7.5

Bufferbloat: A simulation

Mark Santcroos - mark@santcroos.net
Sytse van Genderen - sytsevg@gmail.com

February 27, 2011

Abstract

Bufferbloat can be described by the effect of excessive buffering in routers and hosts. This effect has recently been noticed particularly by end nodes on the network, namely users on a fast local area network (LAN), connected to the Internet (WAN) via a router. The difference in throughput between this fast LAN and slower WAN is causing excessive buffering in routers under certain conditions. Users using applications where timely response of data is critical notice the instability and slow responsiveness when another user on the same LAN is sending a lot of data where timely response is not an issue. In this paper we report on a simulation we did to analyze this problem. We use the INET Framework of the OMNeT++ simulator to reproduce the bufferbloat behavior. We show that by creating the right combination of topology, traffic and buffer sizes the bufferbloat effect becomes apparent.

1 Introduction

The TCP protocol is a transmission service that lies in the transport layer level of the OSI model. It has been designed to transport data reliably over a network. As a network can have congestion, traffic load balancing and unpredictable behavior, TCP has a number of properties to adapt to these problems.

Modern implementations of TCP contain four main phases, Slow-start, congestion avoidance, fast retransmit, and fast recovery[1]. The slow-start phase is when data begins to transmit and the congestion window is exponentially increased by the number of segments acknowledged. This happens until an acknowledgement for a segment is not received or the window reaches the size of the receivers advertised window. At this point it will step over to congestion avoidance.

TCP uses a sliding window flow control to avoid sending the data too fast for the receiver to control or to receive and process it reliably. This sliding window gives the TCP a bursty behavior. Initially the window could not be greater than 2$\hat{1}$6 bytes, but has been expanded to use a scaling option to support up to 1 gigabyte[2]. Support for window scaling has been implemented starting with Windows 2000 / Vista in 2001 and with Linux in 2004.

Modern day routers that connect a user to the Internet have been fitted by very large buffers[3]. This can be explained by the fact that memory is cheap now compared to 20 years ago. Recently a blog published by Jim Getty points out to an effect end users seem to be having in their home

better explanation about buffer bloat: end

networks.

A typical home user connected to their home network via LAN at 100 Mbps or greater, or via Wi-Fi at 54Mbps or greater are bottle necked by their ADSL Internet connection. ADSL supports speeds of around 20 Mbps download and 1Mbps upload. The effects are best seen because the local area connection is typically much faster than the Internet connection, and with a router which has a large buffer, queues the outgoing packets.

The remainder of this paper is as follows, in the following section we will describe the experimental setup of our simulation, in the next section we will give the results of our experiments and explain some of the observed effects. In the section thereafter we will bring the theory and our results together in a discussion. The paper ends with conclusions and recommendations.

goede introductie, probleem, waarom, context uitgelegd

2 Simulation

2.1 OMNeT++

For our simulations we make use of the OMNeT++ simulator. OMNeT++ is an extensible, modular, component-based C++ simulation library and framework, primarily for building network simulators[4]. The INET Framework [5] is an extension of OMNeT++ that implements many of the TCP/IP related protocols. We will use the PPP, IP, TCP and UDP protocols in our setup.

2.2 Network Topology

The topology we used for the simulation is depicted in figure 1.

All links in out setup are PPP links. The end nodes are connected with a speed of 1Gbps and a 0.1us delay to their respective router. The link between the two routers is a 1Mbps link with a 20ms delay. The network protocol IPv4 and its addressing is statically configured. This network tries to mimic the end-users connected via LAN to the Internet via a router with a large buffer. All systems are configured with the Reno TCP stack, this is the default and most mature in OMNeT++.

Heldere opzet

2.3 Network Events

In our simulation we make use of the following events in various combinations:

2.3.1 Event A: TCP Data Transfer 1

TCP transfer of 20 MB in one session from Client 1 to Server 1. Starts at time = 0s and finishes at time = 200s. The server side is implemented as a sink, so the traffic is asymmetric.

2.3.2 Event B: TCP Data Transfer 2

TCP transfer of 2 MB in one session from Client 2 to Server 2. Start at time = 40s and finishes at time = 80s. The server side is implemented as an echo, so the traffic is symmetric.

2.3.3 Event C: UDP Data Transfer

A continuous and timed stream of UDP packets from Client 2 to Server 2. The transfer starts at time = 0s, for every t. The transfer is asynchronous as the server is implemented as a a sink. The receiving side measures and records the delay variation of the incoming packets.

