Project 3 Part 3 - UDP Berryessa Report

Nicholas Myers ID: 917617010 Section: A01

Filenames: **receiver_solano.py**
sender_solano.py

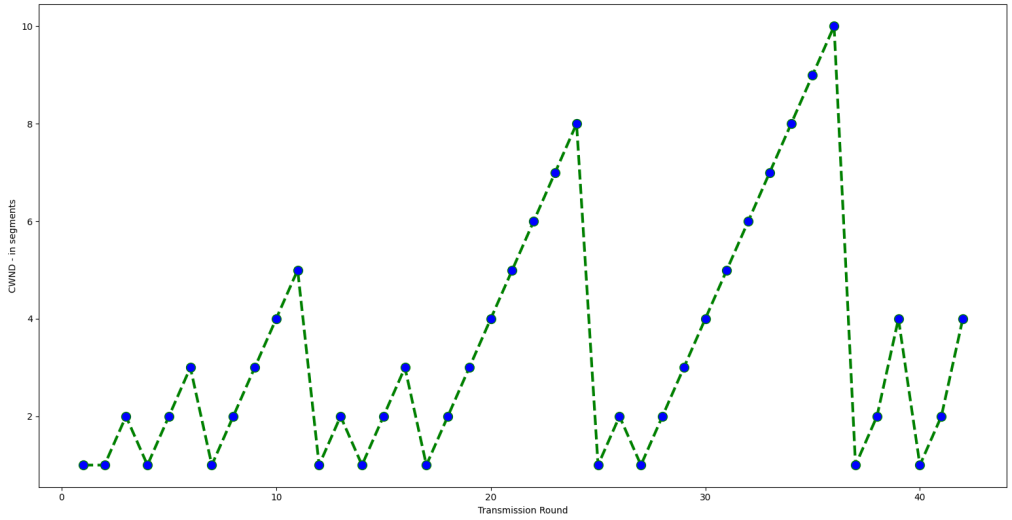
**** Tahoe/Reno graphs also located after questions ****

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

- In part 3 I only added a cwnd header field as this would be needed to keep track of the current window size as it changed. Having this also allowed the server to keep track of how many bits were being sent all at once and adjust its congestion control if this rate exceeded the input BDP. Also since cwnd and ssth are essential aspects of tcp Tahoe/Reno, I of course needed to add cwnd into the header to keep track of it and adjust accordingly.

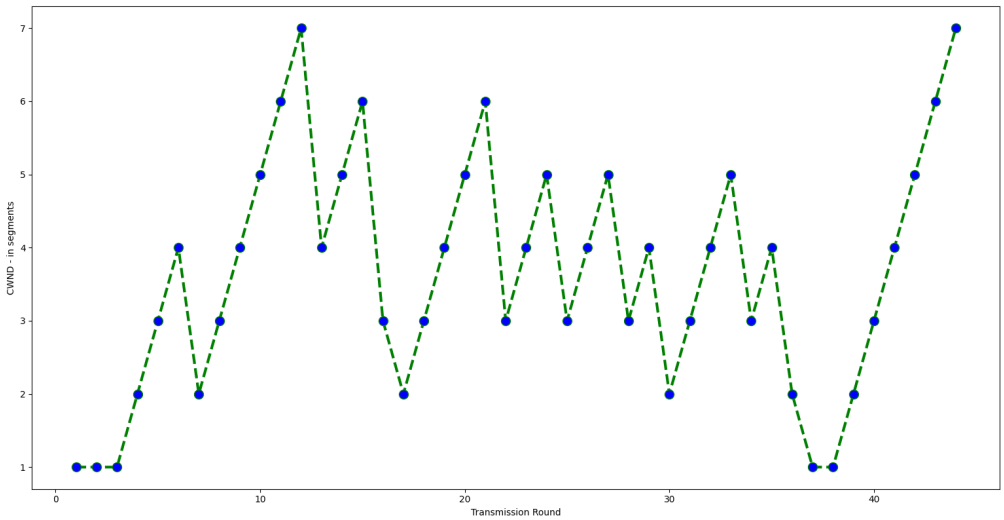
2. Based on the graphs that you generate, list down each time the state of your network changes (from a slow start to congestion avoidance and vice versa), explain the reason behind the change, and report the values of CWND and Slow Start Threshold (SSThreshold) before and after the state change. (12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

ALICE29.txt TAHOE:



