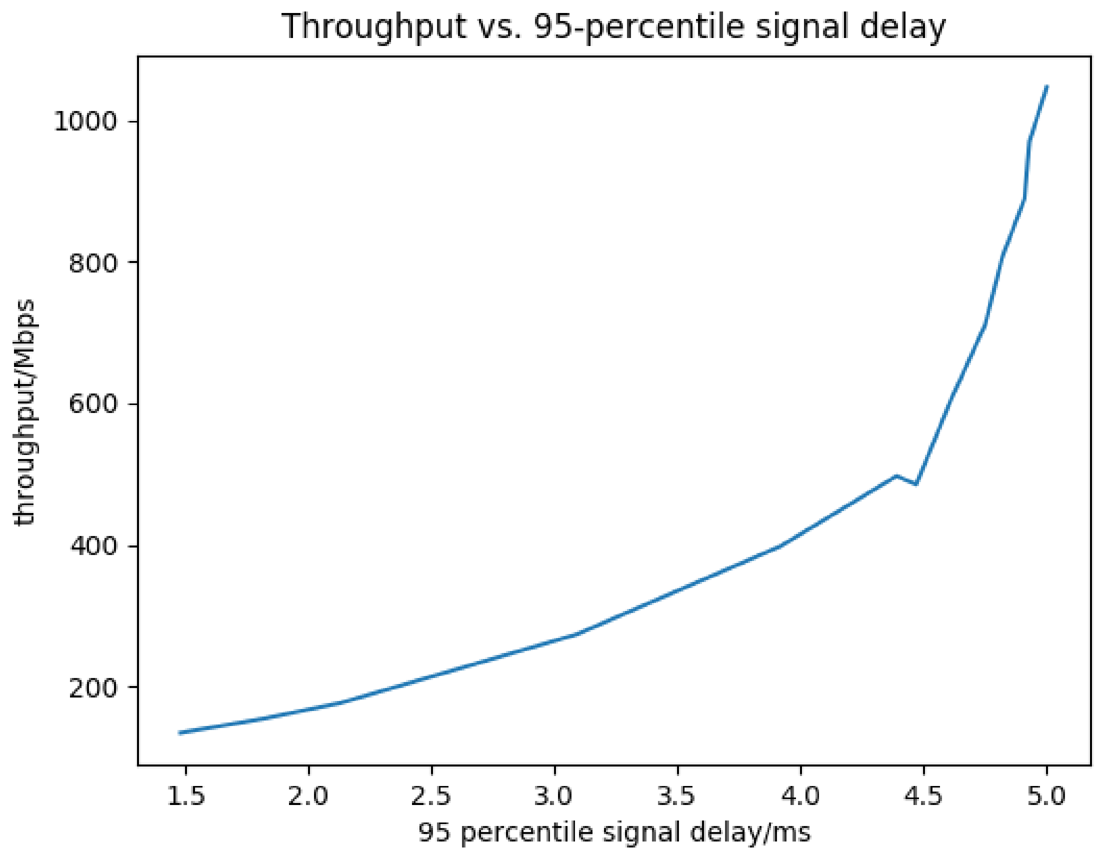
REPORT ASSIGNMENT _3

Muhammad Abu Bakar Aziz **20100117**
Nabeel Rizvi **20100049**

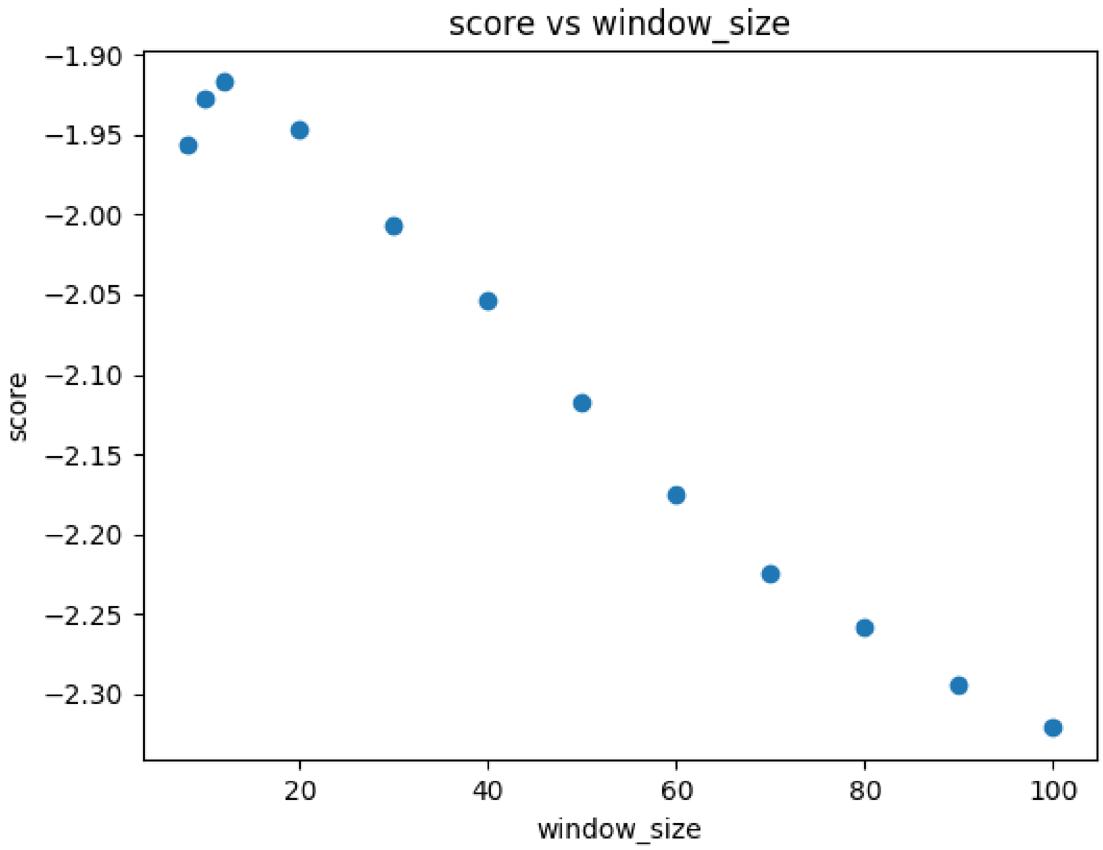
Congestion Control Protocols.

Part 1:



For different window sizes, I plotted graph of throughput vs 95 percentile signal delay. It is evident from the graph that as throughput increases, signal delay also increases. I ran tests for window sizes starting from 10 and increased till 100 with 10 size intervals. As throughput increases, the congestion starts to happen in the network. More and more queues start to build up at the buffer in the router which significantly increases the queuing delay.

To find the maximum score (log(throughput /delay) vs window_size), I plotted the graph of score vs window size which is attached below.



It is seen that score was greater for lower window values compared to higher window values. So, on window size=10 the score was (-1.94). So, I conduced further tests around this window size to find the window size at which the score was maximum. Window size 8 further decreased the score. I found out that the maximum score happened at window size 12 and score was -1.92. As we further increase the window size from 12, even though the throughput increases, the signal delay increases at a greater rate and hence (throughput/delay) decreases. Below the window size of 12, even though the signal delay decreases, but throughput decreases at a faster rate, and hence throughput/delay decreases. Hence, window size of 12 gives the max score.

<u>Part :-2 AIMD:</u>

In this part, as mentioned, I tried to implement simple AIMD scheme and selected values that give the best score. I ran multiple tests for the scheme. Below is the table containing some of the tests with their results. In AIMD, initial phase is slow start and once window size reaches ssthresh, congestion avoidance phase is entered where window size is additively increased. However, I didn't only additive increased the window size, but also changed values on different tests to check for increase in score.

No:	Thresh on timeout	window_size on timeout	Congestion avoidance increase in window size		On Timeout/s	Throughput /Mbits/s	Delay /ms
1	Constant =20	w_size/2	1/w_size	+	85	1.72	155
2	Constant =40	w_size/2	1/w_size		80	1.78	175
3	w_size/2	3	1/w_size		105	2.33	97
4	w_size/2	2	2/w_size		105	2.66	113
5	w_size/2	3	2/w_size		105	2.81	110
6	w_size/2	2	2.5/w_size		100	2.43	103
7	w_size/2	3	2.5/w_size		105	2.79	118

