CS5222 Computer Networks and Internets Tutorial 6 (Week 6)

Prof Weifa Liang

Weifa.liang@cityu.edu.hk

Slides based on book *Computer Networking: A Top-Down Approach.*

Revision

Flow Control vs. Congestion Control

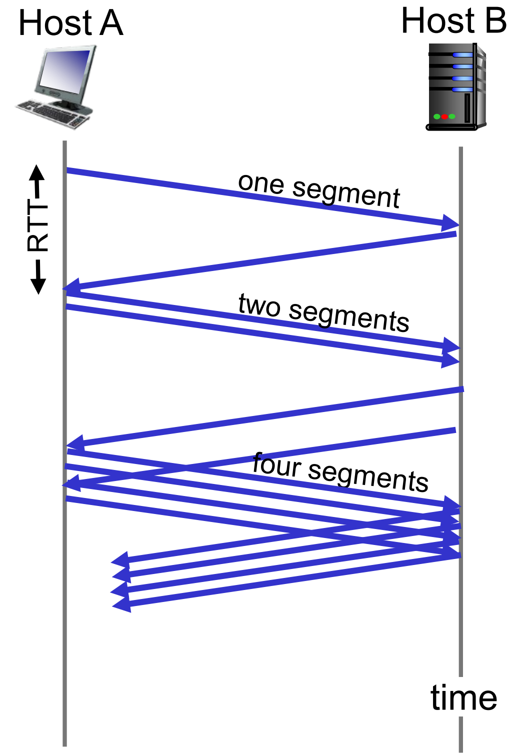
- Flow control
  - Keeping *one fast sender* from overwhelming *a slow receiver*

- Congestion control
  - Keep a *set of senders* from overloading the *network*
    - E.g., persuade hosts to stop sending, or slow down
    - Typically has notions of fairness (i.e., sharing the pain)

- Different concepts, but similar mechanisms
  - TCP flow control: receiver window
  - TCP congestion control: congestion window
  - TCP window: *min{congestion window, receiver window}*

TCP slow start

- when a 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:* the initial rate is slow, but ramps up exponentially fast



TCP: detecting, reacting to loss

- loss indicated by timeout:
  - **`cwnd`** set to 1 MSS;
  - window then grows exponentially (as in slow start) to threshold (the half the previous **cwnd**), then grows linearly

- loss indicated by 3 duplicate ACKs: **TCP RENO**
  - dup ACKs indicate network capable of delivering *some* segments
  - **`cwnd`** is cut in half window then grows linearly

- **TCP Tahoe** always sets **`cwnd`** to 1 (timeout)

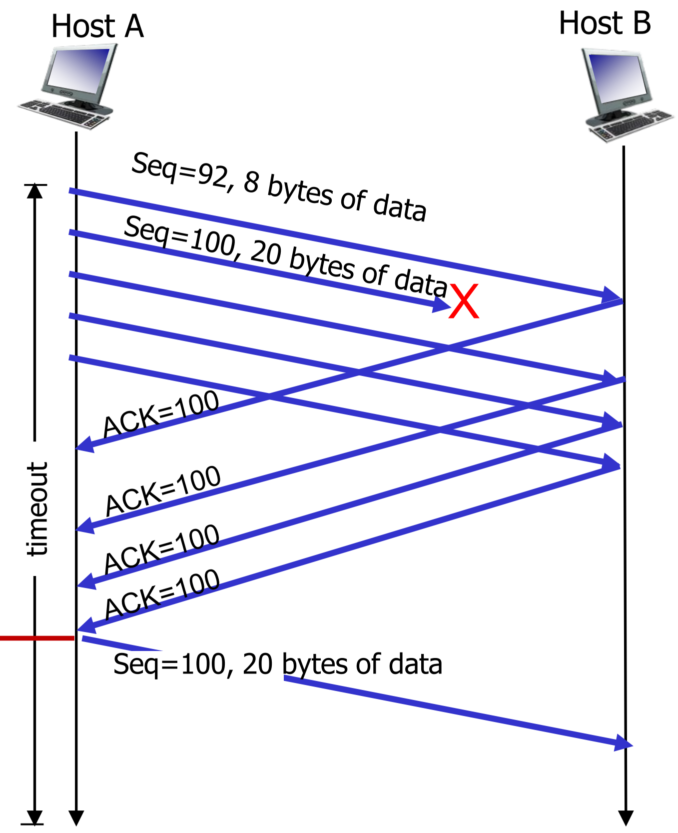
TCP fast retransmit

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, so don't wait for timeout

Receipt of three duplicate ACKs indicates 3 segments received after a missing segment – lost segment is likely. So retransmit!



