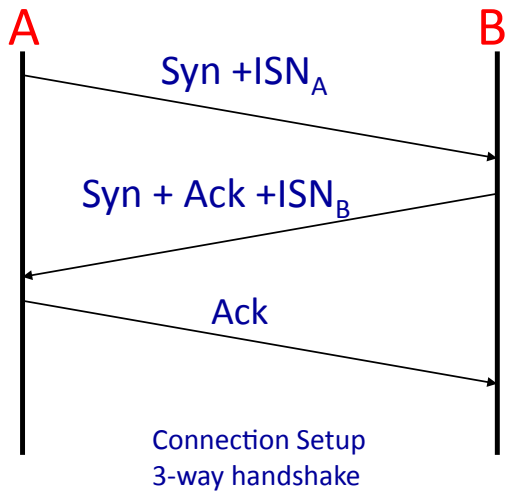


The Unique ID of a TCP connection

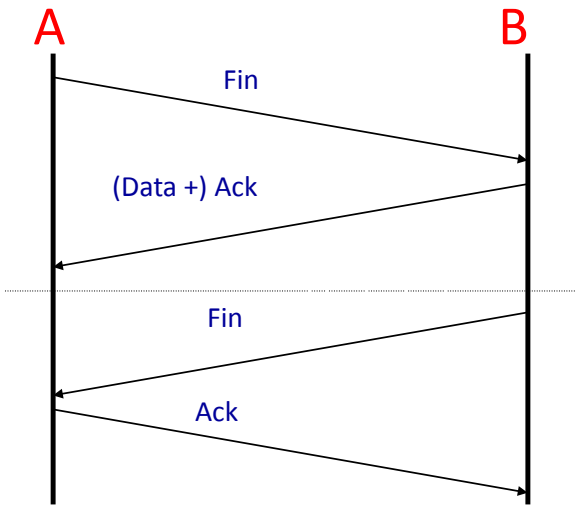


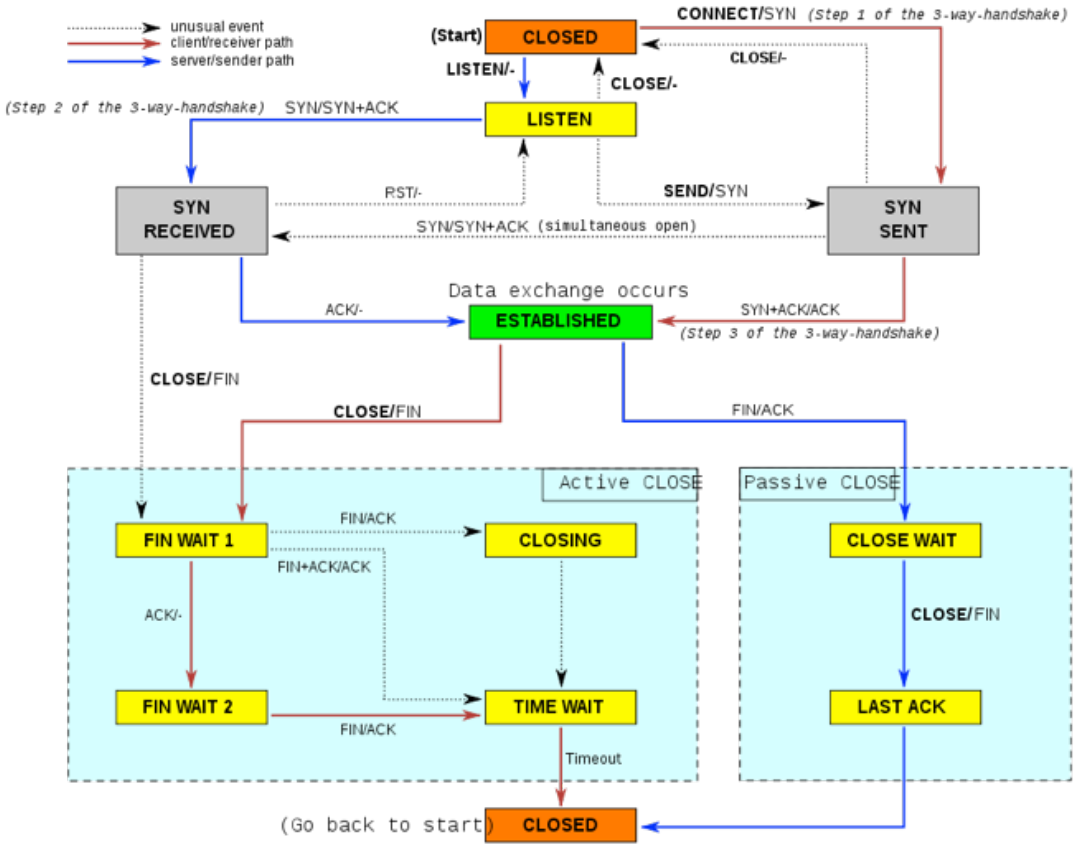
1. Host A increments source port for every new connection

2. TCP picks ISN to avoid overlap with previous connection with same ID.



Connection teardown





Why is there a timeout for TIME WAIT?

Why not for LAST ACK?