Computer Communication Networks

Transport Layer

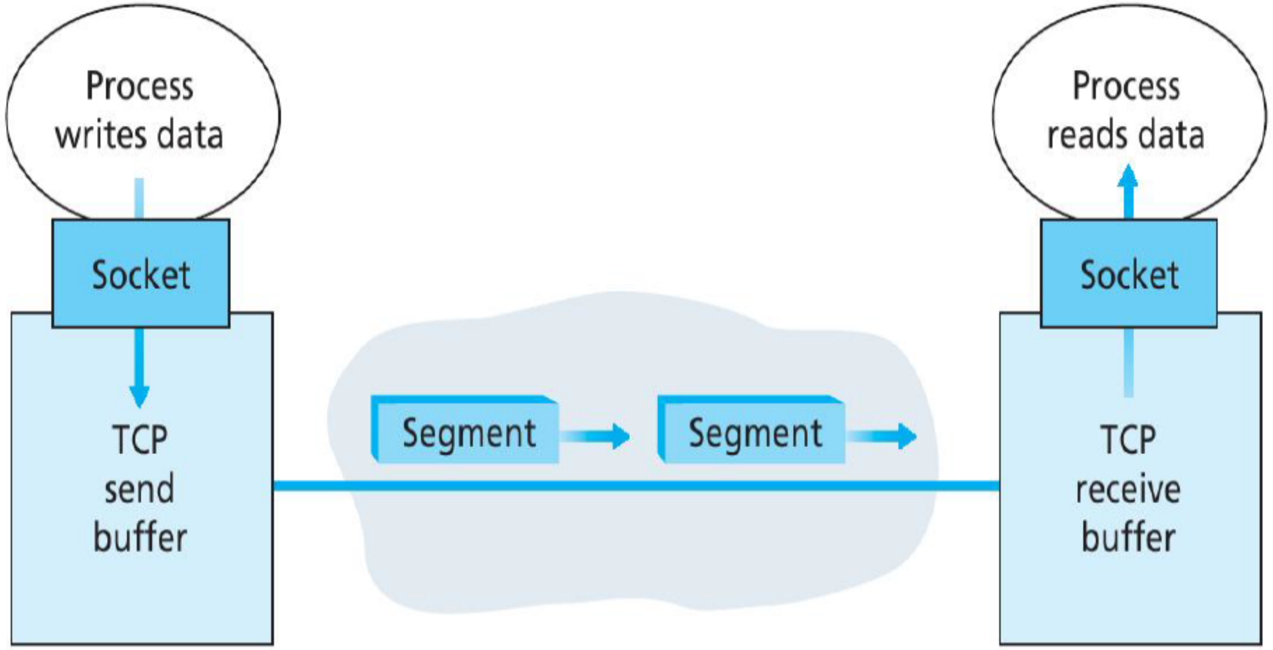
Dr. Raja Vara Prasad

Assistant Professor

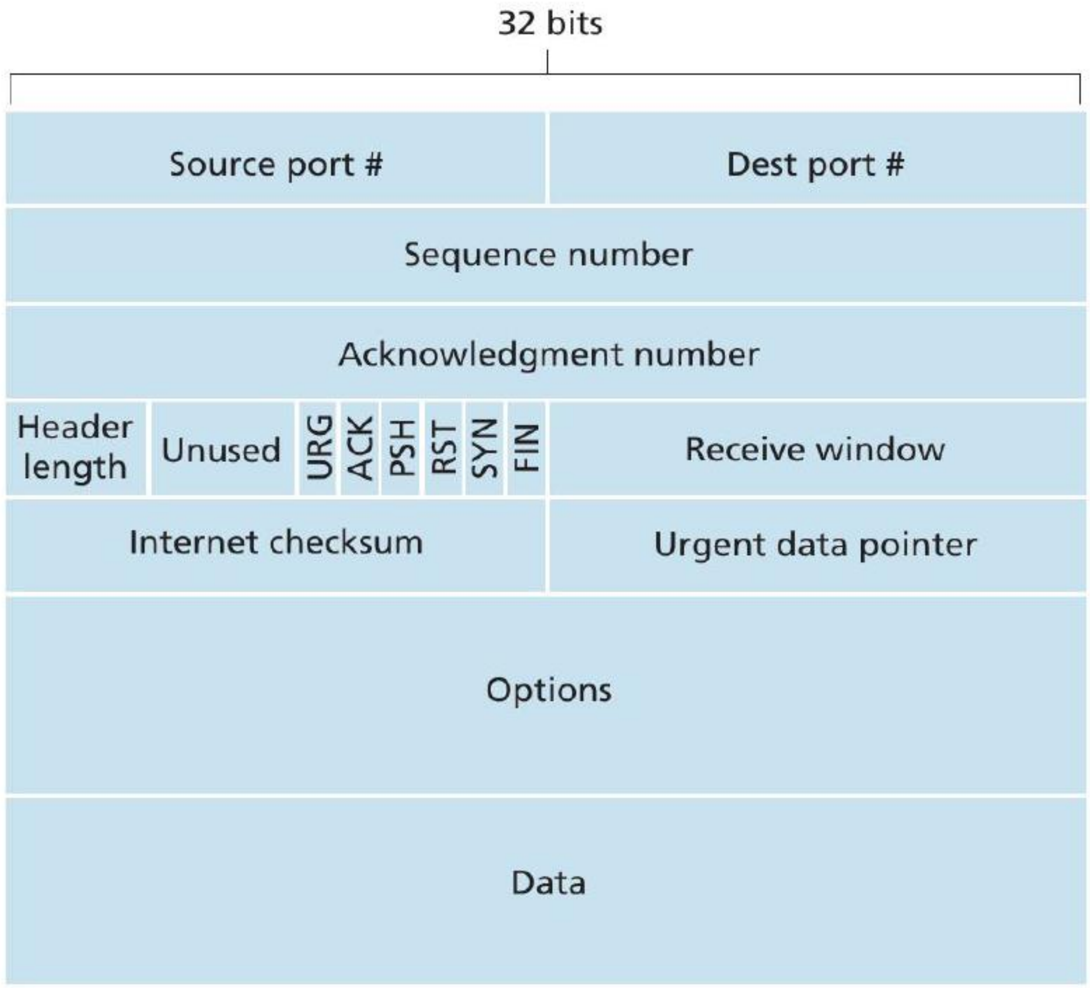
IIIT Sri City

TCP

- TCP is a full duplex service
- No multicasting
- Maximum segment size (MSS) is the maximum amount of data that a TCP segment can contain.



TCP Segment

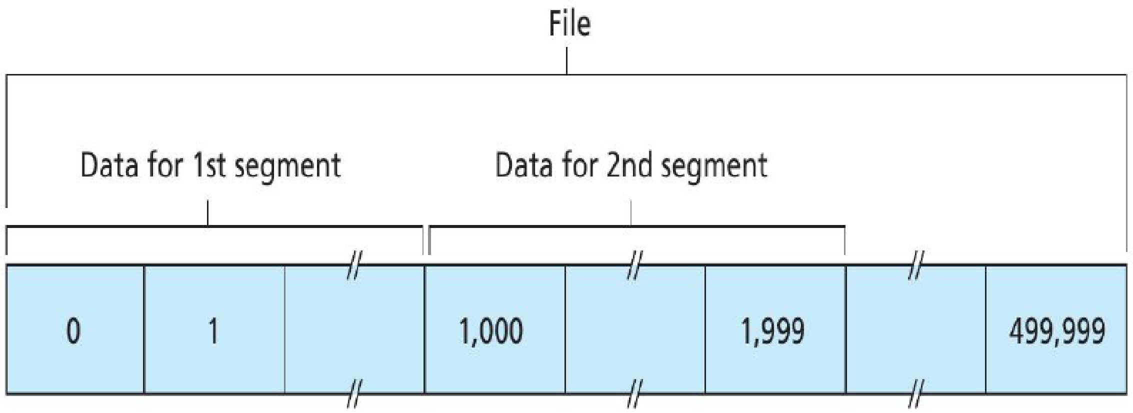

32 bits
Source port #
Dest port #
Sequence number
Acknowledgment number
Header length
Unused
URG
ACK
PSH
RST
SYN
FIN
Receive window
Internet checksum
Urgent data pointer
Options
Data

TCP Segment

- The 16-bit receive window indicates the number of bytes that a receiver is willing to accept
- Header length filed is 4-bytes, specifies the length of the TCP header in 32-bit words.
- Options are used to negotiate MSS, include time-stamping, etc.
- The flag field contains 6 bits, RST, SYN, FIN are used for connection setup and teardown.
- PSH indicates that data has to be sent to upper layers immediately.
- URG is used to mark the segment as urgent, when it is on there will be a 16-bit urgent data pointer filed at the end of urgent data.

