Bufferbloat

Kanu Monga
Dept. of CSE, SBSSTC, Ferozepur

Sahil Kumar
Dept. of CSE, SBSSTC, Ferozepur

Abstract

In order to prevent packet drop, network devices are configured with much larger buffer sizes than before because of dramatic price drop of memory. This misleads the assumptions of the TCP congestion avoidance algorithm which depends on packet loss for detecting and preventing network congestion. When the link is saturated then large buffer sizes cause dramatic reduction of throughput rates for applications.We investigate the effects of this bufferbloat phenomenon on throughput and latency by using RRUL test for default queue DropTail.This test is run using the netperf-wrapper testing harness. Our results show that increasing buffer sizes reduces throughput and sharply increases latency.

Keywords—*Bufferbloat, Queueing delay, Dark Buffers*

---

1. Introduction

Modern computer networks are incredibly complex, and decades of research has gone into ensuring they work accurately on the global internet scale in the face of unreliable transport links and varying bandwidth and latency. We depend on computer networks to transport huge quantities of data across the globe every second, and this works well, but not always without drawbacks. In recent years, Internet has observed an dramatic increase in its end to end latency.

Recent research has shown that this is due to buffers which are built in to every piece of network equipment and this problem has been named as Bufferbloat. The term bufferbloat has been coined by Jim Gettys in 2010 to describe the phenomenon of increase in latency when network links are saturated. “Bufferbloat is the existence of excessively large (bloated) buffers in systems, particularly network communication systems." [1] [2] In this work, bufferbloat problem has been seen by using RRUL test designed by bufferbloat community. This test can be run by using tool netperf-wrapper.

2. Bufferbloat

Over the past few years, there has been a massive increase in the Internet usage and that's why network management and control is becoming even more challenging. Recently, there has been much discussion of the bufferbloat problem in the network research community. The bufferbloat problem occurs when buffers are extremely large and unmanaged. Some buffering is required for the correct functioning of network to handle bursty traffic and to achieve

highest link utilization. But when buffer size is significantly larger than it needs to be, it leads to problem like Bufferbloat [3] [4] .

Because price of memory has decreased drastically over the past few years, network devices are designed with much larger buffer sizes than before in order to prevent packet drop and this defeats the TCP congestion control algorithm. TCP is responsible for carrying most of Internet traffic and TCP depends on packet loss to adjust its rate. It keeps increasing its rate until packets get dropped. Ideally, it speeds up and slows down until it finds an symmetry equal to the speed of the link. However, for TCP to work correctly, the packet drops should occur in a timely manner. If the buffers are too large [5], packets sit in the buffer queues instead of being dropped, and no signals reach the endpoints; means they do not slow down in a timely manner, and the buffers stay full. Furthermore, each new packet that enters the buffer has to wait for all packets queued before it to be transmitted before it can continue on its way. These two effects combine to create what is referred to as bufferbloat [6] [7] and the consequence is a loss of interactivity that can cause users to experience huge delays in their network usage and applications to time out.

Due to Bufferbloat [8] [9], many applications can fail completely or partially. Bufferbloat can cause your network to be ten or even a hundred times slower than it should be when suffering from bufferbloat. Applications like networked gaming, voice calls (Voice over IP), video chat programs and other interactive applications such as remote login and instant messaging suffer most. Any type of a service which demands consistently low latency or jitter-free transmission whether in low or high traffic bandwidths can be severely affected. When bufferbloat is present and the network is under load, even normal web page loads can take many seconds to complete, or simple DNS queries can fail due to timeouts.

3. TCP's relationship with bufferbloat

TCP congestion control [10] [11] has a slow start method in which the initial rate of sending data is low. It increases the speed of sending data until a packet drop is detected. To get an optimum data rate which does not cause congestion, TCP increases and decreases the speed of data transmission according to the packet loss. Thus TCP relies on the timely detection of packet loss in order to prevent congestion and achieve an optimal transmission rate.

Bufferbloat means that packets are buffered instead of dropped, sometimes for lengthy periods of time. In the meantime, TCP continues sending packets, causing even further queue buildup at the bottlenecks. When packet drop then occurs, more packets are dropped than necessary, and the exponential back-off mechanism of TCP causes a sharp drop in transmission speed, freeing up bandwidth. This in turn causes TCP to increase its bandwidth again and, because of bufferbloat, no timely congestion notification reaches the sending host, so the bandwidth is once again increased to too high a level, causing another exponential back-off.

