COMPUTER COMMUNICATION NETWORKS

Department of Electronics and Communication Engineering

COMPUTER COMMUNICATION NETWORKS

TCP Congestion Control - I

Dr. Arpita Thakre

Department of Electronics and Communication Engineering

TCP Congestion Control

- Congestion is informally: “too many sources sending too much data too fast for network to handle”

- Congestion is manifested by:

1. long delays (queueing in router buffers)
2. packet loss (buffer overflow at routers)

- TCP provides a congestion-control service to its applications.

- TCP sender may get throttled due to congestion in the network. TCP implements mechanism to avoid throttling of sender – such mechanism is called congestion control.

Cost of Congestion

- Large queueing delays at nodes when incoming packet arrival rate nears the link capacity of outgoing link.
- Buffers at nodes are of finite length, leading to packet drop (loss) at nodes, leading to retransmission of packets, thereby decreasing effective throughput.
- When a packet is dropped along a path, the transmission capacity of upstream links that were used to send the packet from sender up-to the point of drop gets wasted, which decreases effective throughput further.

TCP Congestion Control

- Congestion is detected by a TCP sender based on two events
  - Timeout
  - Reception of three duplicate ACKs

- TCP provides end-to-end congestion control since IP does not provide explicit support to TCP layer for congestion control

TCP Congestion Control

- Congestion window, **cwnd**, maintained at sender
- **cwnd**, imposes a constraint on the rate at which a TCP sender can send traffic into the network. Specifically, the amount of unacknowledged data at a sender may not exceed the minimum of cwnd and rwnd, that is:
- LastByteSent – LastByteAcked ≤ min{cwnd, rwnd}

TCP Congestion Control

- Roughly, at the beginning of every RTT, the constraint permits the sender to send **cwnd** bytes of data into the connection; at the end of the RTT the sender receives acknowledgments for the data.
- Thus the sender's send rate is roughly cwnd/RTT bytes/sec. By adjusting the value of cwnd, the sender can therefore adjust the rate at which it sends data into its connection.
- cwnd = 500 bytes, RTT 200 msec, data rate = 20 kbps
- How does TCP perform congestion control?
- by dynamically adjusting **cwnd** in response to observed network congestion after each round (RTT).

TCP Congestion Control

- LastByteSent – LastByteAcked <= min{cwnd, rwnd}
- Segment Loss, 3 duplicate ACKs – indicate congestion in the path → decrease cwnd
- ACKs received regularly and at high rate – congestion is less -> increase cwnd, in fact cwnd can be increased fast
- TCP self-clocking mechanism

TCP Congestion Control

- Congestion window size, **cwnd** is expressed as number of TCP segments
- TCP segments in a transmission round have same length (bytes)
- TCP segment length is referred to as *maximum segment size* (MSS)
  - Note that MSS is negotiated by sender and receiver using SYN and SYNACK segments via Options field in the TCP header
  - Packet length (L) transmitted = MSS + IP header + link layer header
- Congestion control algorithm modifies the congestion window size based on the congestion in the network
  - When no congestion, then the sender increases the value of cwnd
  - When congestion, then the sender decreases the value of cwnd
- Two versions of TCP will be covered: TCP Reno and TCP Tahoe

TCP Congestion Control

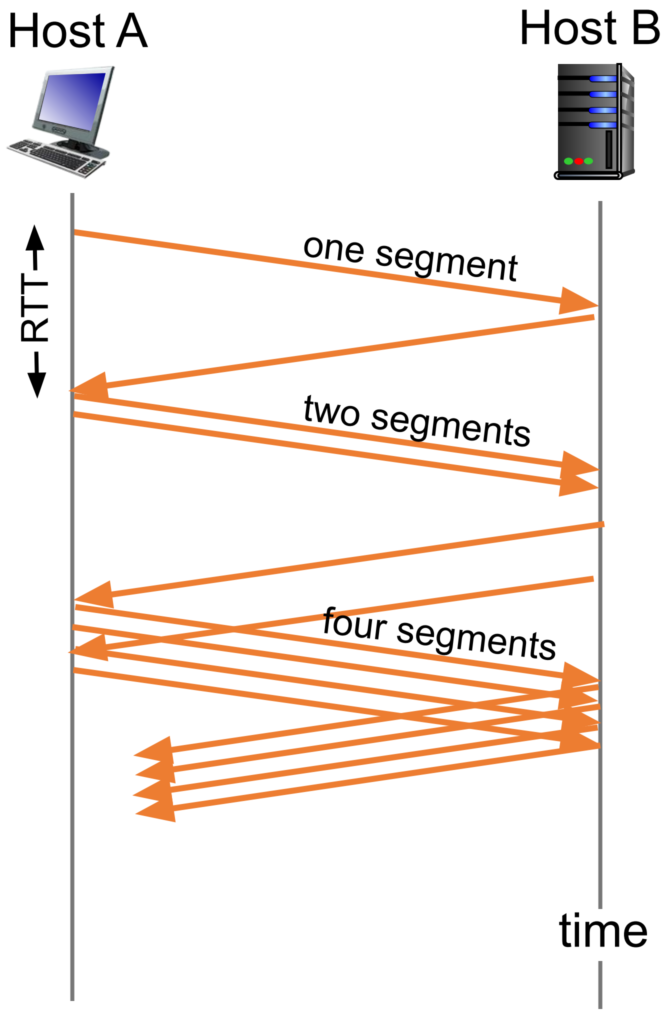
1. Slow Start Phase :
2. Congestion Avoidance Phase :
3. Fast Recovery Phase :

