Exercise 8

December 20th, 2012

Chapter 4 outline

- 3.1 Transport-layer services
- 3.2 Multiplexing and demultiplexing
- 3.3 Connectionless transport: UDP
- 3.4 Principles of reliable data transfer
- 3.5 Connection-oriented transport: TCP
  - segment structure
  - reliable data transfer
  - **flow control**
  - **connection management**
- **3.6 Principles of congestion control**
- **3.7 TCP congestion control**

Flow vs. congestion control

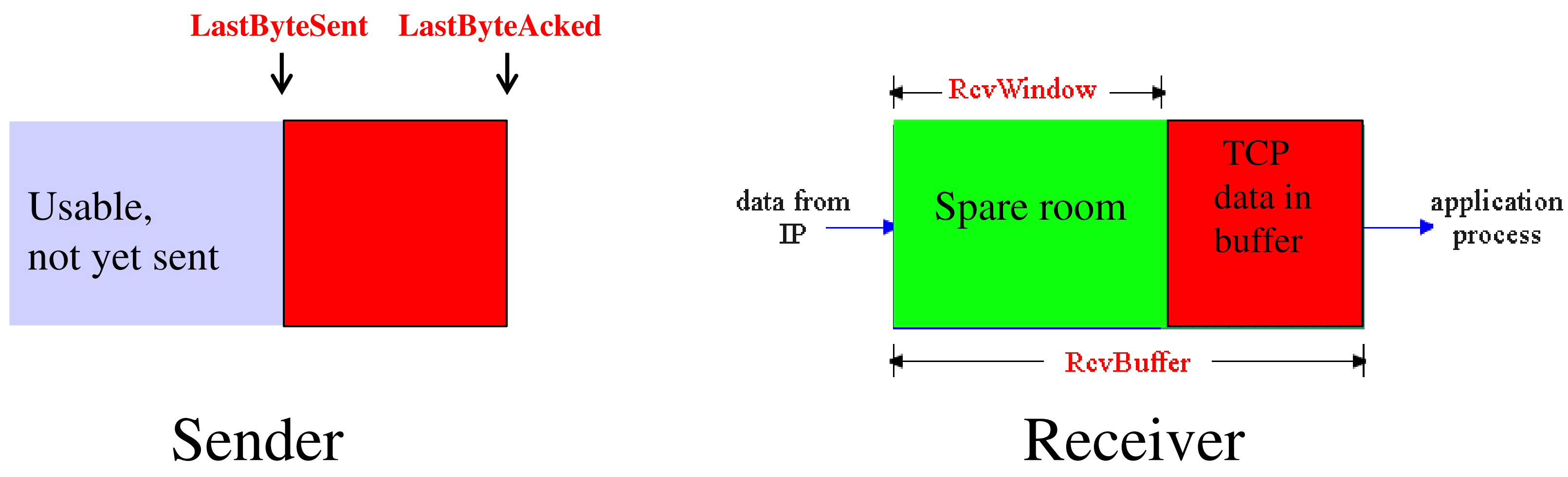
Question 7:

- What is the difference between flow control and congestion control?

Flow control

Flow Control:

- Prevent overwhelming the receiver by sending too much data.



LastByteSent – LastByteAcked <= rwnd

Flow vs. congestion control

Answer 7:

Flow Control:

- Prevent overwhelming the receiver by sending too much data.

Congestion Control:

- React on congestion in the network (on the path to the receiver).

TCP congestion control

Question 1:

- Suppose that in TCP, the sender window is of segment size N = 200, the base of the window is at sequence number 600, and the sender has just sent a complete window size of segments. Let RTT be the sender-to-receiver-to-sender round trip time of 200 ms and Maximum Segment Size MSS = 1 000 bytes.

a). Assuming no loss, what is the **throughput** (in terms of MSS and RTT and in terms of Megabit/s) of this message exchange?

b). Suppose TCP is in its congestion avoidance phase. Assuming no loss, what is the **window size** (in terms of segment) after the N = 200 segments are acknowledged?

