5. This problem deals with TCP congestion control. For the data flow from Source to Sink shown, assume the following: Source is either in multiplicative decrease or additive increase (i.e., congestion window decreases by half upon timeout and increases by 1 upon receiving ack of previous window). Timeout value at the Source is constant at 250 ms. Segment size is 1 KByte. Roundtrip time consists entirely of the delay encountered in the router queue (i.e., zero delay encountered by acks). The router queue holds at most 7 packets and needs 10 ms to serve a packet. Queue overflow is the only cause of packet loss. No other traffic enters the router queue.

a. How many seconds does it take to transfer a 20Kbyte file, starting with congestion window size of 2 packets.

b. How many seconds does it take to transfer a 10 Mbyte file, starting with congestion window size of 2 packets.

We first look at the behavior for a long transfer. Below, time stands for elapsed time (in ms), W_c stands for congestion window, RTT stands for measured roundtrip time (in ms), and tns stands for time to next send (in ms).

time	cong window	roundtrip time	time to next send	packets sent	
0	2	20	20	0..1	
20	3	30	30	2..4	
50	4	40	40	5..8	
90	5	50	50	9..13	
140	6	60	60	14..19	
200	7	70	70	20..26	
270	8	timeout	250	27..34	(34 lost)
520	4	40	40	34..37	
560	5	50	50	38..42	
610	6	60	60	43..48	
670	7	70	70	49..55	
740	8	timeout	250	56..63	(63 lost)
990	4	40	40	63..66	

Thus, Wc evolves in "saw-tooth" fashion, cycling through 4,5,6,7,8. Each "saw-tooth" cycle lasts 470 ms (= 740-270) and transfers 29 packets (30 packets are sent but 1 is lost).

Transfer time for 20 Kbyte file

a. 20 packets suffice (since each packet carries 1 Kbyte). From above, we see that the transfer (including the ack time) is completed by time 200 ms; no timeouts are involved.

Transfer time for 10 Mbyte file

b. 10 Mbyte file requires 10240 packets, which means 10240/29 cycles (ignoring the particulars of the starting and ending cycle), which means 10240*470/29 ms, which is approximately 160 seconds (since 470/29 is a bit more than 470/30, which is 15.7).

1. Consider sending a large file from a host to another over a TCP connection that has no loss. Suppose TCP uses AIMD for its congestion control without slow start. Assuming cwnd increases by 1 MSS every time a batch of ACKs is received and assuming approximately constant round--trip times

a. how long does it take for cwnd to increase from 5 MSS to 11 MSS (assuming no loss events)?

b. What is the average throughput (in terms of MSS and RTT) for this connection up through time = 6 RTT?

Solutions:

a. It takes 1 RTT to increase CongWin to 6 MSS; 2 RTTs to increase to 7 MSS; 3 RTTs to increase to 8 MSS; 4 RTTs to increase to 9 MSS; 5 RTTs to increase to 10 MSS; and 6 RTTs to increase to 11 MSS.

b. In the first RTT 5 MSS was sent; in the second RTT 6 MSS was sent; in the third RTT 7 MSS was sent; in the fourth RTT 8 MSS was sent; in the fifth RTT, 9 MSS was sent; and in the sixth RTT, 10 MSS was sent. Thus, up to time 6 RTT, 5+6+7+8+9+10 = 45 MSS were sent (and acknowledged). Thus, we can say that the average throughput up to time 6 RTT was (45 MSS)/(6 RTT) = 7.5 MSS/RTT.

3. Suppose two TCP connections A and B are sharing a link of capacity 10 Mbps. They both have the same round-trip time (RTT) of 100 ms and the same segment size of 1 Kbits. They both follow an additive increase and multiplicative decrease congestion control algorithm, increasing the window size by one segment every RTT and halving the window size whenever they detect loss. You can ignore the slow start phase in answering the questions below.
   a. Suppose initially flow A is sending at rate of 4 Mbps while B is sending at a rate of 2 Mbps. By drawing a carefully labeled 2-D plot of the evolution of their rates with time or otherwise, argue that in the long run they will each get the same share of the link capacity. Hence, TCP is fair.

   b. What is the long run throughput each connection gets? (You can assume, for this part only, that loss is immediately detected when the aggregate transmission rate of the two connections exceeds the link capacity.) How does this performance compare to that under an explicit rate congestion control algorithm where the available bandwidth is explicitly allocated to the connections?

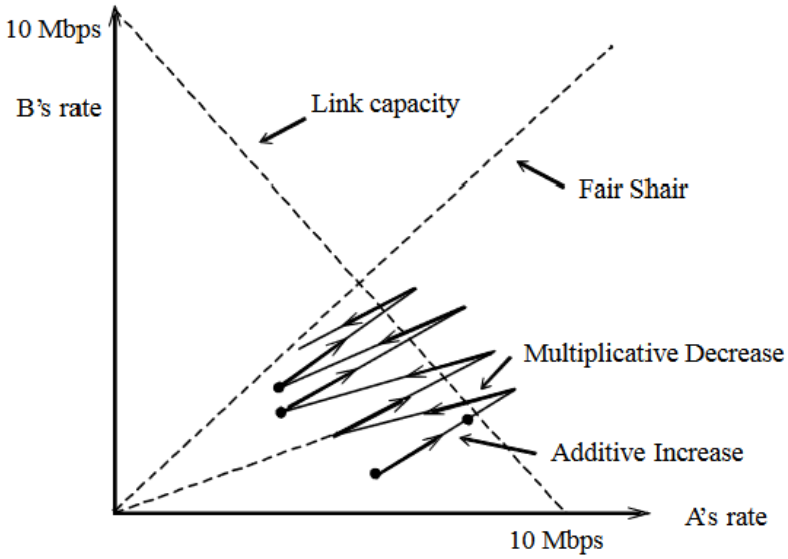
   c. Suppose the window size is reduced by one third instead of halved every time there is a packet loss. Is the congestion control algorithm still fair? Explain.

