Networks Lab 4 – TCP Congestion Control

Name: Dylan Tan
ID: 1004385

What is the normal time required to download the webpage on h1 from h2?

The normal time required to download the webpage on h1 from h2 is 1.0s as can be seen in figure 1.

```
mininet> h2 wget h1
--2021-10-29 15:30:03--  http://10.0.0.1/
Connecting to 10.0.0.1:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 177669 (174K) [text/html]
Saving to: 'index.html.1'

     0K .......... .......... .......... .......... .......... 28%  182K 1s
    50K .......... .......... .......... .......... .......... 57%  176K 0s
   100K .......... .......... .......... .......... .......... 86%  172K 0s
   150K .......... .......... ...                              100%  181K=1.0s

2021-10-29 15:30:04 (177 KB/s) - 'index.html.1' saved [177669/177669]
```

Figure 1 – Time to download webpage on h1 from h2

What was your initial expectation for the congestion window size over time?

Since the download of a webpage is short flow, it will most likely be in the slow-start state throughout. Hence, my initial expectation is that the congestion window will start from size 1, then increase exponentially by doubling the value of previous congestion window size every RTT, until the webpage has been downloaded completely. For example, it will go from 1, 2, 4, 8, 16, 32 etc.

After starting iperf on h1, did you observe something interesting in the ping RTT?

Yes, the ping RTT became very large with an average of around 582ms (Figure 3), whereas before starting iperf on h1, it only had an average of around 32ms (Figure 2).

```
--- 10.0.0.2 ping statistics ---
30 packets transmitted, 30 received, 0% packet loss, time 29043ms
rtt min/avg/max/mdev = 30.450/32.263/39.958/2.457 ms
```

Figure 2 – ping RTT without iperf

```
--- 10.0.0.2 ping statistics ---
30 packets transmitted, 30 received, 0% packet loss, time 29033ms
rtt min/avg/max/mdev = 473.177/582.220/684.702/60.462 ms
```

Figure 3 – ping RTT with iperf

After starting iperf on h1, why does the web page take so much longer to download?

Before starting iperf on h1, it only took 1.0s (Figure 1), but now it takes 4.7s (Figure 4). The reason for this is because the bandwidth is now shared between multiple connections and due to the TCP fairness property, the effective throughput of the connection for the download is reduced. This is evident as the throughput of downloading was initially at 177 KB/s in Figure 1 but became much lower with only 37.2 KB/s in Figure 4.

```
mininet> h1 ./iperf.sh
started iperf
mininet> h2 wget h1
--2021-10-29 15:49:01--  http://10.0.0.1/
Connecting to 10.0.0.1:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 177669 (174K) [text/html]
Saving to: 'index.html'

     0K .......... .......... .......... .......... .......... 28% 40.7K 3s
    50K .......... .......... .......... .......... .......... 57% 65.1K 1s
   100K .......... .......... .......... .......... .......... 86% 59.2K 0s
   150K .......... .......... ...                              100% 12.9K=4.7s

2021-10-29 15:49:07 (37.2 KB/s) - 'index.html' saved [177669/177669]
```

Figure 4 – Time to download webpage on h1 from h2 with iperf

Please provide the figures for the first experiment (with qlen 100).
*** Please comment on what you can see in the figures.**

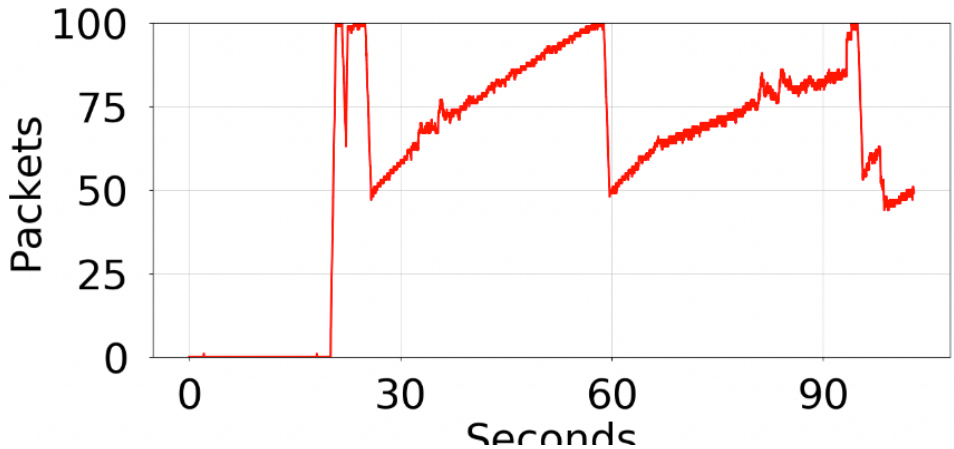

Figure 5 - Switch Queue Occupancy (with qlen 100)

