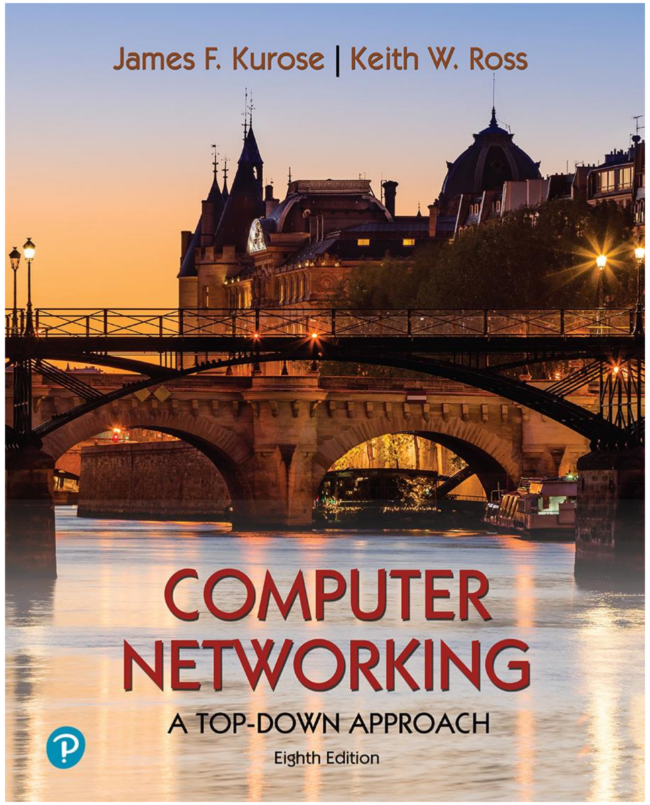
Computer Networks

Jong-won Lee

Handong University

Chapter 3: roadmap

- Transport-layer services
- Multiplexing and demultiplexing
- Connectionless transport: UDP
- Principles of reliable data transfer
- Connection-oriented transport: TCP
  - segment structure
  - reliable data transfer
  - flow control
  - connection management
- Principles of congestion control
- TCP congestion control

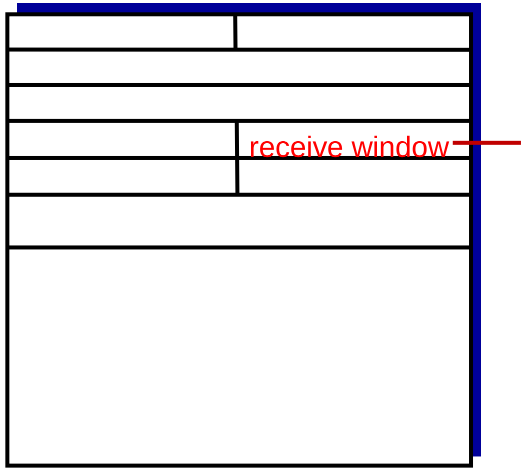


TCP flow control

Q: What happens if network layer delivers data faster than application layer removes data from socket buffers?

flow control

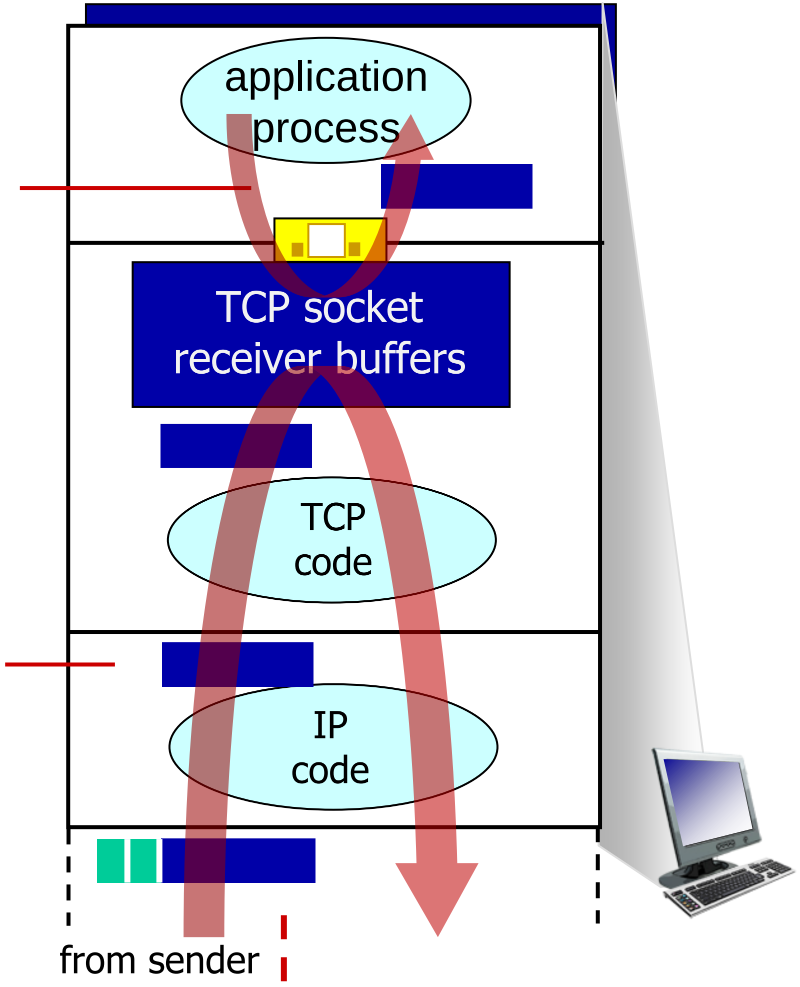
receiver controls sender, so sender won't
by transmitting too much, too fast



