Computer Networks
COL 334/672

TCP Reliability and Connection Establishment

Slides adapted from KR

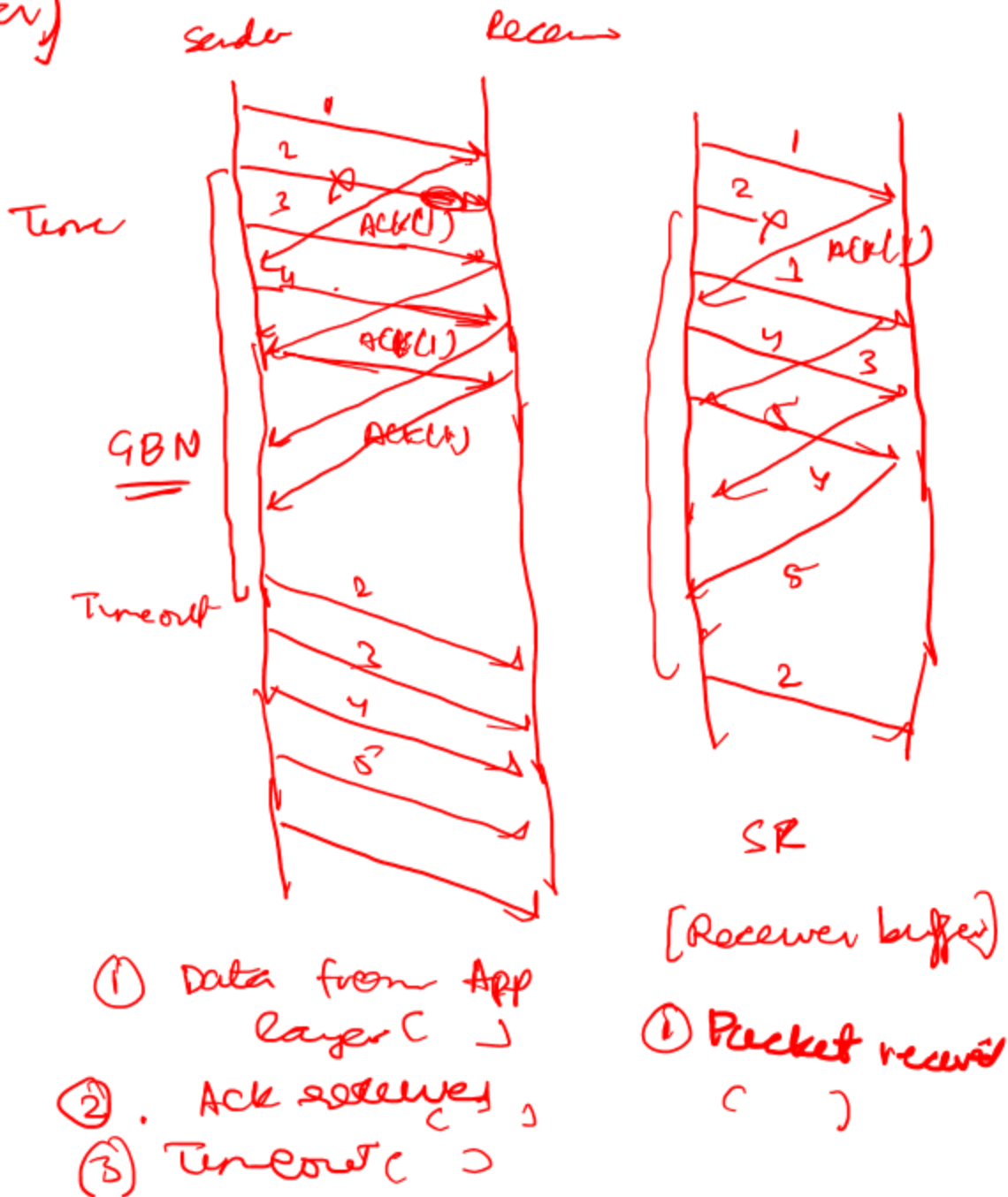
Sem 1, 2025-26

Recap

[Reliability in Transport layer]

(Negative Acknowledgement)

- Sliding window protocols are more efficient than simple stop-and-wait
  - Go-back-N -- cumulative ACKs
    - simpler mechanism at the receiver, but unnecessary retransmissions
  - Selective Repeat – each packet is acked individually
- Event-driven paradigm
- This class: How does TCP implement reliability



How does TCP implement reliability?

