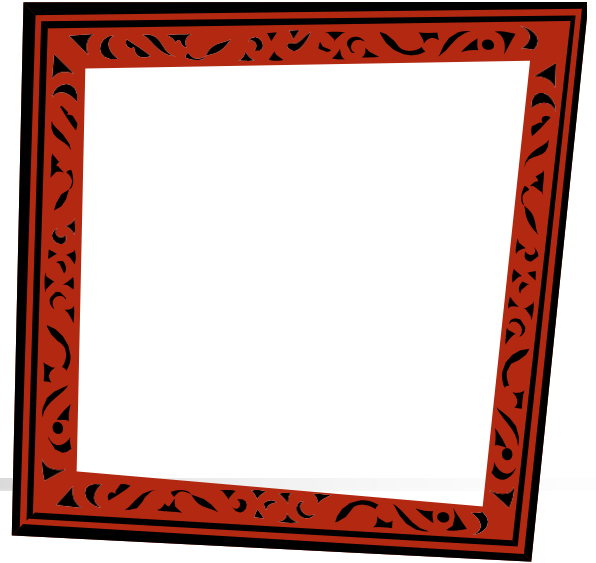


Discussion #14

EE450



Sample Problems
-TCP Congestion Control



Review: TCP Congestion Control

- $\text{MaxWindow} = \text{MIN} (\text{CongestionWindow}, \text{AdvertisedWindow})$
- $\text{EffectiveWindow} = \text{MaxWindow} - (\text{LastByteSent} - \text{LastByteAcked})$
- $\text{Increment} = \text{MSS} \times (\text{MSS} / \text{CongestionWindow})$
- $\text{CongestionWindow} + = \text{Increment}$



Problem #4

- Consider a simple congestion-control algorithm that uses linear increase and multiplicative decrease but not slowstart, that works in units of packets rather than bytes, and that starts each connection with a congestion windows equal to one packet.
- Give a detailed sketch of this algorithm. Assume the delay is latency only, and that when a group of packets is sent, only a single ACK is returned. Plot the congestion window as a function of round-trip times for the situation in which the following packets are lost: 9, 25, 30, 38 and 50. For simplicity, assume a perfect timeout mechanism that detects a lost packet exactly 1 RTT after it is transmitted. “Go-back-N”



