Shahzil Lakhani
CSC458
lakhan77
1004286408

CSC458 A2 – Bufferbloat

Introduction

Bufferbloat is the term that refers to the case where excessive buffering in a network can cause long latency times and unreliable delivery of packets.

Buffering is a mechanism used to avoid overwhelming a network, and to avoid having to drop packets when the flow of information being sent by hosts temporarily becomes more than the network can transmit. It allows for excess packets to remain at the buffers within the networks, to be sent when the network becomes less busy and more free. The advantage to using such a mechanism is that in some cases, the number of packets dropped may be fewer, as the packets can wait in the buffer and be sent at a later time, but one disadvantage may be that packets may have to wait longer before being transmitted.

Consider a hypothetical network with sizable buffers and suppose that it gets busy at some point. Assume that at this point, the total flow of packets that the connected hosts are sending is greater than the network can handle immediately. At this point, due to large buffers in the network, some packets may be stored in buffers to reduce the load on the network, with the plan to send the packets along as soon as the network becomes less busy. Due to the large size of the buffers, the buffers may continue to take on more and more packets if the network continues to be busy, and thus the packets in the buffer may have to wait for a long duration before there is an opportunity to send them.

Other than a long wait time, the other problem is that hosts detect the capacity of the network by increasing their rate of transmission until packets start getting lost in the network, at which point they detect that they are sending too much and drastically reduce their rate of sending. (This is known as the "TCP Sawtooth" and this process is done as hosts do not know the exact capacity of the network that they are sending through). However, when large buffers are used, the packets are not immediately lost/rejected – they are stored in the buffer to be sent later. This means that the routers do not get a notice that they are sending more than the network can handle, and so they continue to increase their rate of transmission by sending more packets. When the buffers in the network finally become full, the hosts realize that they need to send less, but at this point there is a great amount of extra data still in the buffer, and the network can possibly become completely unusable for a period of time as it works through the bloated buffers and sends the packets to the destinations and is unable to accept many new packets from hosts. In some extreme occurrences of bufferbloat, if many packets are dropped at once, the host may believe that the entire connection has been dropped or the network is down, and go in search of another path as well.

Simulation Setup

In order to show the incidence of bufferbloat, a simulation was created in Mininet to emulate a simple network topology including a client, a server, and a router between them. The client node has a connection of 1 Gb/s to the router, while the server node has a slower connection to the router, of

Shahzil Lakhani
CSC458
lakhan77
1004286408

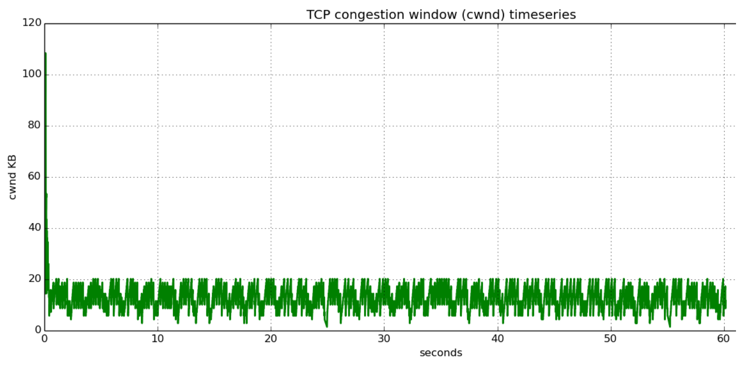
10MB/s. The round trip propagation delay between h1 (the client) and h2 (the server) is 4ms. Finally, the buffer at the router can hold 100 full sized Ethernet frames, which equals about 150KB. During this simulation, there is a long lived TCP flow sending data from h1 to h2 set up using the *iperf* command. There are 10 pings per second being sent from client to the server and their round trip time is being recorded. Finally, a webpage is being downloaded from the server every two seconds, and the time taken for this operation to occur is also being recorded. The factor that is being varied in this simulation is the maximum queue/buffer size, and three different values are to be used: 5, 20, and 100. The simulation is run for 60 seconds each time. The results will all be plotted on graphs, so that the differences can be clearly observed. The summary will be given in the next sections.

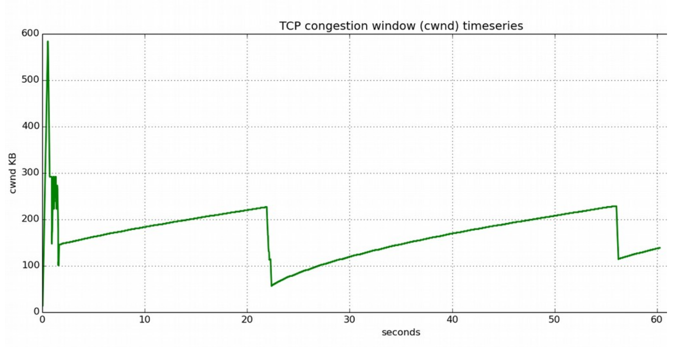
Results

In general, the results of this simulation implied that bufferbloat was occurring when the buffer size increased, and showed the negative effects specific to bufferbloat. Firstly, it was shown that the congestion window took longer to drop when the buffer size was larger. It was also observed that pinging the server took noticeably longer with a larger buffer size. Likewise, the time taken to download a webpage took significantly longer with a larger buffer size due to bufferbloat issues. These observations are explained in the next sections, along with a sample of graphs.