2.4 Scenarios

Table 1 lists all the scenarios that will be used for evaluation of our topology and events in the

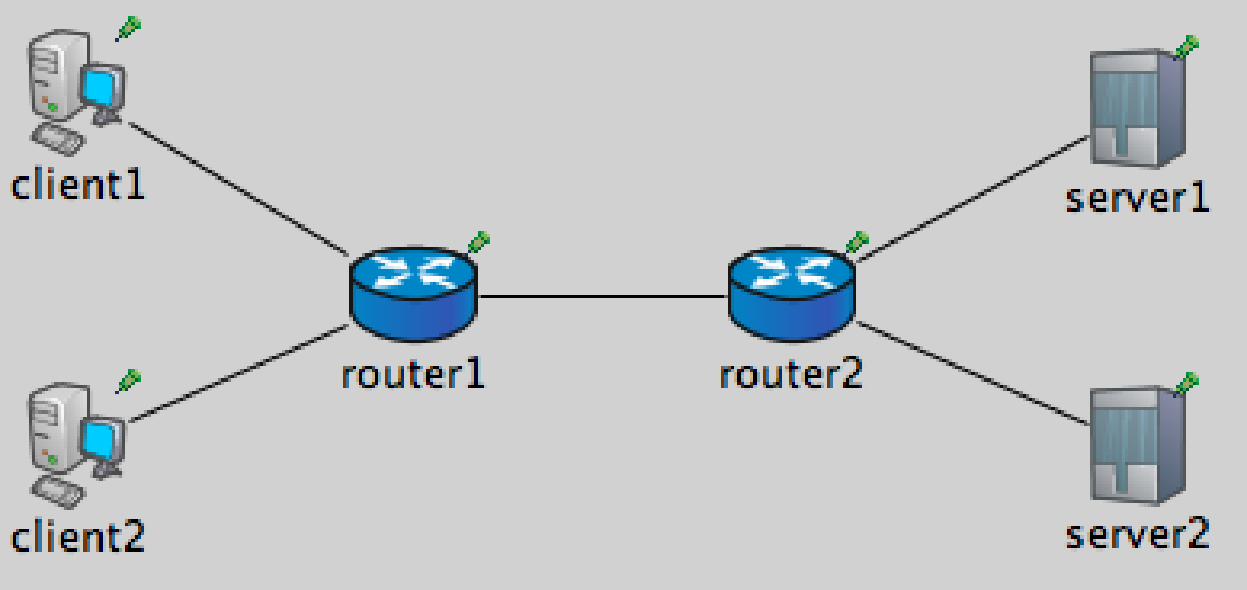

Figure 1: Network Topology

simulator. The following parameters are taken into account:

- Router 1 Queue Size
- Events
- TCP Window Scaling
- TCP Advertised Window

3 Results

3.1 Influence of TCP Advertised Window size

These first series of results try to visualize the effect of the Advertised TCP Window size on the behaviour of the connections.

In Figure 2 we display the growth of the queue on the outgoing interface on router 1. The length of the queue is kept in number of frames. Because the link directly after the outgoing queue is the bottleneck in our network, it is not surprising to see that packets start to queue more as the Window size increases. It can clearly be seen that the initial exponential queue increase is the Slow-Start phase, till it reaches the size of the send window. This excludes the tests where the send window is greater than the maximum 2^{16}, it will be limited by the Slow-Start thresh hold. During the period that both Event A and Event B are taking place we see a doubled rate of increase for all Window sizes except for Window size 2^{16}. The maximum of 262144 2^{18} was chosen as with the given data volume and line speed no difference is made with increasing it further.

As the measured RTT is of course directly dependent on the queuing delay, we see in Figure 3 a strong correlation to the queue length in Figure

voor kleine windows blijft de qeueue gelijk. komt omdat je eigenlijk in het netwerk niet kan bufferen

