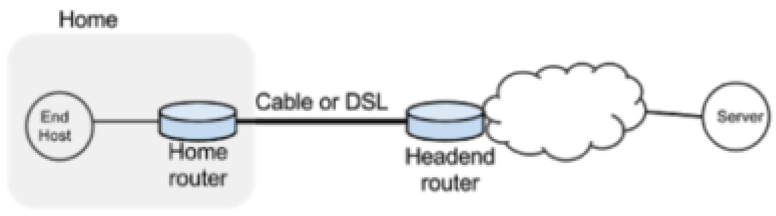
Bryce Montano
Project 6 – Analysis

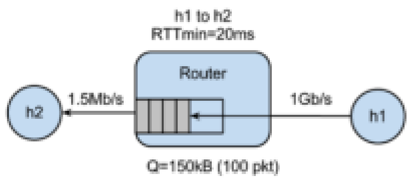


Note: I combined item 1 and 2 is simply a continuation of 1.

Item #1 @ 2: *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. 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!)*

Given the network topology shown in below (Figure A):

Figure A



I would expect to see a cubic shaped graph for the cwnd growth-rate for the host. That is to say, there should be a concave shaped, slowly increasing graph as the cwnd size approaches W_{max}, eventually plateauing and then continuing on a more convex shaped graph as it grows past W_{max} and eventually encounters a packet loss. From there, a multiplicative decreases should occur, dropping the graph to half of it's $W_{max.}$

In this particular network, if TCP Reno were utilized I would expect to see very jagged, 'saw-tooth' shaped window curves with rapid increases of window sizes due to the rapid cwnd practices of TCP Reno. The graphs will look jagged also due to the fact that TCP Reno increases its window size only once per RTT.

Another hypothesis for the TCP Reno flows could be that the overall bandwidth of the home network could be highly underutilized.

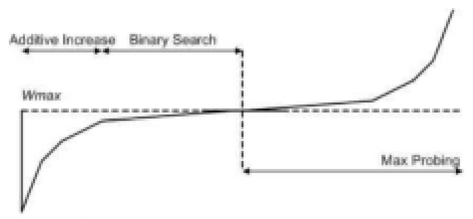
Item #3: With respect to queue occupancy, how are the graphs different? What do you think causes these differences to occur?

The graphs differ in the number of packets sent between failures, the number of window size increases between failures and the number of seconds in between failures. Reviewing the large-queue_tcp_cwnd_iperf.png and small_queue_tcp_cwnd_iperf.png, we see a much more frequent packet loss occurrence in the small_queue and a much smaller cwnd size than those of the large_queue. However, in the case of the large_queue, the time between the growth of the cwnd size from half of it's W_{min} to its W_{max} is a longer period than that of the small_queue due to the overall buffer size.

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 project the 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.

In my cubic results, as expected, I saw cubic graphs, especially in the case of the large-cueue-cubic results where once a packet loss occurred, the initial increase followed a concave curve slowly increasing the window growth and then eventually plateauing and finally changing to a convex growth size where the growth rate is much faster. With regard to the small-queue-cubic results, I saw a rather unexpected growth rate where the graph looked less cubic and smooth and more similar to those of BIC TCP. See below (Figure B):

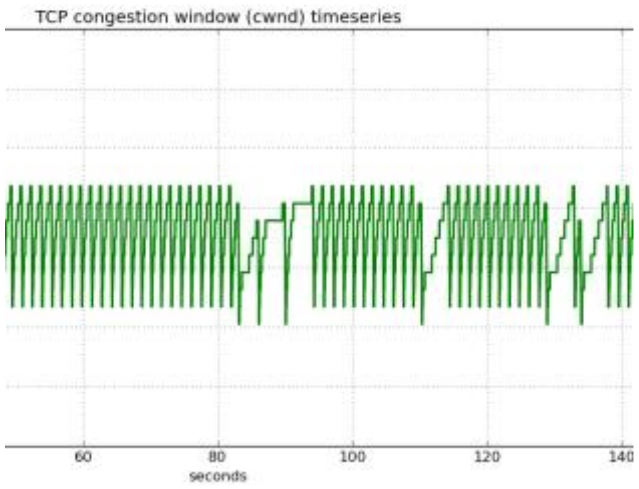
Figure B



(a) BIC-TCP window growth function.

