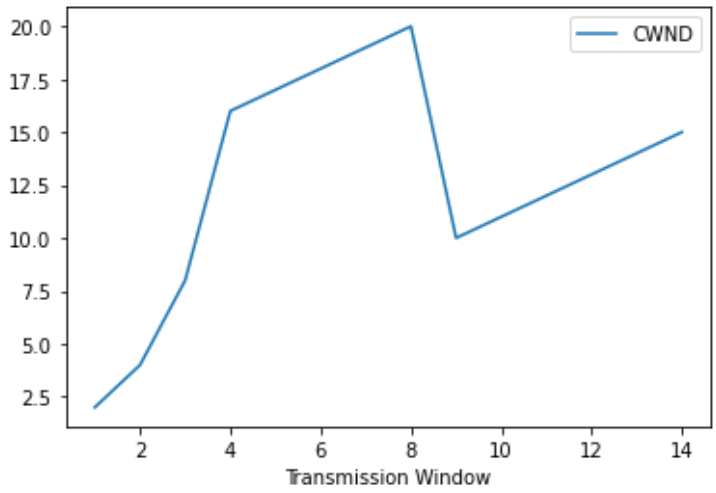
1. Explain and justify the additions to your packet header to implement this part as compared to part 2. (3 points)

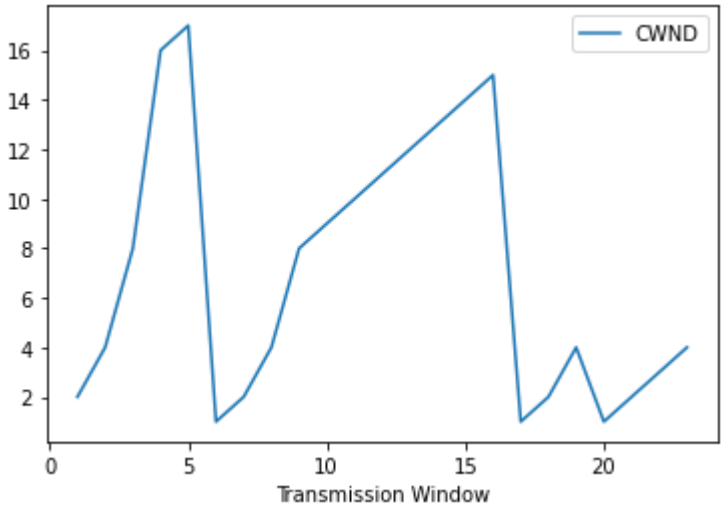
In order to handle the newest requirements for congestion control, I needed to make use of one more header field for CWND. This tell the receiver how much I'm going to be transmitting to it. In our case, it is how the server knows when to simulate congestion.

2. Based on the graphs that you generate, list down each time the state of your network changes (from a slow start to congestion avoidance and vice versa), explain the reason behind the change, and report the values of CWND and Slow Start Threshold (SSThreshold) before and after the state change. (12 points)

Alice – Reno

- Begins at Slow Start
- At CWND = 16 begin AIMD, ssthresh = 16
- CWND = 20 packet loss detected, CWND dops to 10, ssthresh = 10
- Begins fast recovery





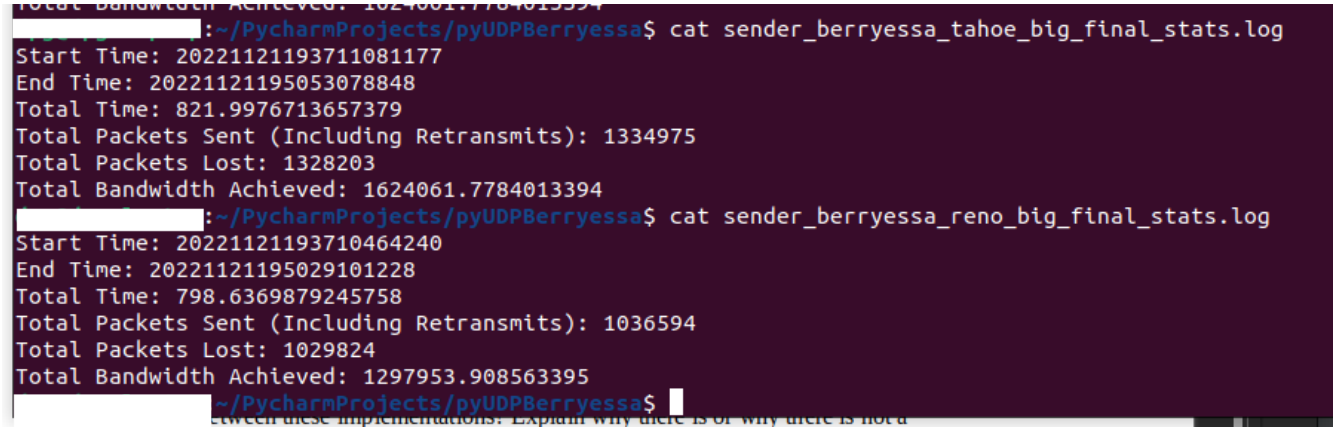
Alice – Tahoe

- Begins at Slow Start
- At CWND 16, ssthresh = 16, begin AIMD
- At CWND 17 packet loss detected, drop CWND to 1, ssthresh = 8, begin Slow Start
- At CWND 8 begin AIMD, ssthresh = 8
- At CWND = 15, packet loss detected again, drop CWND back to Slow Start, CWND = 1, ssthresh = 7

3. Compare the bandwidth difference between TCP Tahoe and Reno in your experiments. Compare it with the bandwidth that you measured in part 2. Do you notice any significant difference between these implementations? Explain why there is or why there is not a difference. (10 points)

Reno did perform significantly better in my tests. It's fast recovery state I believe is the biggest reason why. Watching it transmit at the same time as Tahoe. Reno resent far fewer packages, and when the speeds of both transmissions slowed together, Tahoe drops all the way back to 1 whereas Reno only drops in half and never goes back to Slow Start like Tahoe does.

I noticed a significant performance improvement when I added the functionality to send multiple packets to one connection, instead of waiting on a response for each packet. However, once I enabled Reno and Tahoe, the performance increase was then lost because of the simulated congestion on the receiver. On the receiver I enabled messages to watch it every time it triggers round trip jitter or packet loss. This severely impacted the performance boost we got during this part.



4. Based on your experience and results in designing these different models, what else can be done to improve the bandwidth usage of your network? (5 points)

I would add communication from the receiver when it is in a congestion state, including how much bandwidth it wants. The sender can make use of this information before ramping up past the threshold of the receiver and having to ramp back down.