flow control: # bytes receiver willing to accept

Application removing data from TCP socket buffers

Network layer delivering IP datagram payload into TCP socket buffers

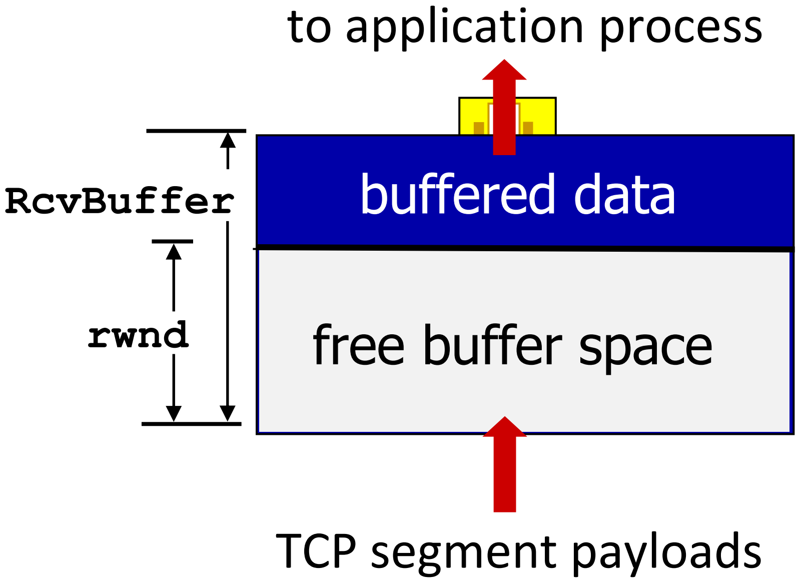


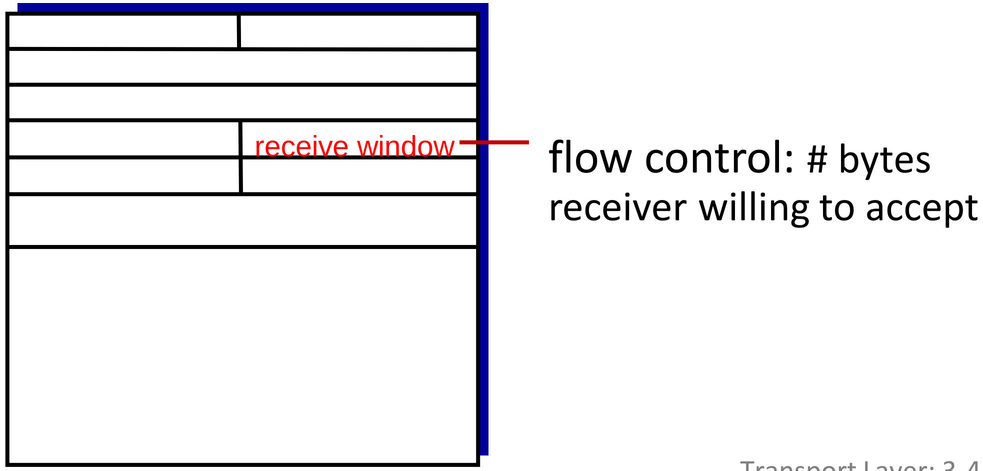
receiver protocol stack

TCP flow control

- TCP receiver "advertises" free buffer space in **`rwnd`** field in TCP header
  - **`RcvBuffer`** size set via socket options (typical default is 4096 bytes)
  - many operating systems autoadjust **`RcvBuffer`**
- sender limits amount of unACKed ("in-flight") data to received **`rwnd`**
  - guarantees receive buffer will not overflow

TCP receiver-side buffering





TCP Transmission Policy

- Sender Buffering
  - ‘Tinygram’ wastes bandwidth
    - a keystroke in telnet session = 41 byte
      (40 byte header + 1 byte data)
  - be able to reduce header overhead by grouping many small data segments into one large TCP segment.