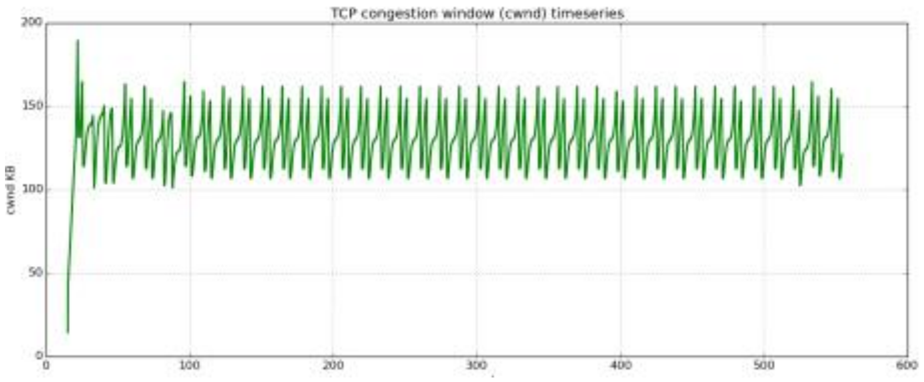
And my results from small-queue-cubic_tcp_cwnd_iperf.png (Figure C):

Figure C



As the paper suggestions, BIC TCP while still has a convex and concave growth rate, does not utilize a cubic growth and instead uses a more aggressive approach to increase the cwnd size (similar to what you see above). The results in the small-queue-cubic_tcp_cwnd_iperf.png were unexpected as I thought they would have employed more of a 'cubic' shaped graph as those in the large-queue-cubic_tcp_cwnd_iperf.png as show below:

Figure D



The results in 'Figure D' clearly show a cubic shape graph shape. This could be due to the fact that ./run-minq-cubic.sh either utilizes a BIC TCP approach (though unlikely) or some other reason I am not sure about.

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?

Compared to my answers in Item #1, some of my predictions were generally correct but I was not sure what to expect when it came to the large queue and small queue sizes.

TCP Reno

With regard to TCP Reno graphs, they looked as expected, displaying the jagged saw-tooth-shape where there is a steady linear growth rate and then a packet loss and a multiplicative decrease. Now, given that prior to the assignment I did not know we would be creating large and small queues, I did not account for (in my ininitial hypothesis) what the differences might look like between the small and large queues. Obviously with the large queue, there were fewer packet losses and much larger cwnd sizes. Overall I would say my hypothesis for TCP reno was spot on.

TCP Cubic

The results of my TCP cubic experiments were similar to what I expected, at least in the case of a large-queue as opposed to a small –queue. With the small-queue graphs, I expected the graphs to display a more cubic shape but instead, these looked extremely similar to BIC TCP shaped graphs reference in the paper where the increases in cwnd were more dramatic.

Item #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.

Figure E – First download
```
100%[=====================================>] 177,669      175KB/s   in 1.0s

2016-10-22 12:42:32 (175 KB/s) - ‘index.html’ saved [177669/177669]
```

Figure F – Second download
```
100%[=====================================>] 177,669     36.5KB/s   in 4.8s

2016-10-22 12:44:47 (36.5 KB/s) - ‘index.html.1’ saved [177669/177669]
```

Figure G – Ping stats (first attempt)
```
--- 10.0.0.2 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9013ms
rtt min/avg/max/mdev = 20.229/20.772/21.572/0.485 ms
```

Figure H – Ping stats (second attempt)
```
--- 10.0.0.2 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9001ms
rtt min/avg/max/mdev = 413.580/633.696/805.367/172.553 ms
```

A long-lived flow on the link affects the flow negatively by increasing the download time of a particular resource on the web. Downloading the index.html file the first time transmitted at 175kb/s and completed in one second. The second time around it downloaded at 36.5 kb/s and took almost 5 seconds (4.8)

The network latency is also substantially increased. Ping ICMP requests, on the first go-around, took on average 20.229ms/packet to return whereas on the second go-around, they on average 413.58 ms/packet.