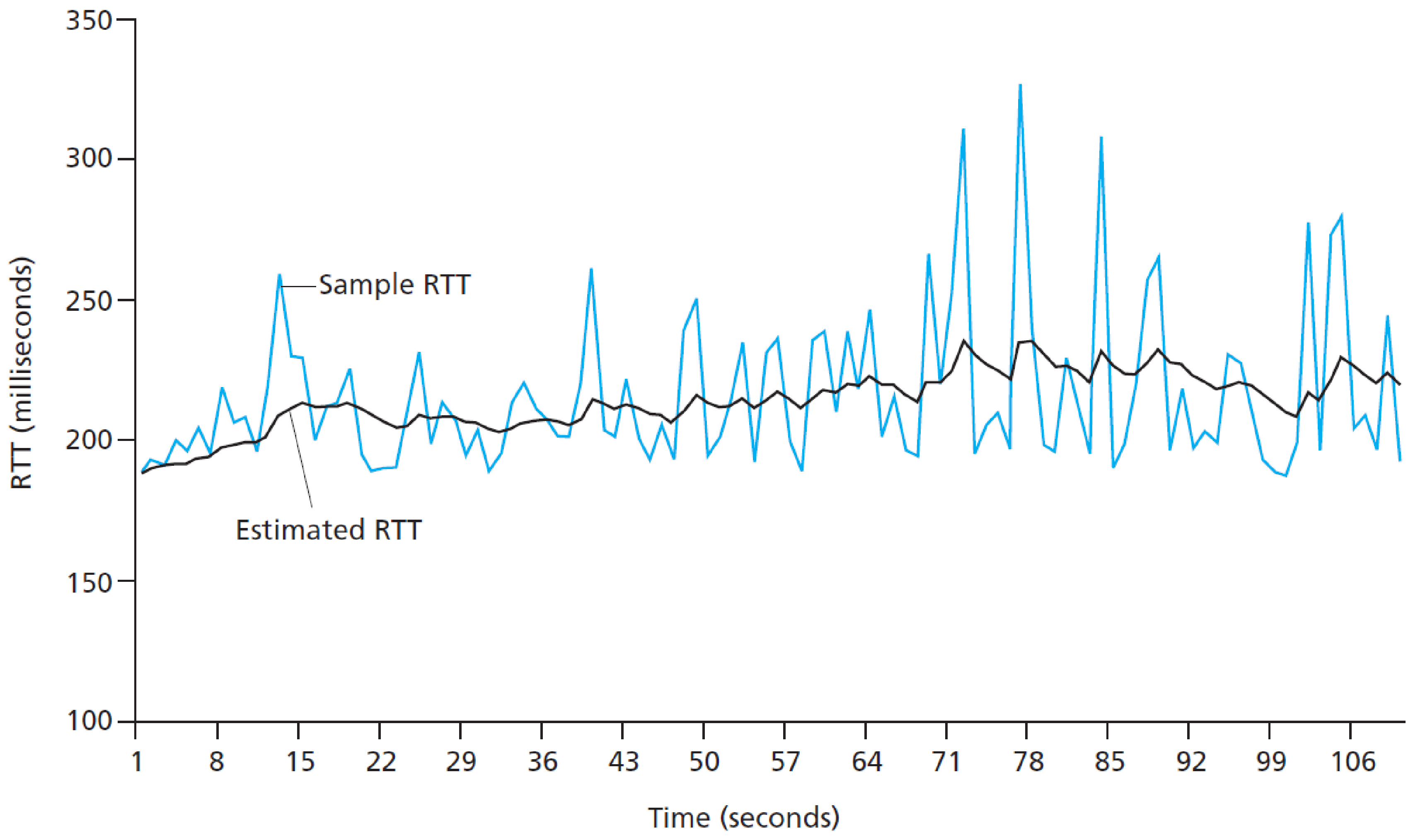
TCP Sequence Numbers

- The sequence number of a segment is the byte-stream number of the first byte of data.
- The acknowledge number is the sequence number of the next byte that is receiver is expecting from source.
- TCP provides cumulative acknowledgments; Out-of-order segements?
- Seqence numbers may not always start from '0'.



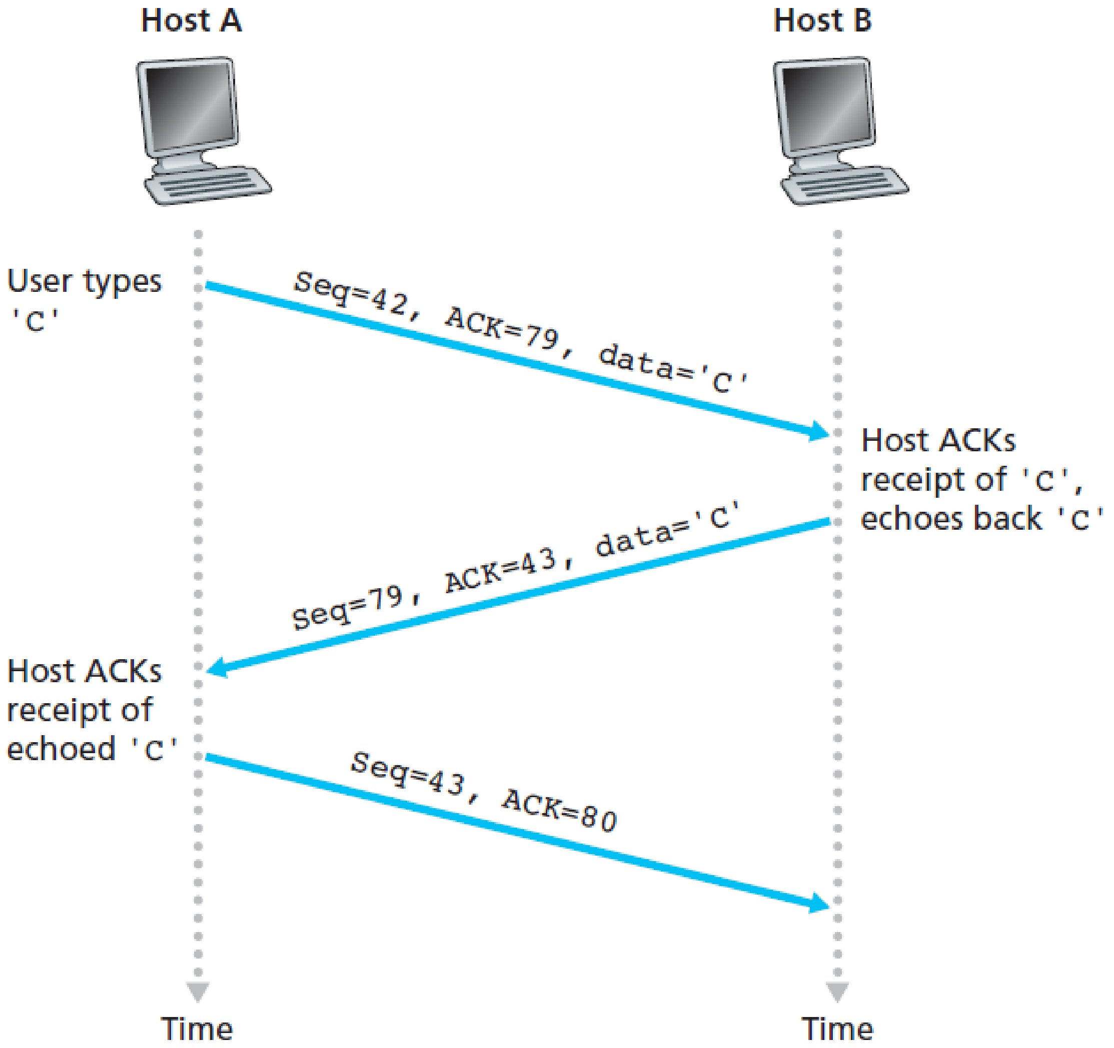
TCP Timeout

- SampleRTT: RTT of a freshly transmitted packet. Computed for each RTT.
- Exponentially weighted moving average: EstimatedRTT $=$ $(1-\alpha)$EstmiatedRTT $+ \alpha$ SampleRTT
- $\alpha = 0.125$
- DevRTT $=$ $(1-\beta)$ DevRTT $+ \beta$ | SampleRTT - EstimatedRTT|
- $\beta = 0.25$
- Timeout $=$ EstimatedRTT $+ 4.$ DevRTT