  - *Nagle*’s algorithm (RFC 896)

    when data come into the sender one byte, send the first byte. Then

    1) buffer all the rest .

    2) Send a new packet

    - better to be disabled if used on mouse movements.

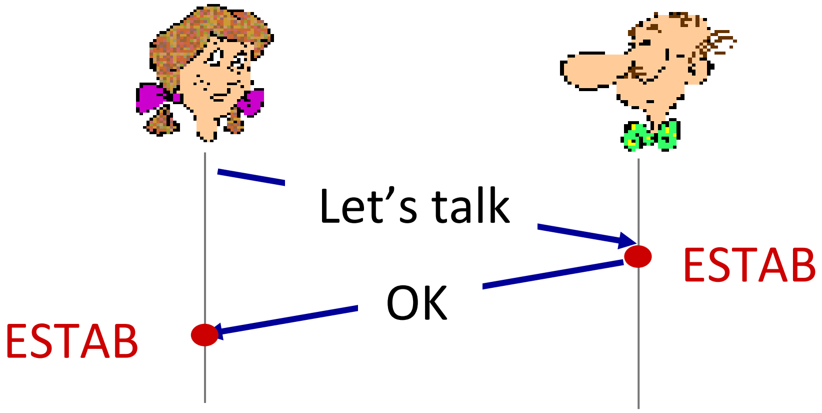
TCP connection management

before exchanging data, sender/receiver “handshake”:

- agree to establish connection (each knowing the other willing to establish connection)
- agree on connection parameters (e.g., starting seq #s)

Agreeing to establish a connection

2-way handshake:

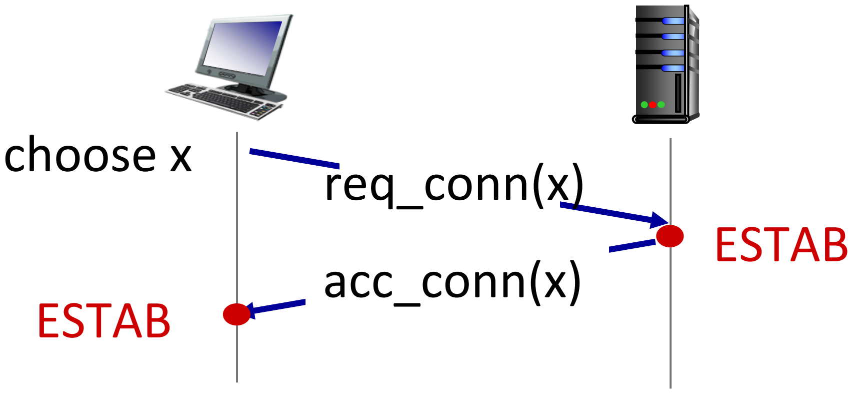


Let’s talk

ESTAB

OK

ESTAB



choose x

req_conn(x)

ESTAB

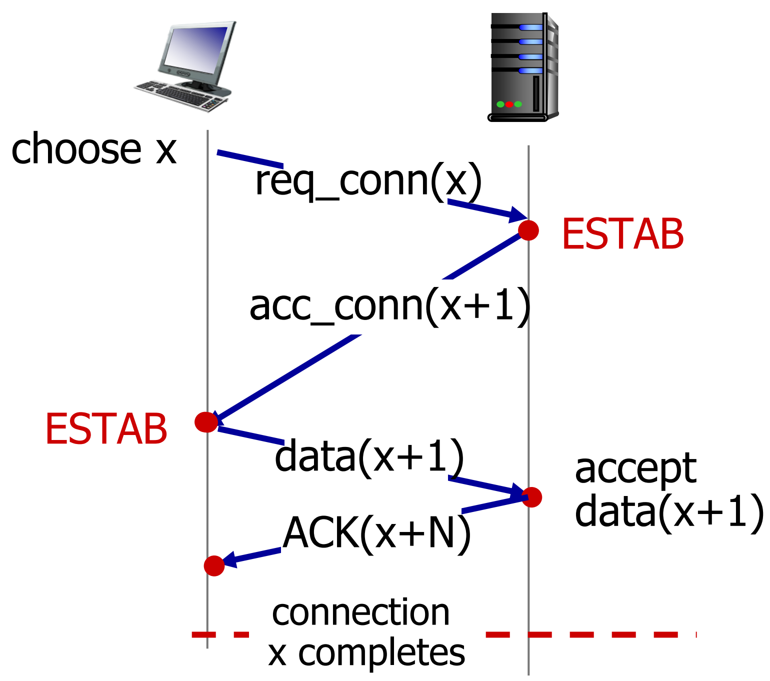
acc_conn(x)

ESTAB

Q: will 2-way handshake always work in network?