Round [2]: Slow Start -> Congestion Avoidance; due to timeout
CWND: 1 -> 1
SSTH: 16 -> 1
Round [37]: Congestion Avoidance -> Slow Start; due to timeout
CWND: 10 -> 1
SSTH: 1 -> 5
Round [39]: Slow Start -> Congestion Avoidance: due to timeout
CWND: 4 -> 1
SSTH: 4 -> 2

ALICE29.txt RENO:



Round [2]: Slow Start -> Congestion Avoidance; due to timeout
CWND: 1 -> 1
SSTH: 16 -> 1

Roughly stays in congestion avoidance throughout the rest of run, as the rest of the issues are triple duplicate ACKs which do not cause Reno to begin slow start.

i.e., Round [6]: Congestion Avoidance stays; due to 3 dupe ACK
CWND: 4->2
SSTH: 1->2
Round [11]: Congestion Avoidance stays; due to 3 dupe ACK
CWND: 7->4
SSTH: 2->4
Round [14]: Congestion Avoidance stays; due to 3 dupe ACK

CWND: 6->3
SSTH: 4->3

and so on…

I was not sure if you wanted me to write all the rest of the rounds where this similar triple dupe ACK behavior occurs since the directions say only to write when it changes congestion state, however, I hope writing these 3 occurrences of the triple dupe ACK rounds demonstrates enough.

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)

PART 2 RESULTS

alice29.txt
Total time taken for file transfer: 92.92850232124329
Total bandwidth for this file: 1622.17184431627 bps
Total packet loss observed for this file: 0.1390728476821192

big.txt
Total time taken for file transfer: 3813.9214372634888
Total bandwidth for this file: 1727.2212100746785 bps
Total packet loss observed for this file: 0.11566484517304189

- ** Part 3's results for Tahoe and Reno are located below question 4 along with their graphs for your convenience.

- Using default parameters, I noticed that my tcpTahoe and tcpReno were capable of achieving higher average bandwidth for the files. However, the time taken to complete the files was not too different besides big.txt where I assume the change in time is due to the variability of getting timeouts in each run and since big.txt is so large, a run with many timeouts will take longer than a run without. This brings me to my next point of how the default parameters we are given to run introduces so many timeouts and jittering that sometimes my tahoe and reno actually performed about the same or even a little worse than my part 2. Since tahoe and reno have to wait for the receiver to finish sleeping, the amount of time virtually stays the same between the two parts and the bandwidth is not able to substantially increase. This does change though if I adjust the

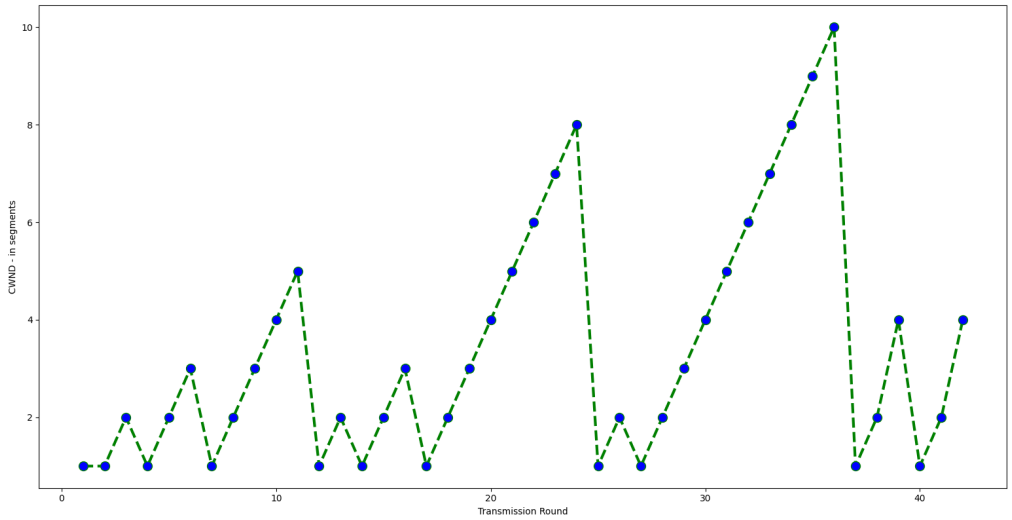
parameters to reduce the amount of timeout and jittering. If I change the jitter parameter to 90 and run alice29.txt, tcpReno now only takes 34 seconds to complete along with a bps of 4386, a much higher amount! Therefore, using a less aggressively set jitter parameter results in a much more accurate tcpTahoe/Reno bandwidth increase comparatively to an aggressive jitter parameter that effectively halts the Tahoe/Reno algorithm as it has to wait so much time for the receiver to stop sleeping.