Maximum Transmission Unit (MTU)

TCP: Maximum Segment Size (MSS)

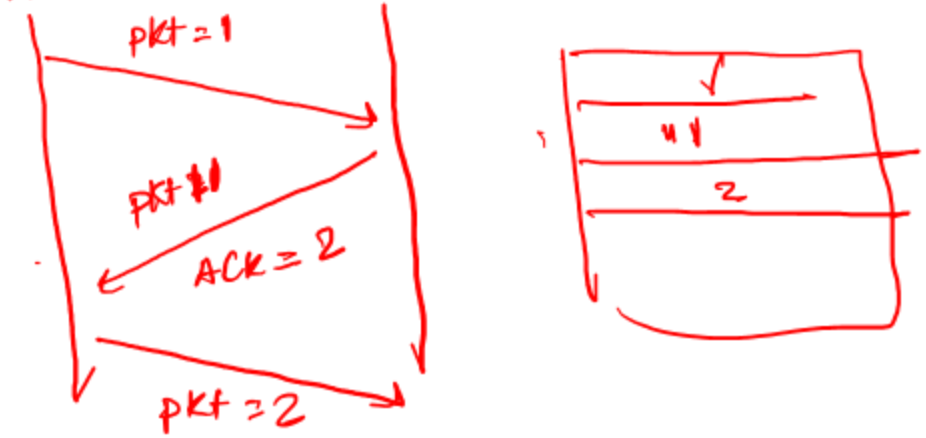
FACT

TCP is bidirectional

(Client Sq #s)

Client

Server (Sq # server)



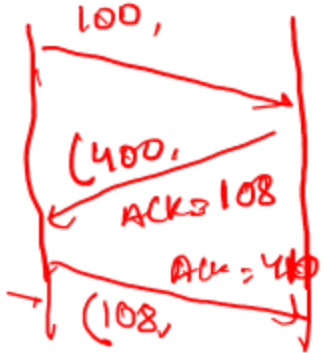
FACT

Byte-stream protocol

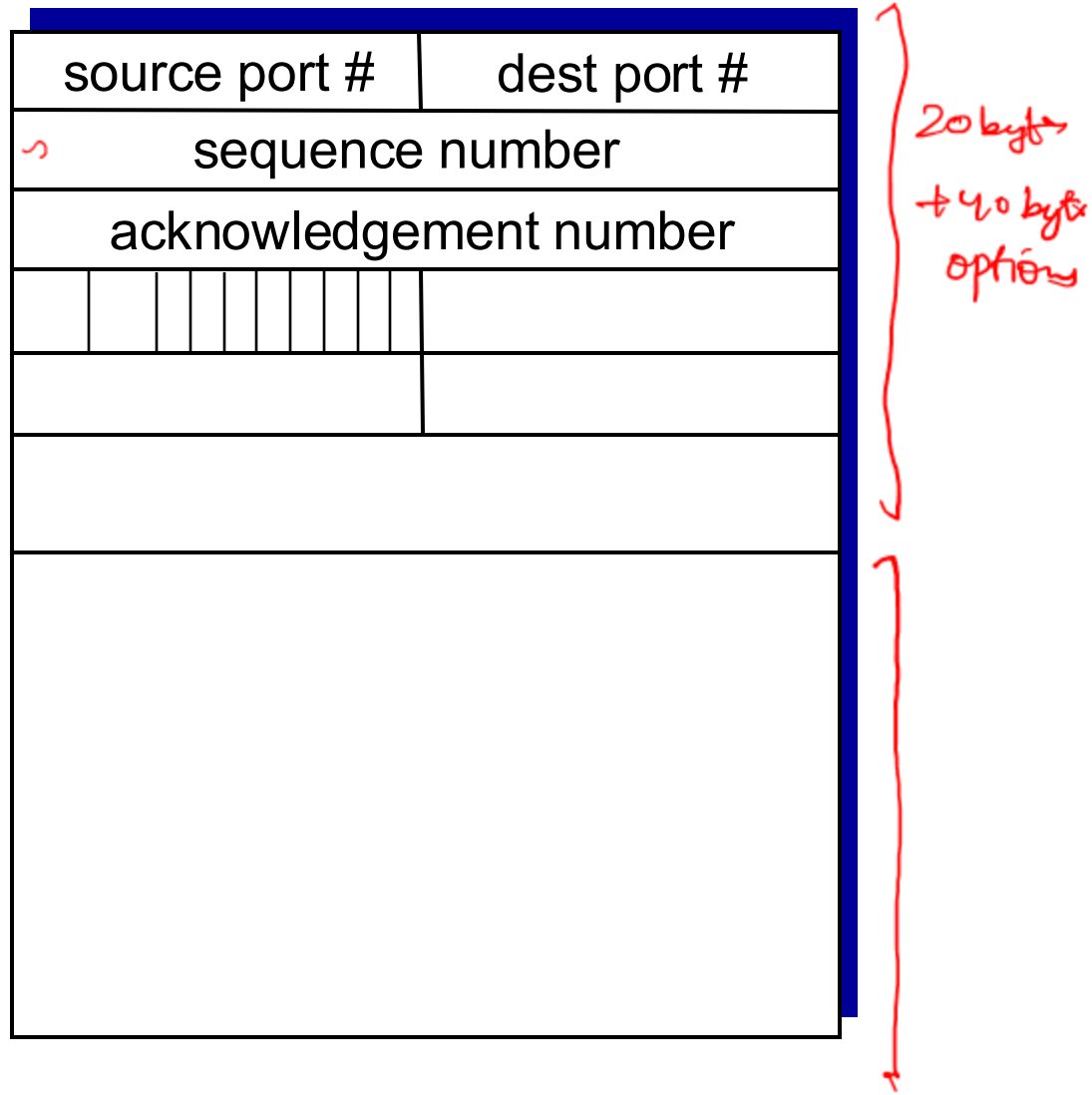
Sequence #s: bytes

sock.send("Hello")
sock.send("world")
Hello World
Hello world

sq # = 100
L = 8

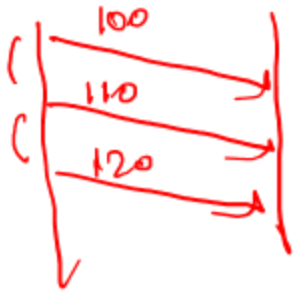


sq # = 400
L = 10



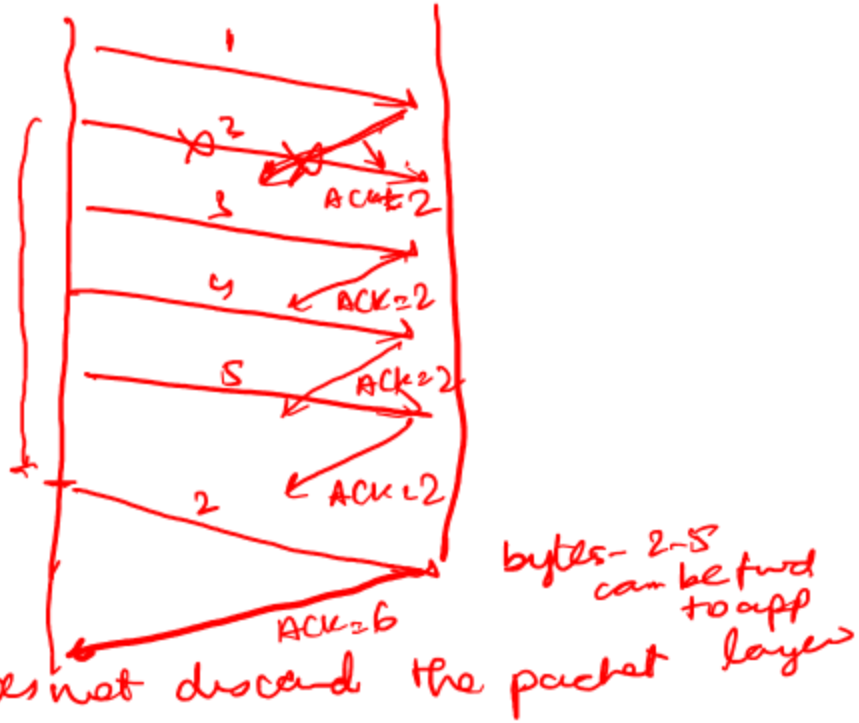
Reliability in TCP

- Uses a sliding window protocol



- ACKs are cumulative

- Specification vary for handling out-of-order delivery
  - Generally, out of order delivery packets within the window are not discarded



TCP is bursty
ws: Maximum # of unacked bytes

Optimization #1: Delayed ACKs

- Reduce the number of ACKs by sending 1 ACK every 2 packets

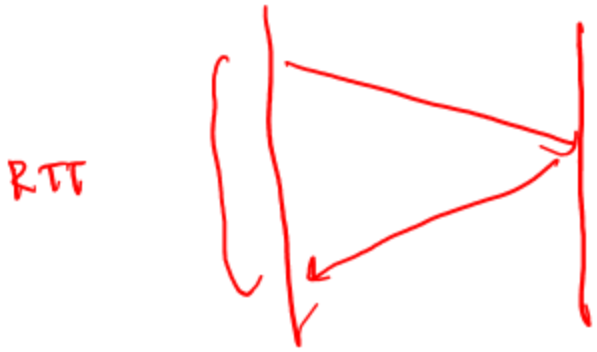
Event at receiver	*TCP receiver action*

How to Set Timeout?