- variable delays
- retransmitted messages (e.g. req_conn(x)) due to message loss
- message reordering
- can’t “see” other side

2-way handshake scenarios



choose x

req_conn(x)

ESTAB

acc_conn(x+1)

ESTAB

data(x+1)

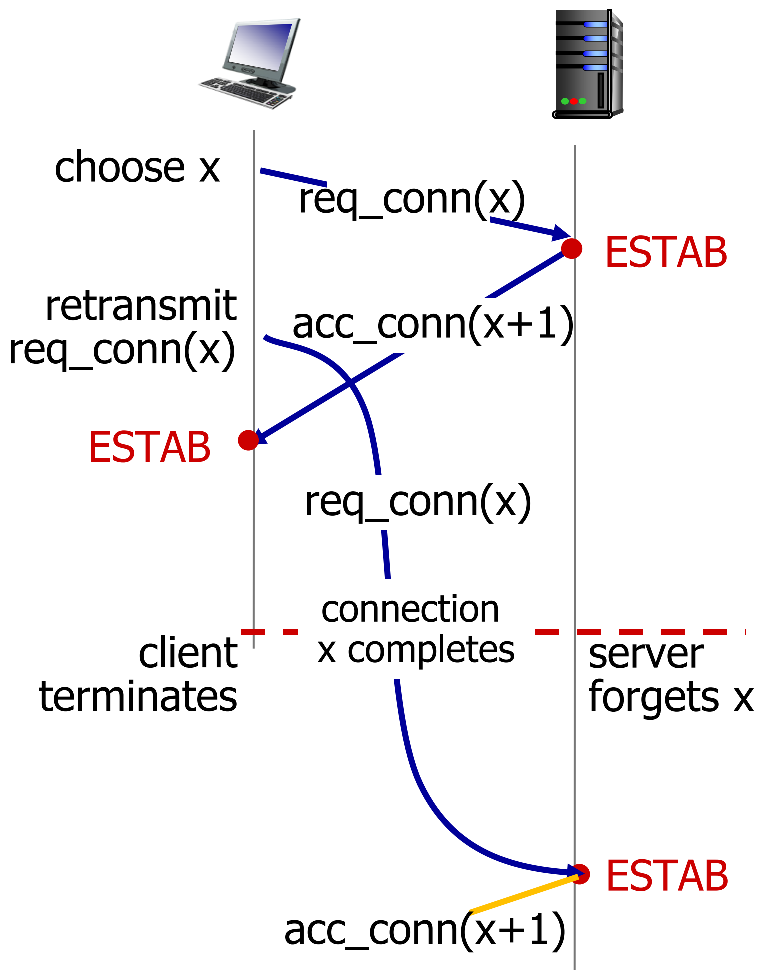
accept data(x+1)

ACK(x+N)

connection x completes

No problem!

✓



choose x

req_conn(x)

ESTAB

retransmit req_conn(x)

acc_conn(x+1)

ESTAB

req_conn(x)

