Problem 1

Host A and B are directly connected with a 100 Mbps link. There is one TCP connection between the two hosts, and Host A is sending to Host B an enormous file over this connection. Host A can send its application data into its TCP socket at a rate as high as 120 Mbps but Host B can read out of its TCP receive buffer at a maximum rate of 50 Mbps. Describe the effect of TCP flow control.

Write your solution to Problem 1 in this box

TCP flow control is defined as the mechanism that the receiver uses to dictate how much data the sender is sending. The receiver controls the sender because the receiver only has a certain amount of available buffer space, called the receive window or rwnd. If the sender sends data at a rate that is faster than the rate at which the receiver processes the data from the buffer, then the buffer will overflow and cause packet drops and retransmissions, which in turn cause network congestion.

In this case, host B is the receiver and it can only process data at a rate of 50Mb/s. As a result, host B will limit the sending rate of host A to 50Mb/s, even though host A can send at a rate of 120Mb/s and the link can handle data at a rate of 100Mb/s.

Problem 2

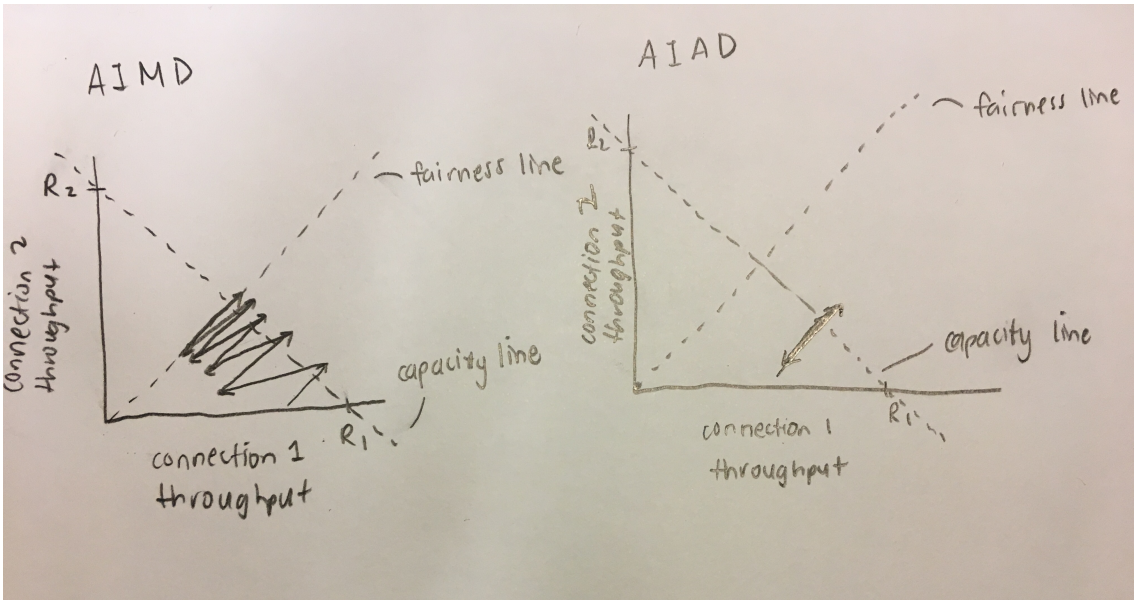
Suppose that instead of a multiplicative decrease, TCP decreased the window size by a constant amount. Would the resulting AIAD algorithm converge to an equal share algorithm? Justify your answer using a graphical diagram similar to Slide 100 of the lecture.

Write your solution to Problem 2 in this box

Instead of a multiplicative decrease, an algorithm using additive increase or AIAD would not converge to an equal share algorithm. This can be shown in a graph of the throughput of two different connections on the same network that are attempting to achieve the highest possible throughput without congestion. The additive increase algorithm causes the both connection's throughputs to increase linearly until it reaches the capacity line.

Then, the multiplicative decrease algorithm causes the line to move towards the origin. From that point, the line will again have a linear growth until it reaches the capacity line. Because it moves towards the origin, this means it will also move closer towards the equal bandwidth or fairness line. This repeats until the line oscillates along the equal bandwidth/fairness line.

In the case of an additive decrease algorithm, the line doesn't move towards the origin, and instead moves back linearly, along its previous path. As a result, it never moves closer towards the equal bandwidth or fairness line. Thus, the AIAD algorithm does not converge to an equal share algorithm.