TCP congestion control

Three questions raised by TCP congestion control approach

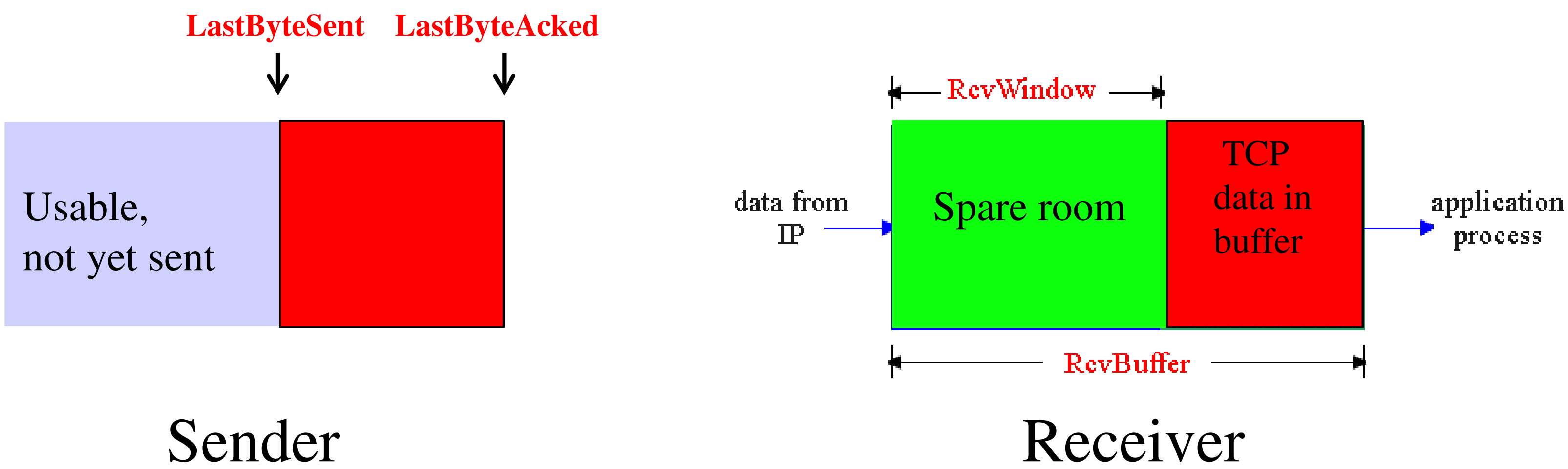
1. How does a TCP sender **limit its send rate** (at which it sends traffic into its connection)?