Obviously the factors that are contributing to these changes in speeds and latency is due to competing resources on the network and having to share bandwidth.

Item #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.

There performance with the smaller queue is much better than with a larger queue. For instance, the wget on the index.html file in the large queue takes about 4.5 seconds to complete whereas with the small queue it takes less than 4 (about 3.5) seconds to complete. Reducing the size of the queue reduces the download time for wget in that it takes less time (less RTT's) for the cwnd to increase in with a buffer of 20 than with a buffer of 10. In other words it takes less time to reach the W_{max} with a smaller buffer size than with a larger buffer size. This is confirmed in the CUBIC paper, "In standard TCP like TCP-Reno, TCPNewReno and TCP-SACK, TCP grows its window one[sic] per round trip time (RTT)." Also, see graphs below for wget times.

Figure I – 20 packet queue

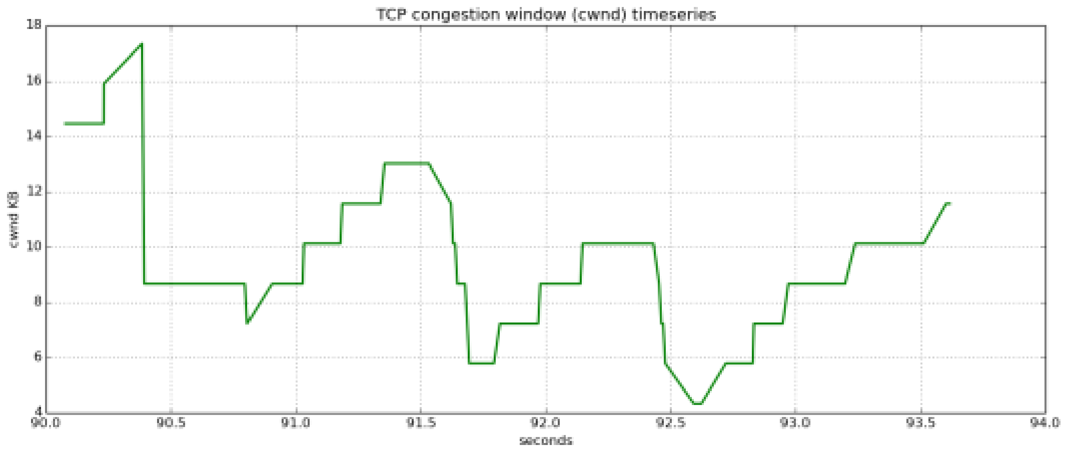
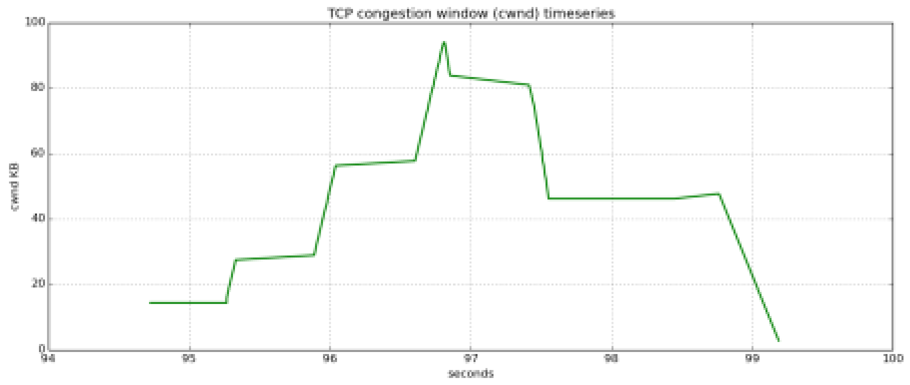


Figure J – 100 packet queue



Item #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.