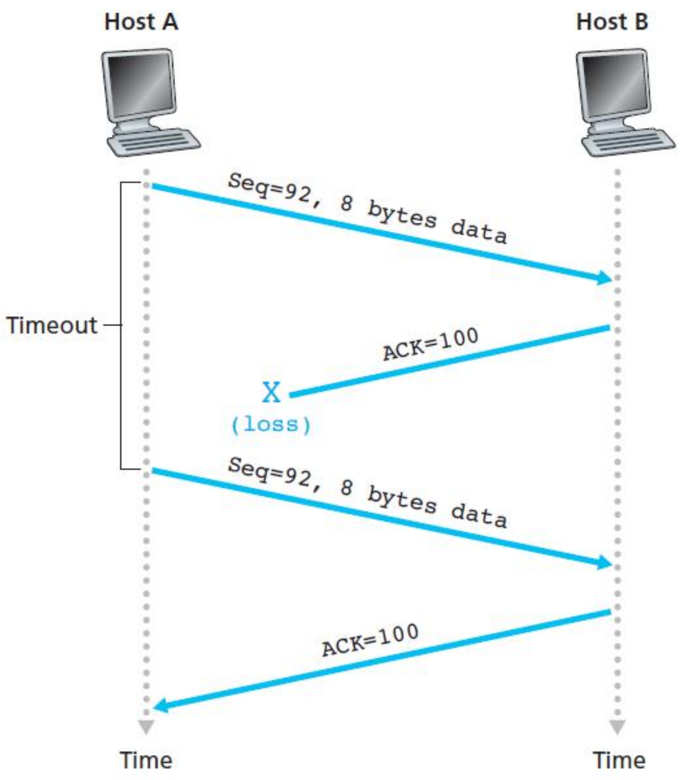


RTT (milliseconds)
350
300
250
200
150
100
Sample RTT
Estimated RTT
1
8
15
22
29
36
43
50
57
64
71
78
85
92
99
106
Time (seconds)

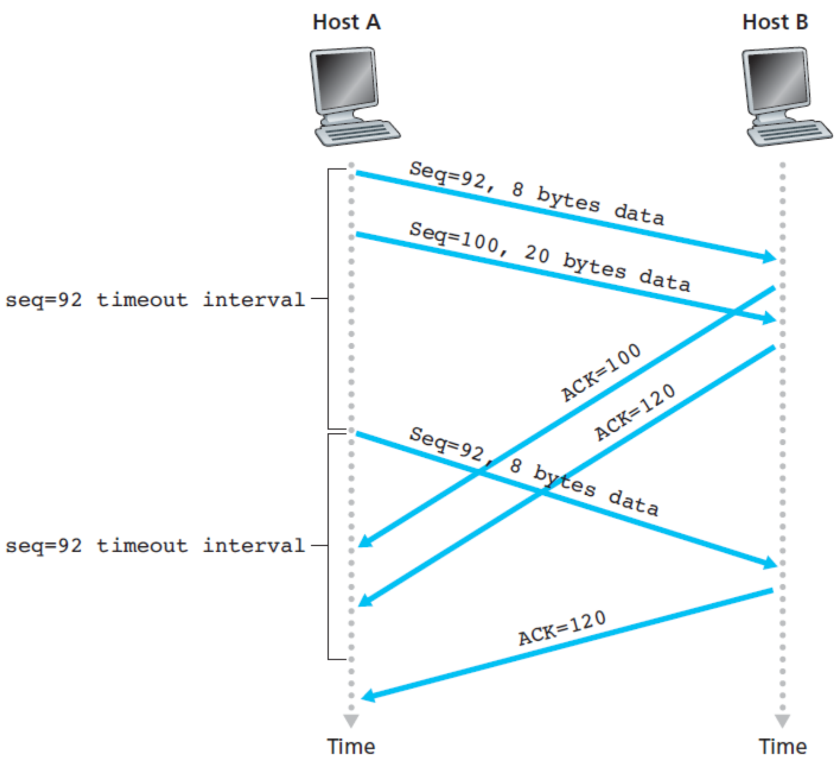
Example: TELNET



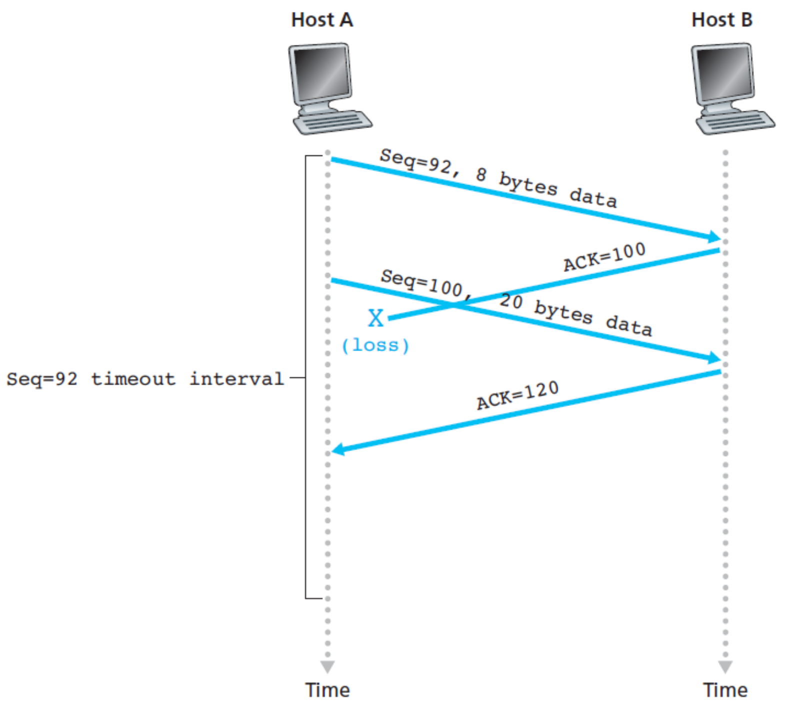
Example: Different scenarios of Reliability aspects



a



b



c

Doubling the Timeout interval:

intervals grow exponentially after each retransmission

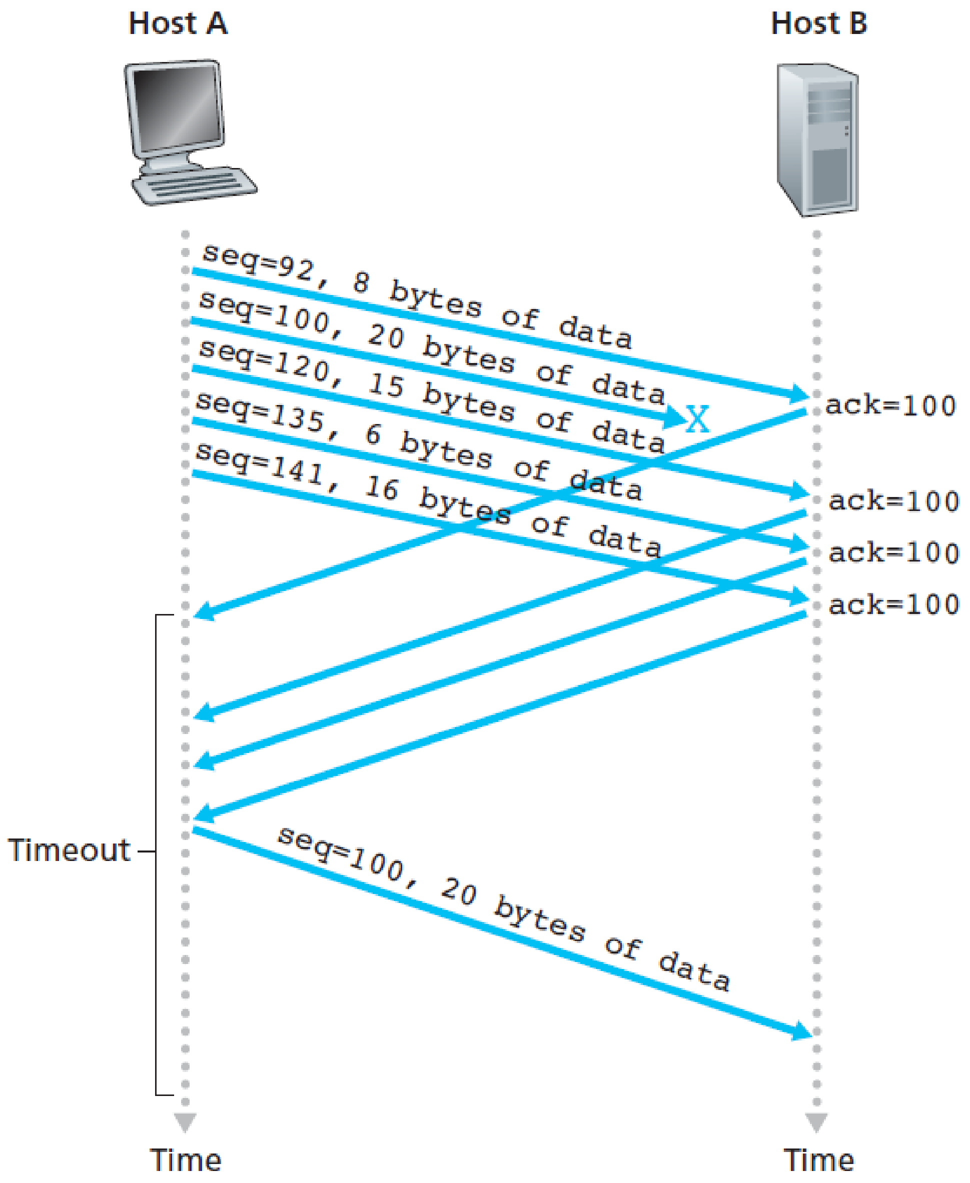
Fast Retransmit:

duplicate ACK multiple times

Example: three duplicate ACKs are received → the TCP sender performs a **fast retransmit**, retransmitting the missing segment *before* that segment's timer expires

Selection: Go-Back-N or Selective Repeat?

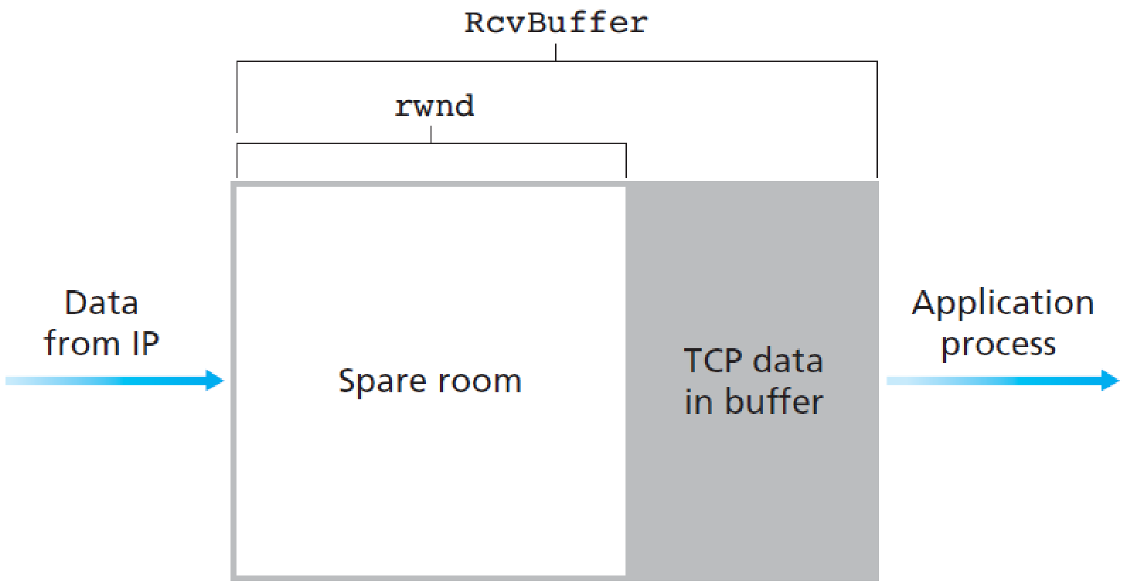
TCP's error-recovery mechanism is probably best categorized as a hybrid of GBN and SR protocols



Flow Control

Flow control is thus a speed-matching service → matching the rate at which the sender is sending against the rate at which the receiving application is reading

receive window: give the sender an idea of how much free buffer space is available at the receiver

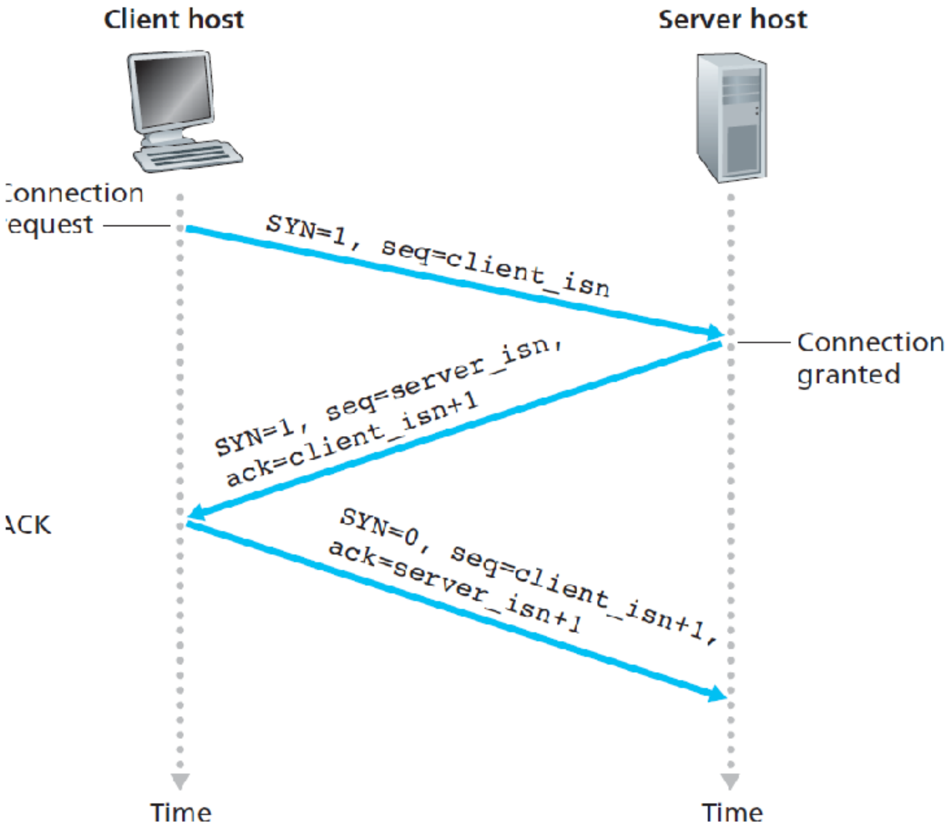


$$\texttt{LastByteRcvd} - \texttt{LastByteRead} \leq \texttt{RcvBuffer}$$

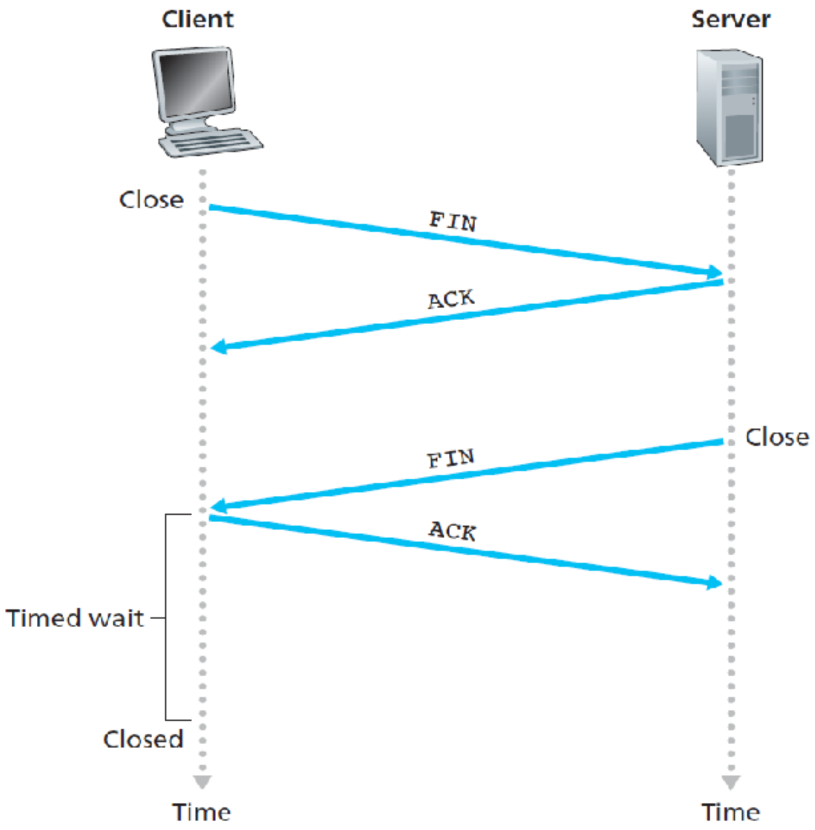
The receive window, denoted rwnd is set to the amount of spare room in the buffer:

$$\texttt{rwnd} = \texttt{RcvBuffer} - [\texttt{LastByteRcvd} - \texttt{LastByteRead}]$$

