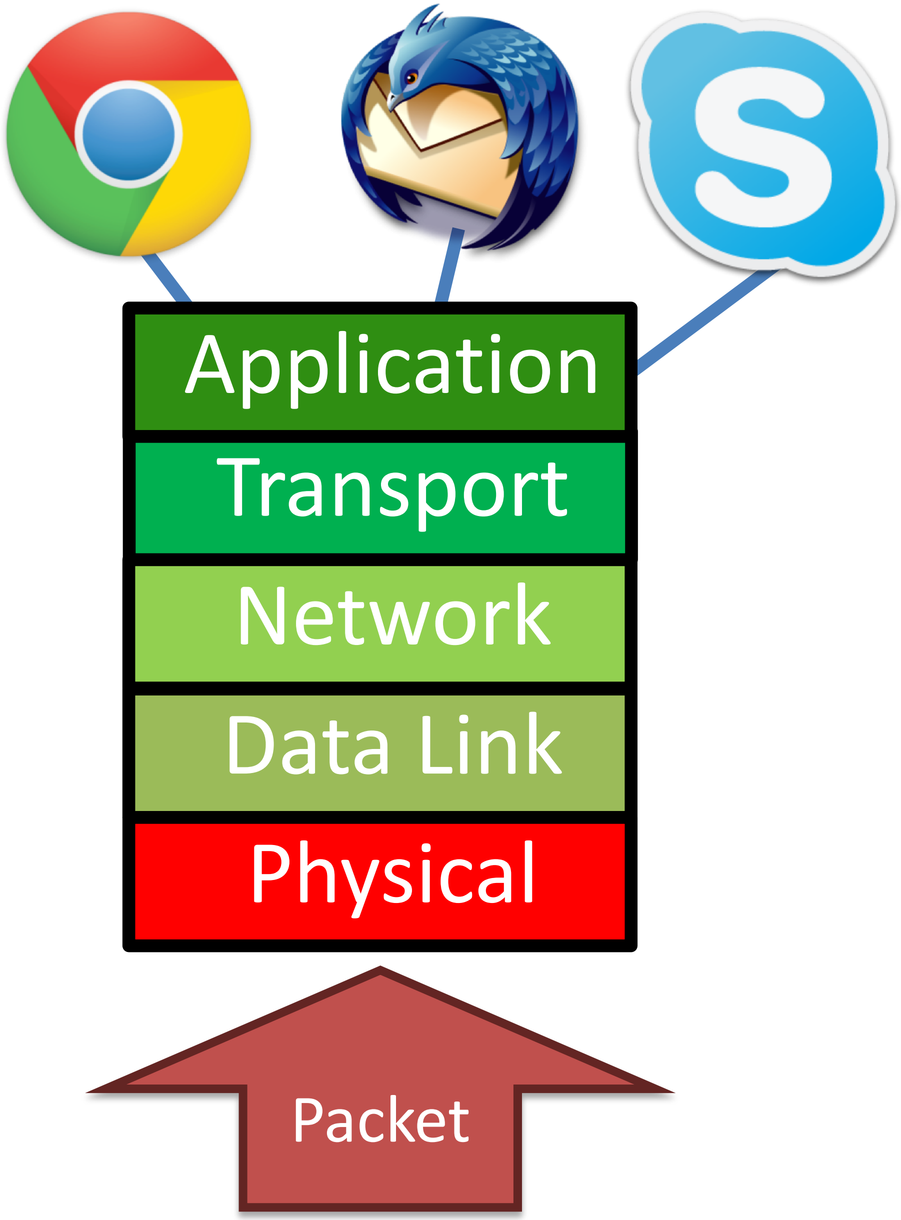
Transport Layer

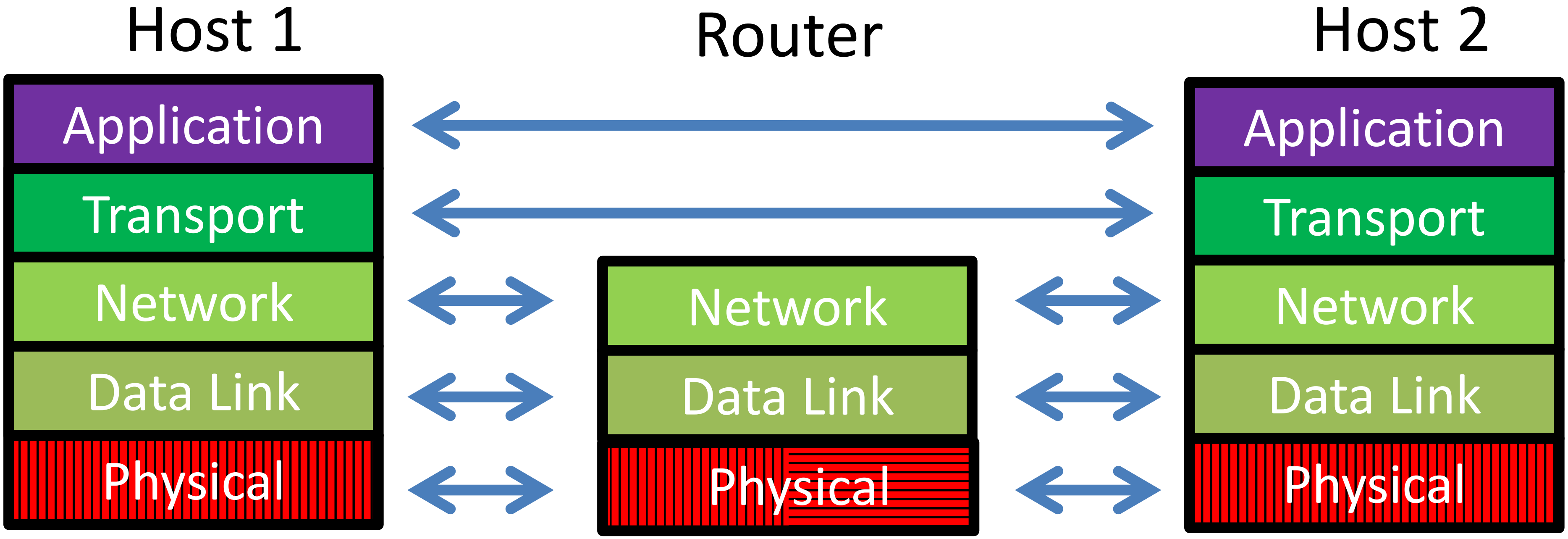
Re-look at the stack

- Headers are "peeled" as you go up the stack

- Headers are added as you go down the stack.