client terminates

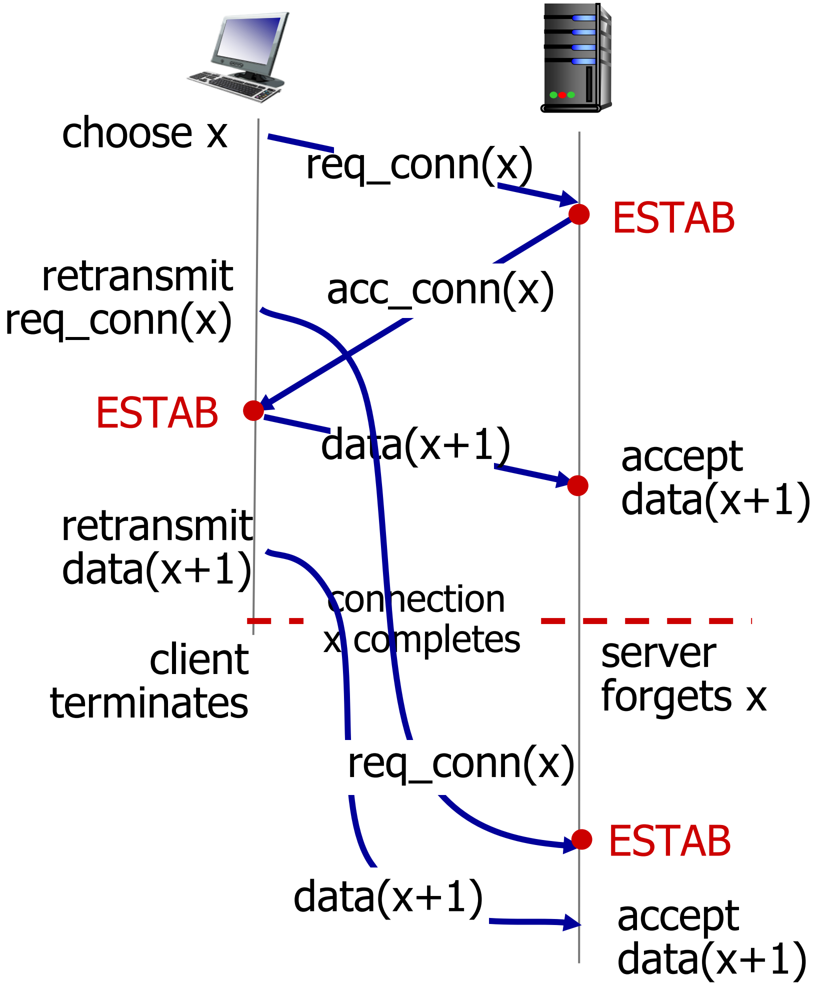
connection x completes

server forgets x

ESTAB

acc_conn(x+1)

✗ Problem: half open connection! (no client)



choose x

req_conn(x)

ESTAB

retransmit req_conn(x)

acc_conn(x)

ESTAB

data(x+1)

accept data(x+1)

retransmit data(x+1)

connection x completes

client terminates

server forgets x

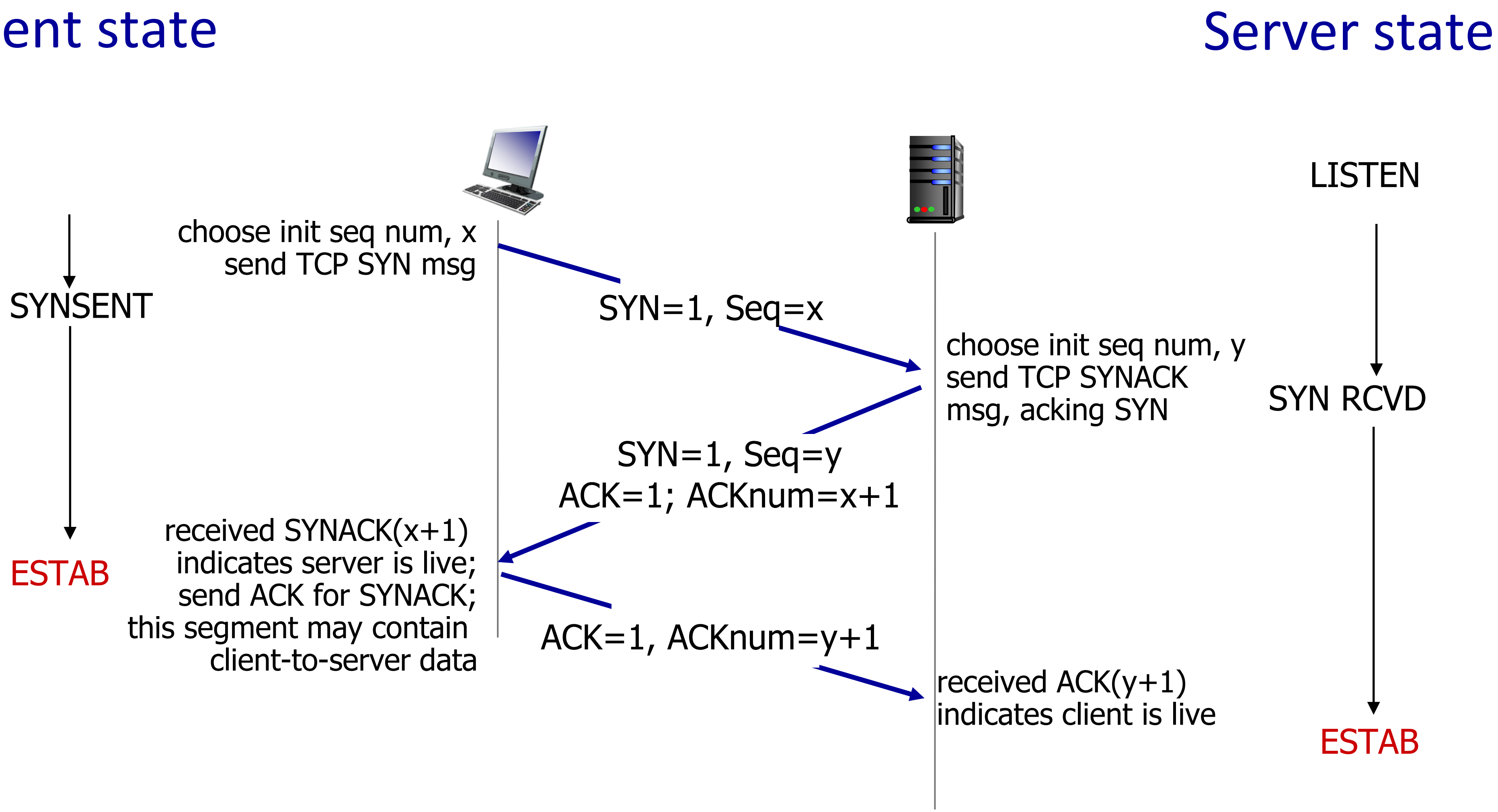
req_conn(x)