Premature timeout : Unnecessary Retx

Long timeout : Higher latency

Timeout ≥ RTT

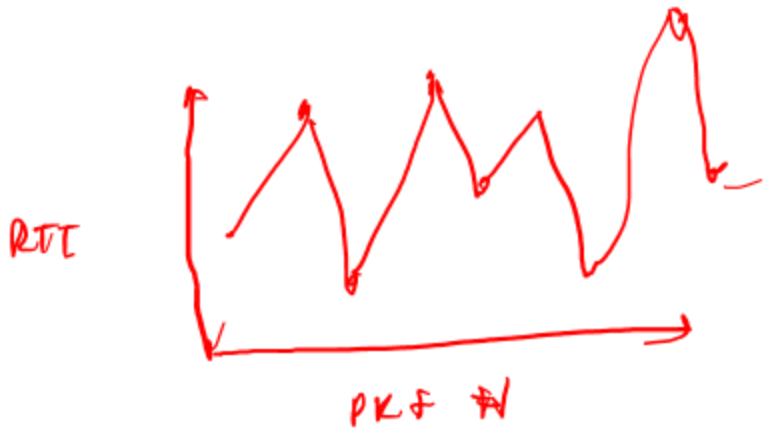


Last 10 packet

last packet RTT

T_o = 1 Second

T_o = RTT + safety margin

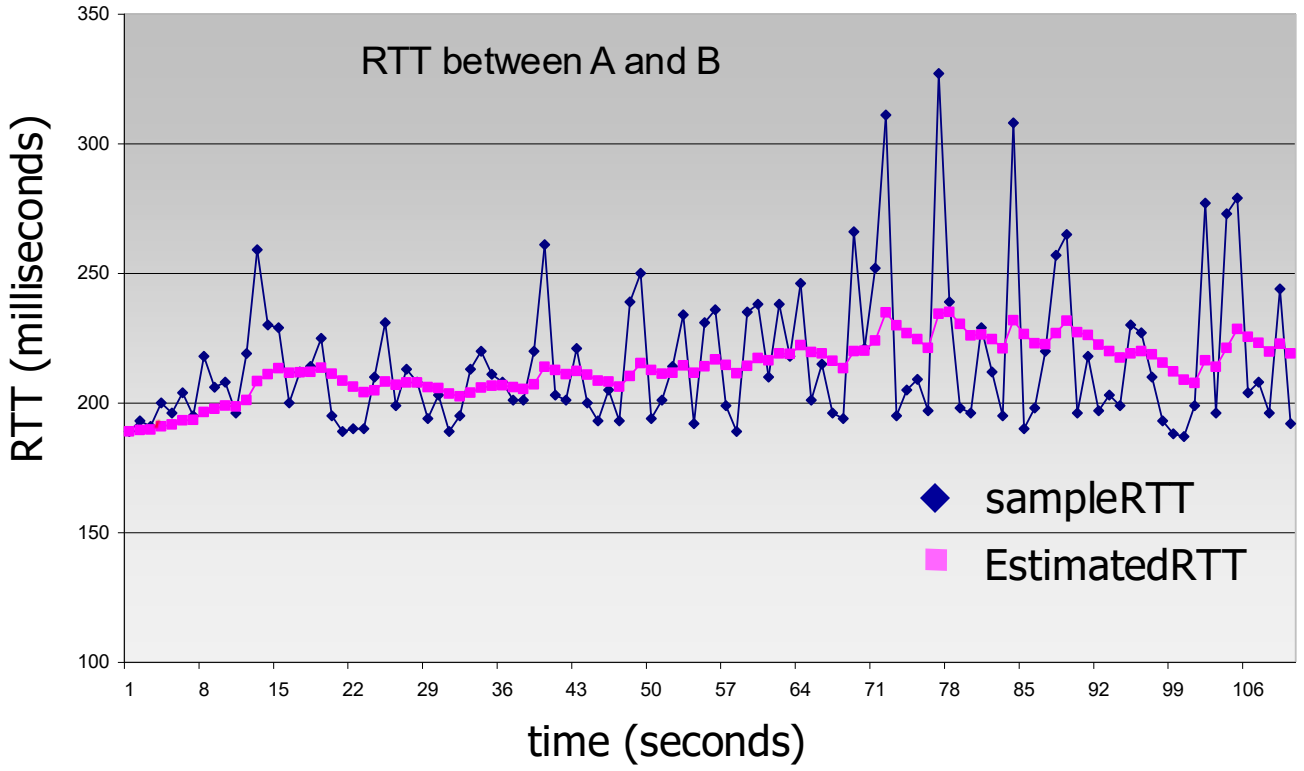


TCP round trip time, timeout

EstimatedRTT = (1- α)*EstimatedRTT + α*SampleRTT

- exponential weighted moving average (EWMA)
- influence of past sample decreases exponentially fast
- typical value: α = 0.125

New RTT_E = (old RTT_E (1−α)) + α Sample RTT



TCP round trip time, timeout

- timeout interval: **EstimatedRTT** plus "safety margin"
  - large variation in **EstimatedRTT**: want a larger safety margin

TimeoutInterval = EstimatedRTT + 4*DevRTT

estimated RTT

"safety margin"

- **DevRTT**: EWMA of (**SampleRTT** deviation from **EstimatedRTT**):

DevRTT = (1-β)*DevRTT + β*|SampleRTT-EstimatedRTT|

EWMA

(typically, β = 0.25)

Timeouts take too long!

- Optimization #2:

3 Duplicate ACK
same data has reached

3 duplicate ACKs
↳ Retransmit

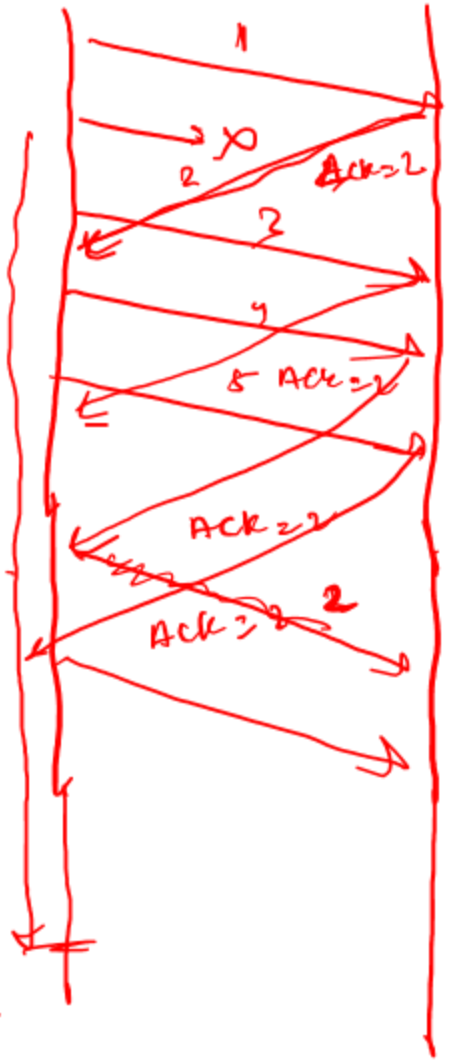
Fast Retransmit

TCP fast retransmit

if sender receives 3 additional ACKs for same data ("triple duplicate ACKs"), resend unACKed segment with smallest seq #

- likely that unACKed segment lost,

TCP options : allow selective Acks



Once there is a timeout
: Double Timeout value (Binary Exp Backoff)

Timeouts take too long!

- Optimization #2:

TCP fast retransmit

if sender receives 3 additional ACKs for same data ("triple duplicate ACKs"), resend unACKed segment with smallest seq #