Solution

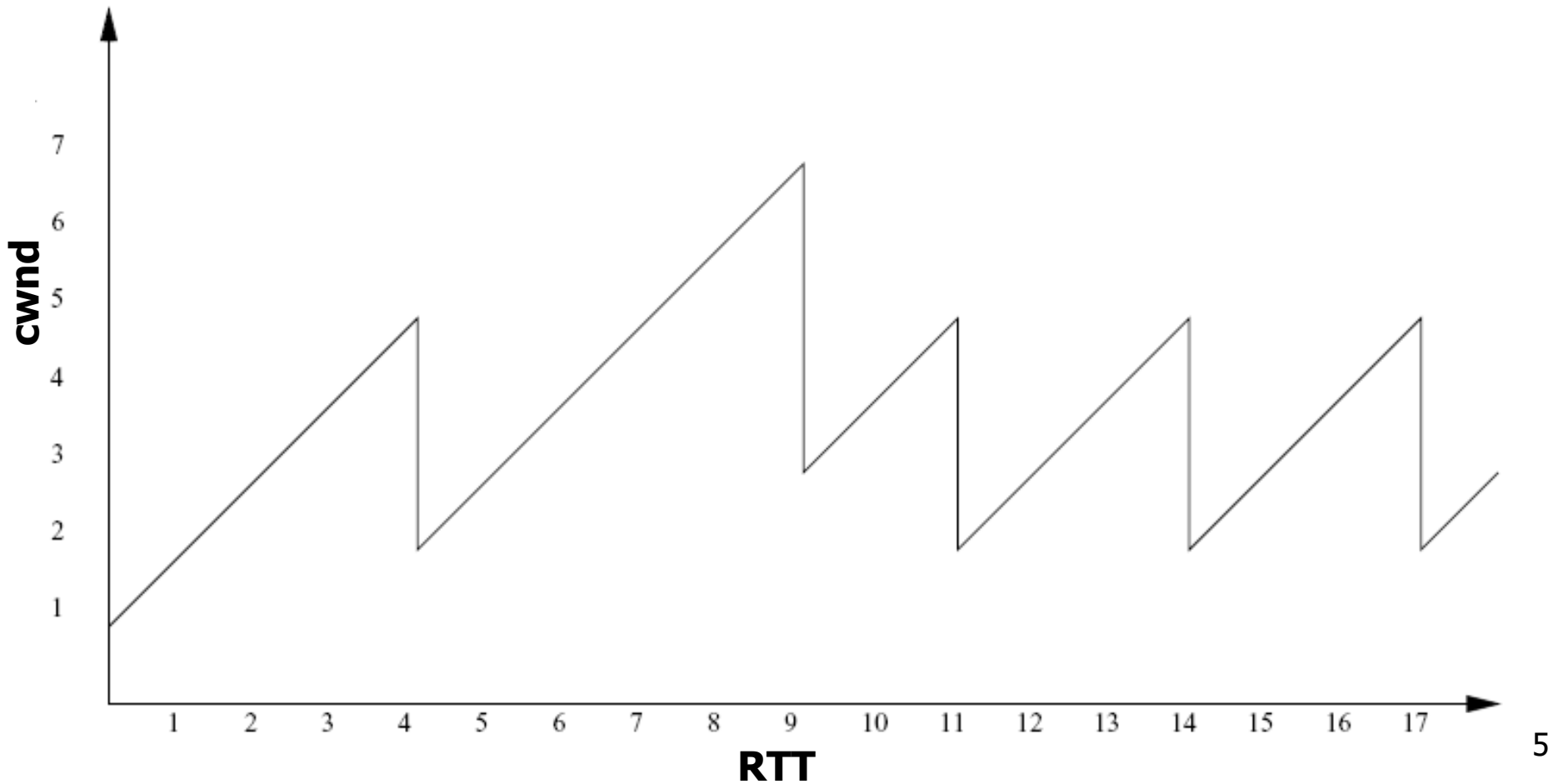
The window size is initially 1; when we get the first ACK it increases to 2. At the beginning Of the second RTT we send packets 2 and 3. When we get their ACKs we increase the window size to 3 and send packets 4, 5 and 6. When these ACKs arrive the window size becomes 4. Now, at the beginning of the fourth RTT, we send packets 7, 8, 9, and 10; by hypothesis packet 9 is lost. So, at the end of the fourth RTT we have a timeout and the window size is reduced to $4/2 = 2$. Continuing we have:

RTT	5	6	7	8	9
Sent	9-10	11-13	14-17	18-22	23-28

Again the congestion window increases up until packet 25 is lost, when it is halved, to 3, at the end of the ninth RTT.

Solution

The plot below shows the window size vs. RTT.





Problem#7: Description

- Assume that TCP implements an extension that allows window sizes much larger than 64 KB. Suppose that you are using this extended TCP over a 1Gbps link with a latency of 100 ms to transfer a 10-MB file. If TCP send 1KB packets (assuming no congestion and no lost packets):
 - (a) How many RTTs does it take until slowstart opens the send window to 1MB?
 - (b) How many RTTs does it take to send the file?
 - (c) If the time to send the file is given by the number of required RTTs multiplied by the link latency, what is the effective throughput for the transfer? What percentage of the link bandwidth is utilized?



Problem#7: Solution, Part(a)

(a) In slowstart, the size of the window doubles every RTT. At the end of the i th RTT, the window size is 2^i KB. It will take 10 RTTs before the send window has reached 2^{10} KB = 1 MB.