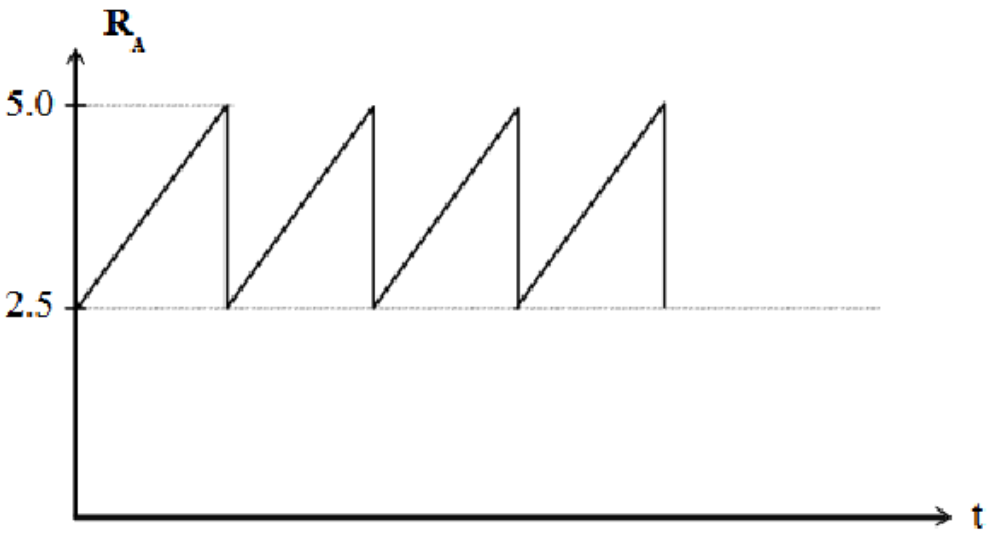
   d. Bob has taken EE555 and decides that a multiplicative decrease is too drastic, resulting in too much loss in throughput. Instead, he suggests that the decrease should be additive as well, decreasing the window size by 1 segment size every round-trip time until there is no loss. Is this algorithm fair, with the initial rates as in part (a)? Justify your answer.

   e. Consider again the basic additive increase multiplicative decrease algorithm, but now the RTT of A is 200 ms and the RTT of B is 100ms. What is the ratio of the long run throughputs of the two connections? Is TCP still fair? Justify your answer.

Solutions: a, b)

In the addition increase phase, the gap between the transmission rates of A and B remains unchanged. Every time this is loss the gap is reduced by $\frac{1}{2}$ So often N such losses event, the gap is reduced by $\left(\frac{1}{2}\right)^N$. As $N \to \infty$, the gap goes to zero. This is also shown in the following figure:



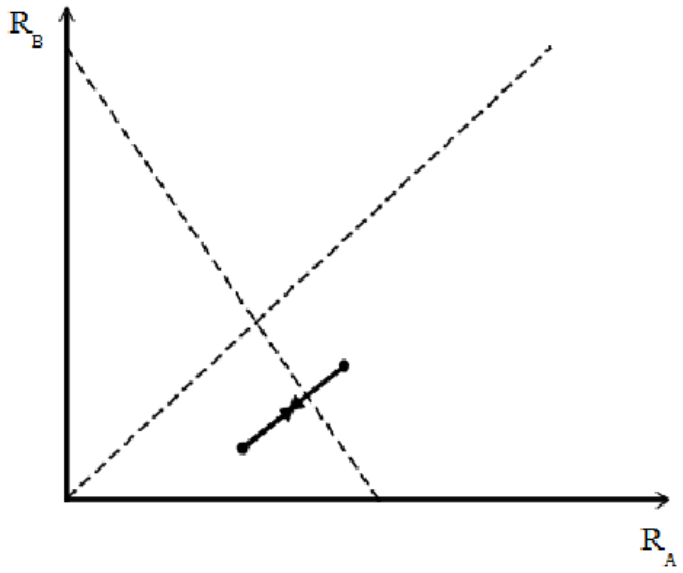


$$\text{Average rate} = \frac{5 + 2.5}{2} = 3.75 Mbps$$

c. Yes, the argument in (a) does not depend on the factor of ½ ; any constant factor $\alpha < 1$ would work.

d.

No, the gap between the users' rates remains constant.



e.

The window of A increases twice as slowly as B's. The rate of A is equal to $\frac{W}{RTT}$, so the rate of A increases $\frac{1}{4}$ as slowly as rate the of B. Asymptotically, throughput of A is $\frac{1}{4}$ that of B. No, TCP is not fair.