2. How does a TCP sender **perceive** that there is **congestion** on the path between itself and the destination?

3. What algorithm should the sender use to **change its send rate** as a function of perceived end-to-end congestion?

TCP congestion control

1. How does a TCP sender **limit its send rate** (at which it sends traffic into its connection)?



LastByteSent – LastByteAcked <= min{cwnd, rwnd}

TCP congestion control

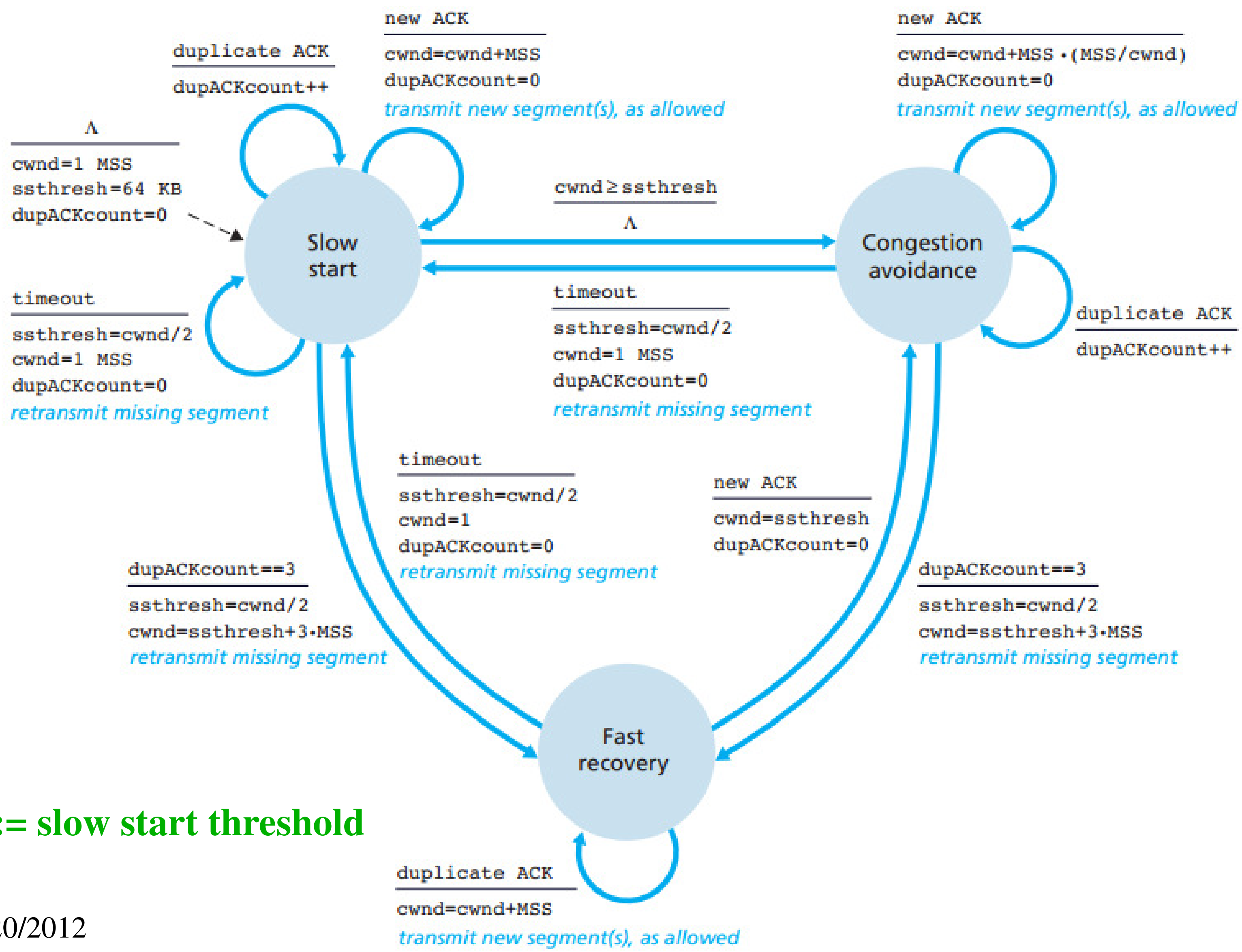
2. How does a TCP sender **perceive** that there is **congestion** on the path between itself and the destination?