Layering, Revisited



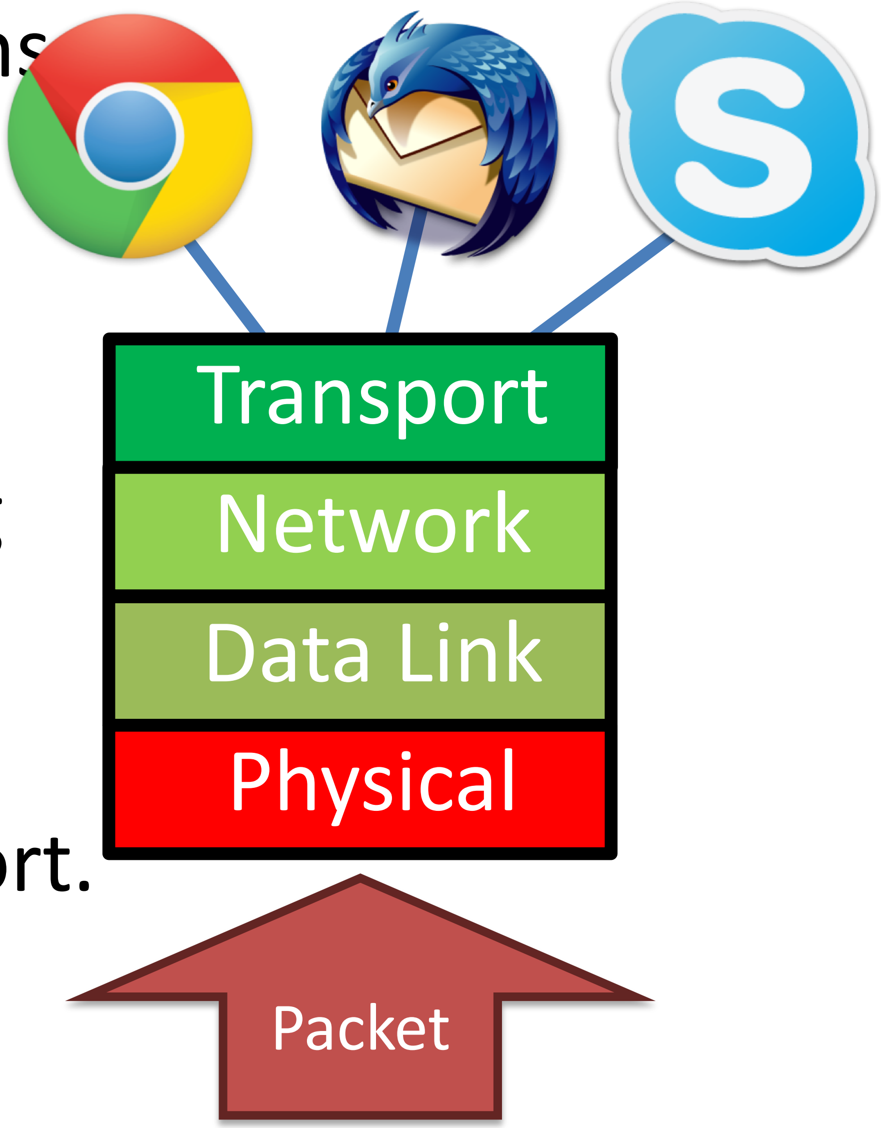
- Lowest level end-to-end protocol
  - Transport header only read by source and destination
  - Routers view transport header as payload
  - Each packet has a Maximum Segment Size (MSS)

Transport layer: TCP

- Transport layer roles
  - “End-to-End” abstraction
  - De-multiplexing

What is de-multiplexing?

- Clients run many applications at the same time
  - Who to deliver packets to?
- Insert Transport Layer to handle demultiplexing using ports
- The end point is identified using an IP address and a port.