TCP: detecting, reacting to loss

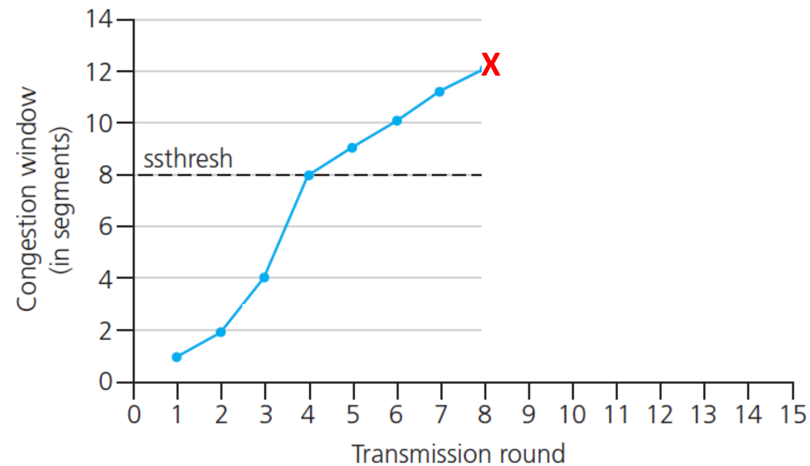
- loss indicated by timeout: **TCP Tahoe**
  - **`cwnd`** set to 1 MSS;
  - window then grows exponentially (as in slow start) to threshold, then grows linearly

- loss indicated by 3 duplicate ACKs: **TCP RENO**
  - dup ACKs indicate network capable of delivering *some* segments
  - **`cwnd`** is cut in half window then grows linearly

TCP: from slow start to congestion avoidance

Q: when should the exponential increase switch to linear?

A: when **cwnd** gets to 1/2 of its value before timeout.



Implementation:

- variable **ssthresh**
- on loss event, **ssthresh** is set to 1/2 of **cwnd** just before loss event

TCP round trip time, 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,
  → we need "smooth" estimated RTT:
  - average several *recent* measurements, not just current `SampleRTT`

Work on your questions

1. Hosts A and B are directly connected with a 200 Mbps link. There is one TCP connection between them, and host A is sending to Host B an enormous file over this connection. Host A can send application data into the link at 100 Mbps but host B can read out of its TCP receive buffer (**RcvBuffer**) at a maximum rate of 50 Mbps. Describe the effect of TCP flow control on the average sending rate at which A can send to B.

Answer:

- Host A sends data into the receive buffer of Host B faster than Host B can remove data from the buffer.
- When the buffer is full, Host B signals to Host A to stop sending data to it by setting RcvBuffer = 0.
- Host A then stops sending data until it receives a TCP segment with RcvBuffer > 0. However, **it keeps sending 1 Byte segments** to B. Why?

  → To give B the chance to advertise a new RcvBuffer.
- Host A will thus repeatedly stop and start sending as a function of the RcvBuffer values it receives from Host B.
- On average, the long-term rate at which Host A sends data to Host B as part of this connection is no more than 50Mbps.

2. At time *t*, a TCP connection has a congestion window of 4,000 bytes. The maximum segment size used by the connection is 1,000 bytes.

Suppose there is one ACK per packet.