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)

- Being able to reduce network jitter vastly improved bandwidth in my tcpTahoe/Reno, which would occur if the server max bdp input was overflowed by the amount of data coming in at once. Thus, my tcp algorithms would benefit in their bandwidth if they were able to efficiently keep their cwnd hovering around the server's max bdp. This of course is difficult due to timeouts and duplicate ACKs which cause cwnd to drop much below that "max efficiency area". Therefore, a possible multi-packet loss detection state could be utilized, much like the triple duplicate ACK detection, which could flag when multiple packet losses occur in a row and make less dramatic changes to cwnd as to not have it stray too far from the "max efficiency area" beneath the max bdp line. This could possibly happen by having this new hypothetical flavor of TCP Reno stay in fast recovery until all the multiple packet loss flagged packets are successfully acknowledged.

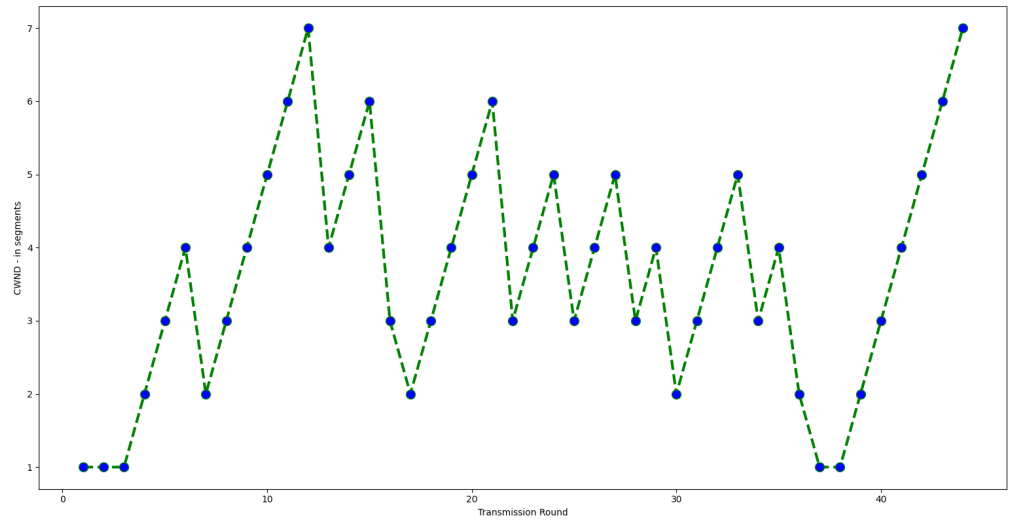
ALICE29.TXT GRAPHS:

TCP TAHOE



Total time taken for file transfer: 84.78161954879761
Total bandwidth for this file: 1781.8130958568418 bps
Total packet loss observed for this file: 0.1935483870967742

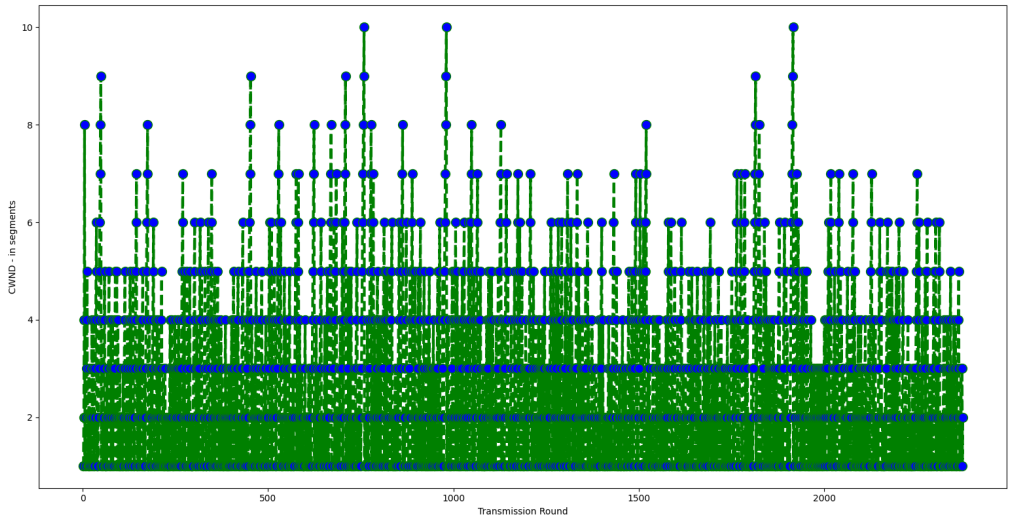
TCP RENO



Total time taken for file transfer: 82.61643242835999
Total bandwidth for this file: 1828.5103285111531 bps
Total packet loss observed for this file: 0.1111111111111111

