CN-Basic
L27

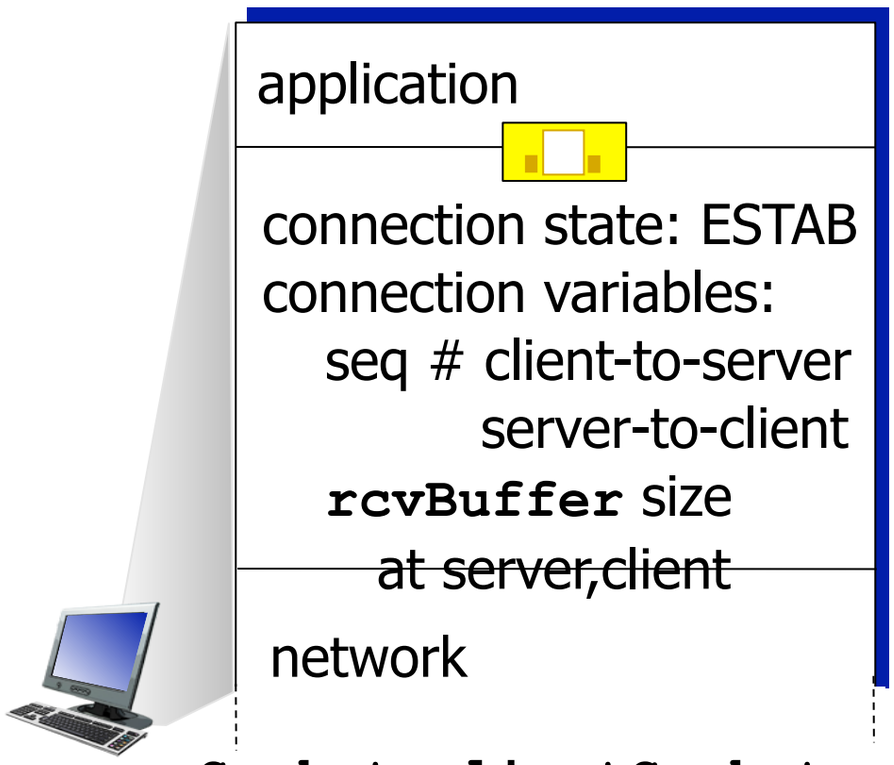
TCP FSM
&
RTT Estimation

Dr. Ram P Rustagi
rprustagi@ksit.edu.in
http://www.rprustagi.com
https://www.youtube.com/rprustagi

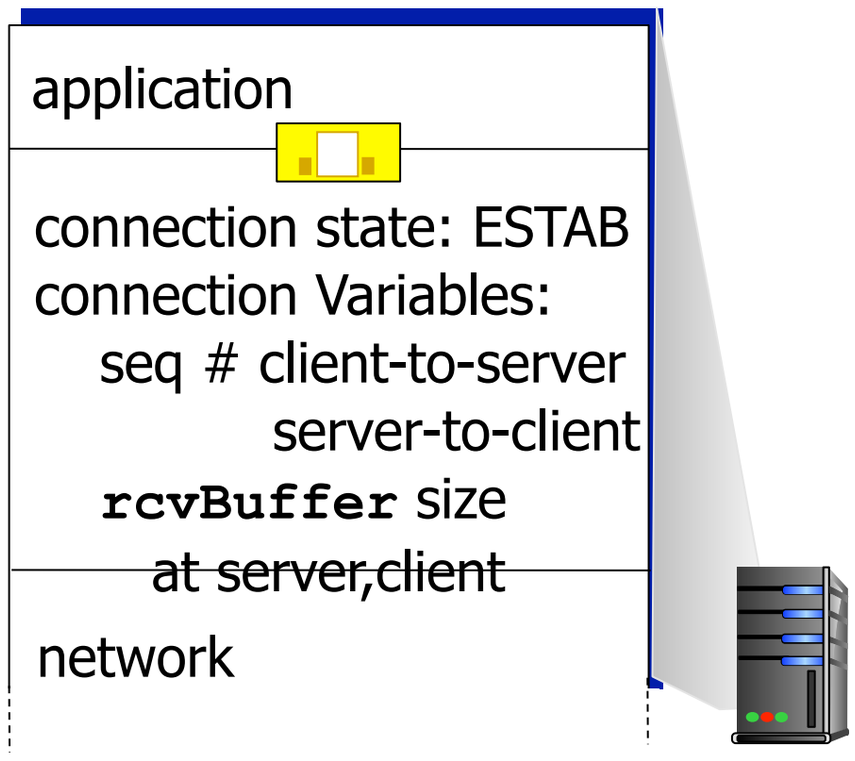
Connection Management

Before exchanging data, sender/receiver "handshake":

- Agree to establish connection (each knowing the other willing to establish connection)
- Agree on connection parameters