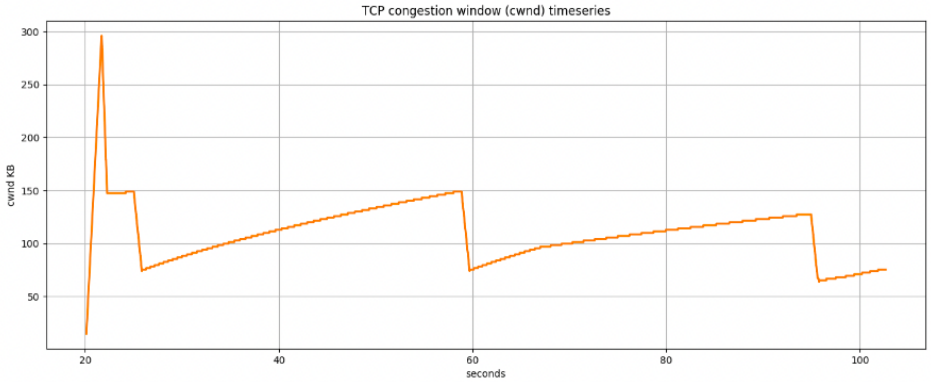

Figure 6 - TCP CWND for iperf (with qlen 100)

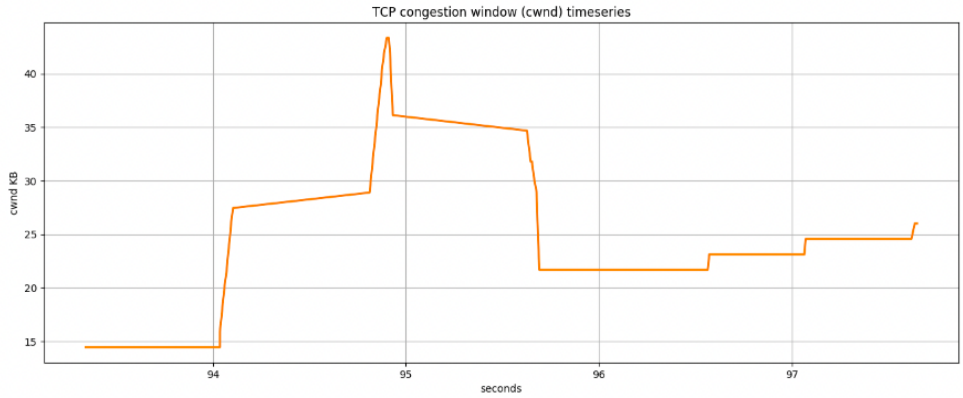

Figure 7 - TCP CWND for wget (with qlen 100)

In figure 5, I can see that it takes about more than 30s to completely fill up the buffer from half the size, which is a very long time. There are also 3 main times when the buffer gets completely filled and experience a packet loss event, which are approximately at times 25s, 55s and 95s.

In figure 6, the tcp cwnd size for iperf is generally between 75 and 150 KB, and similar to figure 5, it is evident that there are 3 main packet loss events at approximately 25s, 55s and 95s. The first packet loss was when the window size was at around 300 KB, then dropped to 150KB, then 130KB. The ssthresh seems to generally be at 75KB. The connection also seems to be at slow start state at around 20-25s, where the congestion window size increases very quickly, and at congestion avoidance state at times 25-55s and 60-95s. The packet loss event also seems to be from duplicate ACK since the window is halved instead of restarting from 1.

In figure 7, the initial packet loss event was when the window size was around 44KB. The tcp cwnd size for wget seems to be between 22 and 44 KB but is difficult to tell since the download is too short. The window size increased exponentially at first from 94-95s, and experienced a packet loss event at 95s where the window size then increased linearly from 95.7s onwards. It seemed to be in slow start state at first then eventually changed to congestion avoidance state. The loss event also seem to be from duplicate ACK since the window did not restart from 1.