ESTAB

data(x+1)

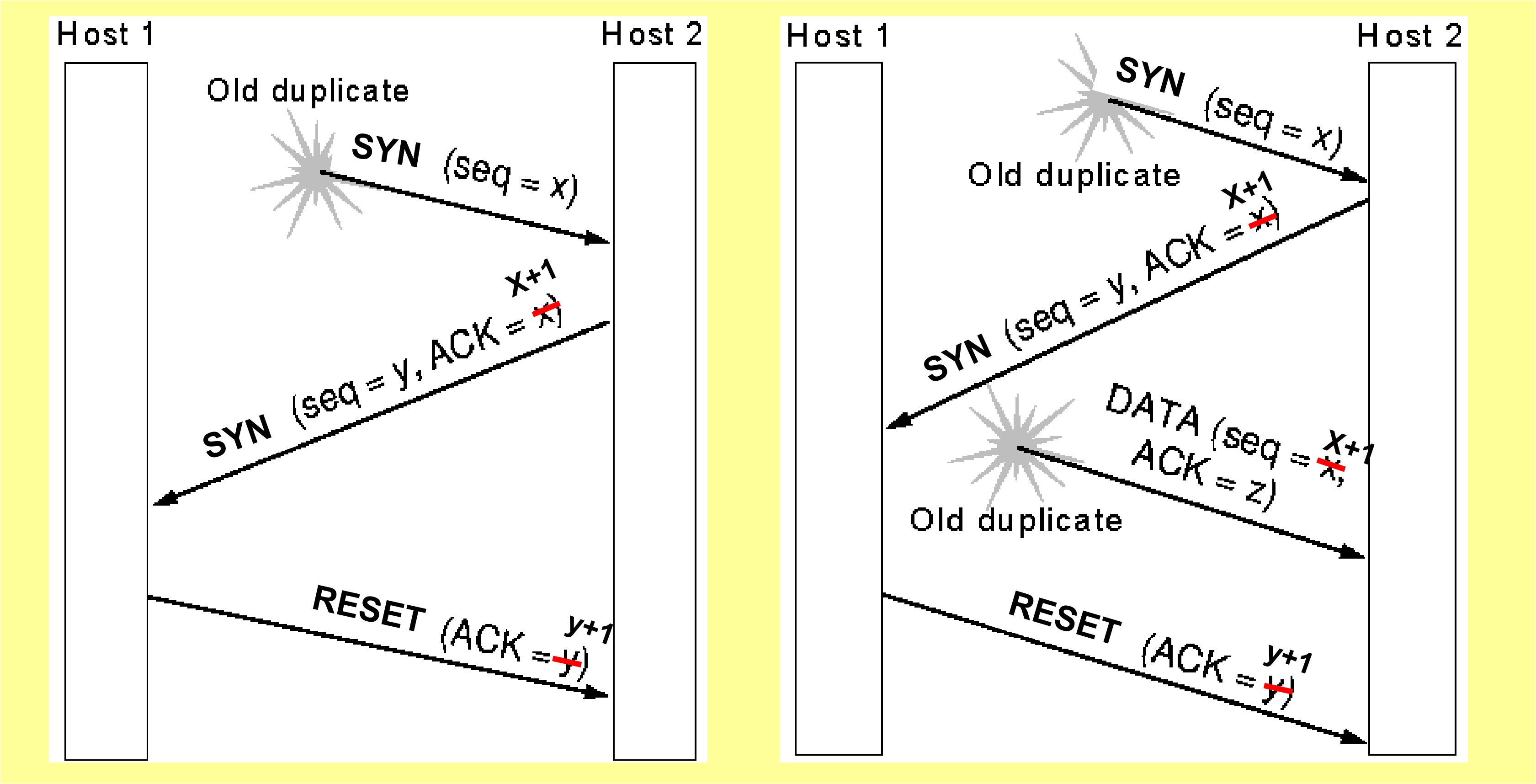
accept data(x+1)