Transport

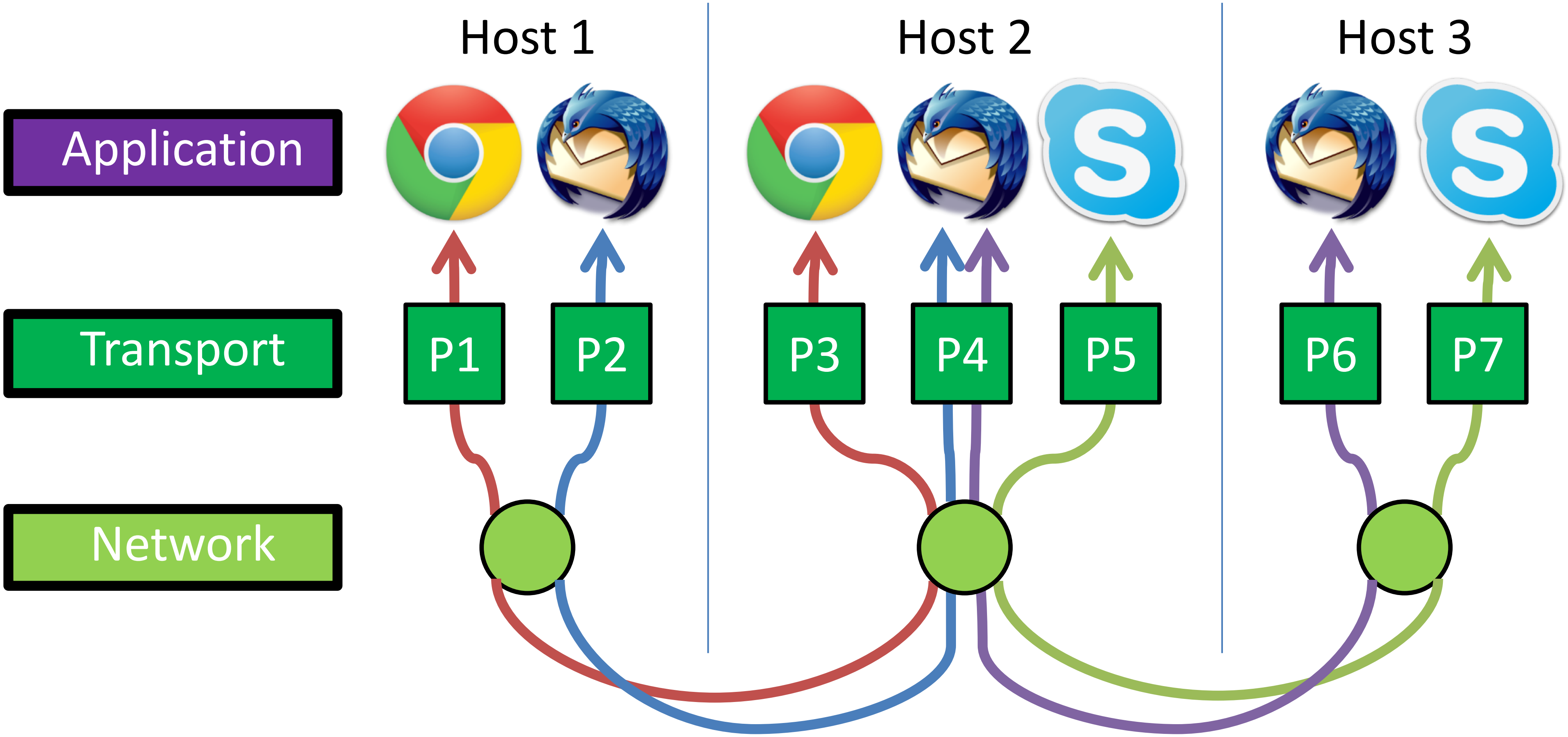
Network

Data Link

Physical

Packet

Demultiplexing Traffic



Endpoints identified by *<src_ip, src_port, dest_ip, dest_port>*

Two types of Transport Protocol

- Transmission Control Protocol (TCP)
  - Connection oriented
  - Masks unreliability.

- User Datagram Protocol (UDP)
  - Connection less
  - Does not mask unreliability.

Socket Programming

- Socket programming provides a way to realize the transport layer abstraction
- Create a socket, connect to the socket, and create a connection.

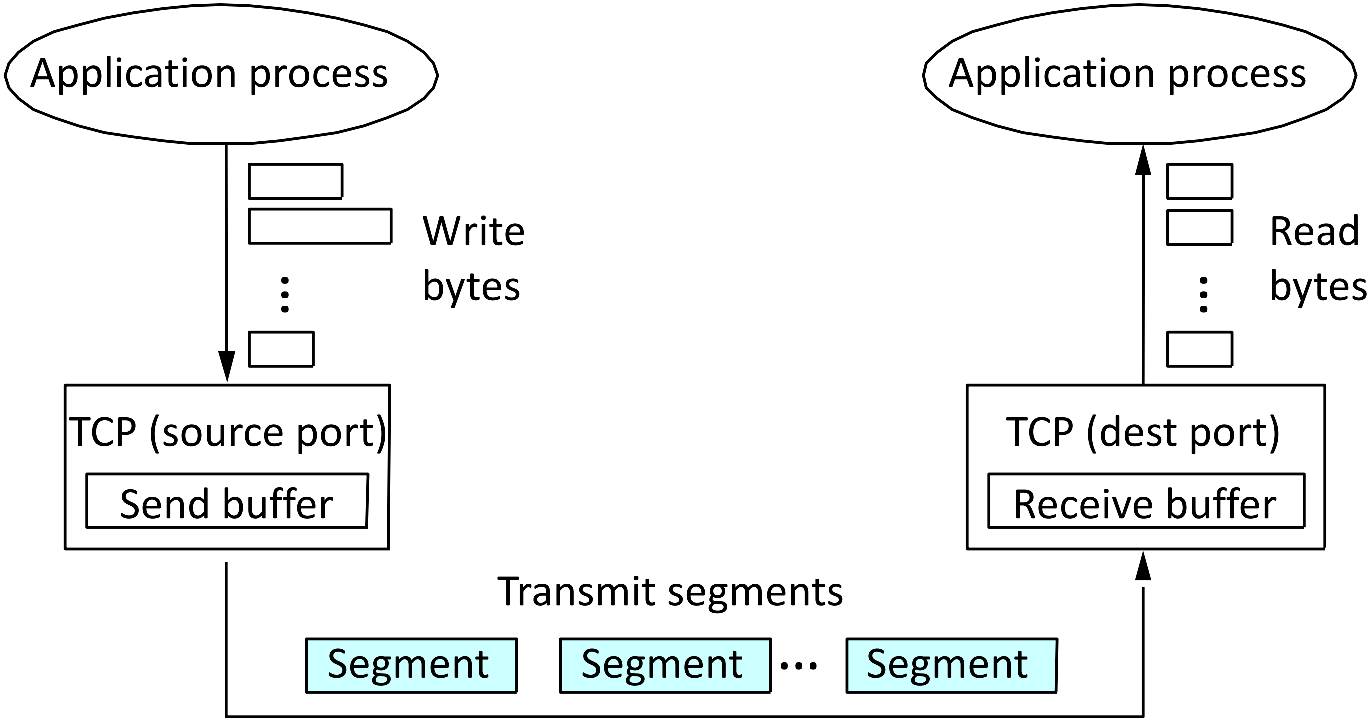
TCP

Transmission Control Protocol

- TCP properties
  - Bi-directional
  - Stream based/connection oriented
  - Maintains state per connection

- TCP provides the following abstraction
  - In-order delivery
  - Reliability
  - End-to-end connectivity*