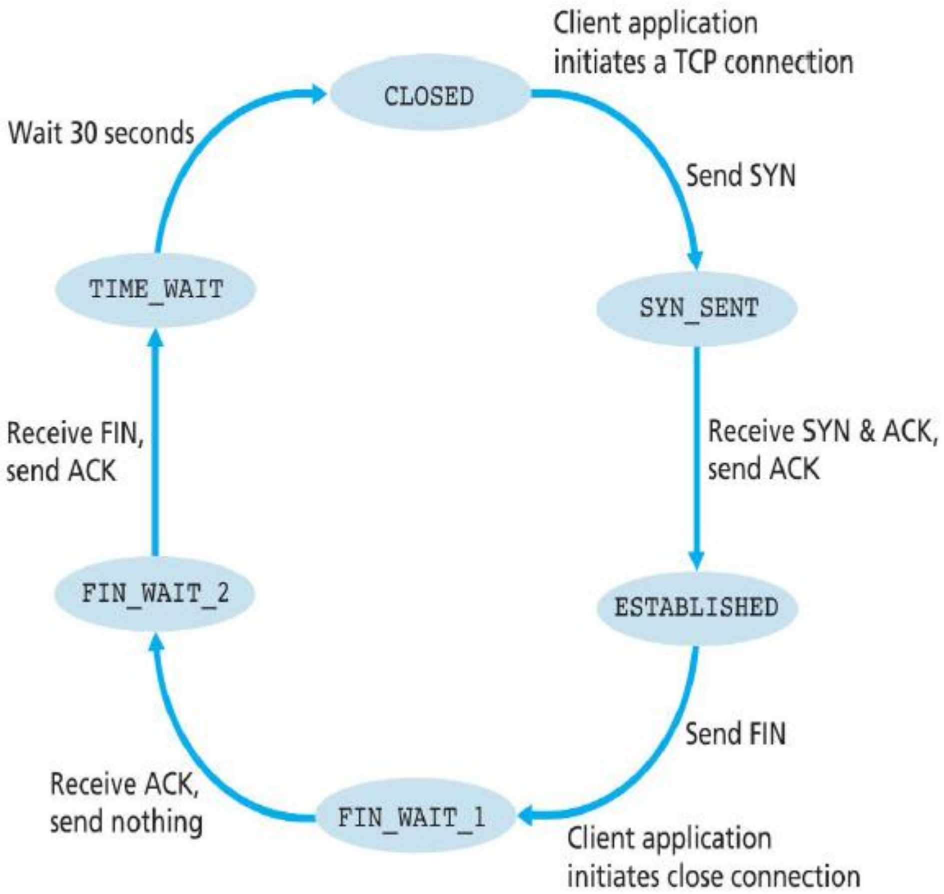
TCP Connection Establishment



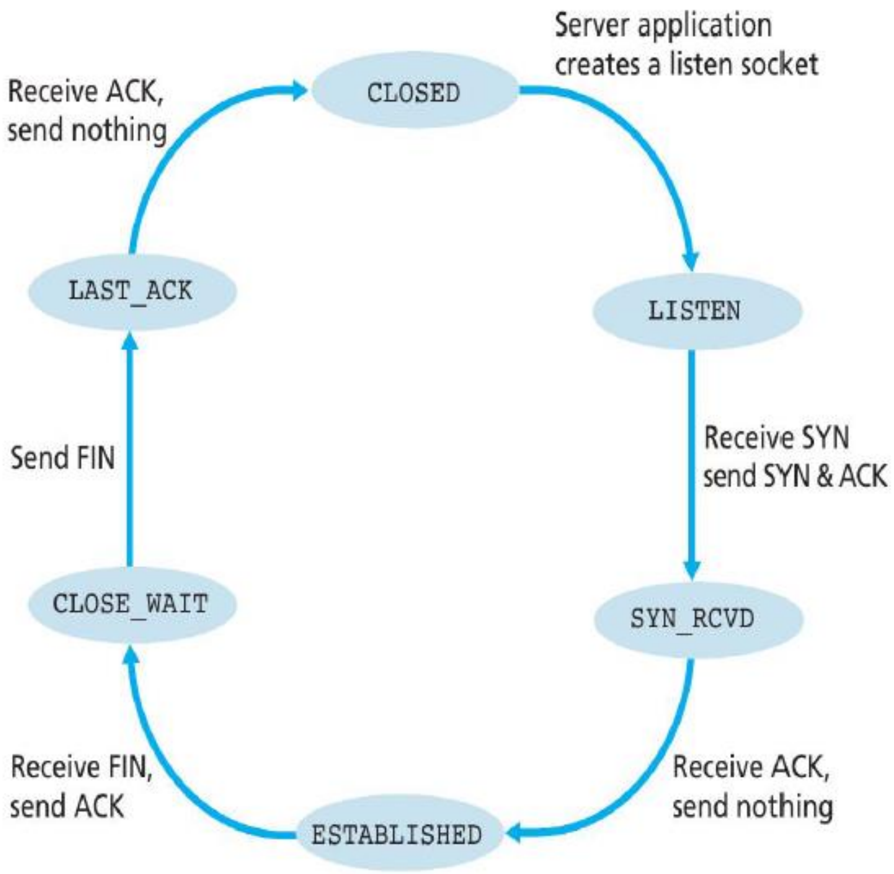
TCP Connection Termination



TCP States at Cleint



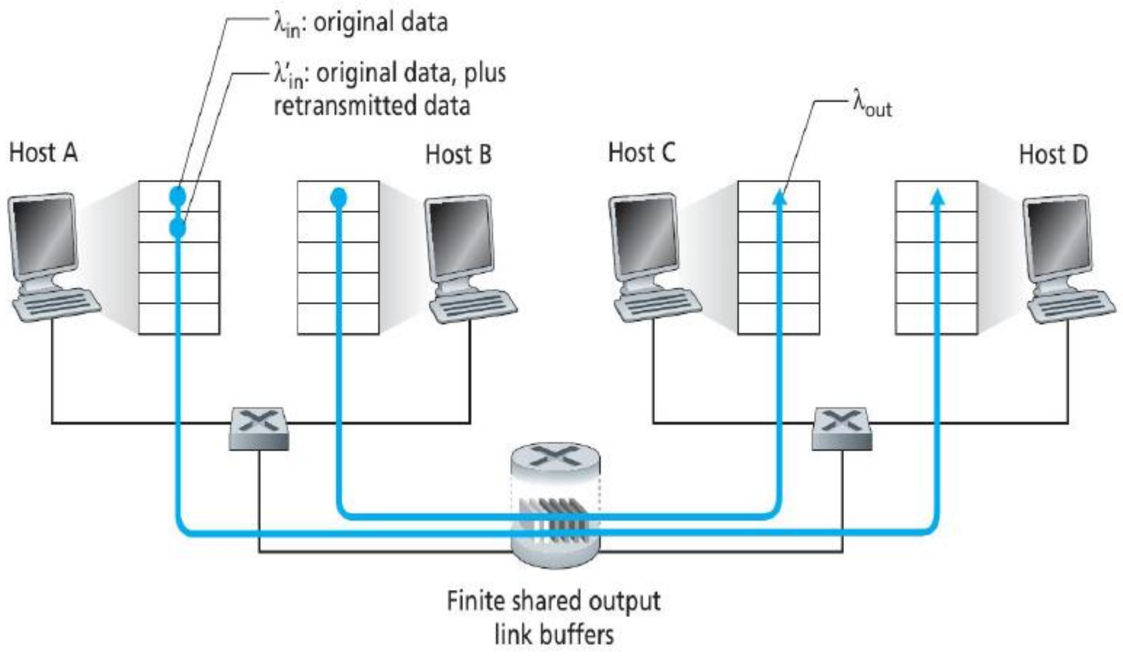
TCP States at Server



Congestion

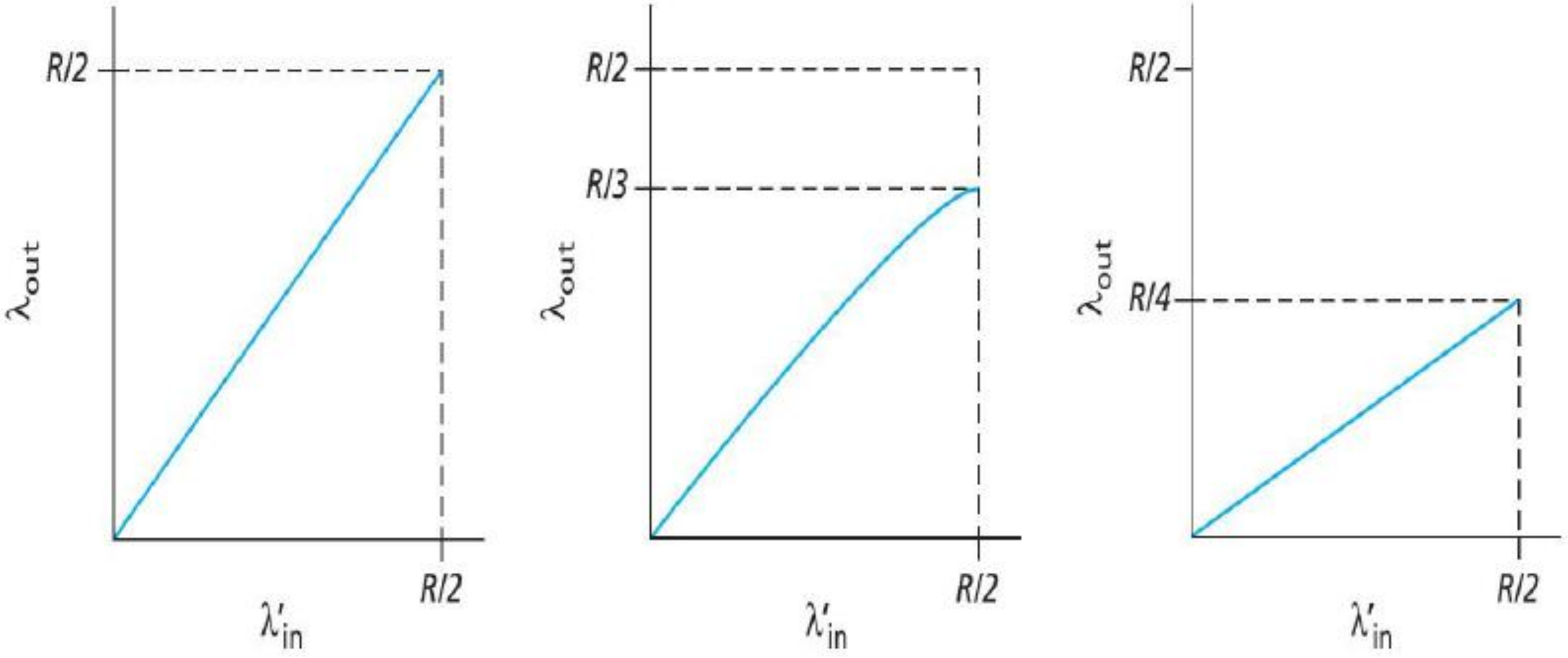
- Why does Congestion occur?
- Packet arrival rate at a router is near or higher than the output link capacity.
- Consequences?
- Buffer overflows, retransmissions to compensate for lost packets
- Unneeded retransmissions