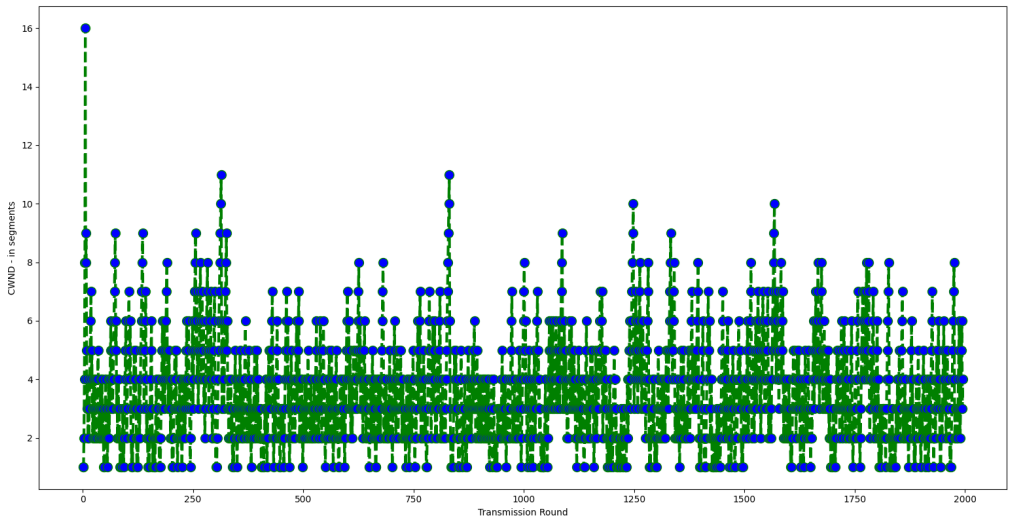
BIG.TXT GRAPHS (FULLY RAN):

TCP TAHOE



Total time taken for file transfer: 3378.92692899704
Total bandwidth for this file: 1953.5441691126855 bps
Total packet loss observed for this file: 0.13879263177092832

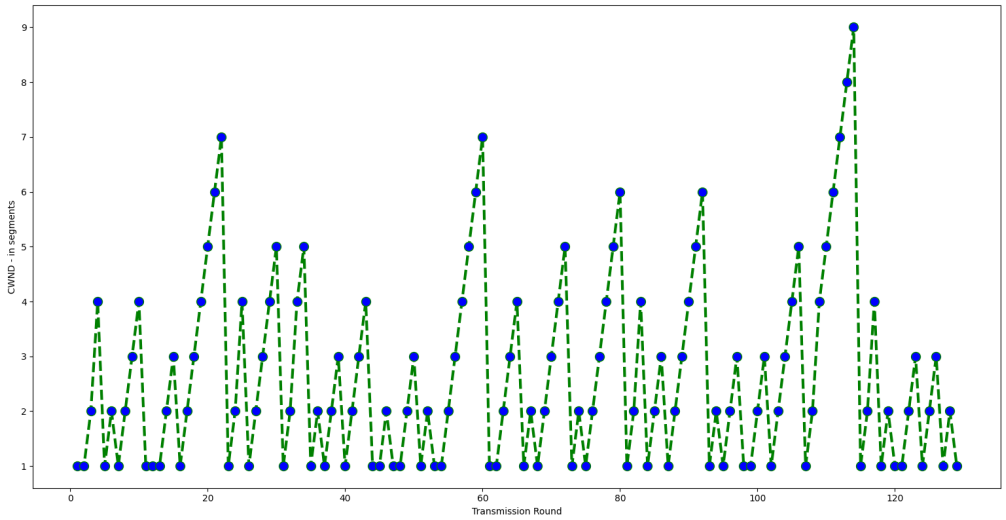
TCP RENO



Total time taken for file transfer: 3436.311883687973
Total bandwidth for this file: 1920.9208079552127 bps
Total packet loss observed for this file: 0.13593866866118176

BIG.TXT GRAPHS (ran for less time for readability):

TCP TAHOE



TCP RENO