COMPUTER COMMUNICATION NETWORKS

Department of Electronics and
Communication Engineering

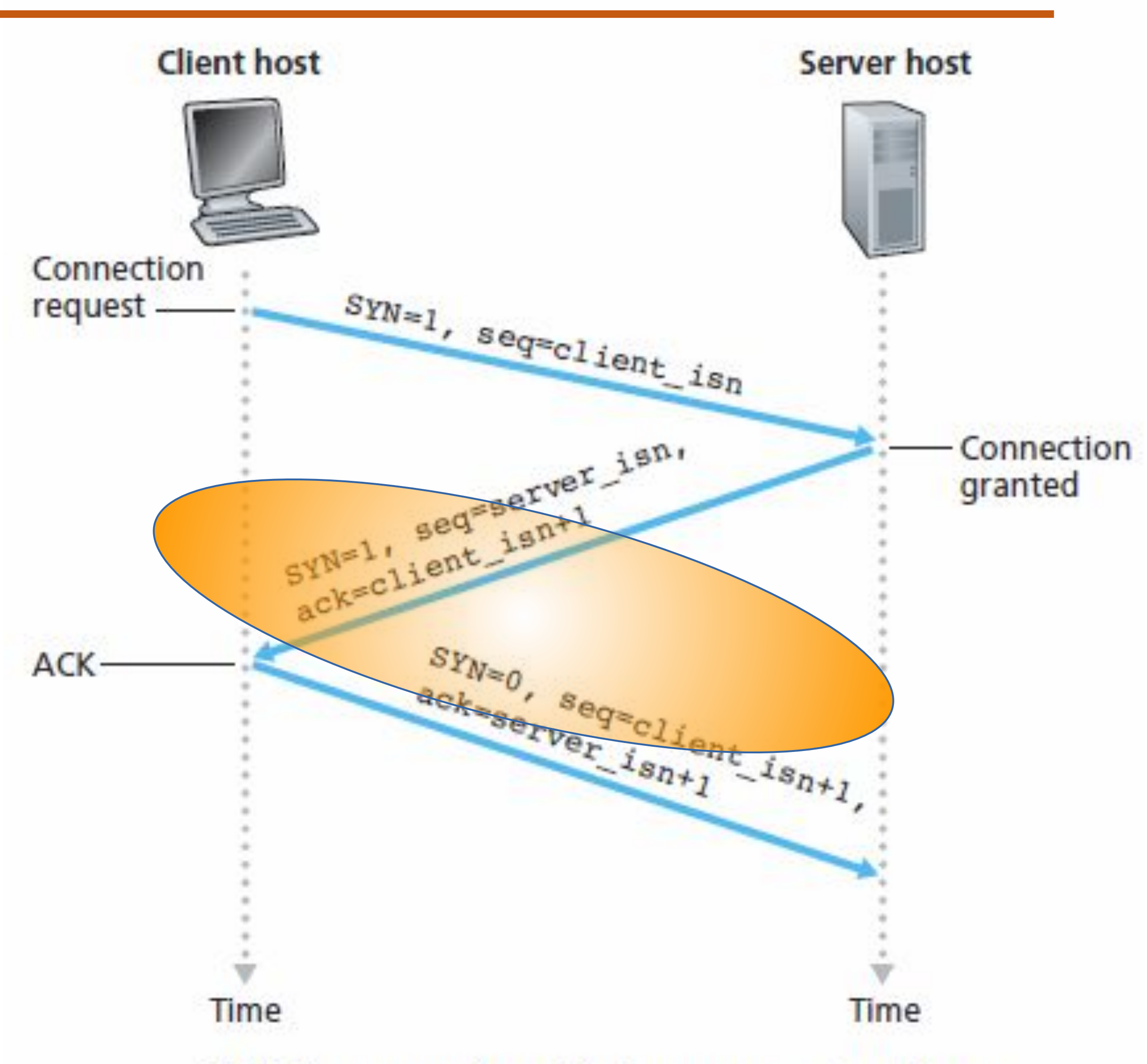
COMPUTER COMMUNICATION NETWORKS

TCP Connection Opening-Closing, Timeout