- likely that unACKed segment lost, so don't wait for timeout

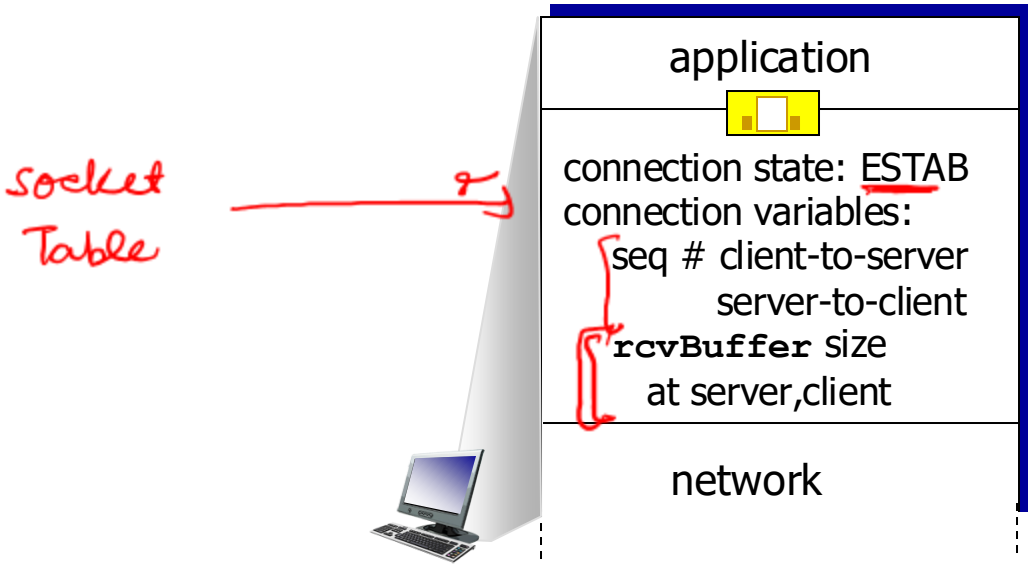
Does TCP implement GBN or SR?

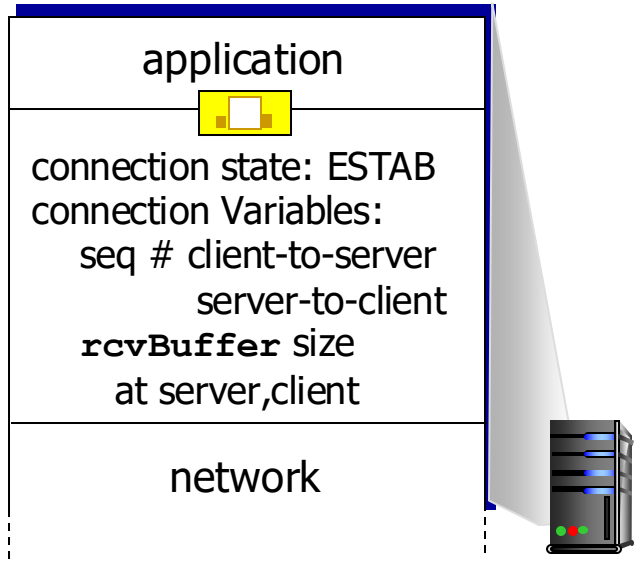
Hybrid

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)



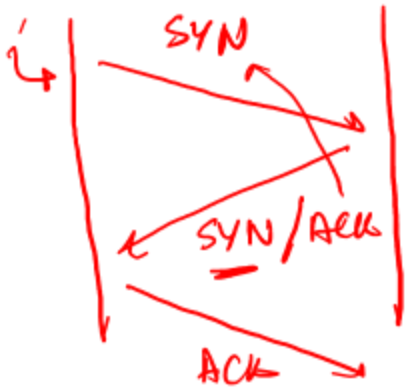



```
Socket clientSocket =
  newSocket("hostname","port number");
```

```
Socket connectionSocket =
  welcomeSocket.accept();
```

Protocol for connection establishment?

Two-way Handshake



3-way Handshake

① Both need to agree on the connection

② Intial Seq #s Need to be exchanged

Challenges

[Delayed Syn / Ack that are roaming around in the network]

② Randomized ISN

[wall clock time]

TCP 3-way handshake

Client state

```
clientSocket = socket(AF_INET, SOCK_STREAM)
```

LISTEN

```
clientSocket.connect((serverName,serverPort))
```

choose init seq num, x
send TCP SYN msg

SYNSENT

SYNbit=1, Seq=x

choose init seq num, y
send TCP SYNACK
msg, acking SYN

SYNbit=1, Seq=y
ACKbit=1; ACKnum=x+1

received SYNACK(x)
indicates server is live;
send ACK for SYNACK;
this segment may contain
client-to-server data

ESTAB

ACKbit=1, ACKnum=y+1

received ACK(y)
indicates client is live

Server state

```
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind(('',serverPort))
serverSocket.listen(1)
connectionSocket, addr = serverSocket.accept()
```

LISTEN

SYN RCVD

ESTAB