Slow start phase

1. Initial cwnd is 1 MSS
2. For every new acknowledgement received in the transmission round, cwnd increases by 1 MSS
3. In other words, when all packets in a given transmission round are acknowledged, then the cwnd value of the next round is double the value of cwnd in the current round (i.e., binary exponential increase)

Slow start phase (contd.):

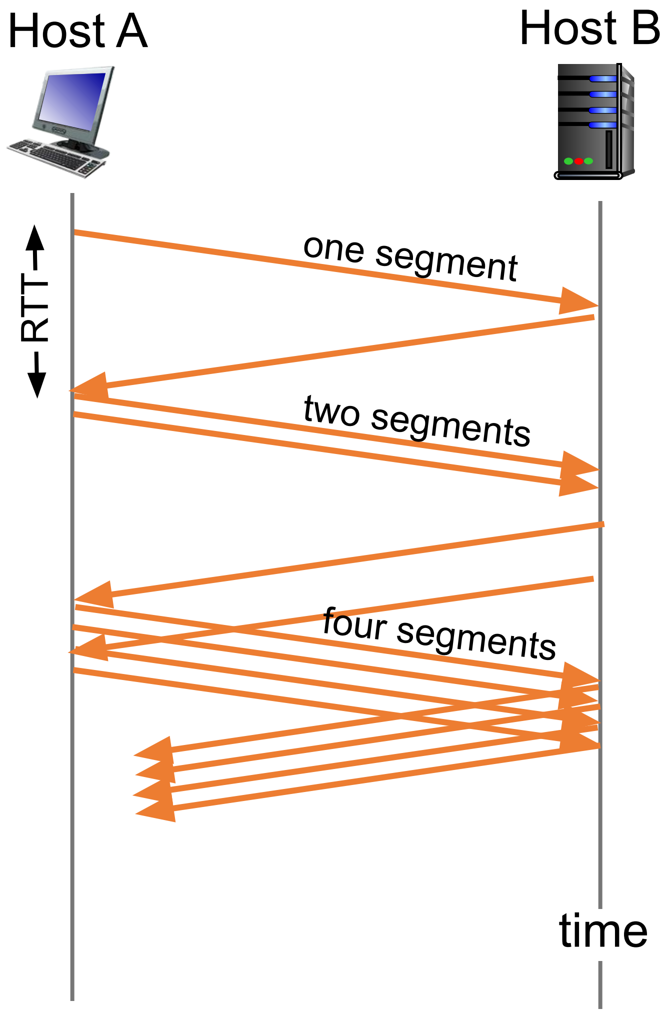
- when connection begins, increase rate exponentially until first loss event:
  - initially **cwnd** = 1 MSS
  - double **cwnd** every RTT
  - done by incrementing **cwnd** for every ACK received
- *summary:* initial rate is slow but ramps up exponentially fast



TCP Congestion Control

Slow start phase (contd.):

- cwnd = 1
- After 1 RTT, cwnd = 2
- After 2 RTT, cwnd = 4
- After 3 RTT, cwnd = 8



Congestion Avoidance phase:

- cwnd = i
- After 1 RTT, cwnd = i+1
- After 2 RTT, cwnd = i+2
- After 3 RTT, cwnd = i+3

Fast recovery phase:

- cwnd = cwnd + 1 for every duplicate ACK received for missing segment that caused TCP to enter into Fast Recovery Phase

TCP Congestion Control

Slow Start Phase -> Timeout occurs (packet loss) -> ssthresh = cwnd/2, cwnd =1 ->Slow Start Phase

Slow Start Phase -> cwnd >= ssthresh -> ssthresh = cwnd/2, cwnd = ssthresh -> Congestion Avoidance