Dr. Arpita Thakre
Department of Electronics and Communication Engineering

COMPUTER COMMUNICATION NETWORKS

TCP three-way handshake



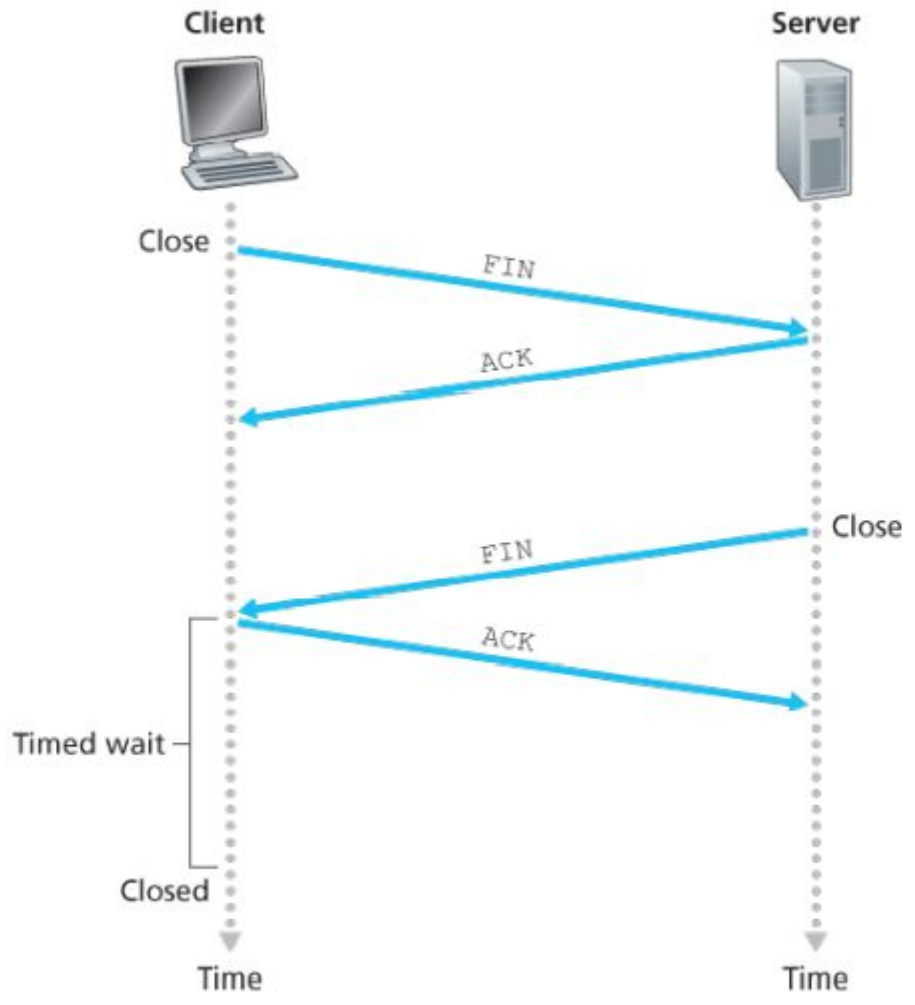
TCP three-way handshake: segment exchange