TCP Sawtooth

The TCP Sawtooth results varied among different buffer sizes. The graphs for a buffer size of 5 (left) and 100 (right) are shown below; the graph of a buffer size of 20 is omitted since it followed the same trend.





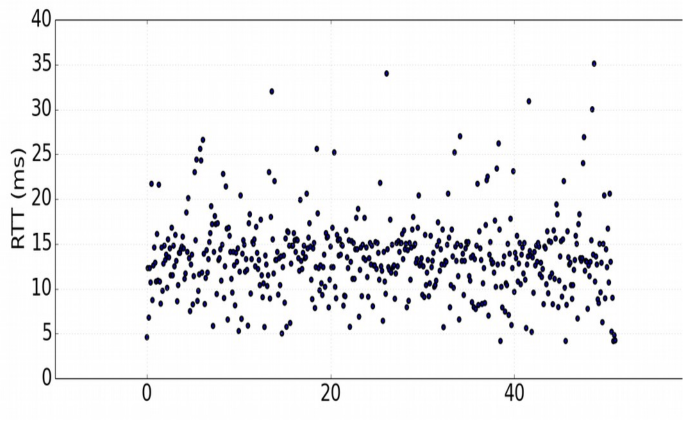
The differences between the two graphs are pronounced. On the graph with the buffer size of 5, we see that the TCP congestion window fluctuates a lot more than the one with the graph with a buffer size of 100. This is because with the buffer size of 5, the network starts dropping packets sooner than the one with the buffer size of 100. This tells the hosts that they should reduce their rate of sending, and so the congestion window decreases more regularly than in the network with a buffer size of 100. The graph with a buffer size of 20 falls in the middle of these two graphs in the sense that the congestion window

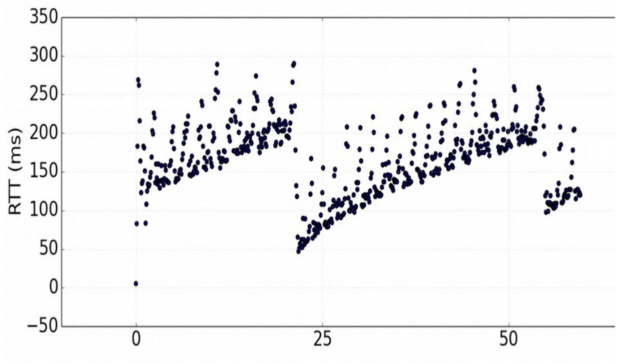
Shahzil Lakhani
CSC458
lakhan77
1004286408

decreases less regularly than the one with a buffer size of 5 and more regularly than the one with a buffer size of 100.

Ping

The results of the ping process also indicate that bufferbloat has occurred. We can see that as the buffer size increases, the round trip time (RTT) increases as well. Graphs of buffer size 5 (left) and buffer size 100 (right) are shown below.





From the graphs shown above, one can see that overall, time for ping increases as the maximum queue size increases. When comparing the graphs for a buffer size of 5 and a buffer size of 100, the buffer size of 5 shows a flat trend for the most part with ping times being very low, and with low variability. For the graph with a buffer size of 100, we see that the ping times are much higher comparatively, and we also see that the ping times are more variable compared to with a buffer size of 5. This is likely because of the bufferbloat problem, which causes packet delivery to be less smooth due to a variable number of packets being stored in the buffer. When the buffer occupancy is reduced, this might be an explanation for a sudden drop in RTT. However note that even after the drop, the RTT is still higher than for that of ping. The relationship between the ping times and the queue sizes is linear. The following table shows the ping time for various queue sizes:

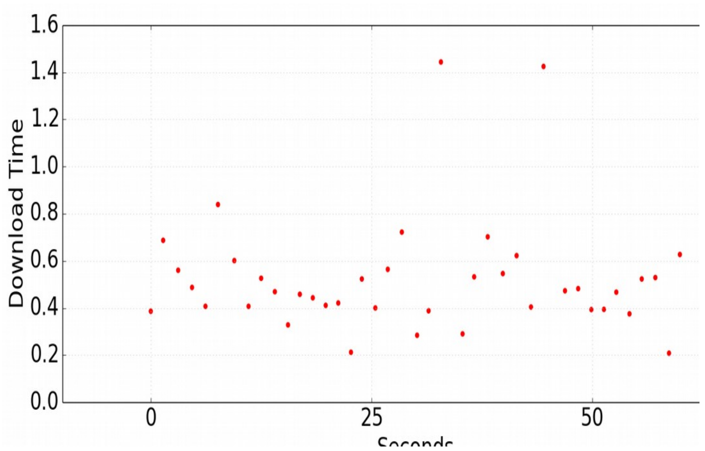
Queue Size	Ping Time
5	12.92
20	37.5603242321
100	169.754806723