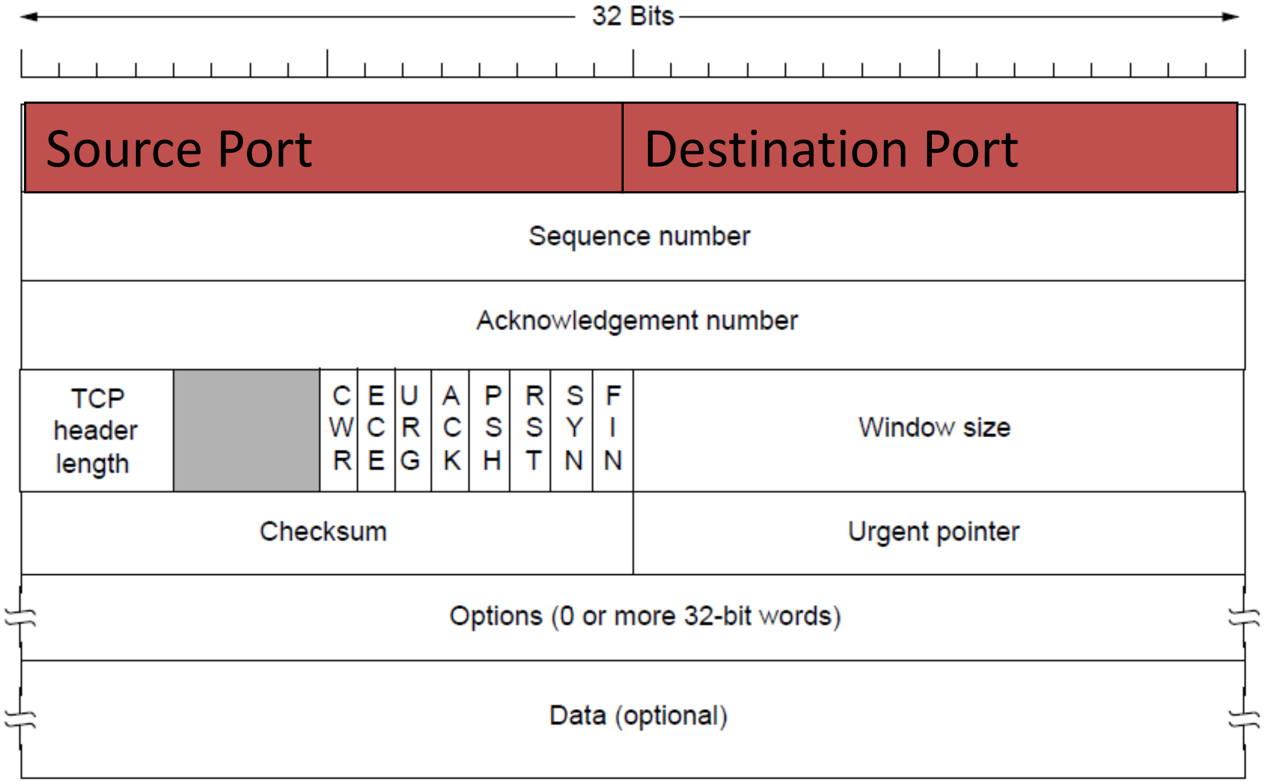
TCP In-Order Delivery



What is the buffer for?