Congestion Scenario - 1

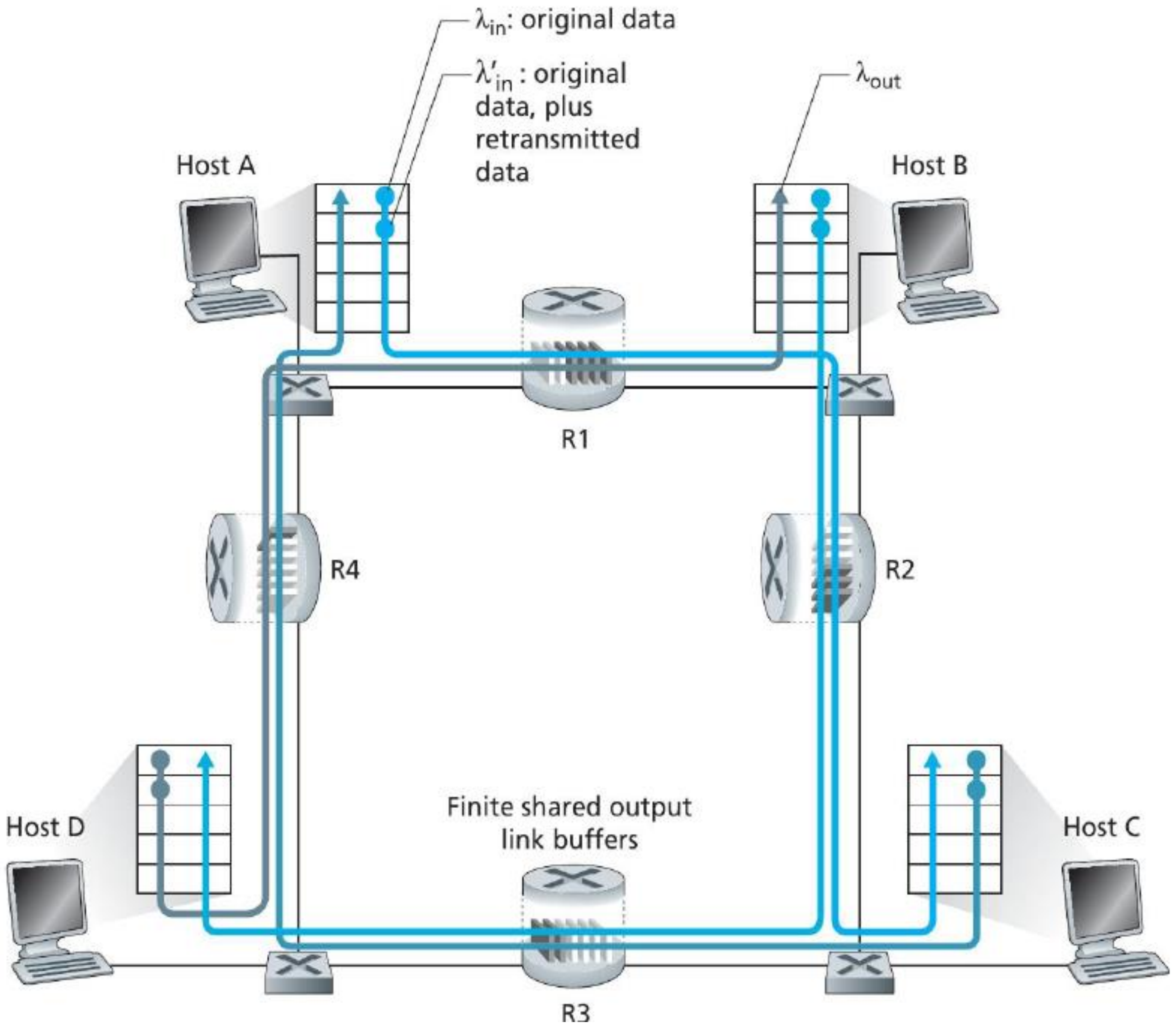


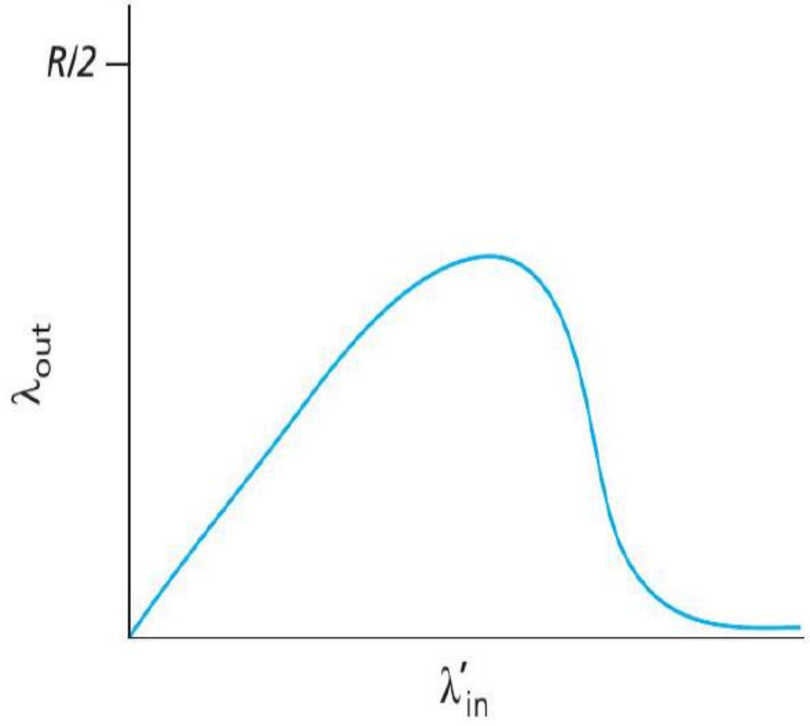
- (a) Host A knows whether buffer in the router has free space or not (Magic!)
- (b) Host A retransmits only if it is sure that packet is lost (Someone has to give this information)
- (c) Host A retransmits on timeouts!

Congestion Scenario - 1



Congestion Scenario - 2

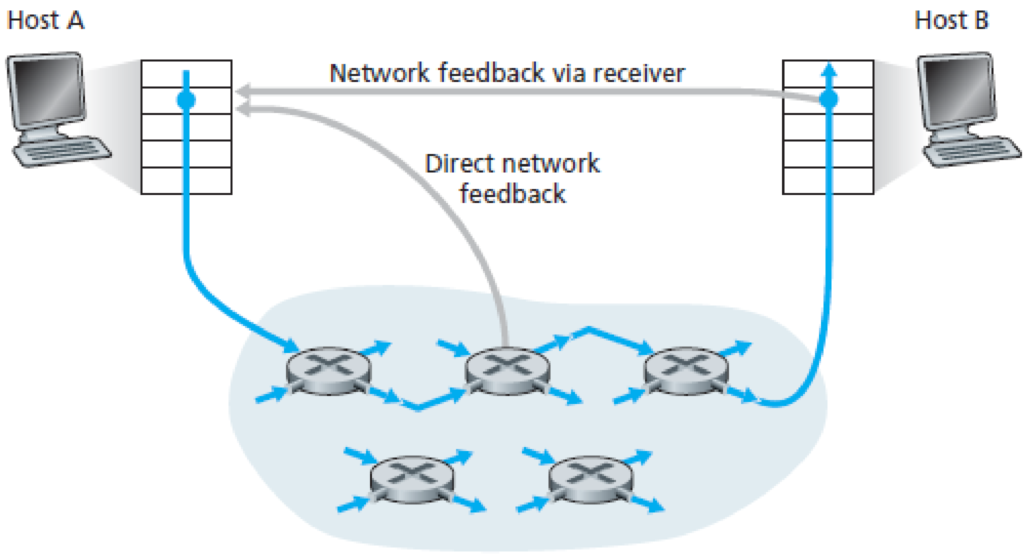




Congestion Control

- End-to-end congestion control: no explicit information about congestion the network
- Network-assisted Congestion Control: Choke packet