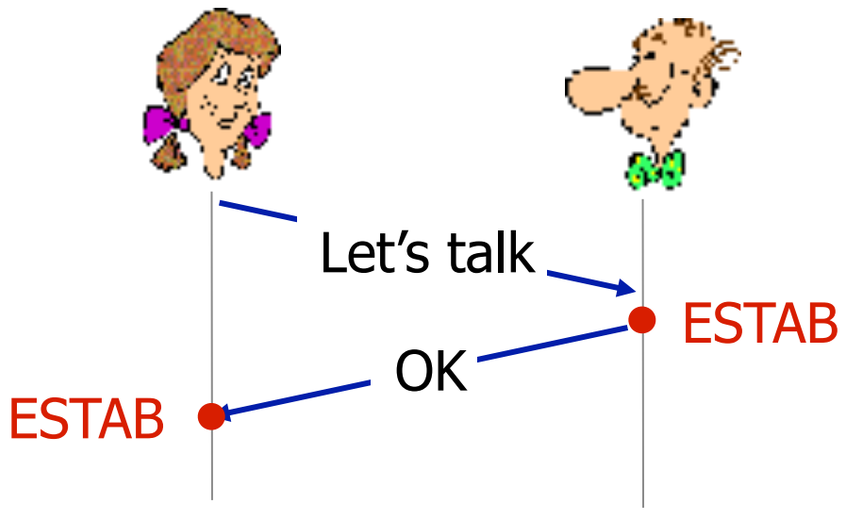

```
Socket clientSocket =
  newSocket("hostname","port
  number");
```

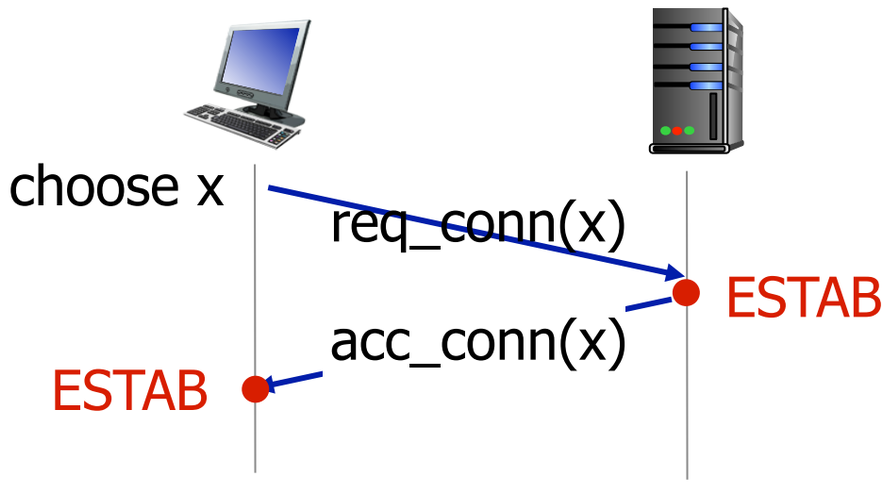



```
Socket connectionSocket =
  welcomeSocket.accept();
```

Agreeing to establish a connection

2-way handshake:



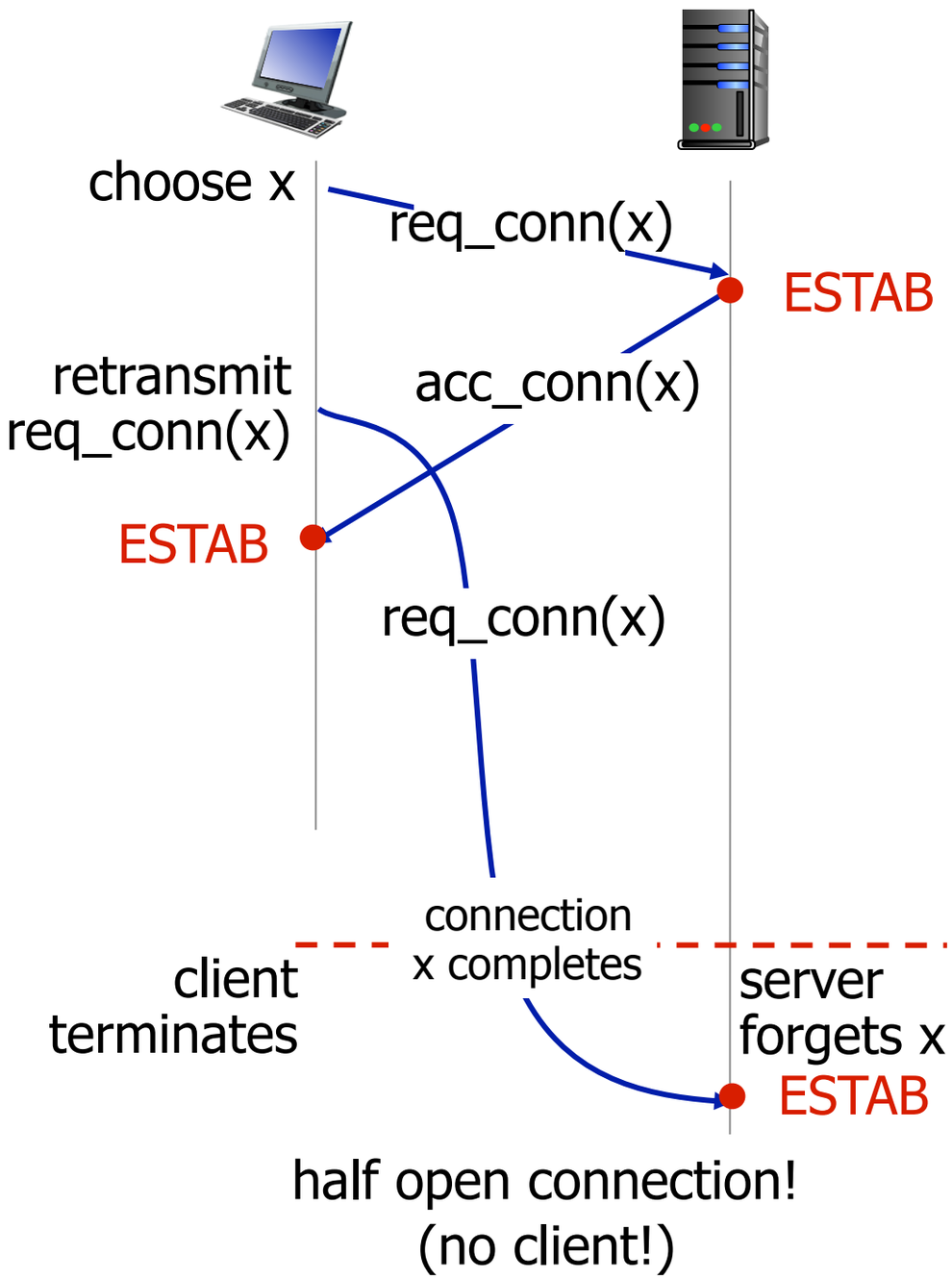


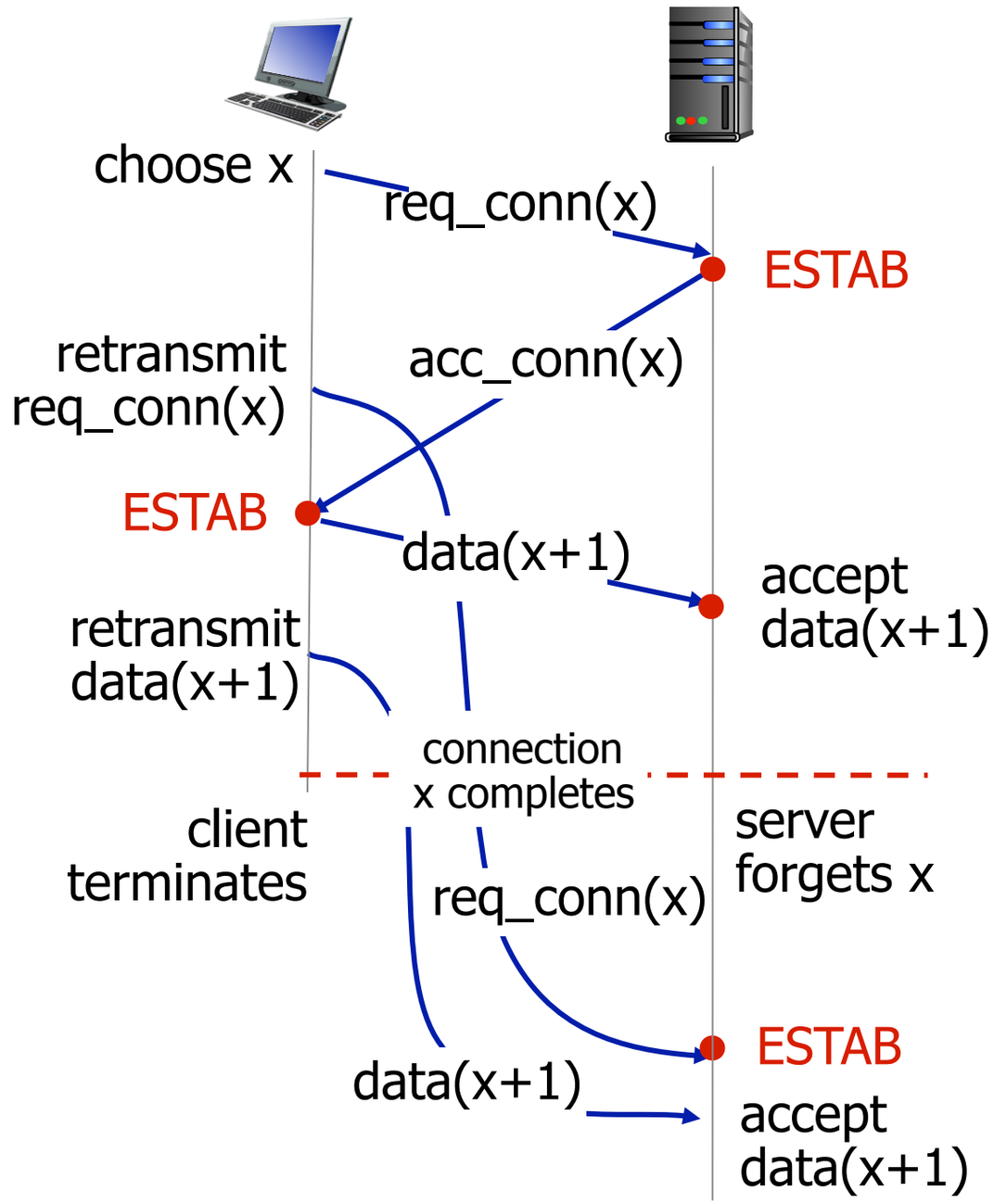
Q: will 2-way handshake always work in network?