Figure K – Ping statistics before long-lived flow initiated

```
--- 10.0.0.2 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9014ms
rtt min/avg/max/mdev = 20.279/20.924/22.077/0.686 ms
```

Figure L – wget before long-lived flow initiated

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

2016-10-22 13:29:56 (175 KB/s) - ‘index.html’ saved [177669/177669]
```

Figure M - Ping statistics after long-lived flow initiated

```
--- 10.0.0.2 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9012ms
rtt min/avg/max/mdev = 20.197/21.139/22.762/0.849 ms
```

Figure N – wget after long-lived flow initiated

```
100%[======================================>] 177,669    87.1KB/s   in
2.0s

2016-10-22 13:39:13 (87.1 KB/s) - ‘index.html.1’ saved [177669/177669]
```

The presence of the long lived flow on the link did affect the download time (though slightly) but it did not appear to affect the overall network latency, when using two queues. As you will notice the ping ICMP requests are almost identical. In fact, on average the ping return times were faster after the long-lived flow than before it was initiated. As for wget times, the average speed / second dropped and the time to completion increased after the initiation of the long-lived flow.

As for my expectations, I expected the latency and the download times to be slower so I was actually midly surprised to see the latency did not increase.

Extra Credit

Item #9: 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.

Figure O –Initial wget of index.html

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

2016-10-22 14:28:33 (175 KB/s) - 'index.html' saved [177669/177669]
```

Figure P – Initial PING (ICMP) requests

```
--- 10.0.0.2 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9012ms
rtt min/avg/max/mdev = 20.220/20.876/21.910/0.609 ms
```

Figure Q – Second PING (ICMP) requests

```
--- 10.0.0.2 ping statistics ---
10 packets transmitted, 8 received, 20% packet loss, time 8998ms
rtt min/avg/max/mdev = 590.038/687.212/788.924/66.092 ms
```

Figure R –Second wget of index.html

```
100%[======================================>] 177,669     27.4KB/s   in 6.3s

2016-10-22 14:29:37 (27.4 KB/s) - 'index.html.1' saved [177669/177669]
```

As indicated by the figures above, not only did the download speeds become much slower (over 600% slower) but ICMP requests slowed by almost 96%. I also noticed a 20% packet loss rate. The factors that are causing this are most likely due to the slow increase in cwnd size that cubic utilizes. As for why the packet losses occurred, I am not quite sure.

Item #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.

Again the smaller queue was much faster than with the larger queue for a download. See below figures:

Figure S – Large queue for wget

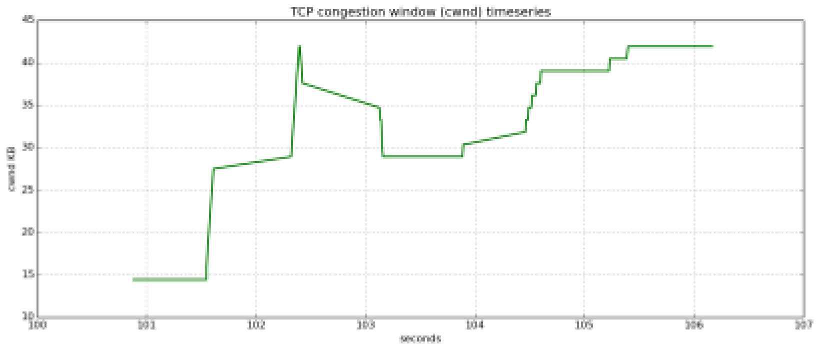
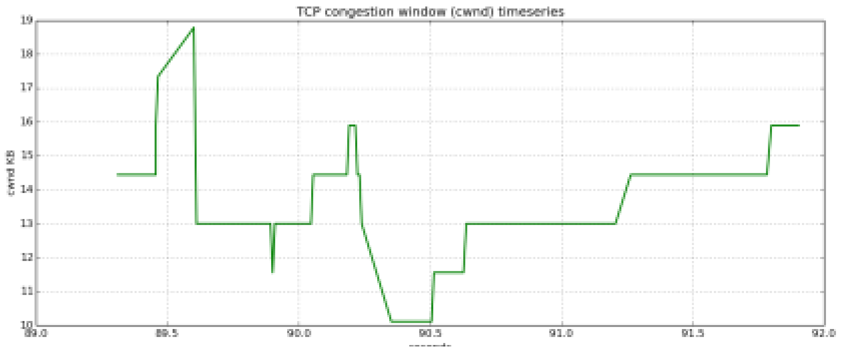


Figure T – Small queue for wget



Again, reducing the queue size allows for fewer RTT's before the window size is increased which allows faster speeds of downloads.