Project 5

Sergiy Palguyev

spalguyev3@gatech.edu

1. Item #1 - In this network, what do you expect the CWND (congestion window) curve for a long lived TCP flow using TCP CUBIC to look like? Contrast it to what you expect the CWND for TCP Reno will look like.

   I would expect initially to see the congestion window to expand before a drop to a stable flow. The TCP Reno congestion window should like a line. The TCP CUBIC congestion window should look like a cubic polynomial curve.

2. Item #2 - Explain why you expect to see this; specifically relate your answers to details in the paper. (We expect you to demonstrate the ability to apply what you have already learned from the paper in your answer to this question, and that you've furthermore thought about what you read in order to make a prediction. So justifying your prediction is important. — It's okay if your prediction ends up being wrong as long as it was based on an understanding and analysis of the paper!)

   As described in the paper, I would expect TCP Reno should look like a linear function as TCP flow versus round-trip-time increases linearly. TCP CUBIC follows a cubic function with a similar shape as TCP-BIC. Cubic uses both, the concave and convex profile of the cubic function for window increase. The CUBIC window growth function is described with the following formula: $W(t) = C(t - K)3 + Wmax$. As the window adjustment algorithm runs if cwnd is above Wmax then CUBIC is in the convex region, otherwise CUBIC is in the convex region.

3. Item #3 - With respect to queue occupancy, how are the graphs different? What do you think causes these differences to occur? (Make sure to include the graphs with your answer).

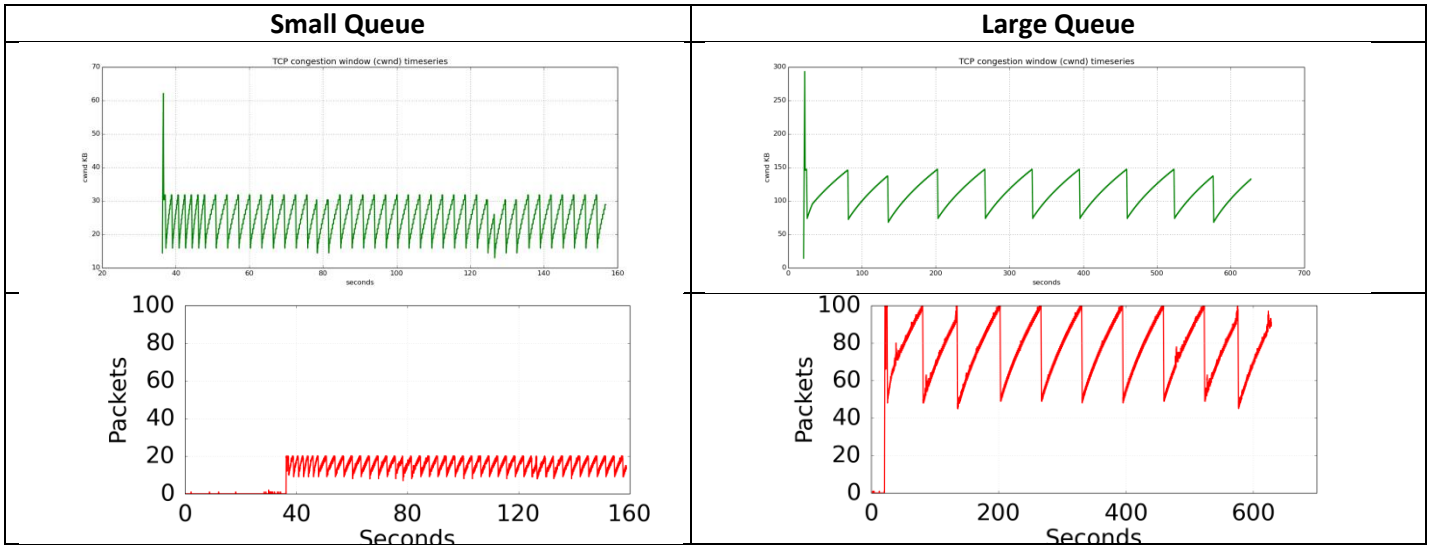

Figure 1. Small Queue and Large Queue performance of TCP RENO function.