In other words, to reach a window size of 1MB, it will take:

$$\lceil \log_2 \frac{\text{SendWindow}}{\text{SegmentSize}} \rceil \text{RTT} = \lceil \log_2 \frac{1\text{MB}}{1\text{KB}} \rceil \text{RTT} = \lceil \log_2 1024 \rceil \text{RTT} = 10\text{RTT}$$



Problem#7: Solution

i	Time slot	Amount of Data sent	Send window in segments At the end of the ith RTT	Send window in bytes At the end of the ith RTT
1	1 st RTT	$2^0 \times 1\text{KB}$	$2^1 = 2$	$2^1 \times 1\text{KB}$
2	2 nd RTT	$2^1 \times 1\text{KB}$	$2^2 = 4$	$2^2 \times 1\text{KB}$
3	3 rd RTT	$2^2 \times 1\text{KB}$	$2^3 = 8$	$2^3 \times 1\text{KB}$
4	4 th RTT	$2^3 \times 1\text{KB}$	$2^4 = 16$	$2^4 \times 1\text{KB}$
5	5 th RTT	$2^4 \times 1\text{KB}$	$2^5 = 32$	$2^5 \times 1\text{KB}$
6	6 th RTT	$2^5 \times 1\text{KB}$	$2^6 = 64$	$2^6 \times 1\text{KB}$
7	7 th RTT	$2^6 \times 1\text{KB}$	$2^7 = 128$	$2^7 \times 1\text{KB}$
8	8 th RTT	$2^7 \times 1\text{KB}$	$2^8 = 256$	$2^8 \times 1\text{KB}$
9	9 th RTT	$2^8 \times 1\text{KB}$	$2^9 = 512$	$2^9 \times 1\text{KB}$
10	10 th RTT	$2^9 \times 1\text{KB}$	$2^{10} = 1024$	$2^{10} \times 1\text{KB}$



Problem#7: Solution, Part(b)

(b) Amount of data sent during the first 10 RTTs:

$$\left(\sum_{i=1}^{10} 2^{i-1}\right) \times 1KB = (2^{10} - 1) \times 1KB = (1024 - 1) \times 1KB \cong 1MB$$

So we need to send 9MB more to completely transfer the 10MB file. Since we have not reached the capacity of the network, the slowstart continues to double the window every RTT.



Problem#7: Solution

The amount of data transferred during each RTT is well below the capacity of the link in one RTT. Therefore the file is completely transferred in 14th RTTs.

i	Time slot	Amount of Data sent
11	11 th RTT	$2^{10} \times 1\text{KB} = 1\text{MB}$
12	12 th RTT	$2^{11} \times 1\text{KB} = 2\text{MB}$
13	13 th RTT	$2^{12} \times 1\text{KB} = 4\text{MB}$
14	14 th RTT	A partial window of 2MB data



Problem#7: Solution, Part(c)

(c) It takes $14 \times \text{RTTs} = 14 \times 100\text{ms} = 1.4 \text{ sec}$ to send the file.

The effective throughput is:

$$(10\text{MB}/1.4 \text{ sec}) = 7.1\text{MBps} = 57.1\text{Mbps}.$$

The utilization is:

$$(57.1\text{Mbps}/1\text{Gbps}) = 5.7\% \text{ of the link bandwidth}.$$



Problem#8: Description

Consider the TCP congestion-control algorithm that uses slow-start but has no fast retransmit/fast recovery. The algorithm starts each connection with a window size equal to one segment. Assume the only delay is latency only, and a single ACK is sent for a group of segments. Assume a perfect timeout mechanism that detects a lost segment exactly 1 RTT after it is transmitted. Round up the fractional numbers encountered in window sizes to greater integers (like 2.5 \rightarrow 3)

a) Fill in the table below assuming segments 7, 12 and 15 are lost.

RTT	1	2	3	4	5	6	7	8	9	10
Sent Segments										

b) Sketch the variation of congestion window with respected to time

Problem#8: Solution, Part(a)

a)

RTT	0	1	2	3	4	5	6	7	8	9	10	11
Sent Packets	1	2,3	4,5,6,7	4	5,6	7,8,9	10,11,12,13	10	11,12	13,14,15	13	14,15
CW (At the end of RTT)	2	4	1	2	3	4	1	2	3	1	2	3
Threshold			$4/2-2$				$4/2-2$			$3/2 \rightarrow 2$		

Remember:

- A single ACK is sent for a group of segments. When a segment is lost, all the segments of that group need to be resent.
- After the threshold is crossed the increase in CW is linear.

Problem#8: Solution, Part(b)

b)

The graph is as follows:

