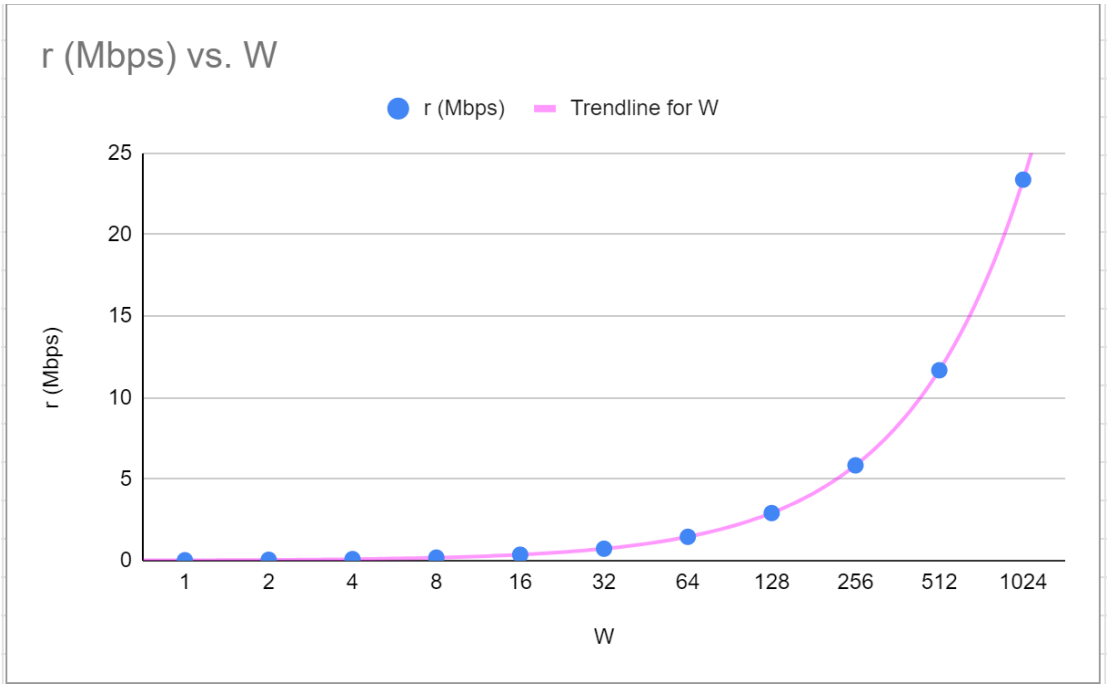
CSCE 463-500

Reliable Data Transfer over UDP - Homework 3.3

The purpose of this assignment was to write a C++ transport-layer service over UDP that can sustain non-trivial transfer rates (hundreds of megabits/sec) under low packet loss and avoid getting bogged down under heavy loss.

1. **Set packet loss *p* to zero in both directions, the RTT to 0.5 seconds, and bottleneck link speed to S = 1 Gbps. Examine how your goodput scales with window size W. This should be done by plotting the steady-state rate r(W) for W = 1, 2, 4, 8, …, 210 and keeping the x axis on a log-scale. Your peak rate will be around 24 Mbps and, depending on your home bandwidth, usage of an on-campus server might be necessary. Using curve-fitting, generate a model for r(W). Discuss whether it matches the theory discussed in class.**

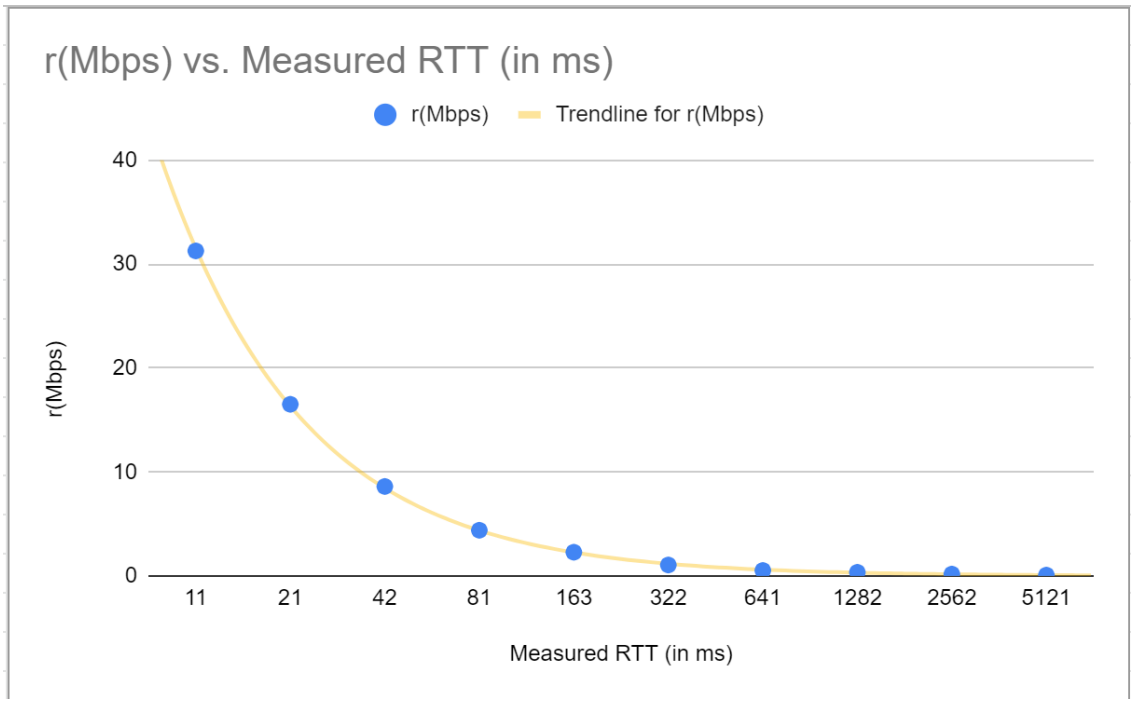
    a. The model for r(W) obtained is included in the image below, as well as the best fit function for the curve.



    b. We can see that the generated r(W) plot is linear, which matches the theory that we learned in lecture (and that is included in the class slides from 3/1/22) , which says that U$sender$ = (3L/R)/(RTT + L/R).

2. **Expanding on the previous question, fix the window size at W = 30 packets and vary the RTT = 10, 20, 40, …, 5120 ms. Plot stable rate r(RTT), again placing the x-axis on a log scale. Perform curve-fitting to determine a model that describes this relationship. Due to queuing/transmission delays emulated by the server and various OS kernel overhead, the 1 actual RTT may deviate from the requested RTT. Thus, use the measured average in your plots and comment on whether the resulting curve matches theory.**

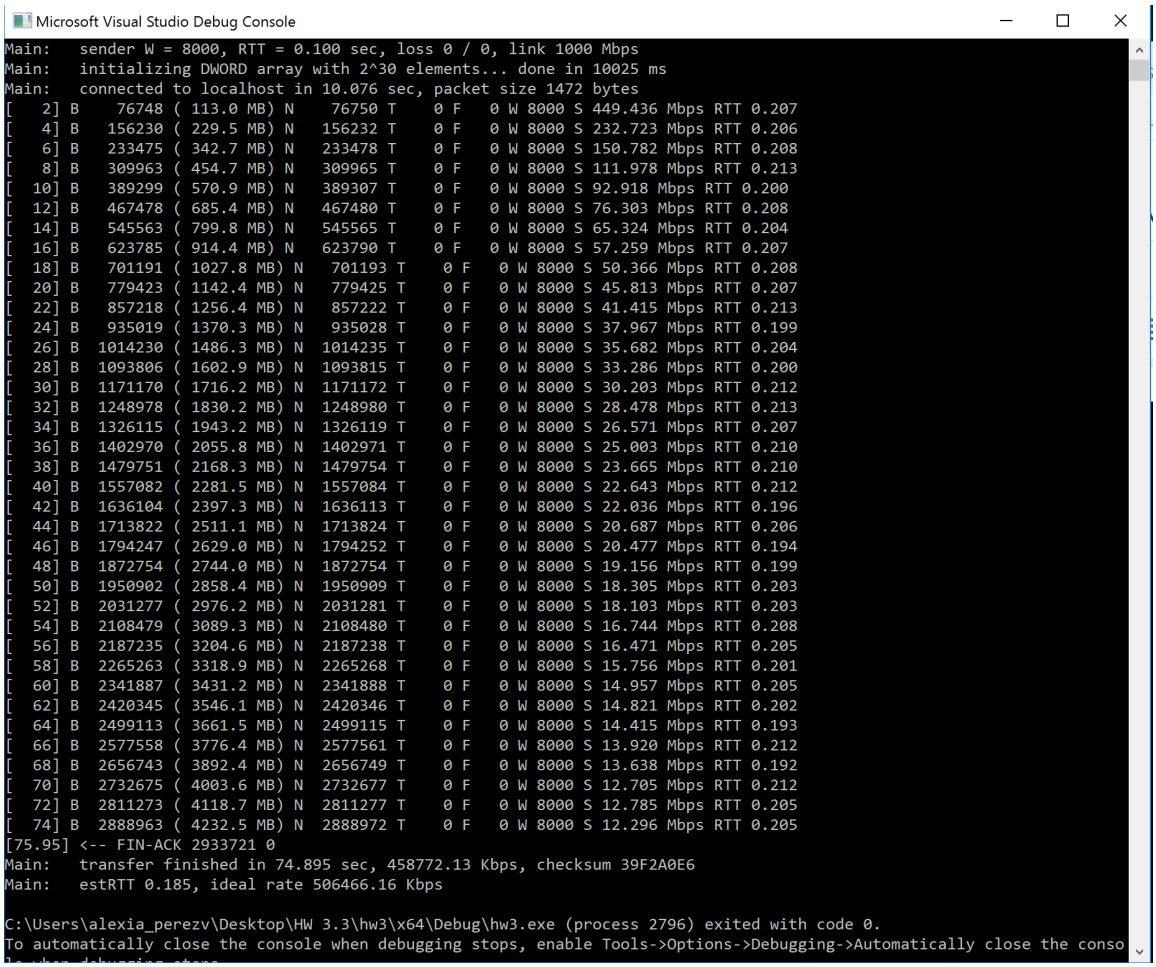
    a. The generated model for r(RTT) obtained is included in the image below, as well as the best fit function for the curve.



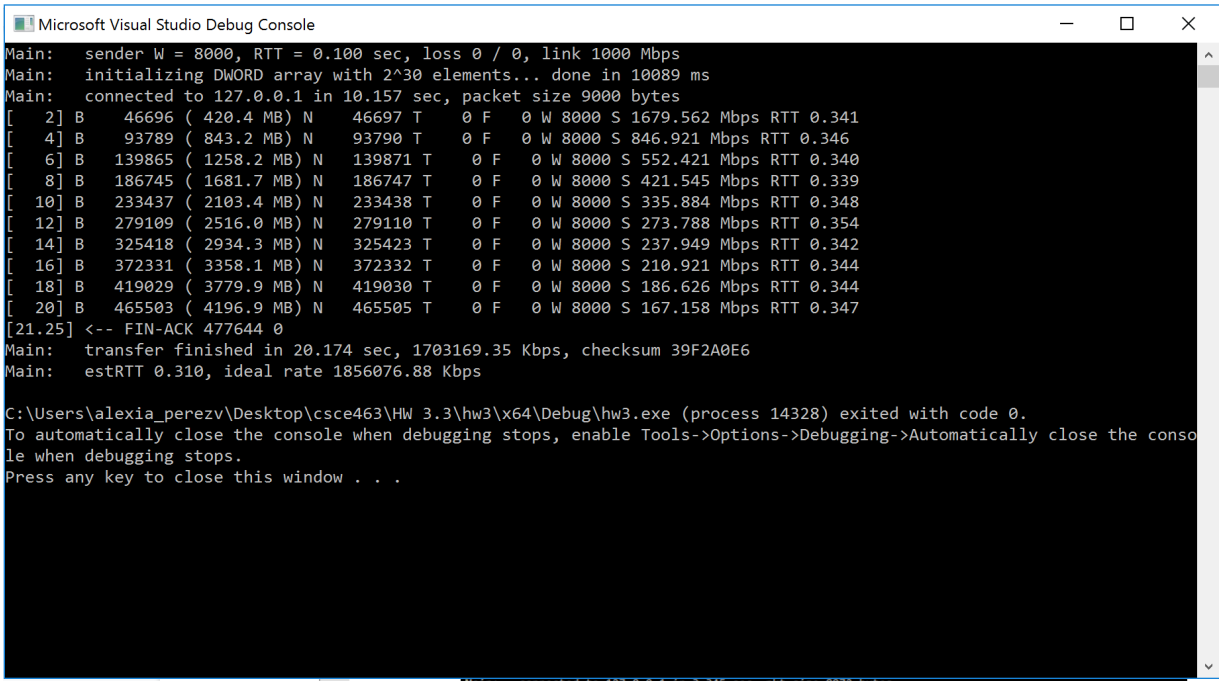
    b. We again can see that the curve is linear and it matches the theory discussed in lecture.