Slow Start Phase -> 3 duplicate ACKs -> Fast Recovery

TCP Congestion Control

Congestion Avoidance -> Timeout occurs (packet loss) -> ssthresh = cwnd/2, cwnd =1

-> Slow Start Phase

Congestion Avoidance -> 3 duplicate ACKs -> Fast Recovery

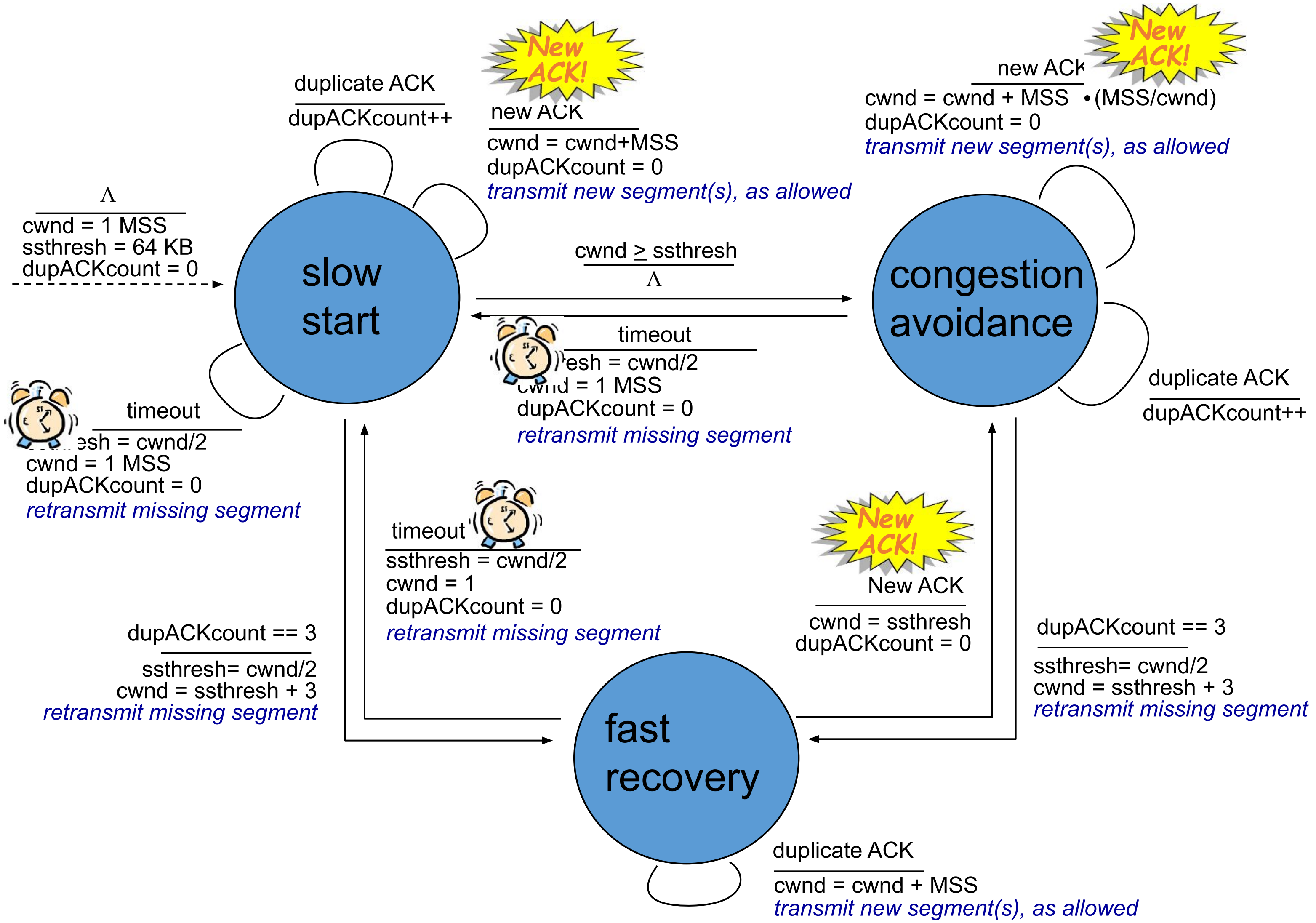
TCP Congestion Control

Fast Recovery -> Timeout occurs (packet loss) -> ssthresh = cwnd/2, cwnd =1

-> Slow Start

COMPUTER COMMUNICATION NETWORKS

TCP Congestion Control



THANK YOU

Dr. Arpita Thakre

Department of Electronics and Communication Engineering