   The large queue has a higher occupancy since it produces between 50 and 100 packets while the small queue produces between 10 and 20 packets per cycle. Thus, the difference in the graphs is mainly due to the packet count per the queue sizes. This correlates to the differences observed of the congestion windowed (CWND) as the large queue CWND varies between 75 to 150 KB, while the small queue CWND varies between 15 to 30KB due to queue size.

4. Item #4 - What did you actually see in your CUBIC results? What anomalies or unexpected results did you see, and why do you think these behaviors/anomalies occurred? Be sure to reference the paper you read at the beginning of

the project to help explain something that you saw in your results. Your hypothesis doesn't necessarily have to be correct, so long as it demonstrates understanding of the paper.

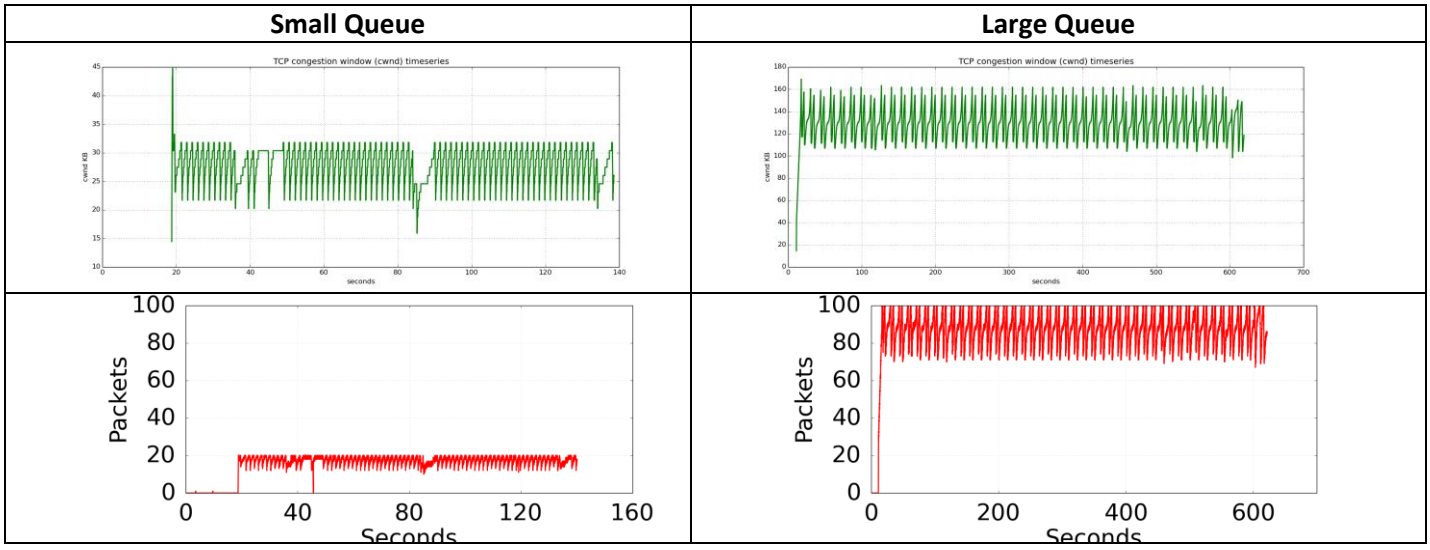

Figure 2. Small Queue and Large Queue performance of TCP CUBIC function.

The graphs producing with the CUBIC function look similar to the TCP RENO with very interesting anomalies especially visible in the small queue but also present in the large queue graphs. Similarly to the TCP Reno graphs, the graphs mainly differ by the sizes of their congestion window. The anomalies observed around 20ms, 70ms and 120ms are most likely due to lost packets. What is being observed is a drop in congestion window range as well as amore prolonged increase in window size. As described by ha, Rhee and Xu [1], when a CUBIC function encounters a packet loss, CUBIC reduces its window size by a factor of Beta [1] which is usually 0.2, which in turn leads to slower convergence. Both of these behaviors seem to describe well the anomalies observed in the graphs. The authors call this behaviors “multiplicative decrease”.