- Timeout
- Three duplicated ACK's

TCP congestion control

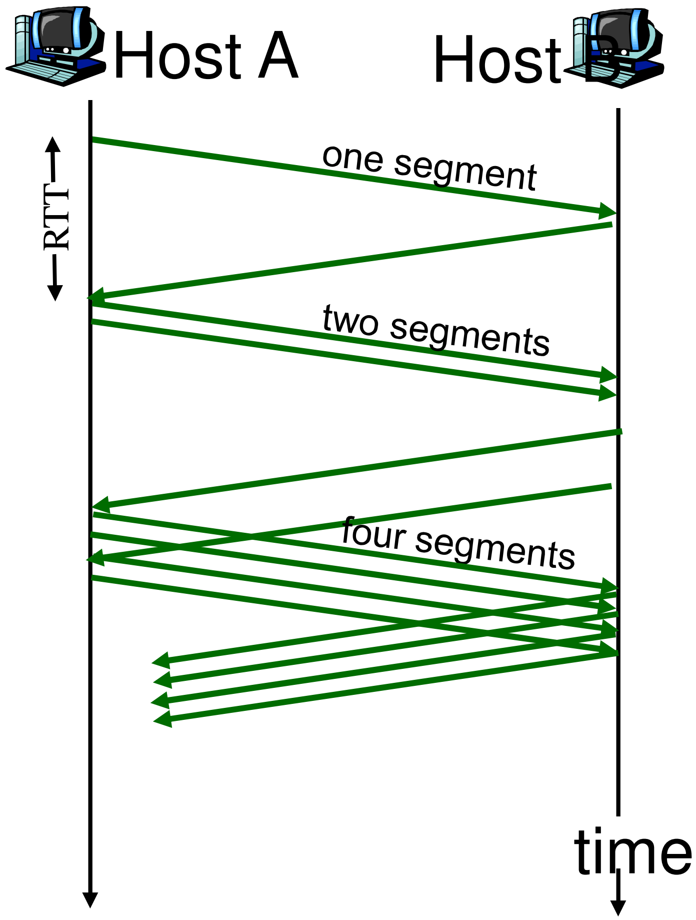
3. What algorithm should the sender use to **change its send rate** as a function of perceived end-to-end congestion?

TCP sender congestion control



ssthresh := slow start threshold

Why Slow start is quick?



TCP congestion control

Answer 1:

N=200, RTT=200ms, MSS=1000 bytes, sender just sent a complete window!

a). Assuming no loss, what is the throughput (in terms of MSS and RTT and in terms of Megabit/s) of this message exchange?

$$throughput = \frac{segments \cdot MSS}{RTT} = \frac{200 \cdot 8000 Bit}{0.2s} = 8000000 \frac{Bit}{s} = 8 \frac{MBit}{s}$$

TCP congestion control cont'd

Answer 1:

b). Suppose TCP is in its congestion avoidance phase. Assuming no loss, what is the window size (in terms of segment) after the N = 200 segments are acknowledged?

- Congestion Avoidance, in one RTT:

  Here we define cwnd := Congestion Window.

$$\mathrm{cwnd} = \mathrm{cwnd} + \mathrm{MSS} \cdot \left(\frac{\mathrm{MSS}}{\mathrm{cwnd}} \right)$$

- Each ack increases the cwnd by MSS/cwnd, which is 8000Bit/200=40Bit. As 200 acks arrive, the window is increased by 8000Bit which is exactly 1MSS, therefore cwnd=200+1! Note cwnd = 200+1 MSS!

TCP-Reno and Tahoe

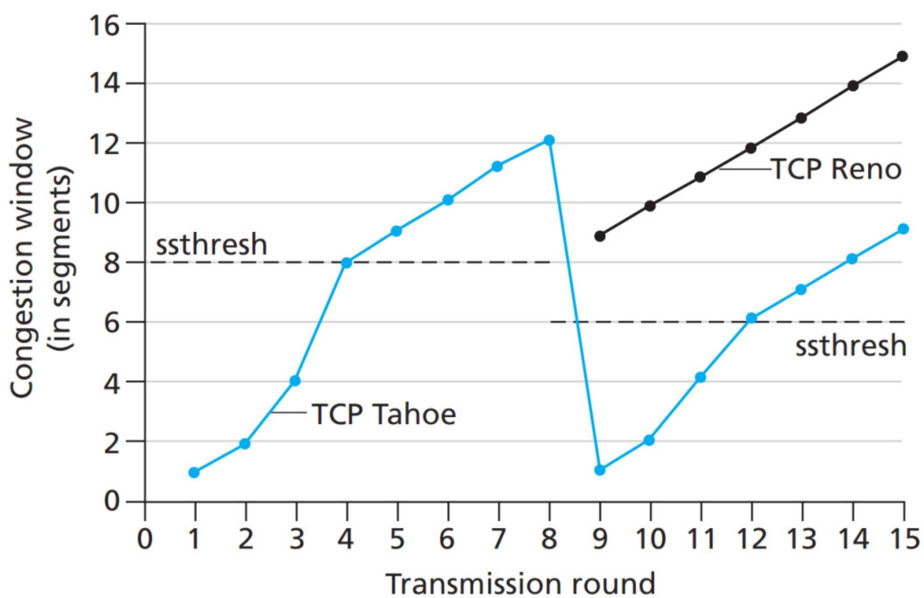
Question 2:

- What is the difference between the two congestion control algorithms TCP-Tahoe and TCP-Reno?

TCP-Reno and Tahoe

Answer 2:

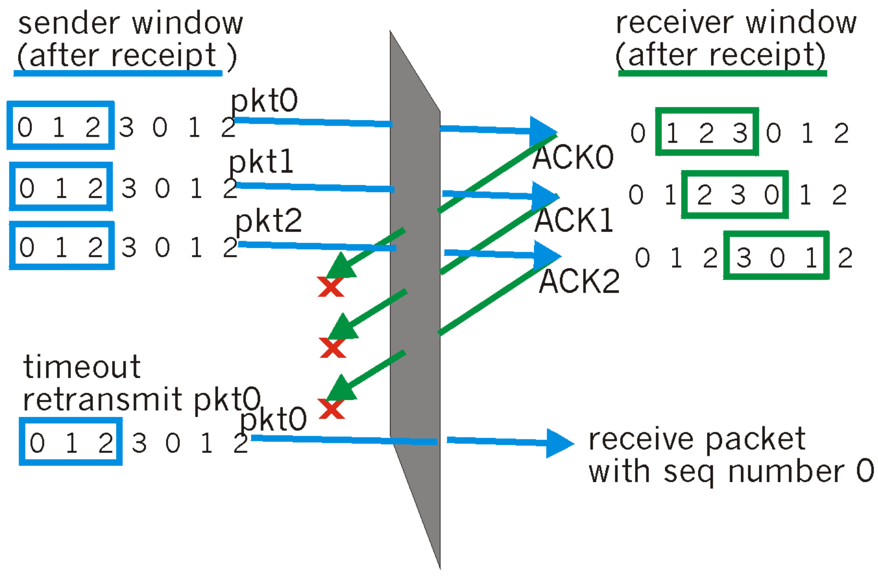
- Difference in handling timeouts and triple duplicate acks!
  - **Tahoe** always down to 1MSS,
  - **Reno** distinguishes:
    - 3 duplicate ACKs -> go down to 50% then CA,
    - timeout -> go down to 1MSS



Selective Repeat

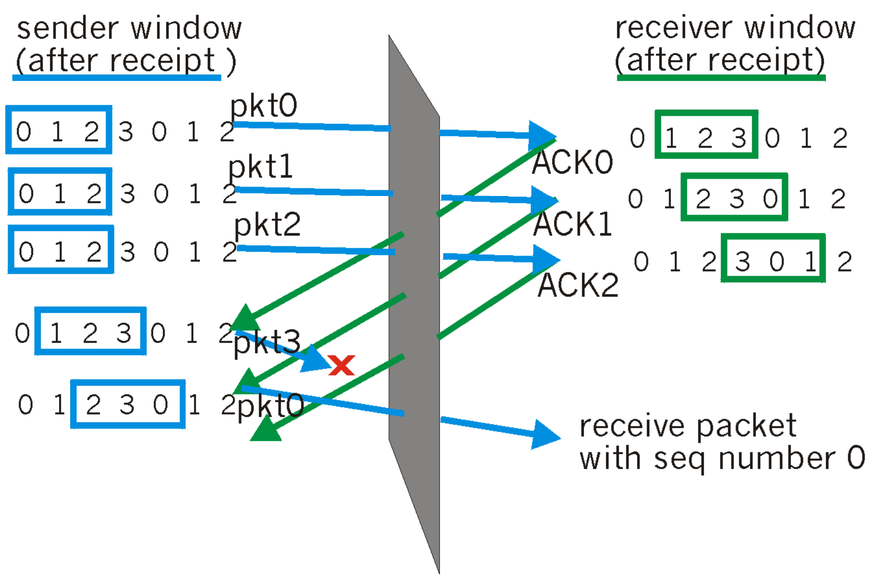
Question 3:

- Please explain the selective repeat dilemma and name a solution to prevent its occurrence.



sender window
(after receipt)
receiver window
(after receipt)
0 1 2 3 0 1 2
pkt0
pkt1
pkt2
ACK0
ACK1
ACK2
0 1 2 3 0 1 2
timeout
retransmit pkt0
receive packet
with seq number 0

(a)



sender window
(after receipt)
receiver window
(after receipt)
0 1 2 3 0 1 2
pkt0
pkt1
pkt2
pkt3
pkt0
ACK0
ACK1
ACK2
receive packet
with seq number 0

(b)

Selective Repeat

Answer 3:

- Dilemma occurs on a limited sequence range and large window size. Solution: Window size should be maximally half of the sequence range!

TCP vs. UDP

Question 4:

Please name at least three differences between UDP and TCP.

TCP vs. UDP

Answer 4:

1. TCP is connection oriented, UDP is not
2. TCP is a reliable data transfer protocol, UDP is not reliable
3. TCP enables in-order delivery, UDP does not guarantee in-order deliver
4. UDP has less overhead (lightweight) compared to TCP (heavy load due to ordering, window maintenance etc...)
5. TCP uses flow control, UDP does not
6. TCP uses congestion control, UDP does not

Choosing a protocol

Question 5:

- If you would like to transfer a file, which transport protocol would you use? Which protocol would you use for voice traffic?

Choosing a protocol

Answer 5:

- File: TCP as it is reliable, in-order delivery. Receiver can directly pipe data contents into file
- Voice: UDP as it is lightweight, small in-orders cannot be heard and reliability has no advantage if delivery takes to long

TCP fast retransmit

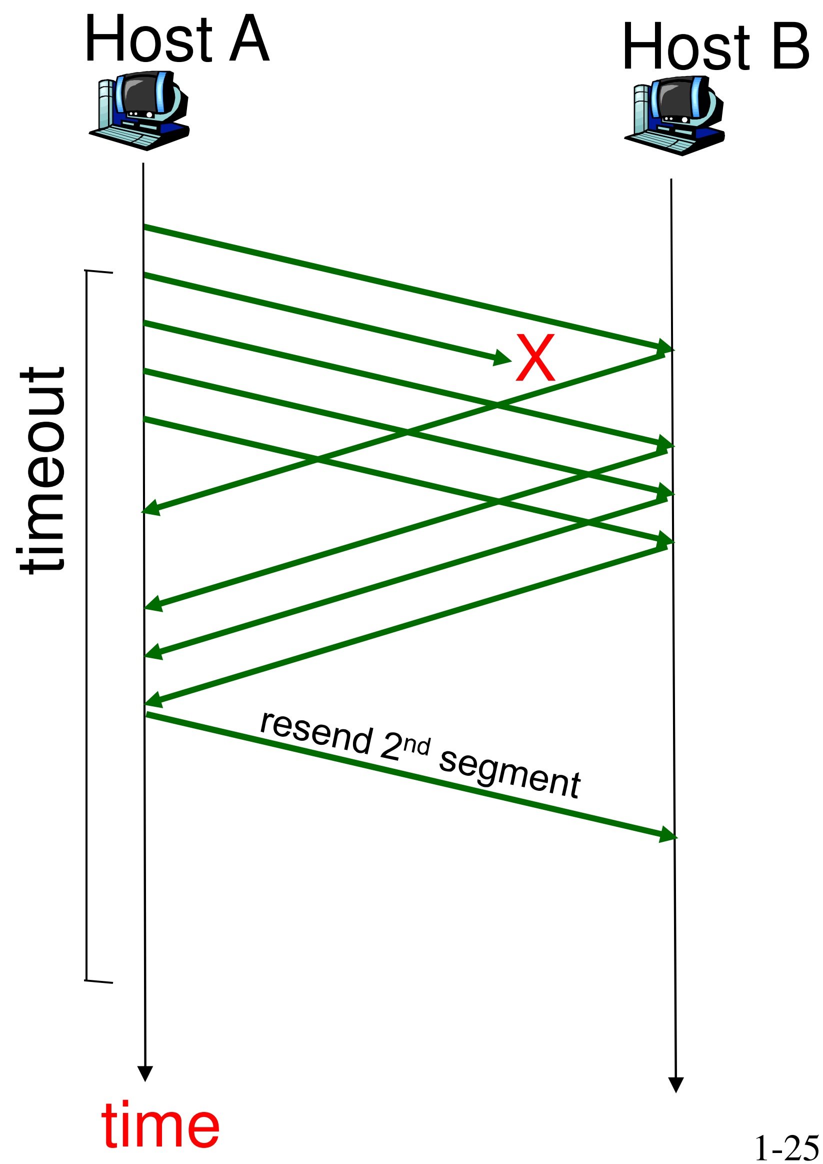
Question 6:

- Please explain TCP fast retransmit.

TCP fast retransmit

Answer 6:

- Time-out period often relatively long:
- Detect lost segments via three duplicate ACKs.
- Fast retransmit: resend segment before timer expires, directly after receiving three duplicate acks

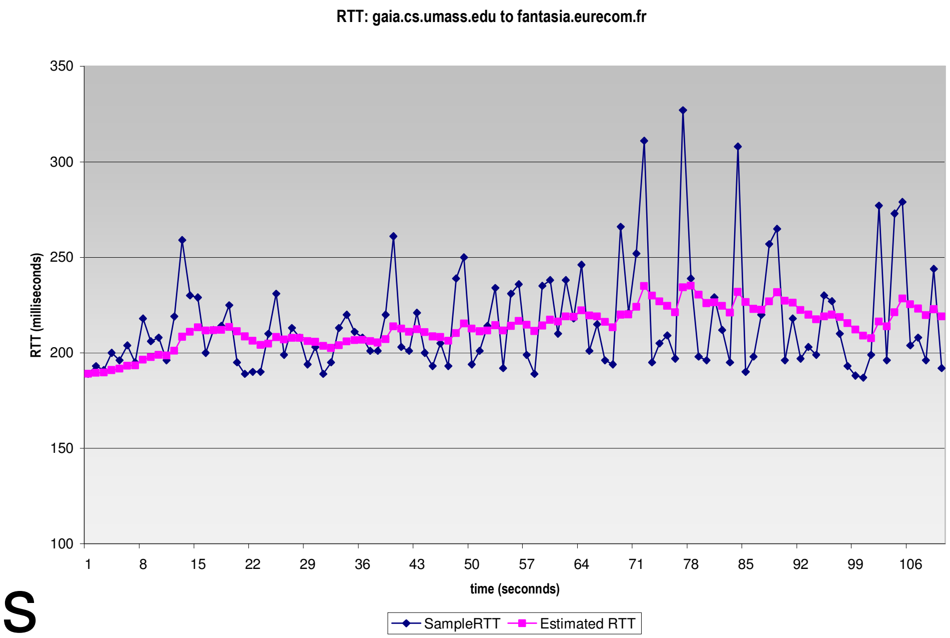


Estimated vs. sampled RTT

Question 8:

- Why is an EstimatedRTT used to calculate the TCP timeout instead of the recently sampled RTT?

Estimated vs. sampled RTT



Answer 8:

- Exponential weighted moving average
- influence of past sample decreases exponentially fast

- SampleRTT fluctuates too much. EstimatedRTT + safety margin is a safer guess to set the timer.

- That’s all and thanks for your attention!