TCP Connection Closing/Termination

- 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

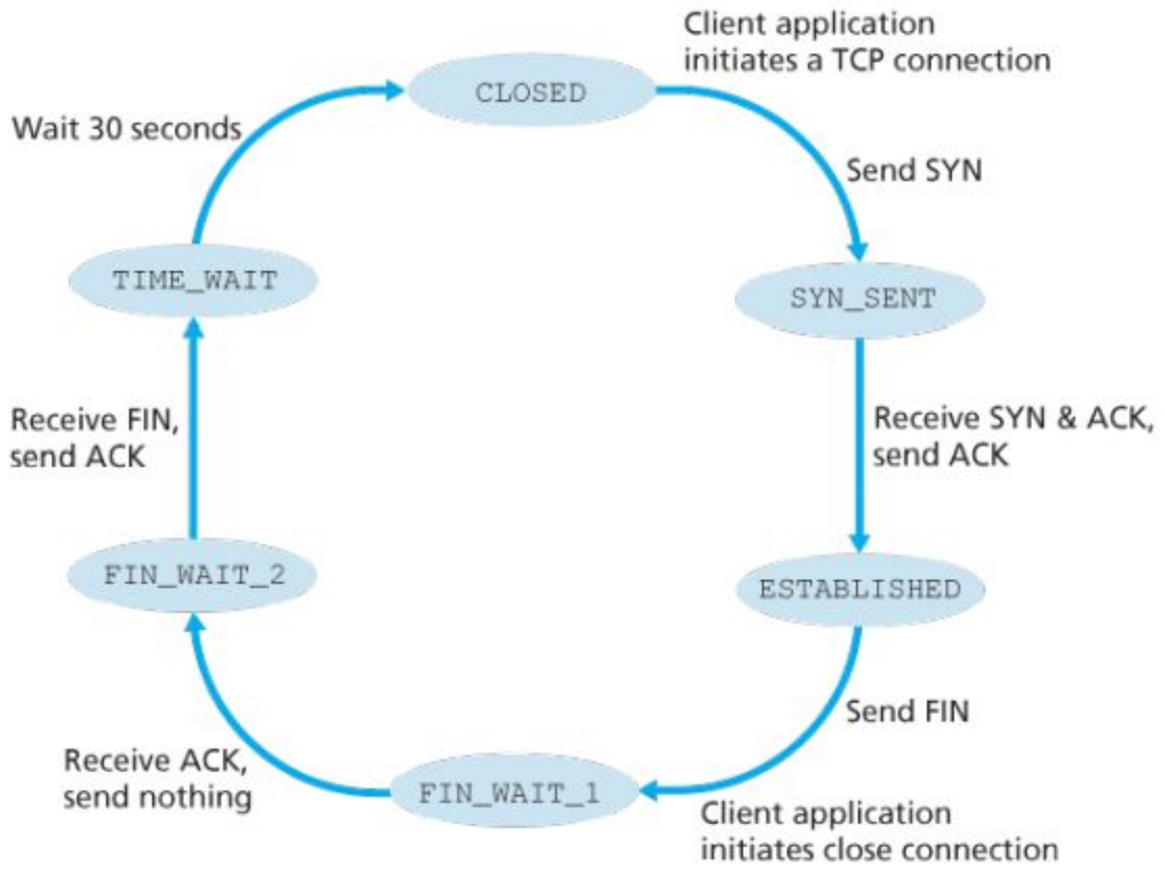
COMPUTER COMMUNICATION NETWORKS

TCP Connection Closing

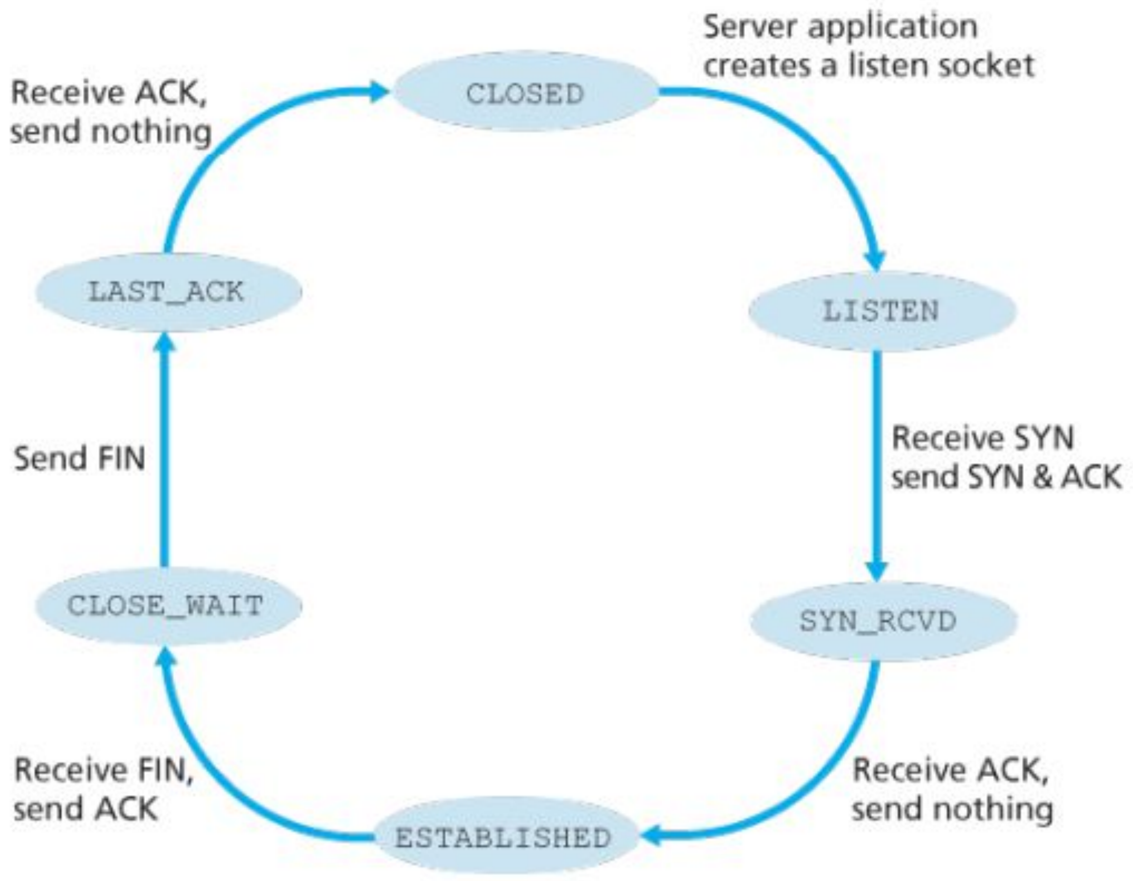


At this point, all the resources in the two hosts are now deallocated.

A typical sequence of TCP states visited by a client TCP


CLOSED
Client application initiates a TCP connection
Send SYN
SYN_SENT
Receive SYN & ACK, send ACK
ESTABLISHED
Send FIN
Client application initiates close connection
FIN_WAIT_1
Receive ACK, send nothing
FIN_WAIT_2
Receive FIN, send ACK
TIME_WAIT
Wait 30 seconds

A typical sequence of TCP states visited by a server-side TCP


Server application creates a listen socket
CLOSED
LISTEN
Receive SYN send SYN & ACK
SYN_RCVD
Receive ACK, send nothing
ESTABLISHED
Receive FIN, send ACK
CLOSE_WAIT
Send FIN
LAST_ACK
Receive ACK, send nothing

TCP : RTT & Timeout

Q: how to set TCP timeout value?

- longer than RTT, but RTT varies!
- *too short:* premature timeout, unnecessary retransmissions
- *too long:* slow reaction to segment loss

Q: how to estimate RTT?

- SampleRTT:measured time from segment transmission until ACK receipt
  - ignore retransmissions
- SampleRTT will vary, want estimated RTT "smoother"
  - average several *recent* measurements, not just current SampleRTT

TCP : Sender Events

event: data received from application

- create segment with seq #
- seq # is byte-stream number of first data byte in segment
- start timer if not already running
  - think of timer as for oldest unACKed segment
  - expiration interval: `TimeOutInterval`

event: timeout

- retransmit segment that caused timeout
- restart timer

event: ACK received

- if ACK acknowledges previously unACKed segments
  - update what is known to be ACKed
  - start timer if there are still unACKed segments

TCP Sender Events

Timeout interval estimation:

- Based on round trip time (RTT)
    - Time taken to get acknowledgement for a TCP segment
- RTT is measured for one of the TCP segments at a time
    - Instantaneous value of RTT is referred to as *SampleRTT*
- RTT is measured only for freshly transmitted segments
    - Segments are chosen randomly
- Time averaged statistics are generated for the RTTs
    - Mean value is denoted as *EstimatedRTT* and standard deviation is denoted as *DevRTT*
- *Timeout interval* is chosen as a random number based on the time averaged mean and standard deviation of the RTTs

TCP Sender Events

Timeout interval estimation(condt.)