Please provide the figures for the second experiment (with qlen 20).
*** Please comment on what you can see in the figures**

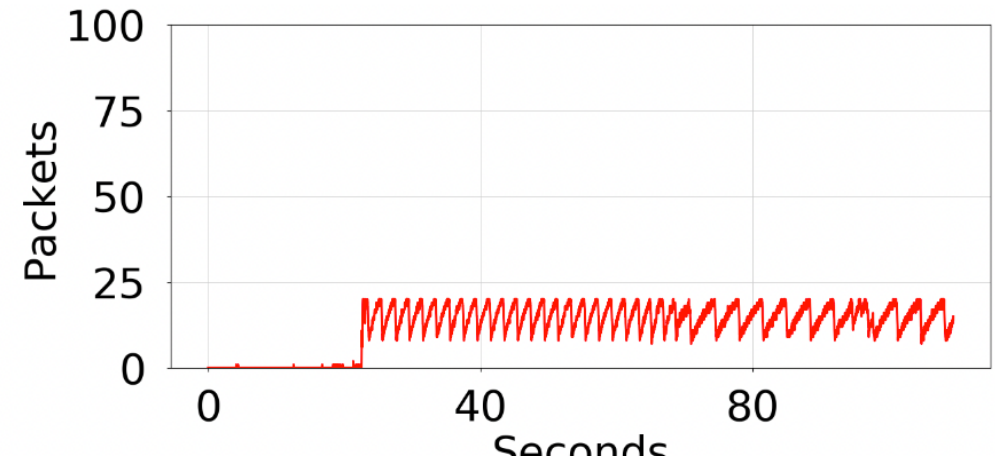

Figure 8 - Switch Queue Occupancy (with qlen 20)

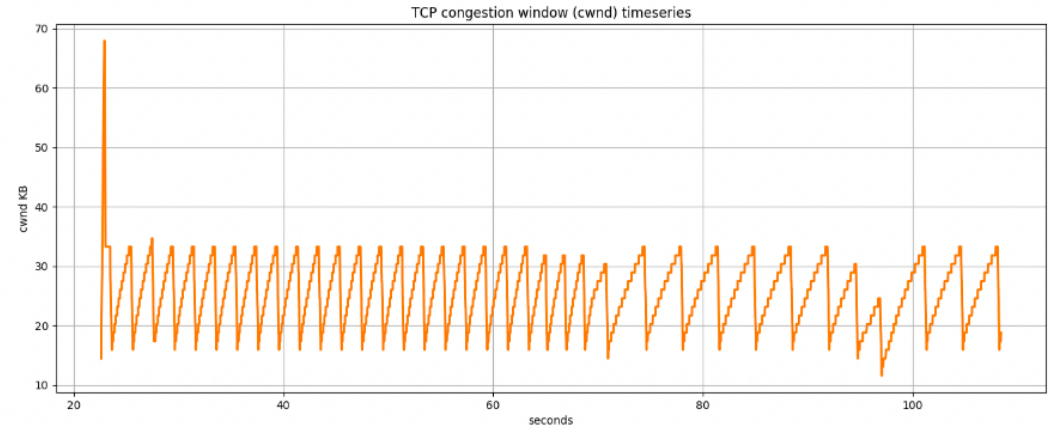

Figure 9 - TCP CWND for iperf (with qlen 20)