Furthermore, the cwnd graph of the large queue has repeating anomalies, A fast growing peak is immediately followed by a slow growing one in a repeating pattern throughout the graph. Following the logic observed in the small queue, one theory may be that the large queue graph is demonstrating packet loss at each pulse, thus the following pulse exhibiting the multiplicative decrease behavior. Since this occurs every time, the behavior repeats itself continuously.

5. Item #5 - Compare your prediction from Item #1 with the congestion window graphs you generated for TCP Reno and TCP CUBIC. How well did your predictions match up with the actual results? If they were different, what do you think caused the differences? Which queue size better matched your predictions, small or large?

The predictions stated in Item #1 appear to have been correctly described. Observing the cwnd increase behavior in cwnd graphs in Item #3, one can see a very discrete and linear step increase from Wmin to Wmax. However, observing the cwnd graphs in Item #4, the congestion window increase reaches an obvious plateau half way through the increase and then continues in what can be seen as a cubic function. The behavior is more pronounced in the small queue graphs but is also evident in the large queue graphs if observed closely.

6. How does the presence of a long lived flow on the link affect the download time and network latency (RTT)? What factors do you think are at play causing this? Be sure to include the RTT and download times you recorded throughout Part 3.

Without Long-Lived Flow:

100%[=====================================>] 177,669 175KB/s in 1.0s

rtt min/avg/max/mdev = 20.440/20.543/20.646/0.103 ms

With Long-Lived Flow:

100%[=====================================>] 177,669 41.4KB/s in 4.2s

rtt min/avg/max/mdev = 689.796/714.148/730.105/12.880 ms

The download time is 1s and average RTT is 20.440ms before the flow and grows to 4.2s download time and 689.796ms RTT after the flow. Since this experiment utilizes TCP Reno, the most likely culprit of such a massive increase in download and RTT times is the fact that the entire bottleneck bandwidth is not being used [1] as well as the congestion window function does not adapt optimally to the increase in packet flow.

7. How does the performance of the download differ with a smaller queue versus a larger queue? Why does reducing the queue size reduce the download time for wget? Be sure to include any graphs that support your reasoning.