TCP 3-way handshake



TCP 3-way handshake

- Three-way handshake : against abnormal cases

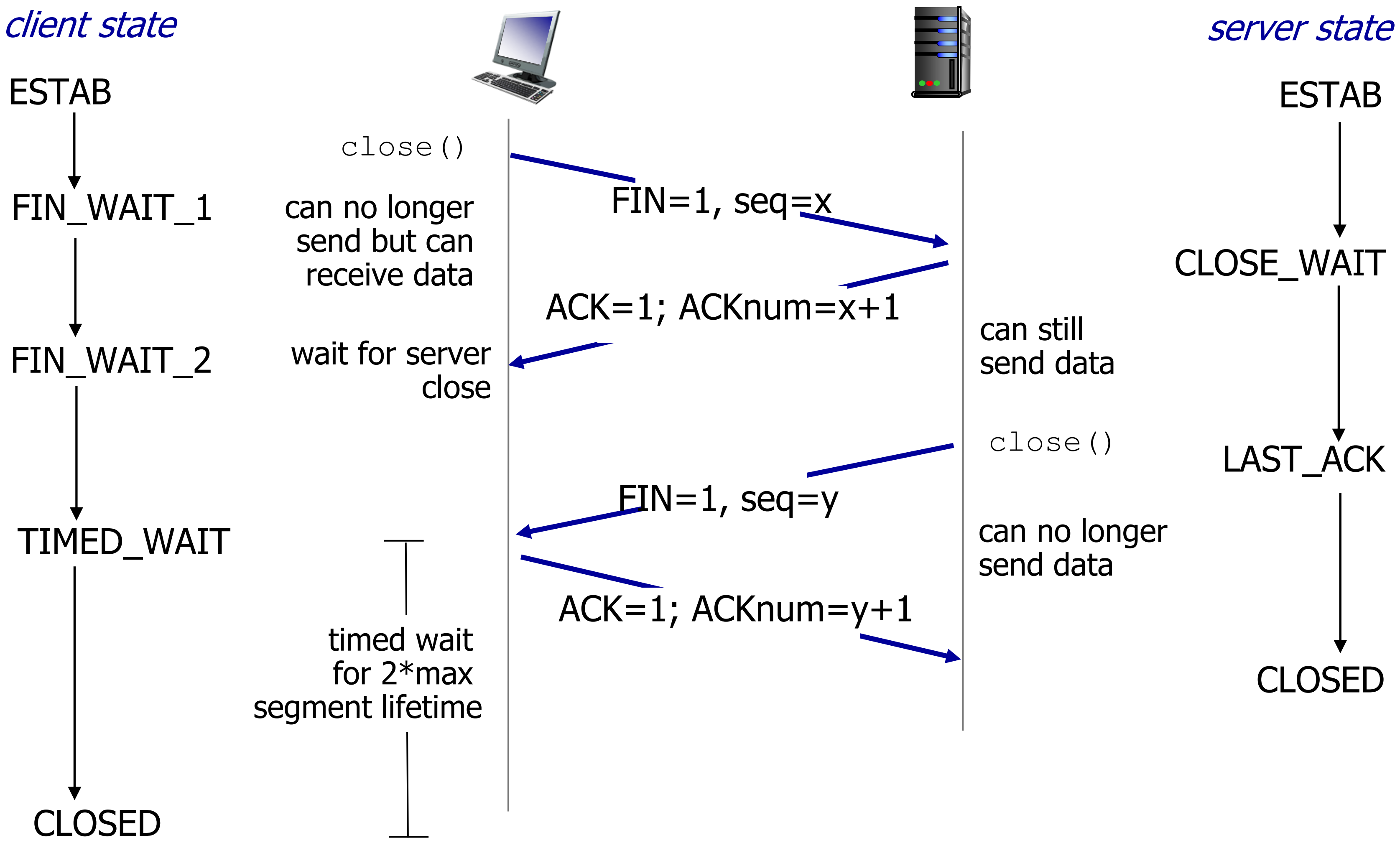


Closing a TCP connection

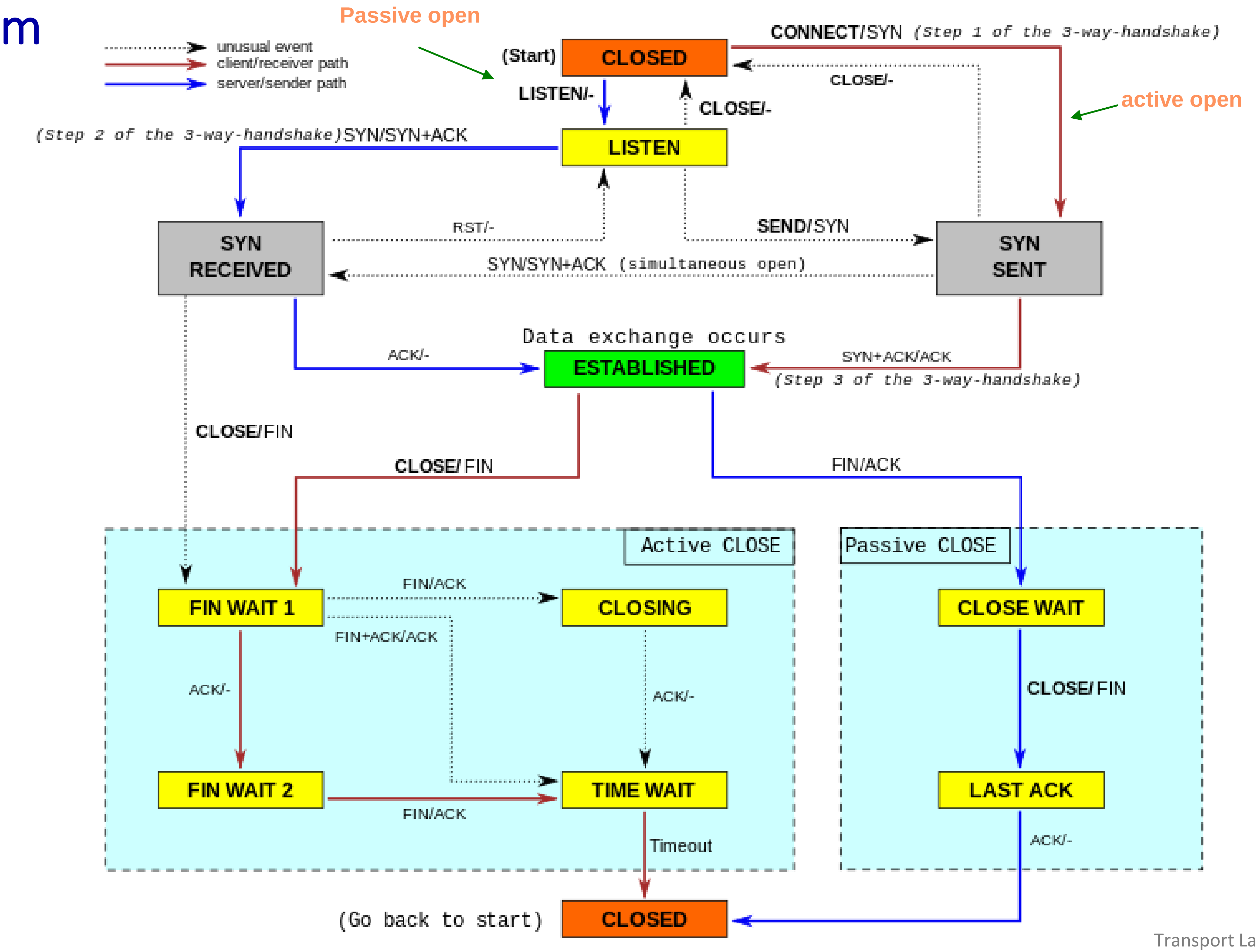
- client, server each close their side of connection
  - send TCP segment with FIN bit = 1
- respond to received FIN with ACK
  - on receiving FIN, ACK can be combined with own FIN
- simultaneous FIN exchanges can be handled

Closing a TCP connection

- Performed separately in each direction.



TCP: State Diagram



FIN WAIT 1: The client has said it is finished.
FIN WAIT 2: The server has agreed to release (half close)

TIME WAIT : Wait for all packets to die off.

TCP: State Transition Diagram

- TIME_WAIT state
  - wait
    - wait for final segment to be transmitted before releasing connection
    - Implementation-dependent (typically 30 sec, 60 sec, 120 sec)
  - 2MSL wait protects against delayed segments from the previous “incarnation” of the connection.
  - 2MSL effects
    - Socket *pair* cannot be
      - If you kill a client and restart, it will get
      - If you kill a server and restart, you may get a bind error.