μ_n : EstimatedRTT in round n

σ_n : DevRTT in round n

r_n : RTT measured in round n

T_n : Timeout interval in round n

Update for μ_n and σ_n are given by

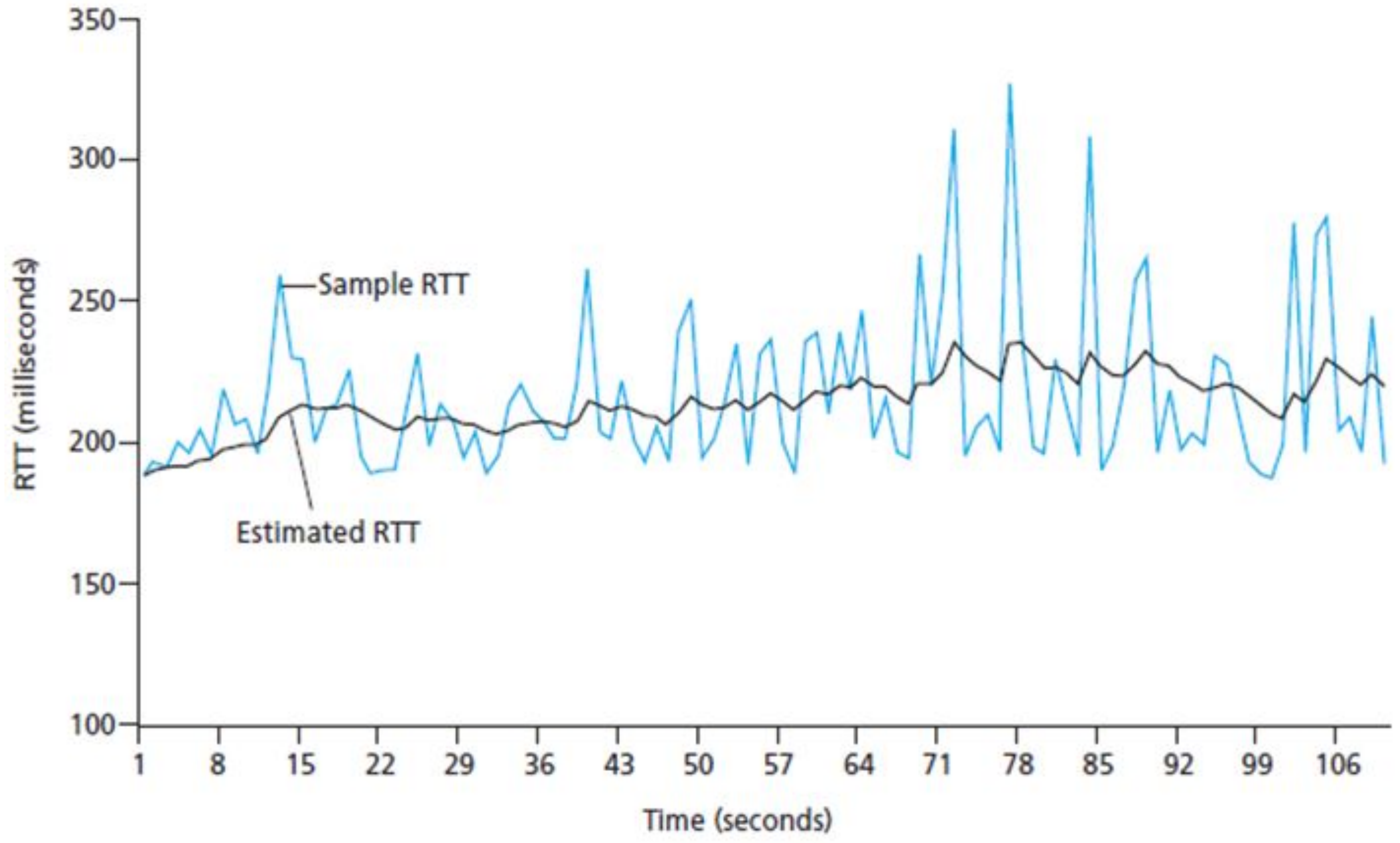
$\mu_n = (1 - \alpha)\mu_{n-1} + \alpha r_n$

$\sigma_n = (1 - \beta)\sigma_{n-1} + \beta|r_n - \mu_n|$

Update of timeout T_n is given by $T_n = \mu_n + 4\sigma_n$

TCP RTT & Timeout

Timeout interval estimation(condt.)