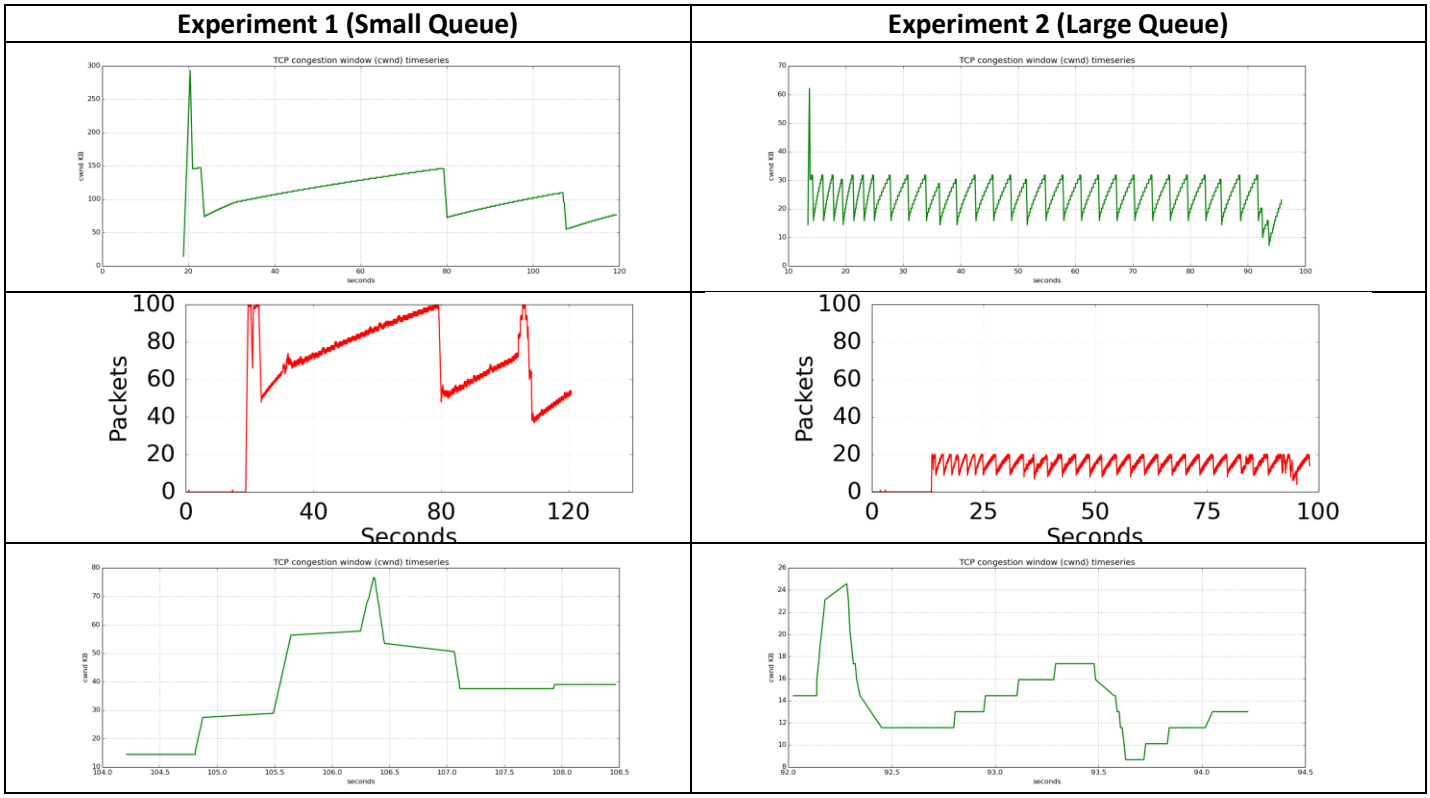

Figure 3. Small Queue and Large Queue TCP-RENO performance.

The smaller queue size visibly improves download performance likely due to fewer packets being buffered by the Headend router. Since the router does not have to buffer as many packets, the flow is not delayed and the download can complete faster. The large queue on the other hand introduces buffer bloat as a larger amount of packets Is delivered.

8. How does the presence of a long lived flow on the link affect the download time and network latency (RTT) when using two queues? Were the results as you expected? Be sure to include the RTT and download times you recorded throughout Part 5.

Without Long-Lived Flow:

100%[=====================================>] 177,669 175KB/s in 1.0s

rtt min/avg/max/mdev = 20.219/20.264/20.309 /0.045 ms

With Long-Lived Flow:

100%[==========>] 177,669 87.3KB/s in 2.0s

rtt min/avg/max/mdev = 20.070/20.538/21.275/0.454 ms

When using two queues, the download time is half as fast while the network latency remains unaffected. This is expected per Ha, Rhee and Xu [1] describing RTT-fairness. The RTT statistics did not change as each flow now receives its own queue. On the other hand, the download took twice as long due to fair queuing, as now each flow receives half of the bottleneck link rate effectively leading to a download time that which takes twice as long.

9. Did the change in congestion control algorithm result in any significant differences in the results?
   a. If so, which algorithm performed better? What aspect of that particular algorithm do you think caused it to perform better?
   b. If there was no change, was this what you expected? Why or why not?