TCP Header Format

- Ports plus IP addresses identify a connection



32 Bits

Source Port

Destination Port

Sequence number

Acknowledgement number

TCP header length

CWR

ECE

URG

ACK

PSH

RST

SYN

FIN

Window size

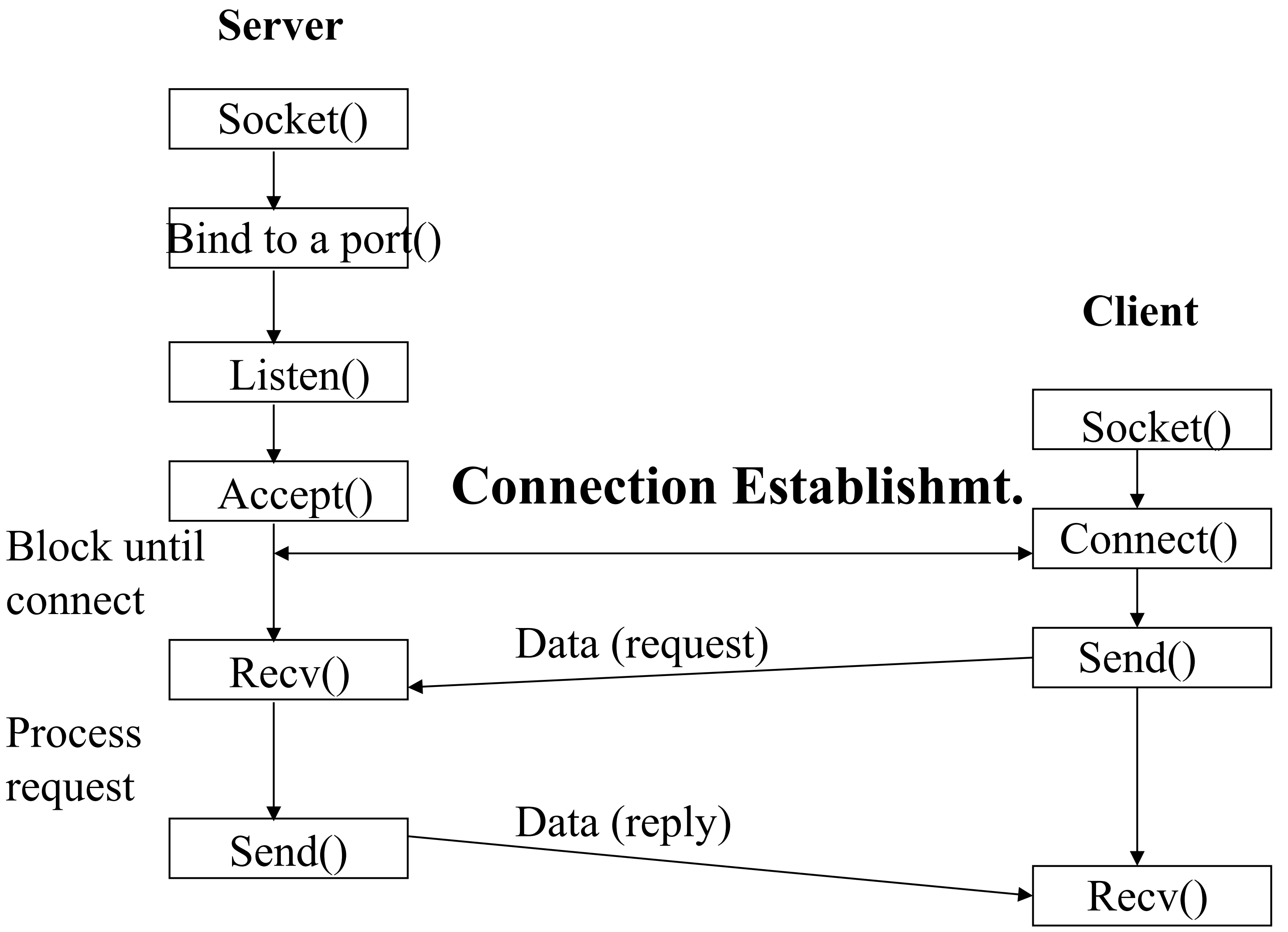
Checksum

Urgent pointer

Options (0 or more 32-bit words)

Data (optional)

TCP connection



Connection Establishment in TCP

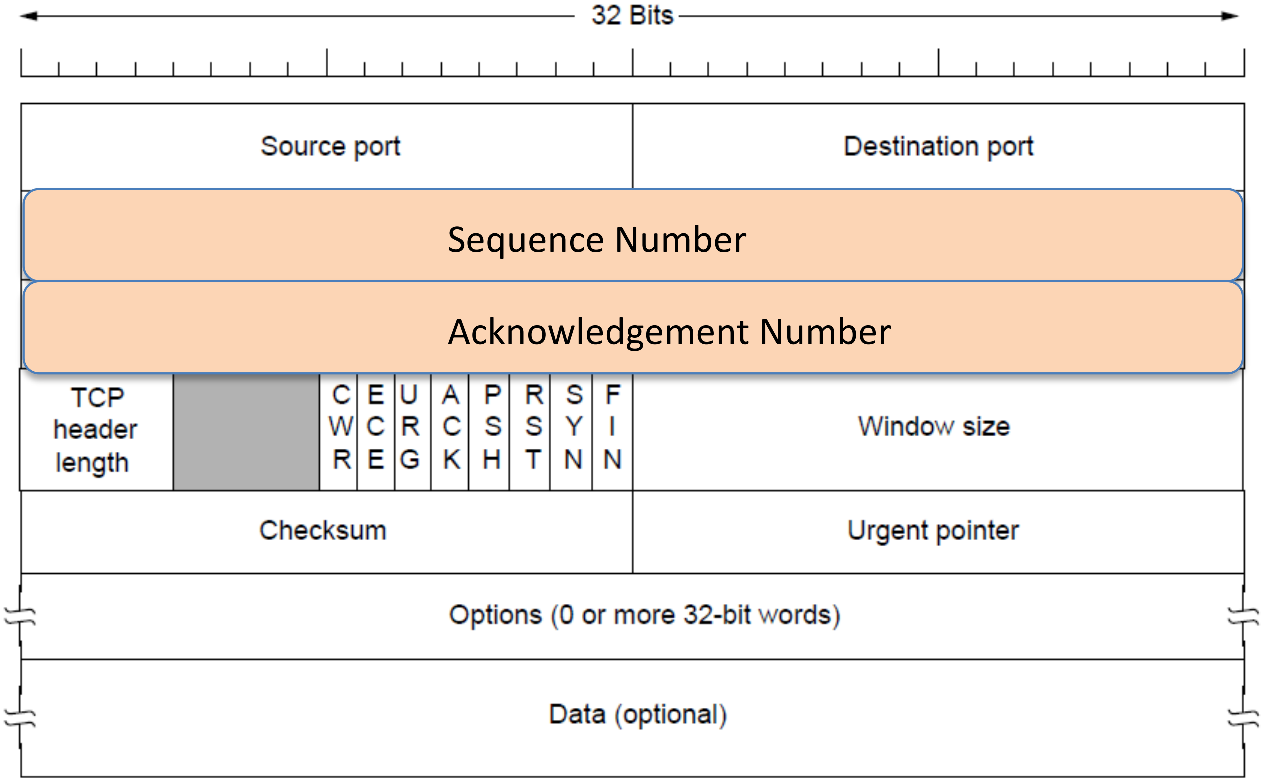
- Both sender and receiver must be ready before we start to transfer the data
- Sender and receiver need to agree on a set of parameters
  - This is signaling. It sets up state at the endpoints
  - Compare to “dialing” in the telephone network

Problems with Connection Establishment

Key problem is to ensure reliability even though packets may be lost, corrupted, delayed, and duplicated

How can we avoid duplicates and delayed packets?

Sequence numbers



32 Bits

Source port

Destination port

Sequence Number

Acknowledgement Number

TCP header length

CWR

ECE

URG

ACK

PSH

RST

SYN

FIN

Window size

Checksum

Urgent pointer

Options (0 or more 32-bit words)