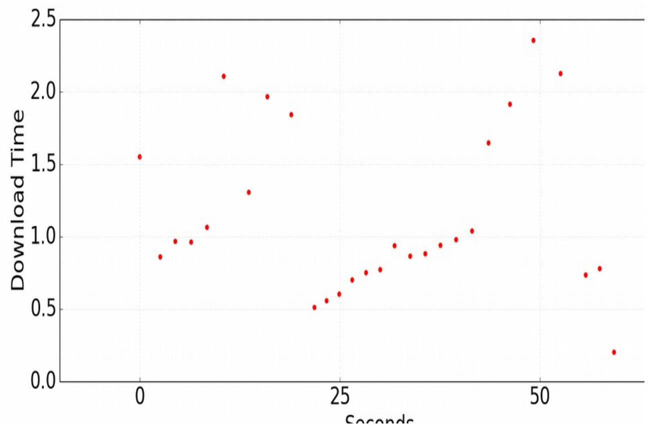
A rough equation that models this relationship is $y = 1.64x + 5.75$. Hence, queue size linearly affects round trip time.

Shahzil Lakhani
CSC458
lakhan77
1004286408

Webpage Download Time

Predictable results also arise from the webpage download time plots. As the buffer size increases, the webpage download time also increases. Graphs of buffer size 5 (left) and 100 (right) are shown below; the graph of buffer size 20 is omitted due to having a similar trend.





From the graphs shown above, we can see that when we compare the graph with a queue size of 100 to the one with a queue size of 5, the simulation with a queue of 5 was much quicker to download the webpage. This may be because bufferbloat occurred with the webpage with the larger queue size, causing packets that were part of the webpage to be stuck in the large buffer for some time instead of being dropped by the network and being resent. This shows the negative impact of bufferbloat to network speed -> users may be able to easily detect this symptom, as it would take a greater amount of time for a webpage to show up on their screen for example, or when streaming audio or video, more issues would be observed by the user because the data flow would be choppier. The graph with a queue size of 20 follows the same trend as the two graphs pictured above in that webpage download time is longer than for a queue size of 5 and shorter than for a queue size of 100, so it is not pictured above. The average and standard deviation of the webpage download times is as follows:

Queue Size	Average Download Time	Standard Deviation Download Time
5	0.6572162162162161	0.6497381649405225
20	0.6794444444444445	0.23927755204694867
100	1.2164814814814815	0.5677394009885541

Further Analysis

Bufferbloat can also occur in other places, such as in the network interface card. In the virtual machine used to conduct the experiment, the maximum transmit queue length is 1000, which is equivalent to 1500kB of data. If the queue drains at 100MB per second (equivalent to 100 000KB per second), then it would take 0.015sec at most before a packet would be cleared from the network interface card.

Therefore, bufferbloat can happen in the network interface card as well and one should check it as well as the other parts of the network when investigating bufferbloat.

Mitigating the Bufferbloat problem

There are several ways in which the bufferbloat problem could be mitigated. Firstly, the network's buffer size could reduce the maximum buffer size down to a reasonably small amount. From the graphs shown above, it is apparent that this helps to reduce the negative impacts of bufferbloat, because this smaller buffer size means that the network doesn't store unnecessarily large amounts of data which will overwhelm it later. Evidently, this leads to overall better performance experienced by the end user as shown from the webpage download times and the ping times.

Another way to mitigate the bufferbloat problem would be to send a message back to the hosts that are sending too quickly whenever the buffers begin to be used more. One of the problems that occurs with bufferbloat is that with large buffers, hosts are not promptly informed when they are sending too much data in the network, so they continue to increase their sending rate until the buffers themselves are overflowed. However, if a message was sent back to inform the culprit hosts to reduce their rate of sending when the buffer begins to be used, they could reduce their sending rate to a more appropriate rate before the buffers get overwhelmed as well. Although this would cause a higher load on the network in the shorter term (more messages being sent back to hosts to tell them to reduce their sending rate) it would mitigate the negative effects that occur when buffers are completely filled.

Sources

F. Casey, "Beginner's Guide to Bufferbloat - How to fix network lag," *Netduma.com*, 18-Jul-2018. [Online]. Available: https://netduma.com/blog/beginners-guide-to-bufferbloat/. [Accessed: 28-Nov-2020].

"Introduction," *Introduction - Bufferbloat.net*. [Online]. Available: https://www.bufferbloat.net/projects/bloat/wiki/Introduction/. [Accessed: 28-Nov-2020].