This behaviour can repeat indefinitely and cause throughput degradation as well as latency problems. When multiple TCP streams share the same buffer (e.g. because they go through the same bottleneck router), the back-off behaviour has a tendency to synchronise because the flows see packet drops at the same time (when the buffer fills). This causes all the flows to throttle back their transmission rates simultaneously, amplifying the effect. The synchronisation behaviour is known as TCP global synchronization.

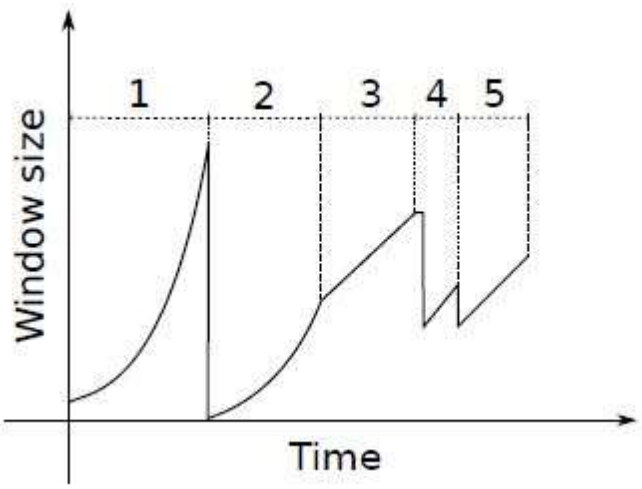

Fig. 1: TCP window Increasing Mechanism

4. Experimental Results and Analysis

4.1 Methodology

In this paper, the bufferbloat problem has been studied and investigated on real time Internet using Realtime Response Under Load (RRUL) test designed by the bufferbloat community. This test can be run by using tool netperf-wrapper.

The netperf-wrapper tool

The main benchmarking tool used for the performance tests is the Netperf benchmark [12]. It supports various modes of sending TCP and UDP streams between a client and server and measuring the throughput and/or roundtrip time. However, diagnosing bufferbloat requires several such streams be run at the same time (at minimum, a roundtrip time measurement while another stream loads up the link), and no tools existed for running such tests and aggregating the results into a single data set. To remedy this lack, the tool, netperf-wrapper [13], is implemented as a Python program that runs various benchmarking tools concurrently, aggregating the results.

Development of the netperf-wrapper tool has been guided by inputs from the bufferbloat online community and the tool is used for diagnosing bufferbloat in various contexts in the community. The main test supported by this tool for testing bufferbloat is RRUL test [14].

The Real-time Response Under Load (RRUL) test

This test is the current prototype of one of the bufferbloat tests under development in the bufferbloat community. It consists of running several concurrent connections: four in each and every direction. It better approximate a real-world busy network where several connections are active simultaneously. The various flows each use a different quality of service marking of the packets, which allows testing for behaviour of routers in the presence of such a marking.

It run a test concurrently running four TCP uploads, four additional TCP downloads and four low-bandwidth workloads, three of which used UDP while the fourth used ICMP ping packets. The graphs in below figures show the throughputs of the TCP streams and the latencies of the low-bandwidth workloads. Here, BE is best-effort (no marking), BK is bulk (class selector 1 (CS1) marking), EF is expedited forwarding, and CS5 is class selector 5 (having higher precedence/priority than CS1).

The RRUL test shows the severely degraded performance of the default DropTail queue when the link is loaded. The top plot shows download speed, the middle one shows upload speed and the bottom plot shows ping times. The bold black lines indicate average values for all streams, while the coloured lines are the individual stream values. The total throughput, as seen, is slightly higher for the upload stream, but much lower for the download stream, leading to a low total throughput and the latency suffers under load.