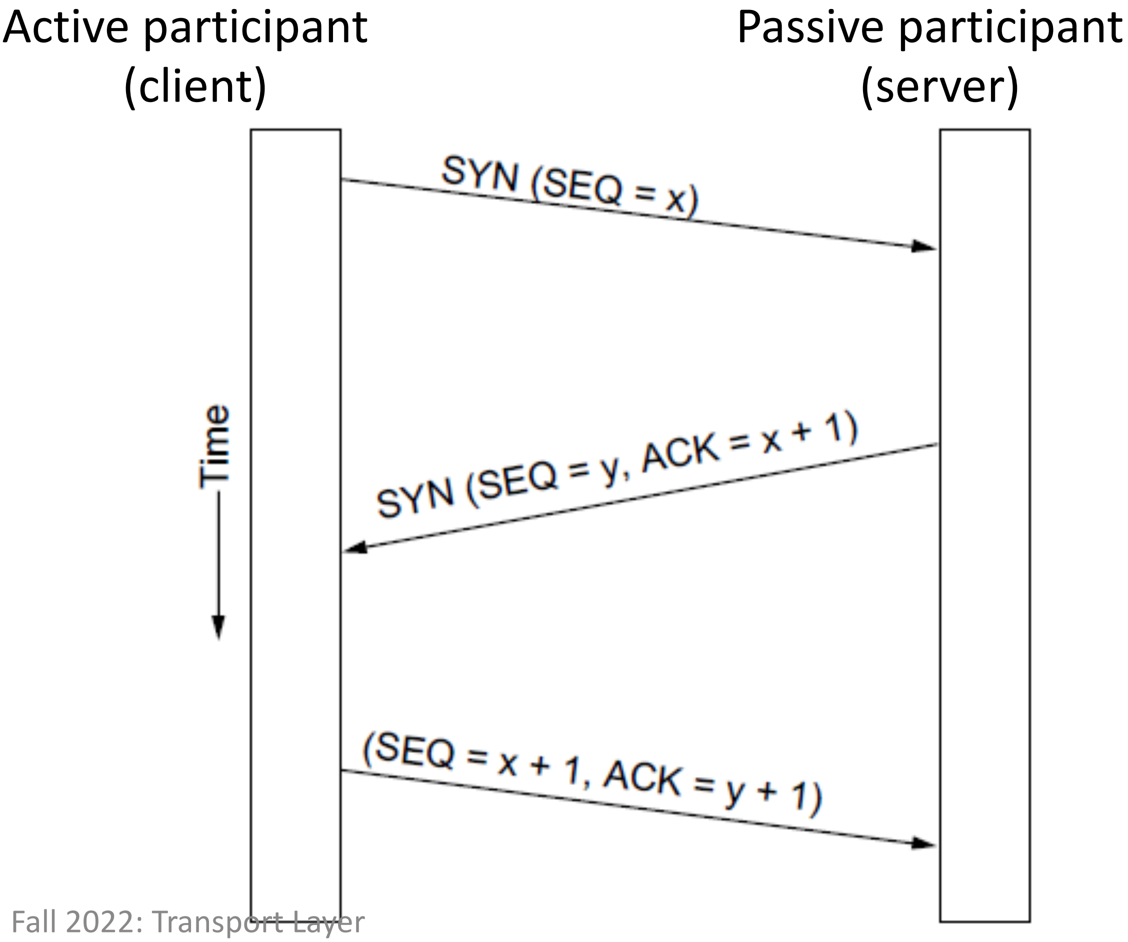
Data (optional)

Use a maximum segment lifetime (MSL)

- Wait until MSL to repeat sequence numbers (120 seconds in the Internet)

Three-Way Handshake

- Opens both directions for transfer


Active participant (client)
Passive participant (server)
SYN (SEQ = x)
SYN (SEQ = y, ACK = x + 1)
(SEQ = x + 1, ACK = y + 1)
Time

Why three way handshake?

- TCP is a bi-directional communication. Both directions have to establish a sequence number to be used during the communication

- What else happens during the handshake?
  - Exchange of connection parameters
  - **TCP is a stateful connections**

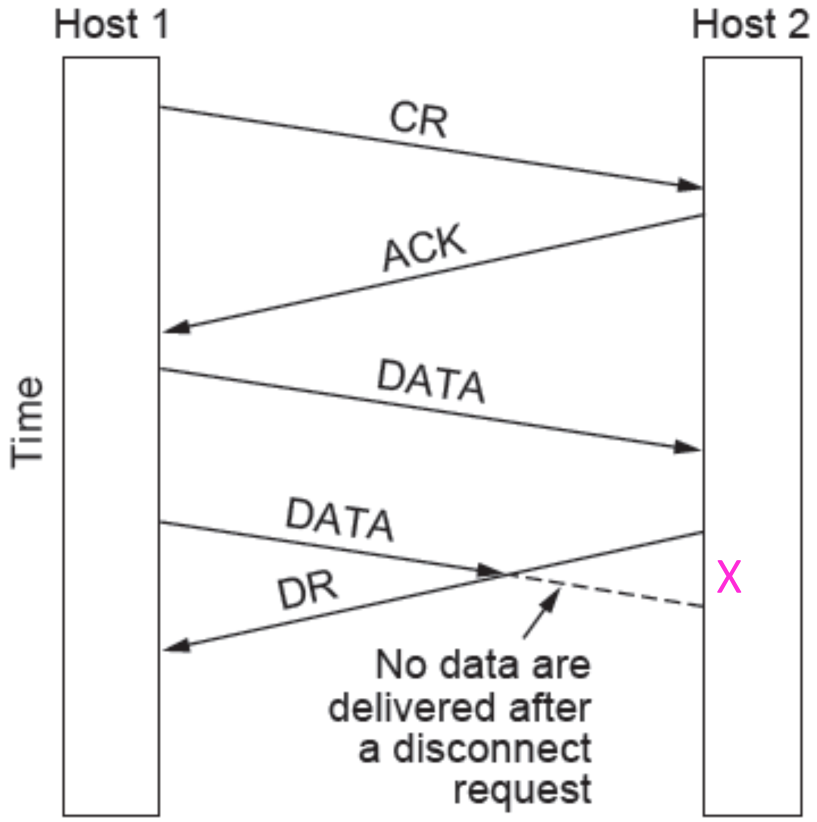
Connection Teardown

- Cleans up state in sender and receiver

- TCP provides a “symmetric” close
  - both sides shutdown independently
  - Why?