a) If the connection is in the slow start mode, or

b) If the connection is in the linear increase mode,

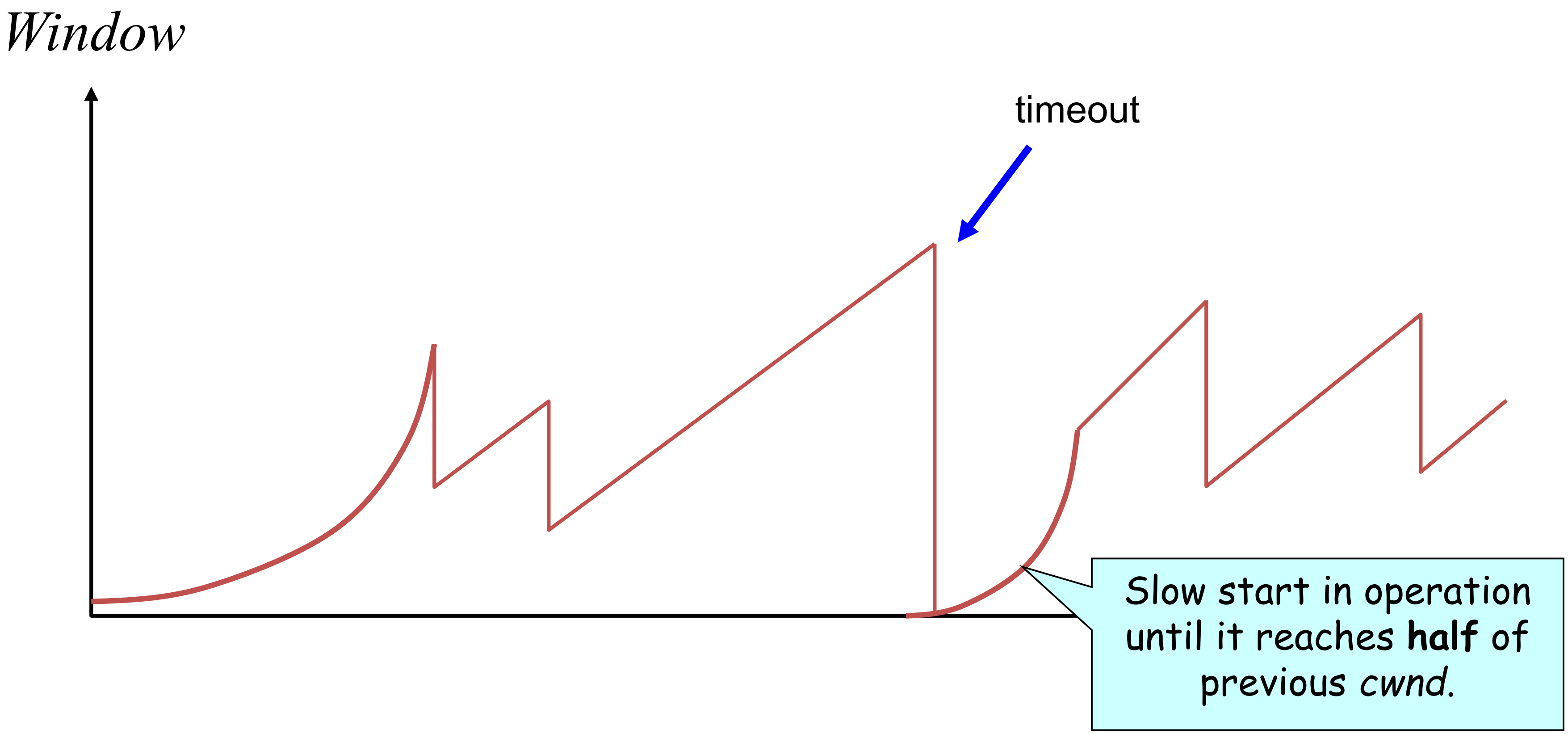
What is the congestion window (**cwnd**) after it sends out 4 packets and receives acks for all of them?

Answer: a) If it is in slow start, after one RTT, **cwnd** will be doubled. Thus, it is 8,000 bytes.

b) If it is in additive (linear) increase mode, after one RTT, **cwnd** will be increased by 1MSS (1,000 bytes in this case).

Thus, **cwnd** = 4,000+1,000= 5,000 bytes.

Repeating Slow Start After Timeout



Slow-start restart: Reset **cwnd** to 1, but take advantage of knowing the previous value of CWND.

3. Consider sending a large file from a host to another over a TCP connection that has **no loss**. Suppose TCP's congestion control **does not use** slow start. We assume that **cwnd** increases by 1 MSS every time a batch of ACKs is received and that the round-trip time is constant.

1) How long does it take for **cwnd** to increase from 5 MSS to 11 MSS assuming no loss events?

Answer: 1)

- after one RTT, **cwnd** becomes 6MSS. After two RTTS, **cwnd** becomes 7RTT, and so on.
- It takes 6 RTTs to make **cwnd** increase from 5MSS to 11MSS.