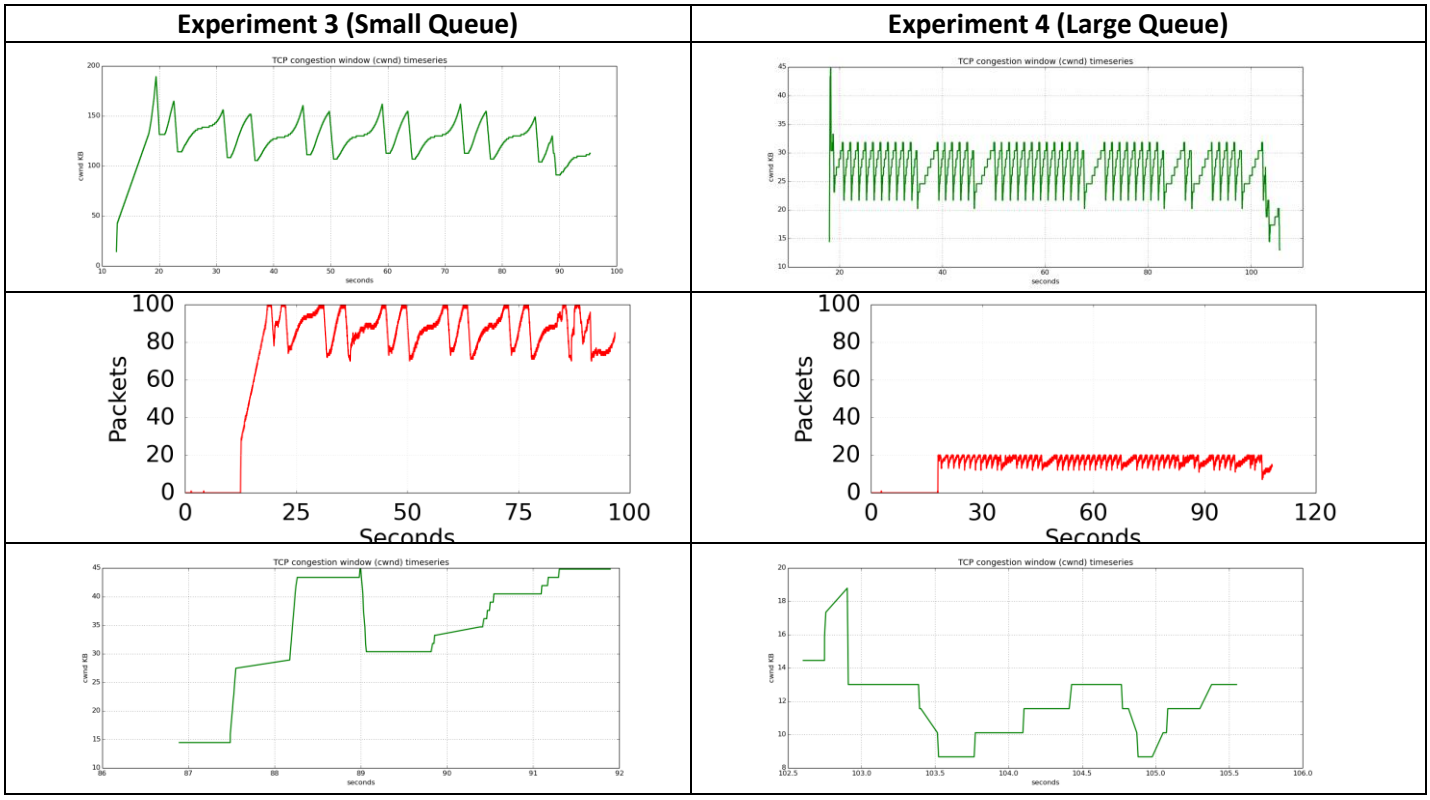

Figure 4. Small Queue and Large Queue TCP-CUBIC performance.

One of the biggest differences observed in using TCP-CUBIC is that the congestion windows for small queue for both wget and iperf quickly recovers and begins expanding again, compared to TCP-RENO where the cwnd slowly decreases. For large queue, there does not seem to be a visible difference in performance.

The reason we see similar performance between TCP-CUBIC and TCP-RENO is because TCP-CUBIC was designed to perform similarly to standard TCP for networks with small bandwidth-delay product as well as those with a short RTT [1]. The behavior is described as TCP-friendliness, where TCP-CUBIC function will maintain the congestion window matched. One other behavior noticed on TCP-CUBIC feature is convergence to congestion window, not observed on the TCP-RENO as cwnd decreases due to lost packets and multiplicative decrease.

The observed behavior is as expected since the introduction of TCP-CUBIC was created to increase TCP-friendliness and TCP-fairness as well as simplifying congestion-window control [1].

References

[1] Ha, S., Rhee, I., Xu L., () ard, M. CUBIC: A New TCP-Friendly High-Speed TCP Variant. *https://www.eecs.umich.edu/courses/eecs589/papers/Cubic08.pdf*