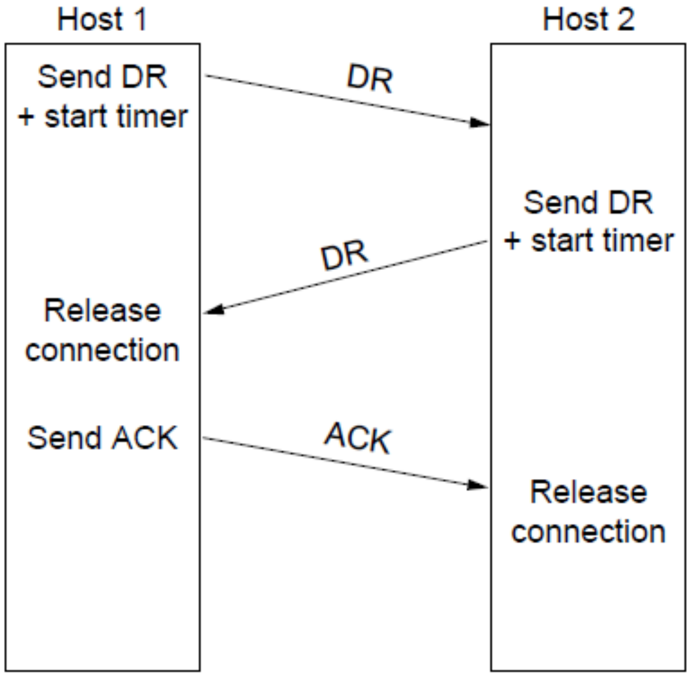
Connection Release problem

- Key problem is to ensure reliability while releasing (DR: Disconnect request)
- Asymmetric release (when one side breaks connection) is abrupt and may lose data

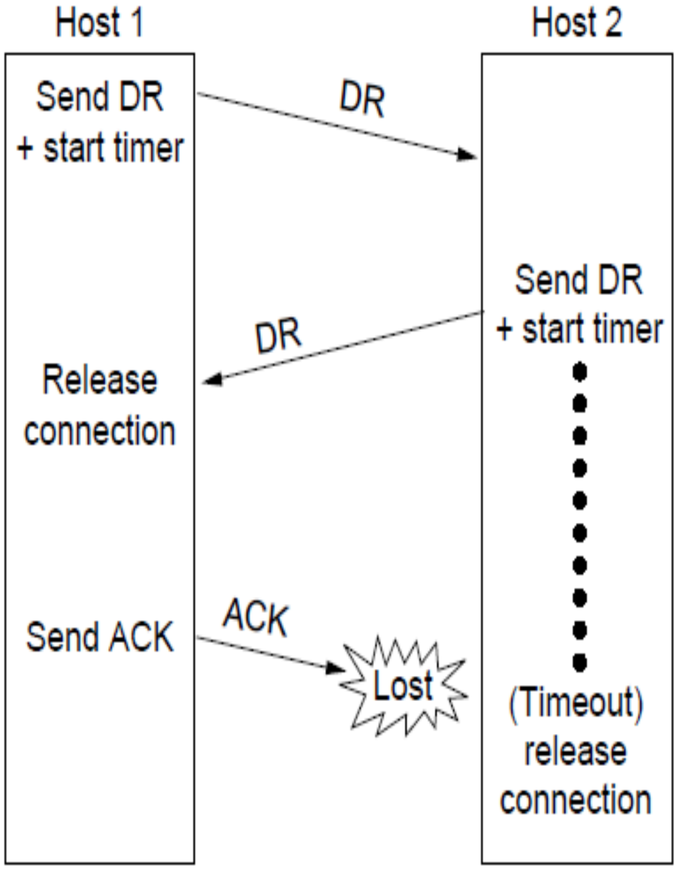


Connection Release

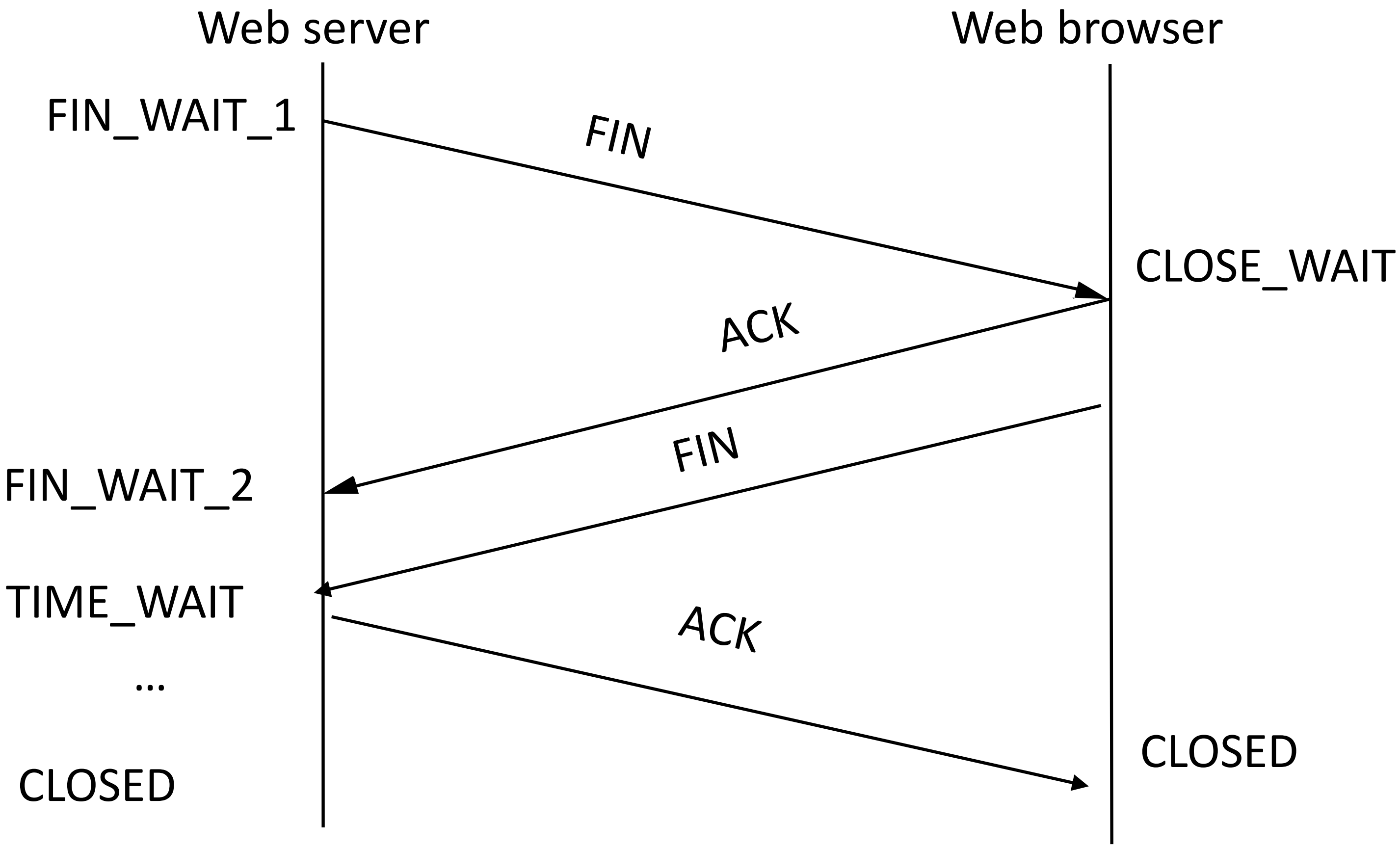
- Normal release sequence, initiated by transport user on Host 1
  - DR=Disconnect Request
  - Both DRs are ACKed by the other side
- What happens if ack is lost?