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

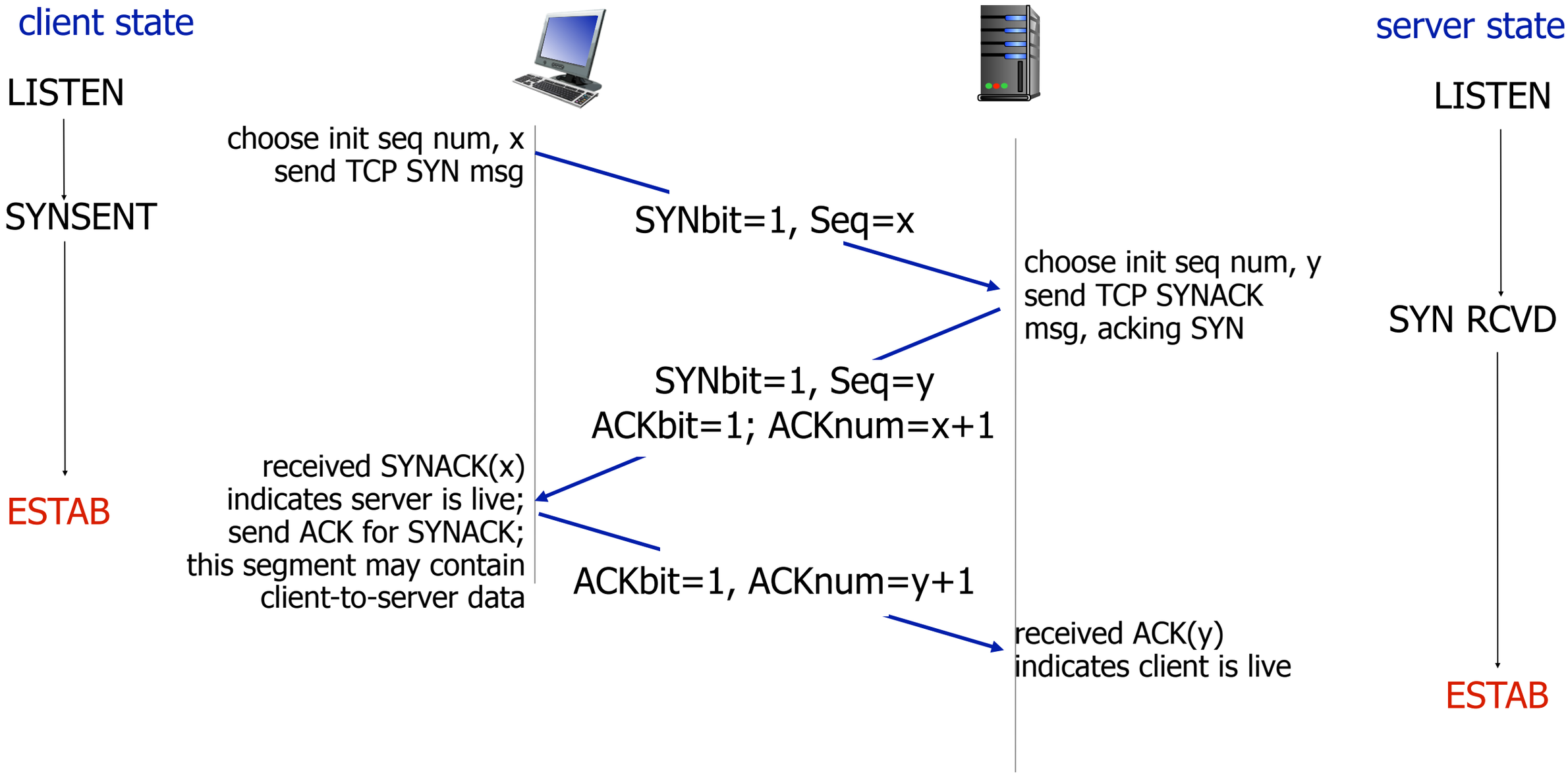
Agreeing to establish a connection

2-way handshake failure scenarios:

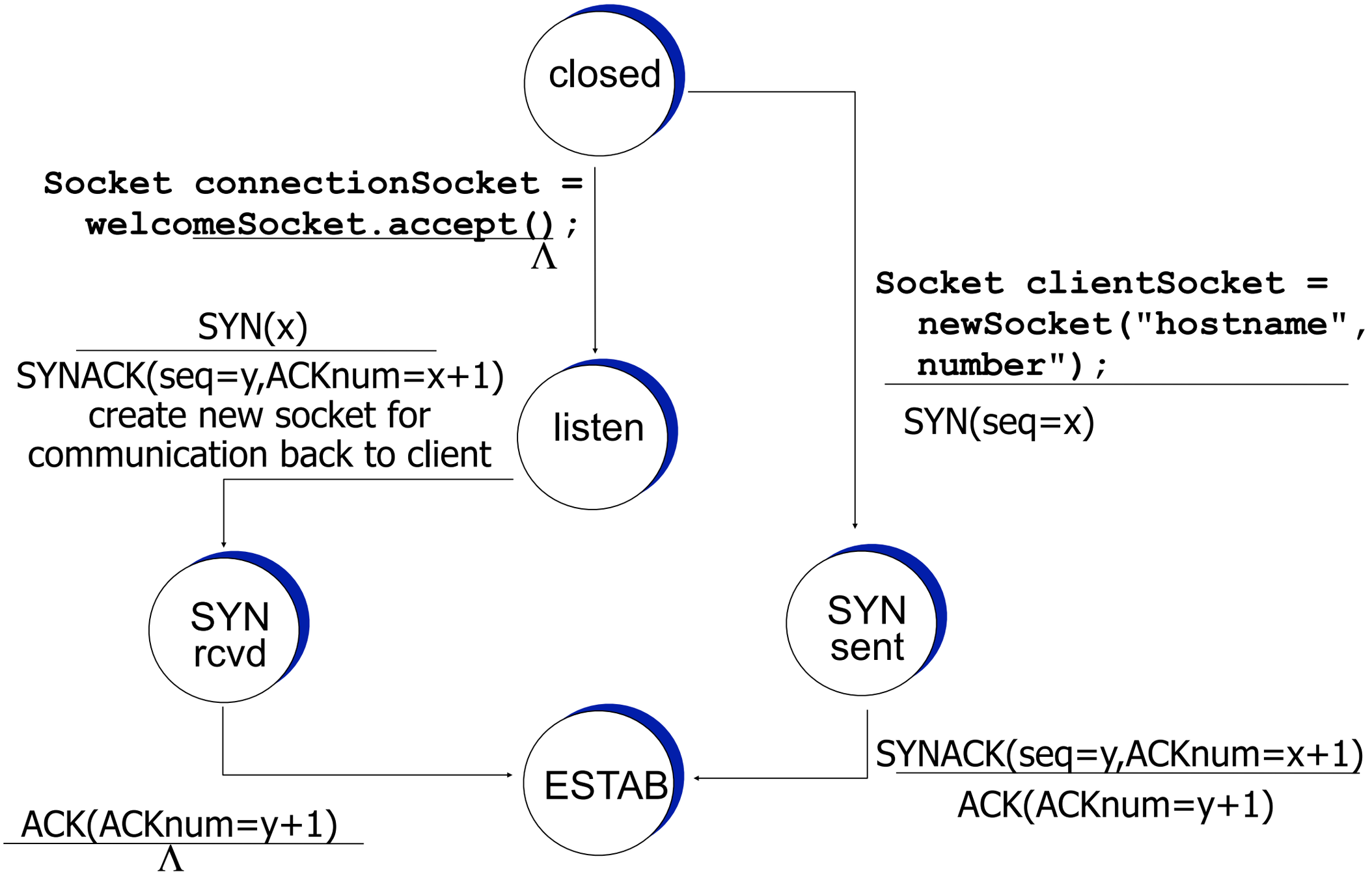




TCP Connection Setup: 3-way handshake



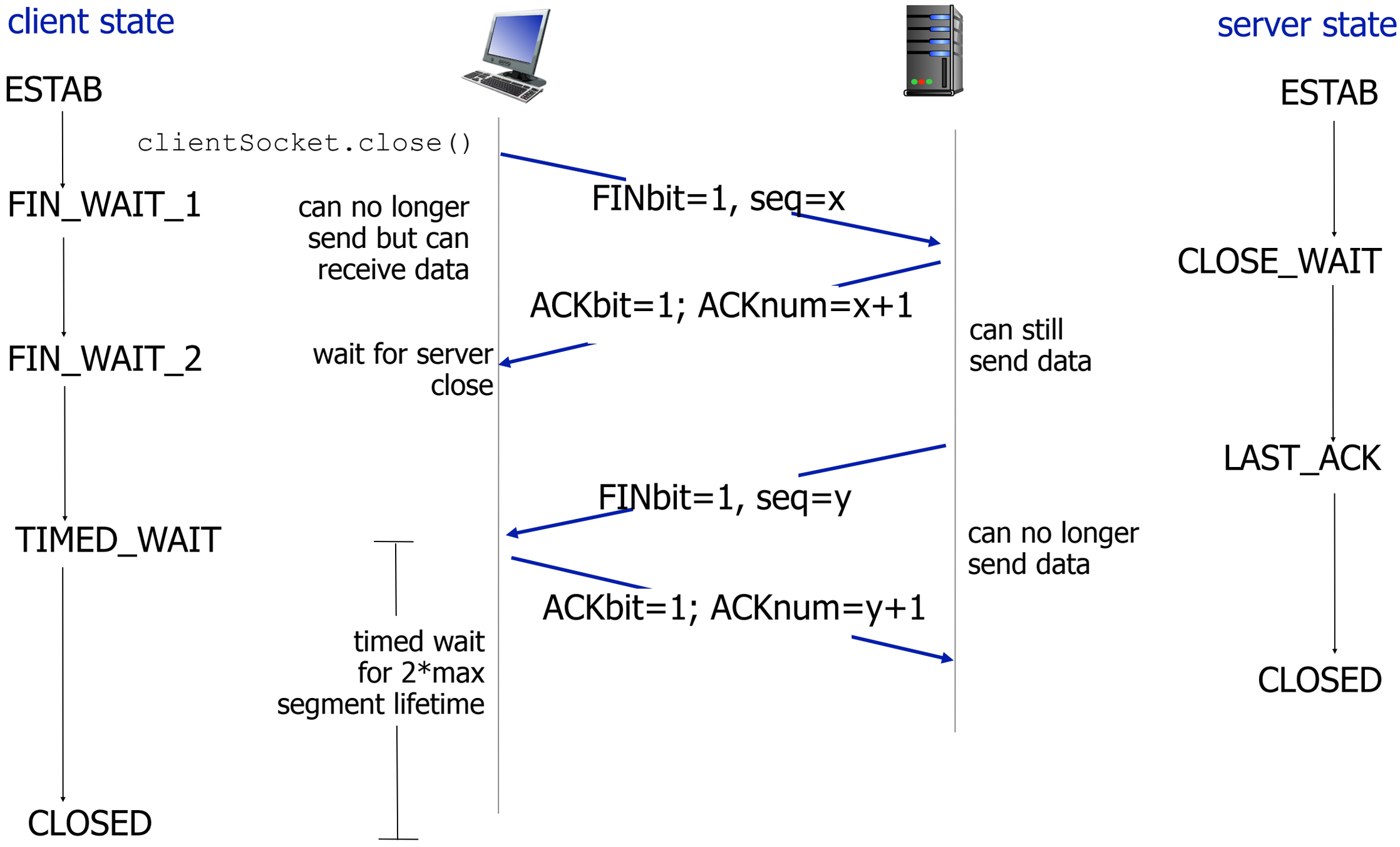
TCP 3-way handshake: FSM

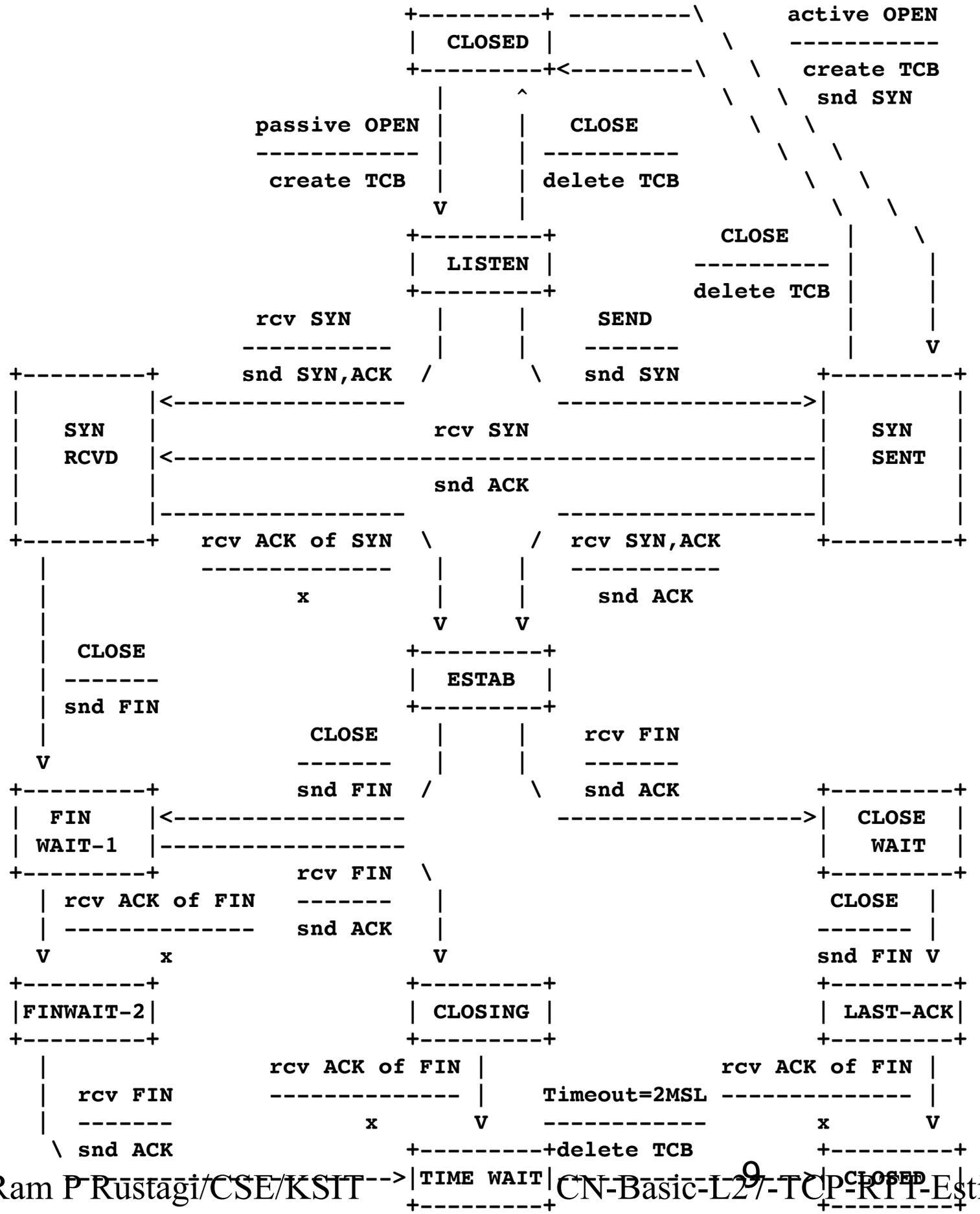


TCP: closing a connection

- 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

TCP: closing a connection





TCP State Transition Diagram
src: RFC 793

TCP round trip time, timeout

Q: how to set TCP timeout value?

- Longer than RTT
  - What if shorter than RTT
- Is RTT constant?
  - Varies per segment
- T*oo short*: premature timeout, unnecessary retransmissions
- T*oo long*: slow reaction to segment loss