I ran tests by keeping changing one variable and keeping other variable constants. Above results are only those results that tend to give the highest results. To improve the score, I fixed value of threshold and didn't change it in once congestion was detected. This however didn't produce good results. For example, in experiment No:1 and 2, the threshold were fixed but the score was very less. It may be because threshold values may be very high and window size keep increasing. The score for these values was very low. After many times, I found that AIMD produces max values when timeout was around 105ms and when ssthreshold decreases to half of the window size on every time out as it is evident from test no 3,4,5,6,7 in the above table. Test 3 produced a higher score compared to test 4. In test No3, on reaching time out, I set the value of window size 3 compared to 4 in test no4. Setting window size to 3 rather than 2 on timeout for an gives better result. Even thoughthe time out for a packet have occurred, the network might still not have become congested as packet may have become lost because of other reasons e.g corrupt. Moreover, in congestion phase, instead of linearly increasing the window size on every ACK received e.g 1/window_size, I realized that doubly increasing the window size e.b 2/window_size increases the throughput more with slightly increase in delay.

<u>Best Results Constant</u>

Test number 5 gives the best results._
Increasing window size by 2/window_size in the congestion avoidance phase give much better results. Any further increase in may result in the increase of throughput, but delay also increases significantly. Moreover, when time out occurs, I set the window size to 3 than 1. This increases the throughput. Moreover, setting window size to value higher than 3 when timeout occurs on ACK received may slightly improve throughput, but also makes the network more congestions and hence delay increases much more.

<u>Limitations of AIMD for this trace:</u>

AIMD was designed to work with wired networks. In wired networks, whenever a loss event happens because of the timeout, there is much higher chance that packet was lost because of the congestion in the network. Thus, AIMD scheme works well for the wire networks. However, in case of the wireless network, AIMD is not efficient. In wireless networks, there is a very high probability that a loss event may occur which doesn’t imply that network is getting congested. Packet could become corrupt or packet might get destroyed because of the interference of the signals. Thus, AIMD scheme which perceives packet loss as congestion in the network performs badly in case of the wireless network. It can be seen from the above results that throughput is utilized around 60 percent because AIMD thinks that network may have become congested. However, loss events might have happened because of the signal interference, packet corruption, or weak signals and not necessarily because of the congestion.

Contest

Part 3 :Throughput_predict_1:

I tried various methods to improve the score. My best score was about 30. Instead of using timeouts to detect congestion, I tried to predict available throughput in the link and then setting the window size (window_size=estimate throughput*delay).
To predict the throughput, I count the number of ACK's received within a time slot e.g 30. After that current throughput was predicted by packet_count/timeslot. I further used Exponentially Weighted Moving Average to estimate throughput. After trial and error, I selected the alpha value to be 0.15. Further more, I also tried to reduce the value of delay if time out occurs for 2 consecutive values. In my approach, if I tried to increase the time slot, I get a very large throughput, but that would increase the delay time significantly.

Best Score:30.75 Throughput 4.09 and delay 133

Limitations:

I ran all tests on my laptop which is very old and I realized that some of the score was coming out to be very less. Perhaps, because I allocated very less RAM to VM (as my RAM of laptop was already very less). Moreover, while calculating the throughput, I divided slot by 2 because otherwise my throughput was being predicted very less. So, if my code is run on different laptops, it is possible that value of slot need not be divided by 2. I also found that on my laptop I was receiving very less number of ACKS received in a given time compared to what other students were receiving. This resulted in relatively lower score. Moreover, my approach could further be improved by dynamically predicting the value of delay and further increasing/decreasing window_size based on throughput. This scheme is better than AIMD because it doesn't depend on time out much for loss of the packet.

FILES:

I am submitting two controller files. One is for part 2 which is (coontroller-AIMD_best.cc) and other is for part 3 which is (throughput_controller.cc). Rename the respective files to to controller.cc before making them.

References:

http://www.sigcomm.org/sites/default/files/ccr/papers/2014/July/0000000-0000005.pdf
http://alfalfa.mit.edu/?fbclid=IwAR0rR4ZV80R73Upysw1Jp8Tasl0iXDnfAzOxRzlx-GvVFPG3iPMmQu7TWTE#paper

For contest I discussed some approaches with Mustafa Abbasi.

BEST RESULT AIMD:

http://cs344g.keithw.org/report/?2010011720100047-1543990099-eipaegei