Error handling in connection release

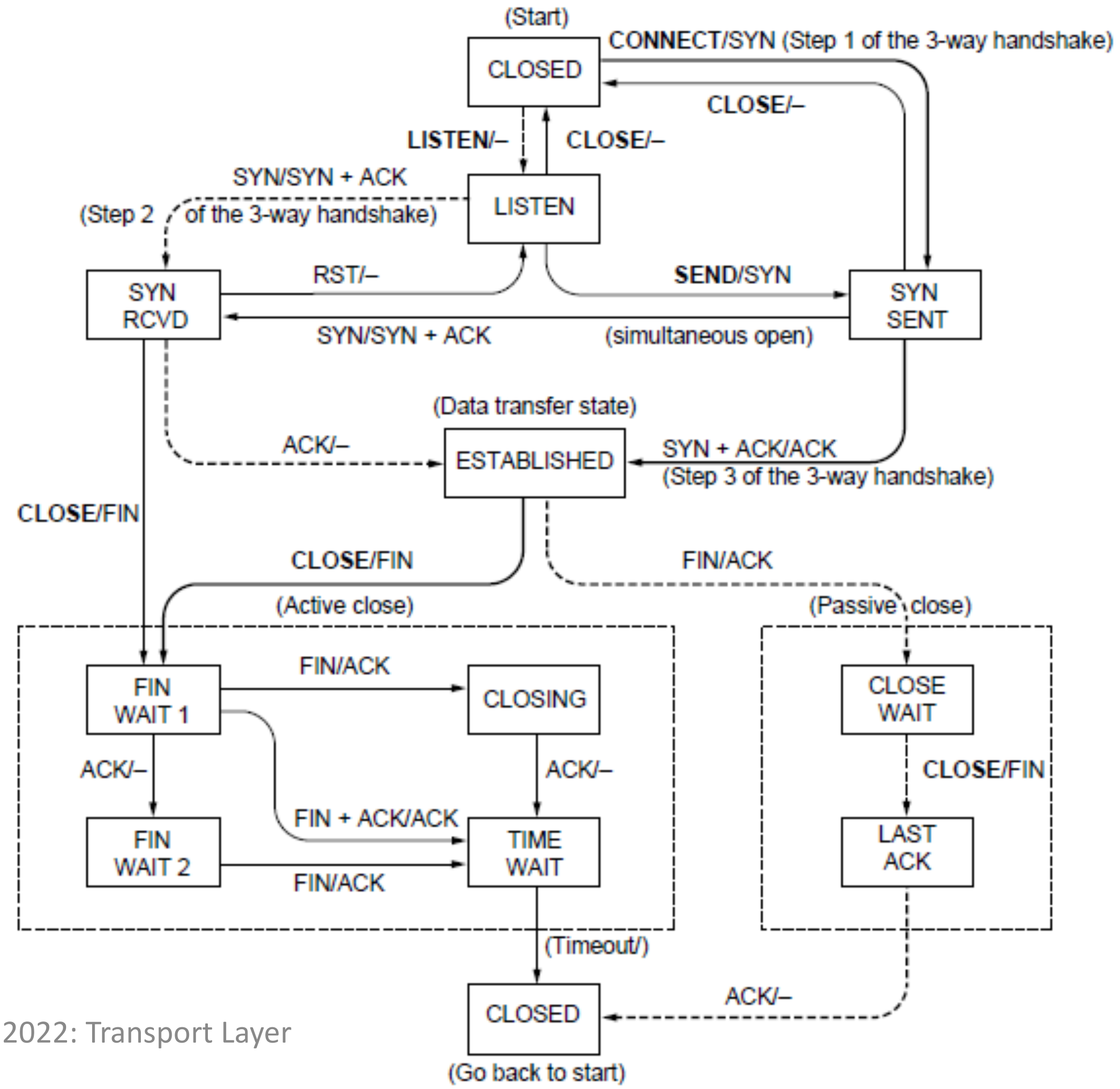


TCP Connection Teardown with states



TCP State Transitions

- Wow!


(Start)
CLOSED
CONNECT/SYN (Step 1 of the 3-way handshake)
CLOSE/–
LISTEN/–
CLOSE/–
SYN/SYN + ACK
(Step 2 of the 3-way handshake)
LISTEN
RST/–
SEND/SYN
SYN RCVD
SYN SENT
SYN/SYN + ACK
(simultaneous open)
(Data transfer state)
ACK/–
ESTABLISHED
SYN + ACK/ACK
(Step 3 of the 3-way handshake)
CLOSE/FIN
CLOSE/FIN
FIN/ACK
(Active close)
(Passive close)
FIN WAIT 1
FIN/ACK
CLOSING
CLOSE WAIT
ACK/–
ACK/–
CLOSE/FIN
FIN + ACK/ACK
FIN WAIT 2
TIME WAIT
LAST ACK
FIN/ACK
(Timeout/)
ACK/–
CLOSED
(Go back to start)