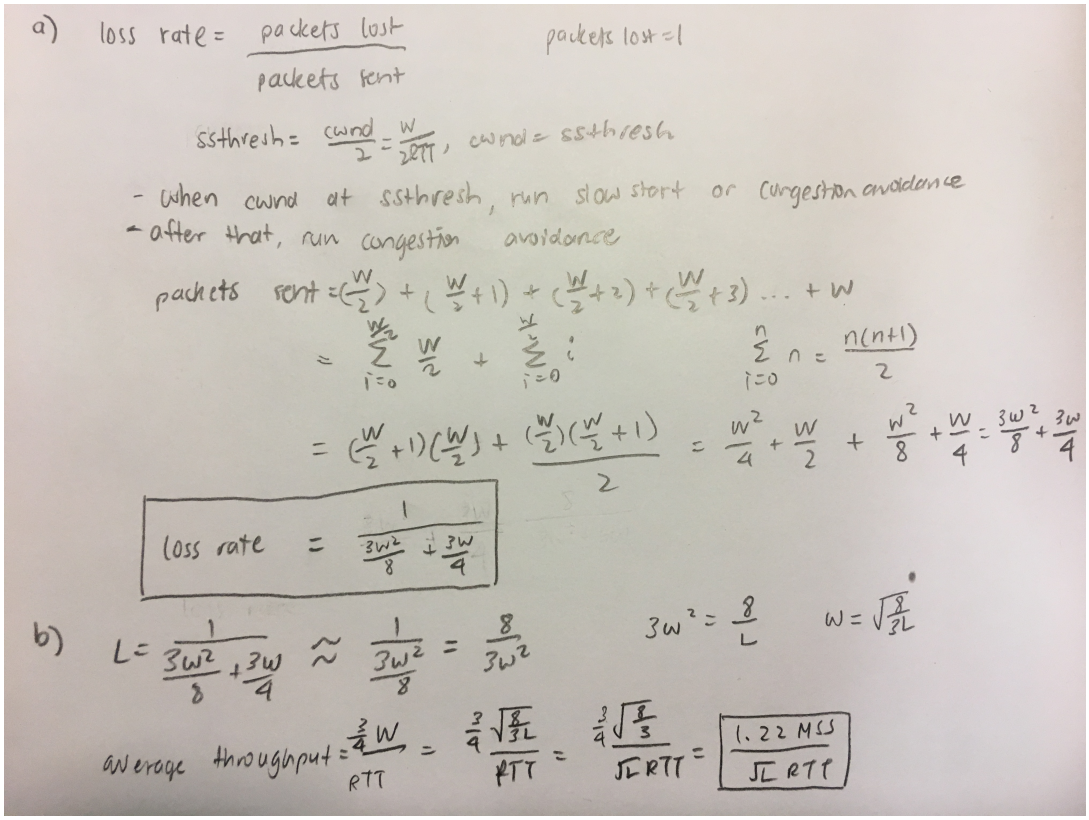
Problem 3

Recall the macroscopic description of TCP throughput, in the period of time from when the connection's rate varies from W/(2 RTT) to W/RTT, only one packet is lost (at the very end of the period).

(a) Show that the loss rate (fraction of packets lost) is equal to L=lossrate= $1/(3/8W^2 + 3/4W)$

(b) Use the result above to show that if a connection has loss rate L, then its average rate is approximately given by $\simeq 1.22 \times MSS/(RTT \times \sqrt{L})$

Write your solution to Problem 3 in this box



Problem 4

You are designing a reliable, sliding window, byte-stream protocol similar to TCP. It will be used for communication with a geosynchronous satellite network, for which the bandwidth is 1 Gbps and the RTT is 300 ms. Assume the maximum segment lifetime is 30 seconds.

(a) How many bits wide should you make the ReceiveWindow and SequenceNum fields? (ReceiveWindow is also called "Advertised Window" in some other textbooks.)

(b) If ReceiveWindow is 16 bits, what upper bound would that impose on the effective bandwidth?

Write your solution to Problem 4 in this box

a)
Receive-window should be larger than the number of packets in-flight or in the network at a given time. In order to maximize the utilization of the network, receive-window should be larger than the product of delay and bandwidth:

Delay * bandwidth
= RTT * bandwidth
= 300ms * 1Gb/s
= 0.3s * 1 000 000 000 bits/s
= 300 000 000 bits/s
= 300Mb/s
= 37.5MB/s

We need at least 26 bits in the receive-window in order to handle a delay-bandwidth product of 37.5MB/s.

SequenceNum should be large enough so that the sequence number does not wrap around before delayed or lost segments arrive at the receiver. The maximum segment lifetime is the maximum amount of time that a segment lives in the network. So, the sequence number must be larger than the product of maximum segment lifetime and delay:

Maximum segment lifetime * bandwidth
= 30s * 1Gb/s
= 30s * 1000 000 000b/s
= 30 000 000 000b
= 3 750 000 000B
= 3.75GB

We need at least 31 bits in the sequence number in order to handle a maximum segment lifetime and bandwidth product of 3.75GB.

b)
If the receive-window is bounded to 16 bits, then the maximum possible amount of data in-flight would be 2^16B / RTT = 65 536B / 0.3s = 218kB/s!

Problem 5

Consider the evolution of a TCP connection with the following characteristics. Assume that all the following algorithms are implemented in TCP congestion control: slow start, congestions avoidance, fast retransmit and fast recovery, and retransmission upon timeout. If ssthresh equals to cwnd, use the slow start algorithm in your calculation.

- The TCP receiver acknowledges every segment, and the sender always has data segments available for transmission.
- The RTT is 100 ms for all transmissions, consists of the network latency of 60 ms in sending a segment (header and payload) from the sender to the receiver and 40 ms in sending an acknowledgment (header only) from the receiver to the sender. Ignore packet-processing delays at the sender and the receiver.
- Initially ssthresh at the sender is set to 5. Assume cwnd and ssthresh are measured in segments, and the transmission time for each segment is negligible.
- Retransmission timeout (RTO) is initially set to 500ms at the sender and is unchanged during the connection lifetime.
- The connection starts to transmit data at time t = 0, and the initial sequence number starts from 1. TCP segment with sequence number 6 is lost once (i.e., it sees segment loss during its first transmission). No other segments are lost during transmissions.

What are the values for cwnd and ssthreshold when the sender receives the TCP ACK with number 15? Show your intermediate steps or your diagram in your solution.

Problem 5 continued on next page...

Write your solution to Problem 5 in this box