2) What is the average throughput (in terms of MSS and RTT) for this connection during the time period that **cwnd** increases from 5MSS to 11MSS?

2) During the 6RTTs, the number of bytes sent out is:

5MSS +6MSS….+10MSS= 45MSS.

Thus, the throughput is: 45MSS/6RTT=7.5 MSS/RTT.

4. Consider the figure below in which a TCP sender and receiver communicate over a connection in which the sender->receiver segments may be lost. The TCP sender sends an initial window of 5 segments. Suppose the initial value of the sender->receiver sequence number is 45 and the first 5 segments each contain 462 bytes. The delay between the sender and receiver is 7 time units, and so the first segment arrives at the receiver at t=8. As shown in the figure below, 3 of the 5 segment(s) are lost between the segment and receiver.

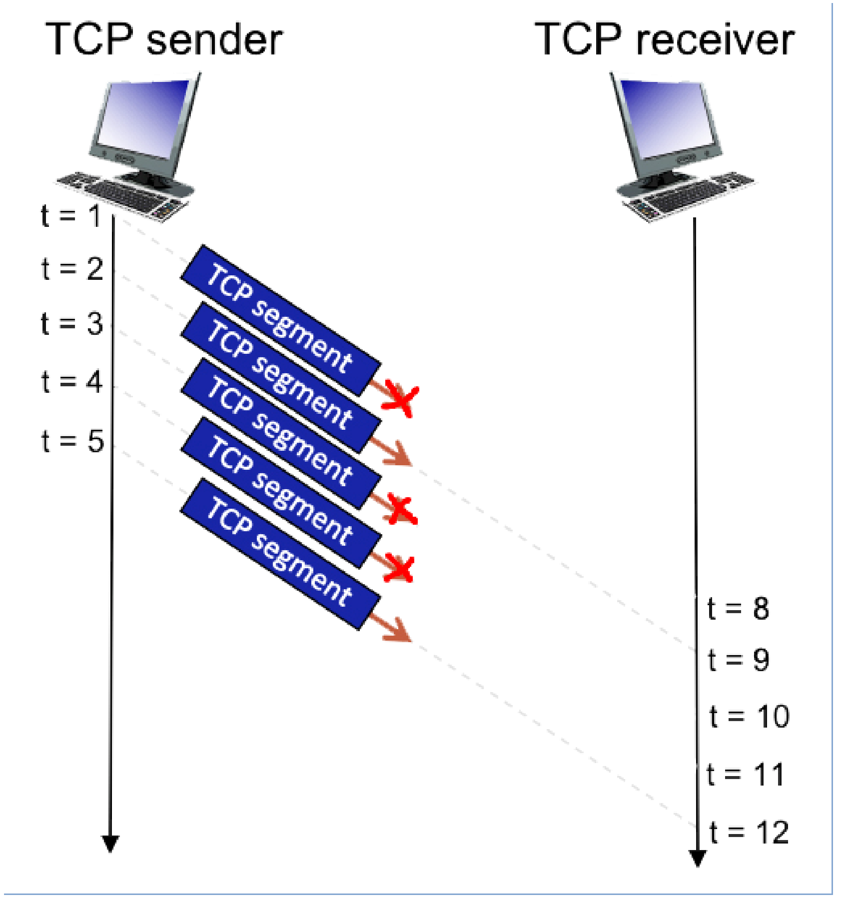
a) Give the sequence numbers associated with each of the 5 segments sent by the sender.

b) Give the ACK numbers the receiver sends in response to each of the segments.

Answer:

a) The sender's sequence numbers are:
   45,
   507,
   969,
   1431,
   1893

b) The receiver's ACKs are: 45 and 45



5. We learned that TCP never measures SampleRTT for retransmitted segments; it only does so for segments that are transmitted for the first time. What is the reason for this?

Answer: Consider a TCP connection between Hosts A and B.

- If Host A retransmits a segment S, the sender sends exactly the same data and the same sequence number as it did in the first transmission.
- Therefore, for the retransmitted segment, Host A expects to receive the same ACK number X as for the first transmission of S.
- When Host A receives ACK X, it cannot distinguish whether this ACK was sent in response to either

  a) the first transmission (because it was slow),

  or

  b) the retransmission.
- Case b) would be ok. But, in Case a) this would result in a wrong estimate of RTT, because it may be much faster than the actual RTT.