Fig. 2 shows how the ping times sharply increase from the initial sub-millisecond times as soon as the TCP flows start (delayed five seconds). When using the default DropTail queue, this increase is particularly marked, where ping times increase rapidly to around 1000 milliseconds, and then continues to increase for the duration of the TCP streams, peaking at seconds of latency. This is a clear sign of bufferbloat. The ping packets have to pass through a large queue in both directions, causing very large delays. The graph of the ping times shown in Fig. 3 makes it easier to directly see the latency behavior. It shows the cumulative distribution of the ping time values from the interval of the test where the TCP streams are active. The default DropTail queue experiences severe latency degradation.

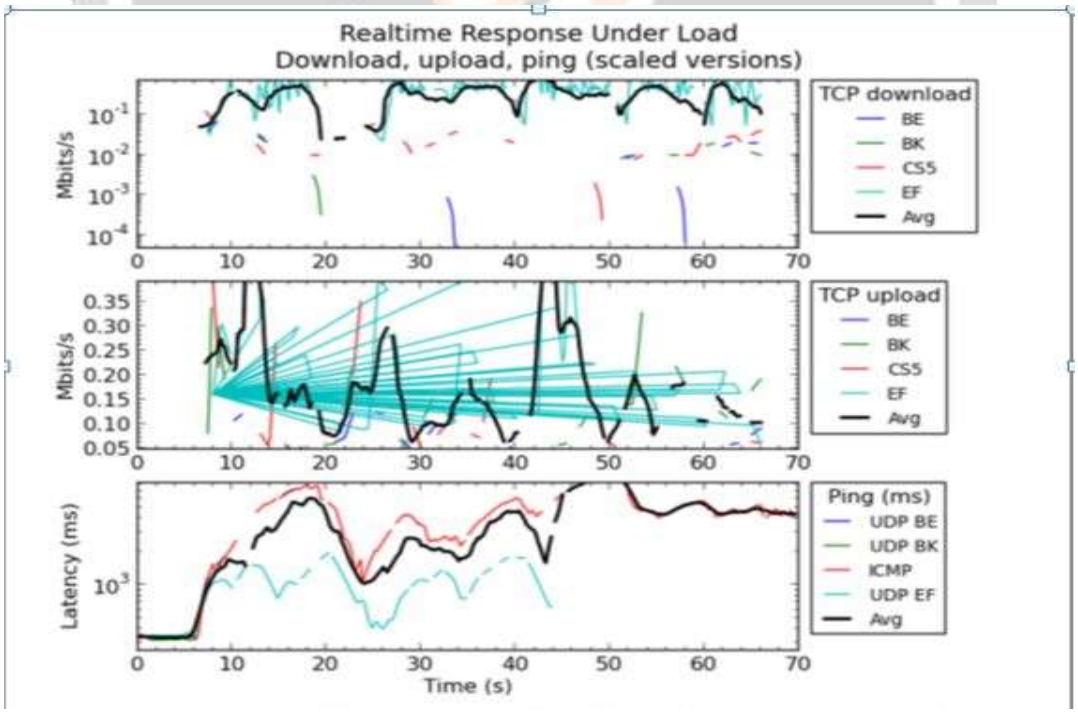

Fig. 2: Bandwidth and ping plots for the RRUL test

5.