TCP RTT & Timeout

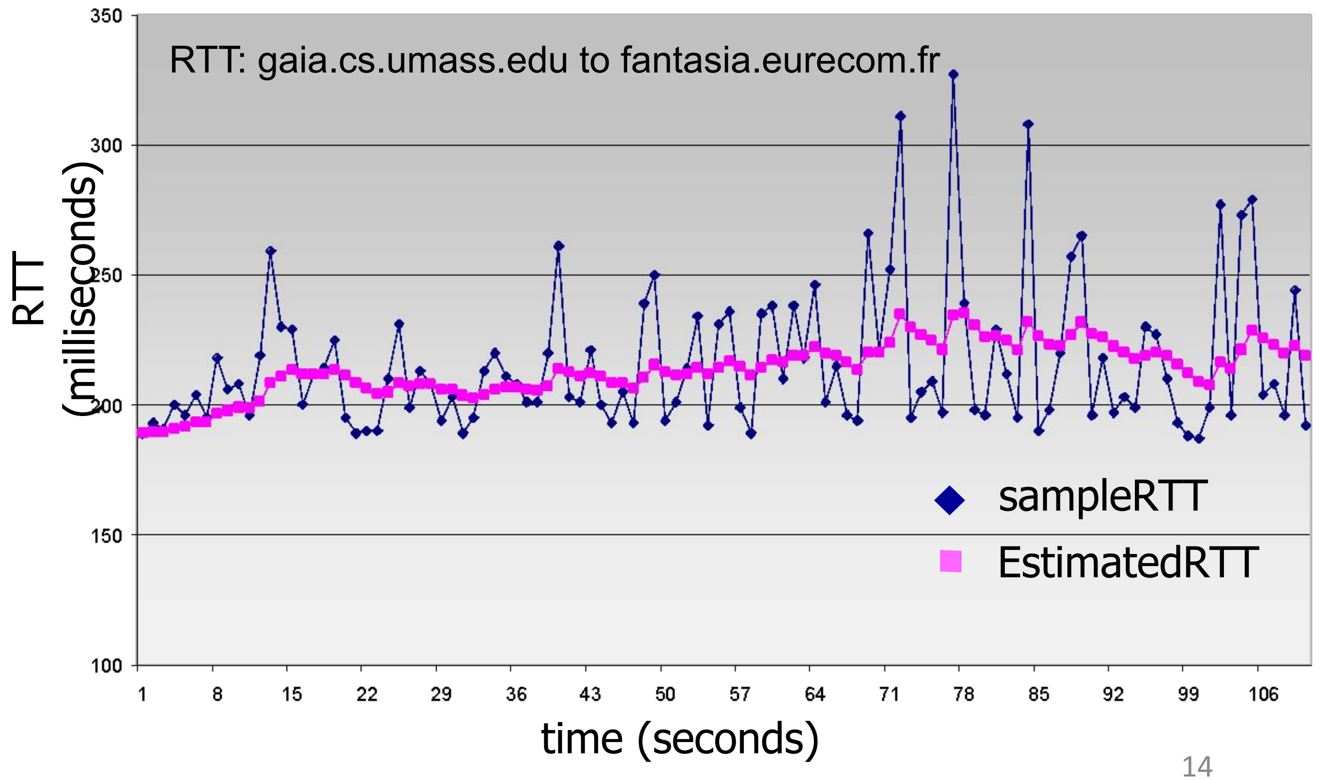
Timeout interval estimation(condt.)

- Recommended value of α is 0.125
- Recommended value of β is 0.25
- Initial value of Timeout interval is 1 sec
- RTT is measured and stored as SampleRTT (x_n) in each round
- After each success (i.e., acknowledgement is received), the Timeout interval for round n is updated using the equations in previous slide
- In case a timeout occurs (i.e., acknowledgements missed), Timeout interval is doubled
- Packets whose acknowledgements were missed in previous round are retransmitted and timer is set to the Timeout interval
- The above steps repeats while data carrying TCP segments are sent

TCP : RTT & Timeout

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

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



TCP : RTT & Timeout

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

```
TimeoutInterval = EstimatedRTT + 4*DevRTT
```

estimated RTT

"safety margin"

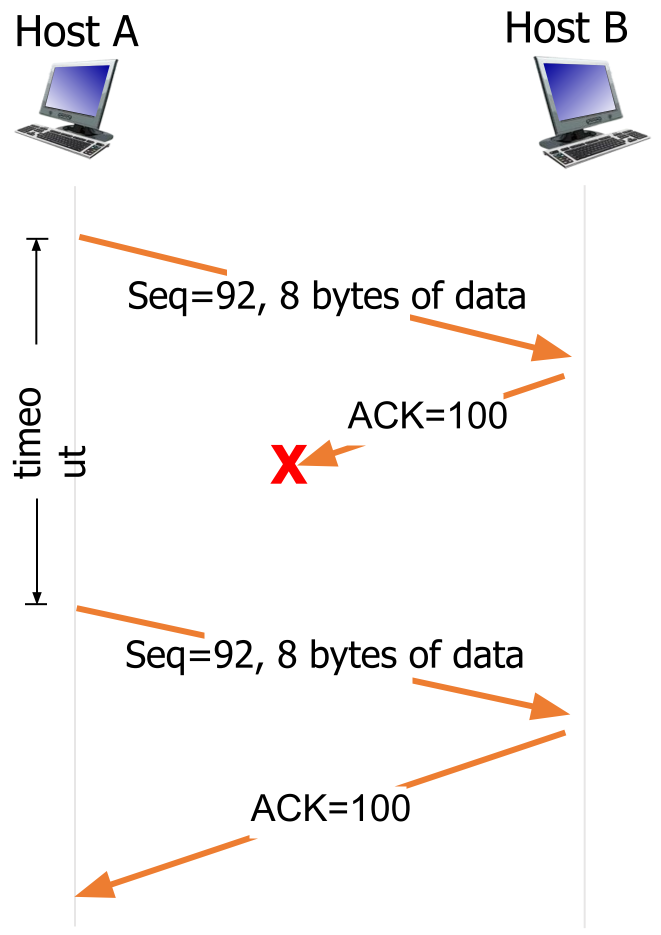
- **`DevRTT`**: EWMA of **`SampleRTT`** deviation from **`EstimatedRTT`**:

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

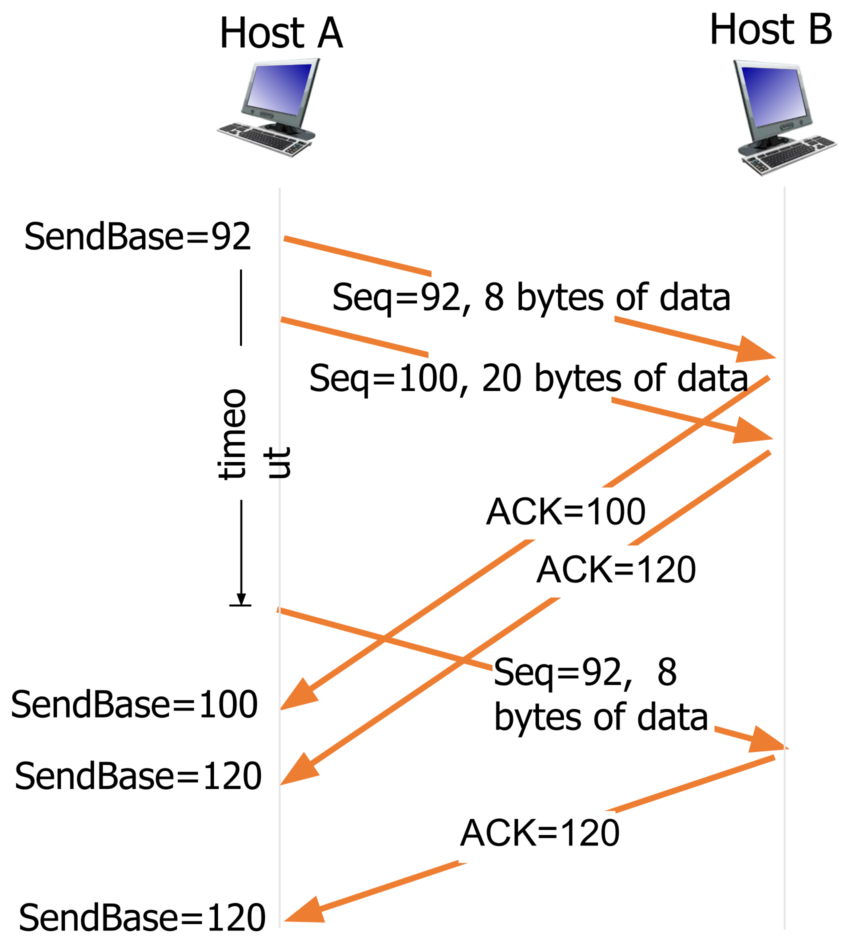
(typically, β = 0.25)

COMPUTER COMMUNICATION NETWORKS

TCP Retransmission Scenarios



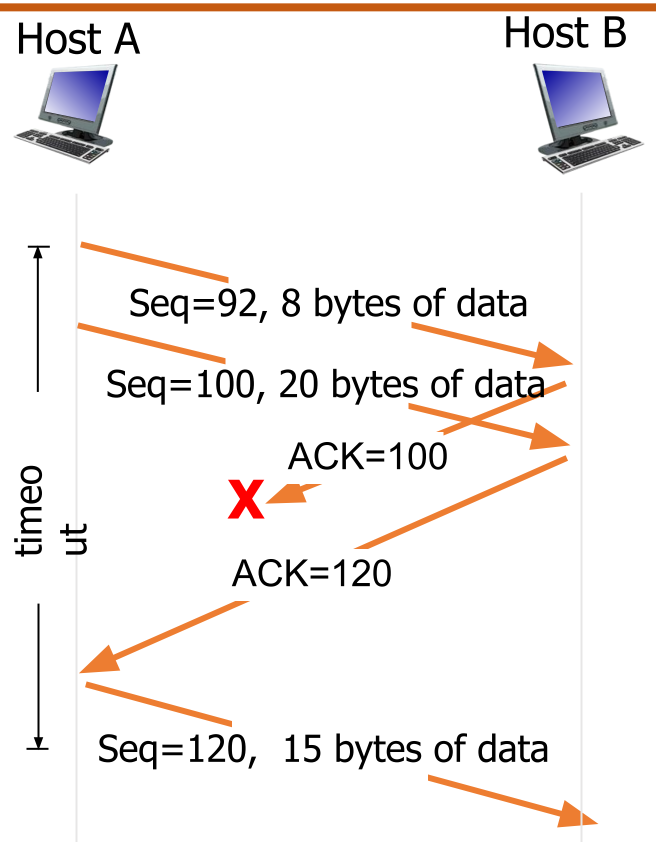
lost ACK scenario



premature timeout

COMPUTER COMMUNICATION NETWORKS

TCP Retransmission Scenarios



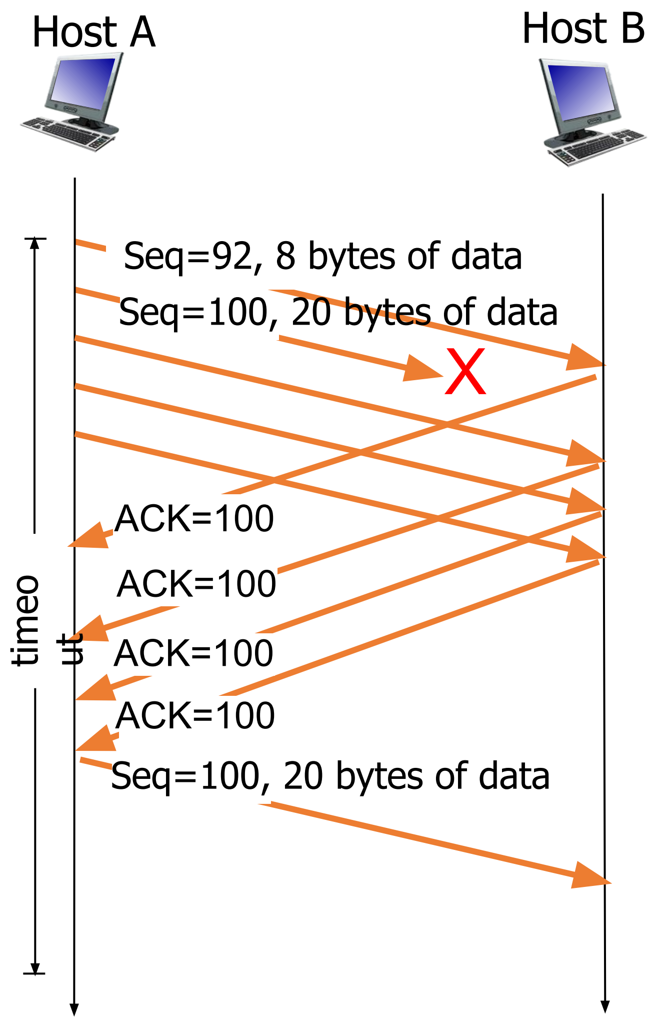
cumulative ACK

TCP ACK Generation

event at receiver	*TCP receiver action*
arrival of in-order segment with expected seq #. All data up to expected seq # already ACKed	delayed ACK. Wait up to 500ms for next segment. If no next segment, send ACK
arrival of in-order segment with expected seq #. One other segment has ACK pending	immediately send single cumulative ACK, ACKing both in-order segments
arrival of out-of-order segment higher-than-expect seq. # . Gap detected	immediately send *duplicate ACK,* indicating seq. # of next expected byte
arrival of segment that partially or completely fills gap	immediate send ACK, provided that segment starts at lower end of gap

TCP Fast Retransmit

TCP fast retransmit



fast retransmit after sender receipt of triple duplicate ACK

THANK YOU

Dr. Arpita Thakre

Department of Electronics and Communication Engineering