3. **Run the dummy receiver on your localhost and produce a trace using W = 8K (the other parameters do not matter as the dummy receiver ignores them, although they should still be within valid ranges). Discuss your CPU configuration and whether you managed to exceed 1 Gbps. How about 10 Gbps using 9-KB packets (see dummy-receiver discussion in Part 1)?**

   a. The trace produced by running the dummy receiver on ts2.cse.tamu.edu with W = 8K is included below:



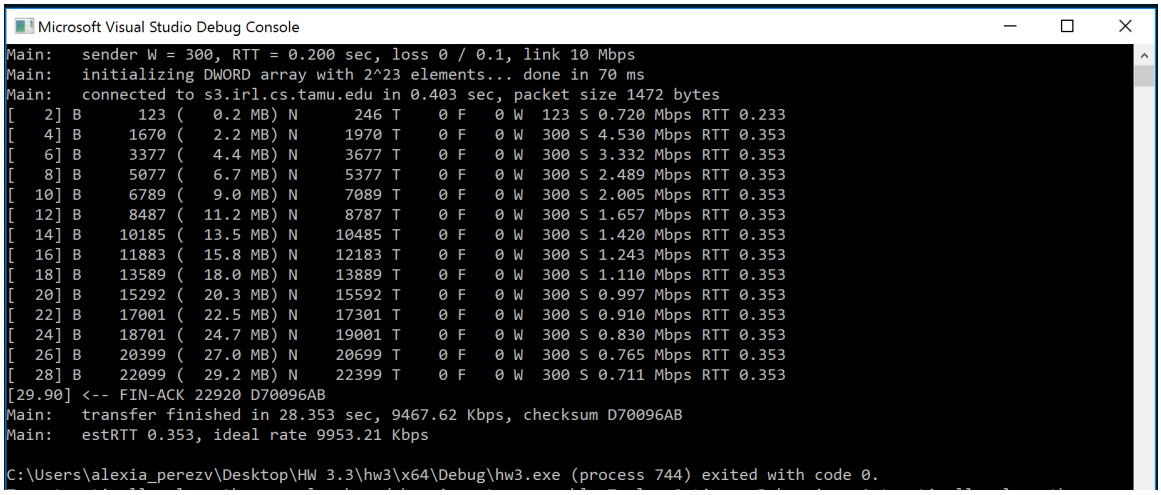
   b. The CPU configuration of the ts2.cse.tamu.edu server is as follows:
      i. Processor: AMD Opteron™ 6172, 2.10 GHz (2 processors)
      ii. Installed RAM: 32GB
   c. I was able to reach 449.436 Mbps when using the ts2 server and running the dummy receiver, but did not reach 1 Gbps at any point in this trace.
   d. The following trace corresponds to running the dummy receiver on the ts2.cse.tamu.edu and adjusting the packet size to 9000 bytes.