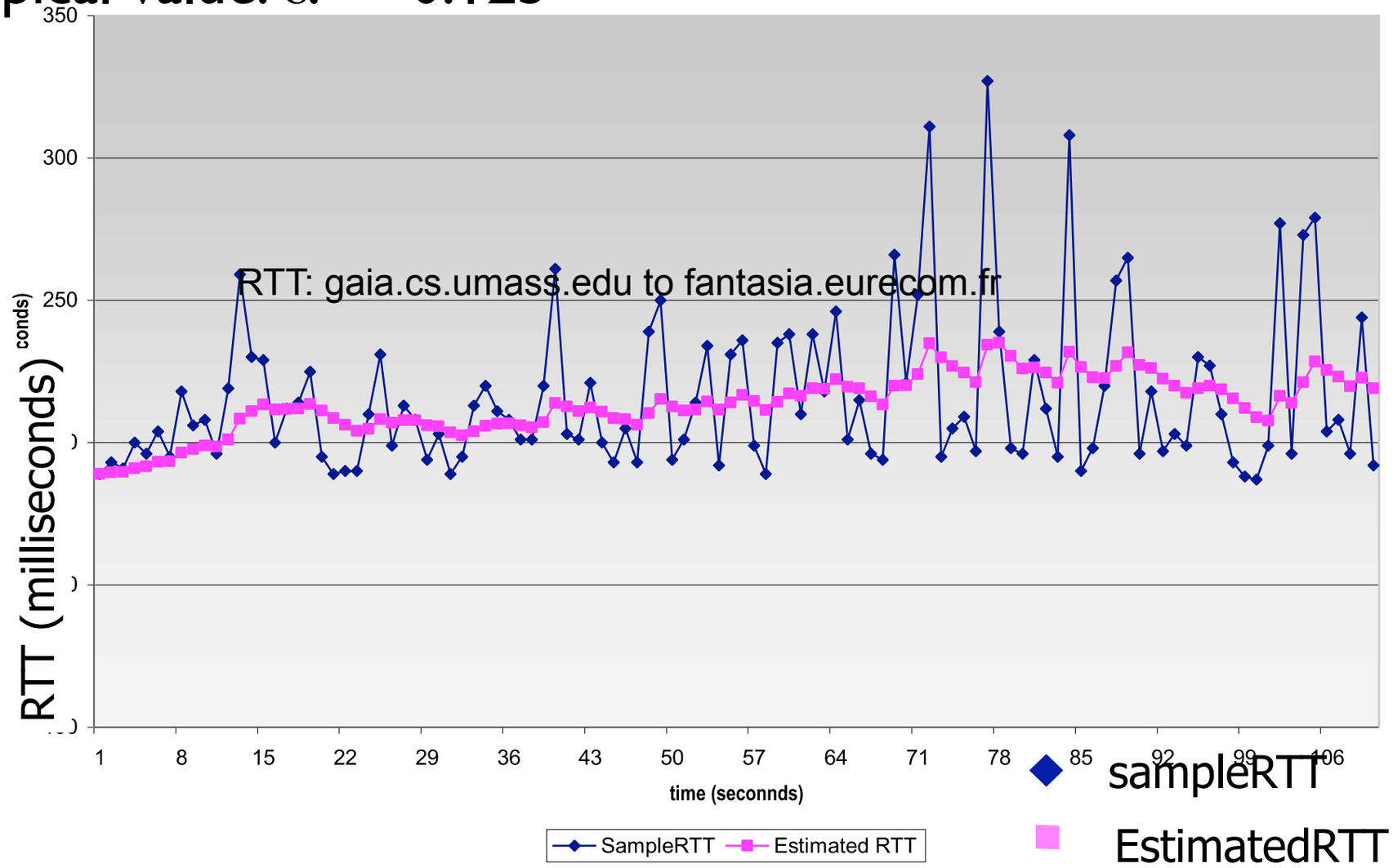
Q: how to estimate RTT?

- **`SampleRTT`**: measured time from segment transmission until ACK receipt
- Ignore retransmissions
- One sampleRTT measured at a time
- **`SampleRTT`** will vary
  - Why?
- Want estimated RTT "smoother"
- Average several *recent* measurements, not just current **`SampleRTT`**
- Uses **`estimatedRTT`**

TCP round trip time, timeout

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

- exponential weighted moving average **(RFC 6298)**
- influence of past sample decreases exponentially fast
- typical value: α = 0.125



between gaia.cs.umass.edu and fantasia.eurecom.fr

TCP round trip time, timeout

- Timeout interval: **`EstimatedRTT`** plus "safety margin"
  - ? Large variation in **`EstimatedRTT`** -> larger safety margin
- Estimate SampleRTT deviation from EstimatedRTT:

$$\texttt{DevRTT} = (1-\beta)*\texttt{DevRTT} + \beta*|\texttt{SampleRTT}-\texttt{EstimatedRTT}|$$

(typically, $\beta = 0.25$)

$$\texttt{TimeoutInterval} = \texttt{EstimatedRTT} + 4*\texttt{DevRTT}$$

estimated RTT

"safety margin"

Note: TimeoutInterval is doubled when timeout occurs
Computed again when ack is received for segment

Example RTT computation

- http://gaia.cs.umass.edu/kurose_ross/interactive/TCP_RTT.php
- Case scenario:
  - `estimatedRTT` = 220ms, `DevRTT` = 33ms
  - next 3 measured RTTs are: 280ms, 390ms, 200ms
  - value of $\alpha = 0.125$, $\beta = 0.25$
- Compute new values of `estimatedRTT`, `DevRTT` and TCP timeout for each of 3 measured RTT

SYN Flood Attack

- 3-way handshake allows DoS attack on server
- Attacker sends a large number SYN packets
  - Without corresponding ACK packets
  - Server resources are allocated (but never used)
- Defense is done using SYN-Cookies
  - Server creates its ISN for SYN using some algo
    - Hash fn. of src/dstn IP/Port + secret number
    - Computed value is called cookie and sent
      - As part of SYN-ACK packet
- Legitimate client will return ACK
  - Server verifies the ACK from its computed value
- Attacker client only ends up wasting computing
  - No harm done, no resources allocated