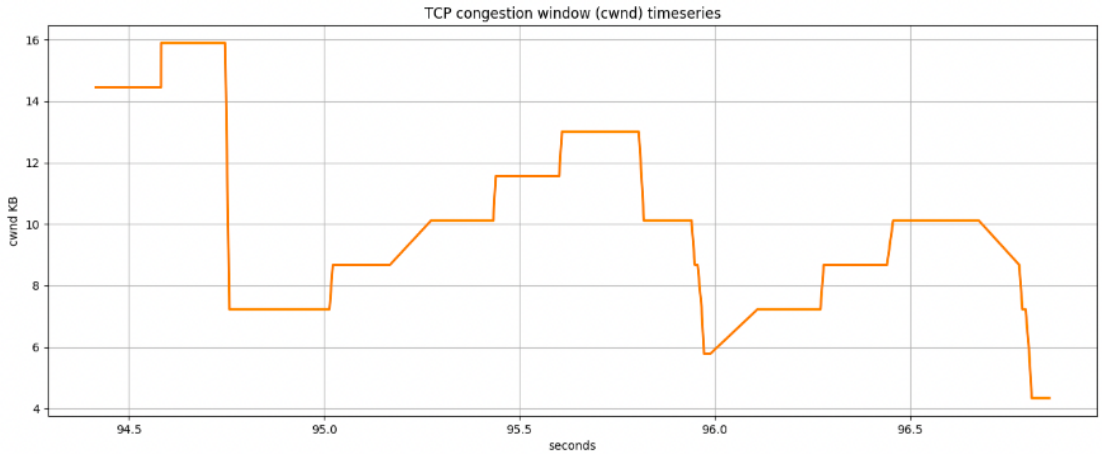

Figure 10 - TCP CWND for wget (with qlen 20)

In figure 8, I can see that it takes a much shorter time to completely fill up the buffer from half the size. There are also many packet loss events within an approximately same time frame as in experiment 1.

In figure 9, the tcp cwnd size for iperf is generally between 17 and 34 KB, and similar to figure 8, it is evident that there are many packet loss events. The first packet loss was at around 68KB, then became around 34KB, thus the ssthresh should generally be about 17KB. The packet loss event also seems to be from duplicate ACK since the window is halved instead of restarting from 1.

In figure 10, the initial packet loss event seems to when the window size was around 16KB and the second packet loss seems to be when the window size was around 13KB. The tcp cwnd size for wget seems to be between 6 and 13 KB but is also difficult to tell since the download is too short.

*** What is difference from the previous experiment. Explain the reason behind the difference.**

```
mininet> h1 ./iperf.sh
started iperf
mininet> h2 wget h1
--2021-10-29 17:00:32--  http://10.0.0.1/
Connecting to 10.0.0.1:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 177669 (174K) [text/html]
Saving to: 'index.html.1'

     0K .......... .......... .......... .......... .......... 28% 68.3K 2s
    50K .......... .......... .......... .......... .......... 57% 58.4K 1s
   100K .......... .......... .......... .......... .......... 86% 49.5K 0s
   150K .......... .......... ...                              100% 74.4K=2.9s

2021-10-29 17:00:36 (59.6 KB/s) - 'index.html.1' saved [177669/177669]
```

Figure 11 – Time to download webpage on h1 from h2 with iperf and qlen 20

```
--- 10.0.0.2 ping statistics ---
30 packets transmitted, 29 received, 3% packet loss, time 29042ms
rtt min/avg/max/mdev = 99.001/141.420/181.421/25.785 ms
```

Figure 12 – ping RTT with iperf and qlen 20

The difference between this and the previous experiment is that the duration for the download is lower, at only 2.9s, the throughput for the connection for downloading is also higher at 59.6 KB/s, as can be seen from figure 11, and the average ping RTT is now lower compared to experiment 1 at around 141ms, as seen from figure 12.

The reason for this is obviously due to the smaller queue length of 20, compared to 100. By comparing figures 5 and 8, we can see that with a smaller queue length, there are more packet loss events, and the time it takes to completely fill the buffer is so much shorter. This is very important because TCP relies on packet loss events as a signal to slow down the sending rate by adjusting the congestion window size. Without packet loss events, TCP will keep testing the network bandwidth by continually increasing the window size.

In experiment 1, the reason for a much longer download time is most likely due to a phenomenon known as bufferbloat, where there is high latency due to excess buffering of packets, usually when the buffer size is large.

Starting iperf simulates long flow where there is another connection sending large segments and hogs the bandwidth (Figure 5). When the network becomes congested, the packets then have to queue for long periods due to the large buffer, especially since it takes a long time for the buffer to be filled and result in a packet loss, which will signal the adjustment of the sending rate. This causes huge amount of delays and therefore, the transfer of packets for downloading the webpage in experiment 1 ends up taking a much longer time due to these delays.

Hence, with a smaller buffer like in experiment 2, packets will not queue for an excessively long time as the buffer gets filled quickly, resulting in more packet losses, sending signals to adjust the window size to get a more appropriate sending rate. Although a smaller buffer can help to minimize the bufferbloat phenomenon, we do have to take note that just as we cannot have a very large buffer, we should not have a very small buffer too.