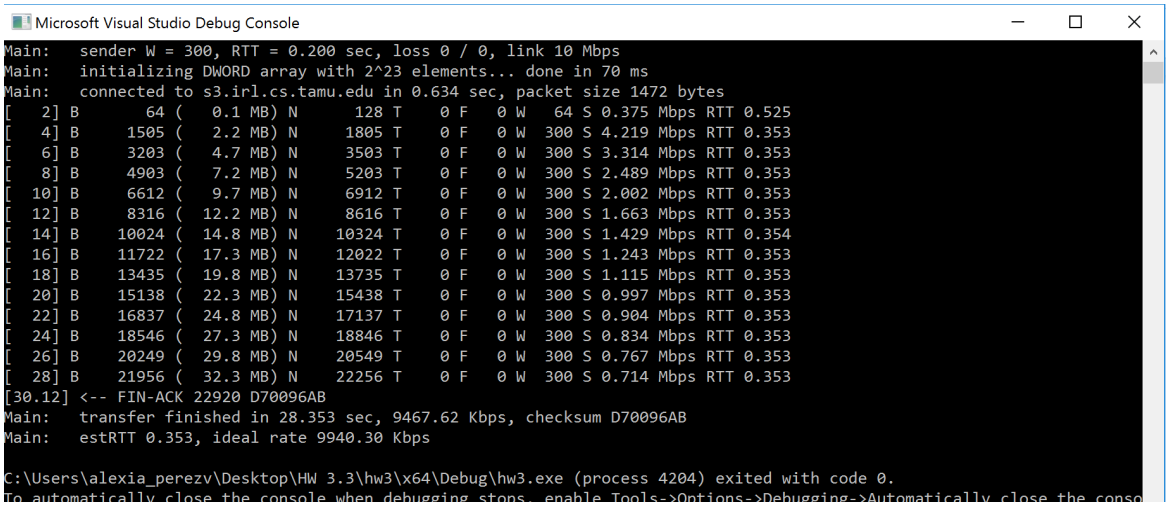
e. Here, we can see that 1679.562 Mbps were reached, however, 10 Gbps were not.
f. I would also like to point out that although I ran these tests on-campus, while connected to the TAMU Wi-Fi network (which should be faster than connecting to the server from my house), the connection took an entire 10 seconds to be established. This is something that I have not experienced throughout the development of my code for this project, and I believe it may simply be due to the Wi-Fi connection being slow rather than my code being incorrect (I have seen my connection times and speeds be much higher even when I am at home, and no changes were made to the code for it to become this much slower).

4. **Use buffer size 223 DWORDs, RTT = 200 ms, window size W = 300 packets, link capacity S = 10 Mbps, and loss only in the reverse direction equal to p = 0.1. Show an entire trace of execution for this scenario and compare it to a similar case with no loss in either direction. Does your protocol keep the same rate in these two cases? Why or why not?**

   a. Loss in reverse direction:



   b. No loss (in either direction):



   c. As it can be seen, both cases do keep the same rate. It is possible that this may be caused by the fact that cumulative ACKs are used in TCP. Therefore, for the case where we have loss in the reverse direction (of 0.1), 3/300 packets will get lost (since window size is 300). Furthermore, if 3 ACKs are sent by the receiver and only the middle one is lost, then when the sender receives the 3rd ACK it automatically knows that the preceding packets must have been received, so the rate is maintained.

5. **Determine the algorithm that the receiver uses to change its advertised window. What name does this technique have in TCP? Hint: the receiver window does not grow to infinity and you need to provide its upper bound as part of the answer.**
    a. The receiver is using TCP Flow Control in order to change its advertised window.
    b. Here, recvWin = recvBuf - [lastOrderedByteRecv + lastByteDelivered]
    c. The diagram below (taken from 3/10/22 lecture slides, page 21) illustrates why recvWin cannot grow to infinity (it is due to the fact that its upper bound is the recvBuf).