- How does TCP identify congestion?
- No assistance form IP
- Identify congestion through timeouts and duplicate ACKs

Congestion Control



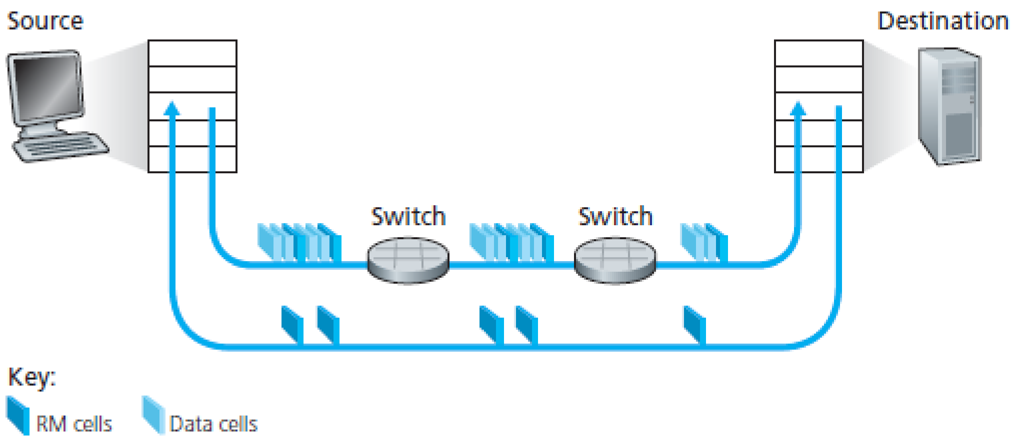


Figure 3.50 ♦ Congestion-control framework for ATM ABR service

EFCI bit. Each *data cell* contains an **explicit forward congestion indication (EFCI) bit**.

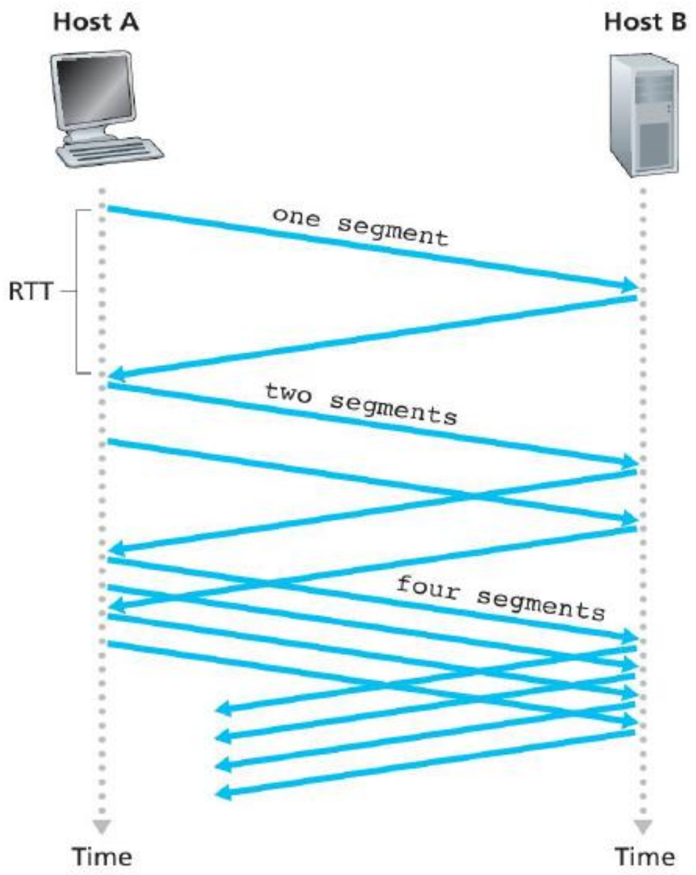
CI and NI bits: **congestion indication (CI) bit** and a **no increase (NI) bit**

ER setting. Each RM cell also contains a 2-byte **explicit rate (ER) field**.