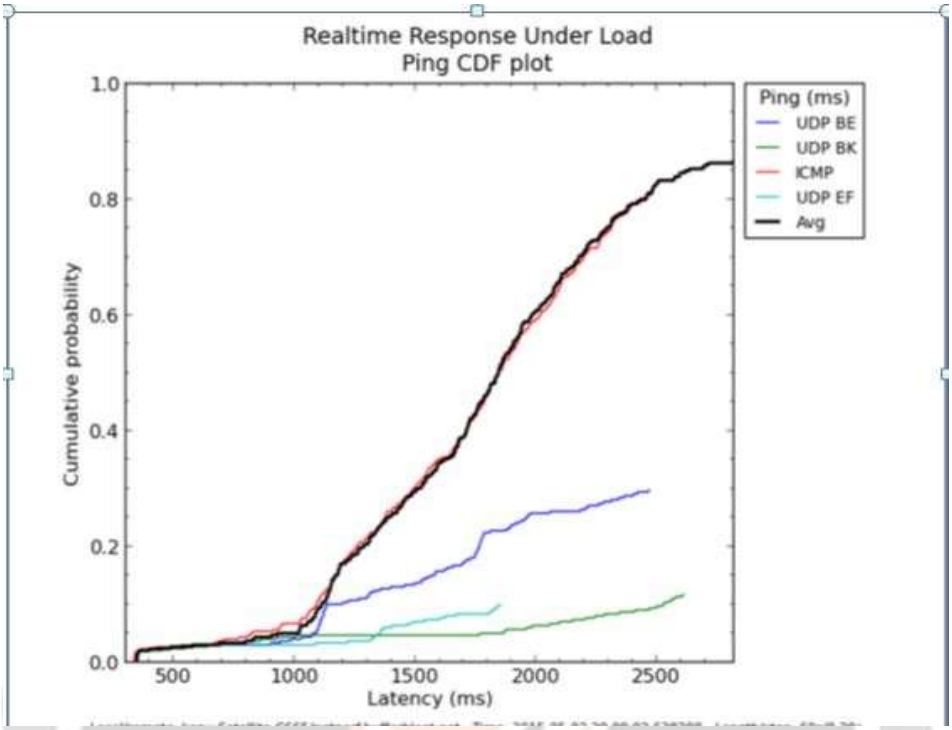

Fig. 3: CDF plot of ping time distributions for the RRUL test

Conclusion & Future Work

Due to growth in the variety of Internet applications, the bufferbloat problems can be easily seen in today's networks. Bufferbloat is a hard problem, with no single right solution It ultimately degrades overall network performance. In this paper, Realtime Response Under Load (RRUL) test is used to study and investigate bufferbloat on real time Internet.

Results show that the default DropTail queue suffers from decreased throughput and significantly increased latency under load when the link is overloaded. The throughput is much lower for the download stream and all this is due to bufferbloat phenomenon.

This work can be extended in many ways. This problem can be tested in varying real-world conditions. Implication of this problem can also be analyzed by varying buffer size. Further, for better understanding this problem deep study and investigation is required.

6. REFERENCES

[1] J. Gettys, "Bufferbloat: dark buffers in the internet," IEEE Internet Computing, vol. 15, no. 3, pp. 95–96, 2011.

[2] J. Gettys and K. Nichols, "Bufferbloat: Dark Buffers in the Internet," ACM Queue, Nov. 29, 2011, http://queue.acm.org/detail.cfm?id=2071893.

[3] J. Gettys and K. Nichols, “Bufferbloat: dark buffers in the internet,” Communications of the ACM, vol. 55, no. 1, pp. 57–65, 2012.

[4] C. Chirichella and D. Rossi, “To the Moon and back: are Internet bufferbloat delays really that large?” in Proc. IEEE INFOCOM Workshop on Traffic Measurement and Analysis, 2013.

[5] G. Appenzeller, I. Keslassy, and N. McKeown, “Sizing Router Buffers,” in Proc. ACM SIGCOMM, 2004.

[6] M. Allman, “Comments on Bufferbloat,” ACM SIGCOMM Computer Communication Review, vol. 43, no. 1, pp. 31–37, 2013.

[7] H. Jiang, Y. Wang, K. Lee, and I. Rhee, “Tackling Bufferbloat in 3G/4G Networks,” in Proc. 2012 ACM IMC, 2012.

[8] Kanu Monga and Krishan Kumar, “Performance Evaluation of Various Active Queue Management for Bufferbloat,” International Journal of Engineering and Advanced Technology (IJEAT), vol. 5, no. 4, 2016

[9] O. Hohlfeld, E. Pujol, F. Ciucu, A. Feldmann, and P. Barford, “BufferBloat: How Relevant? A QoE Perspective on Buffer Sizing,” Technische Universit¨at Berlin, Tech. Rep., 2012.

[10] J. Postel, "Transmission Control Protocol", STD 7, RFC 793 , September 1981.

[11] V. Jacobson, “Congestion avoidance and control,” ACM SIGCOMM Computer Communication Review, vol. 18, no. 4, pp. 314–329, 1988.

[12] R. Jones, “Netperf open source benchmarking software," .2012.

[13] T. Hoiland-Jorgensen, “Netperf wrapper, source code repository,testing harness,” 2012.

[14] T. Hoiland-Jorgensen, “Flent: The flexible network tester"