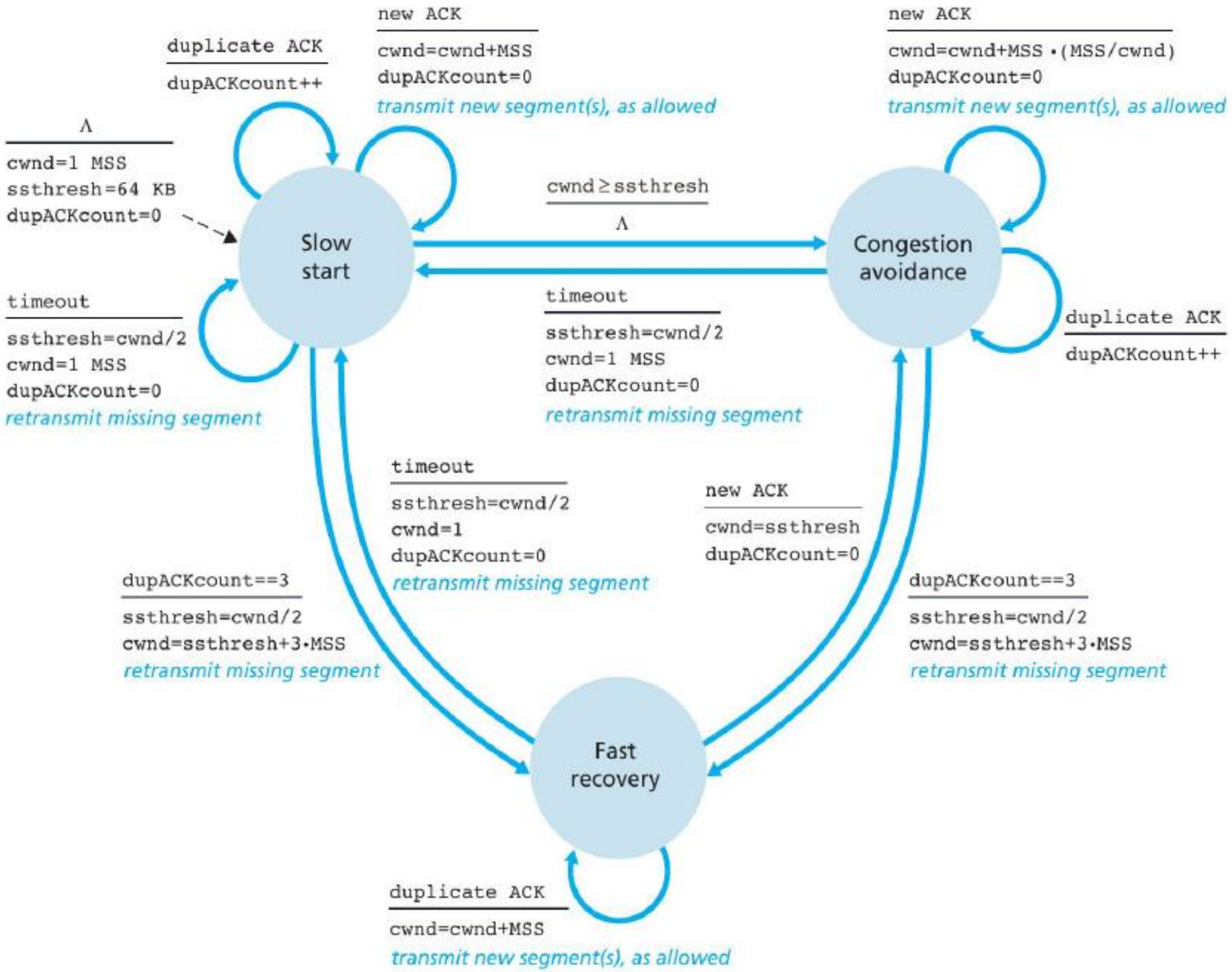
TCP's Congestion Control

- How does TCP control the sending rate?
- Defines a new variable called cwnd.
- LastByteSent - LastByteAcked $\leq$ min{cwnd,rwnd}
- Sends cwnd bytes of infomration per RTT (approximately)
- Can we adjust the speed? Slef-clocking
  - A lost segment triggers the sender to reduce rate of transmission
  - An acknowledgment indicates all is well! Increase the rate
  - Bandwidth Probing

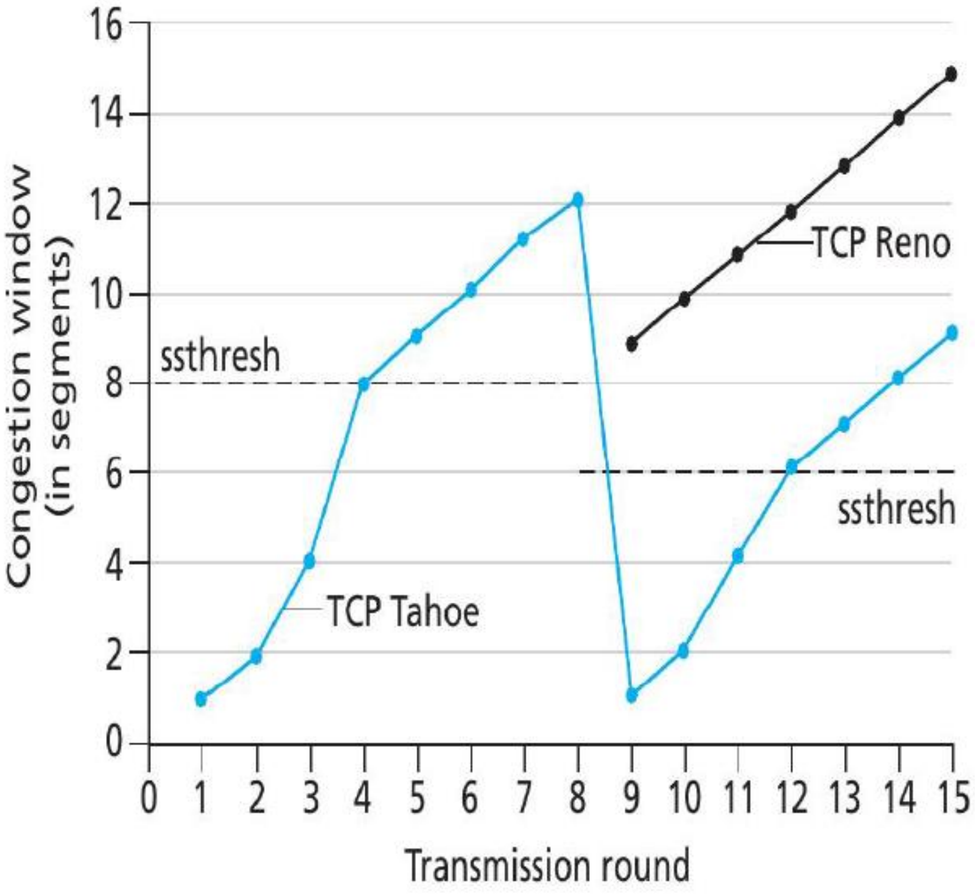
TCP Slow Start



TCP Congestion Control



TCP Congestion Window



AIMD

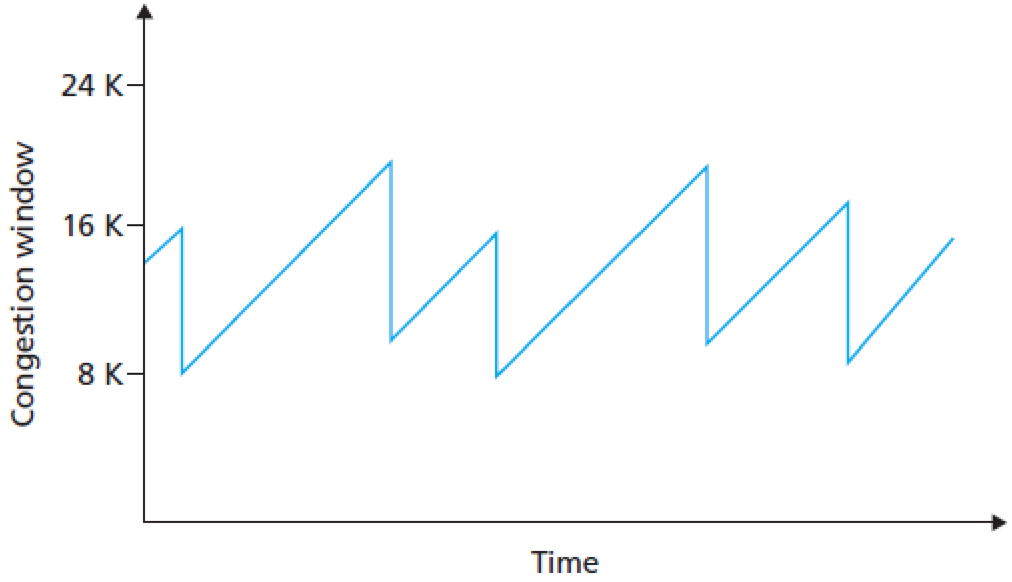


Figure 3.54 ♦ Additive-increase, multiplicative-decrease congestion control

additive-increase, multiplicative decrease (AIMD) form of congestion control:

TCP linearly increases its congestion window size (and hence its transmission rate) until a triple duplicate-ACK event occurs.