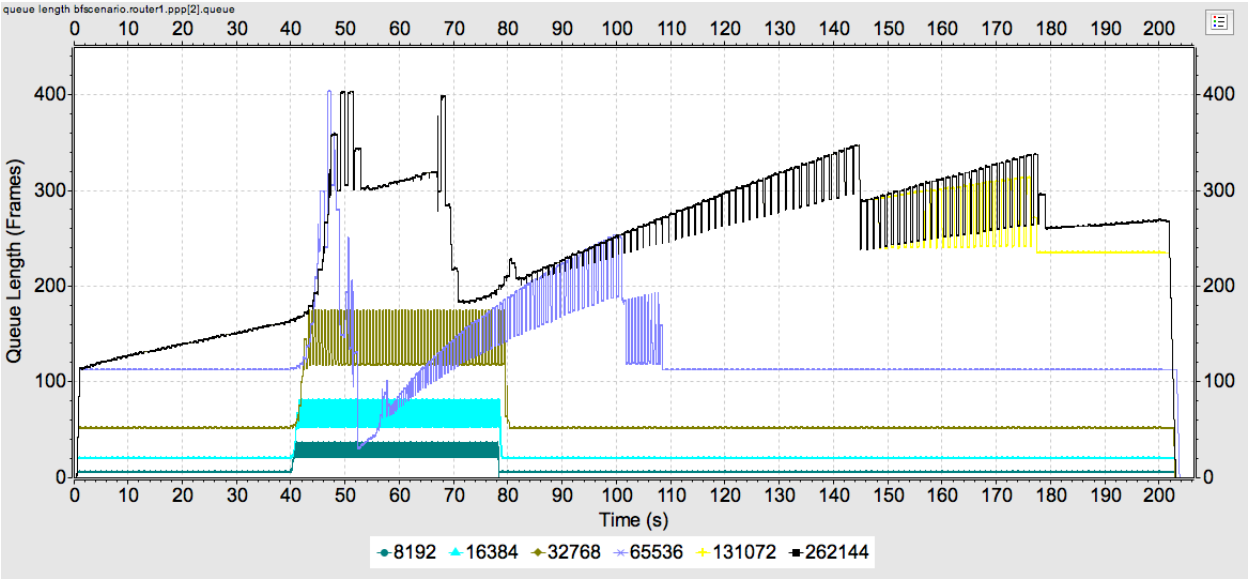

Figure 2: Queue Length for varying Advertised Window sizes

Window size bepaald grotendeels de capaciteit tot buffering en daarmee ook de latency

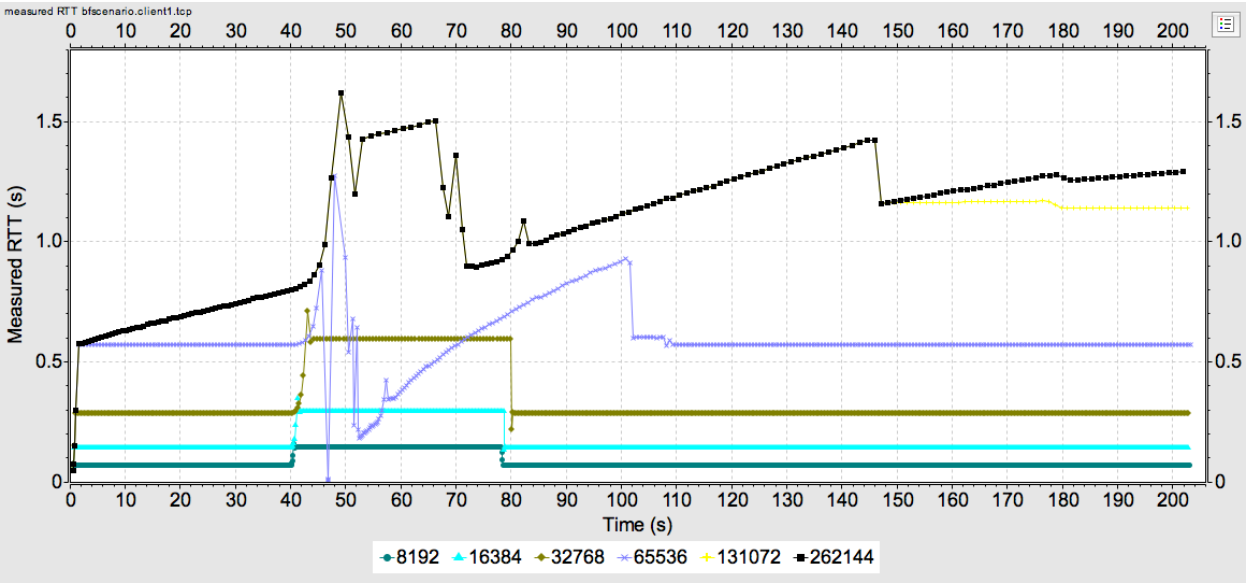

Figure 3: Measured RTT for varying Advertised Window sizes

Jitter wordt er ook mee beinvloedt

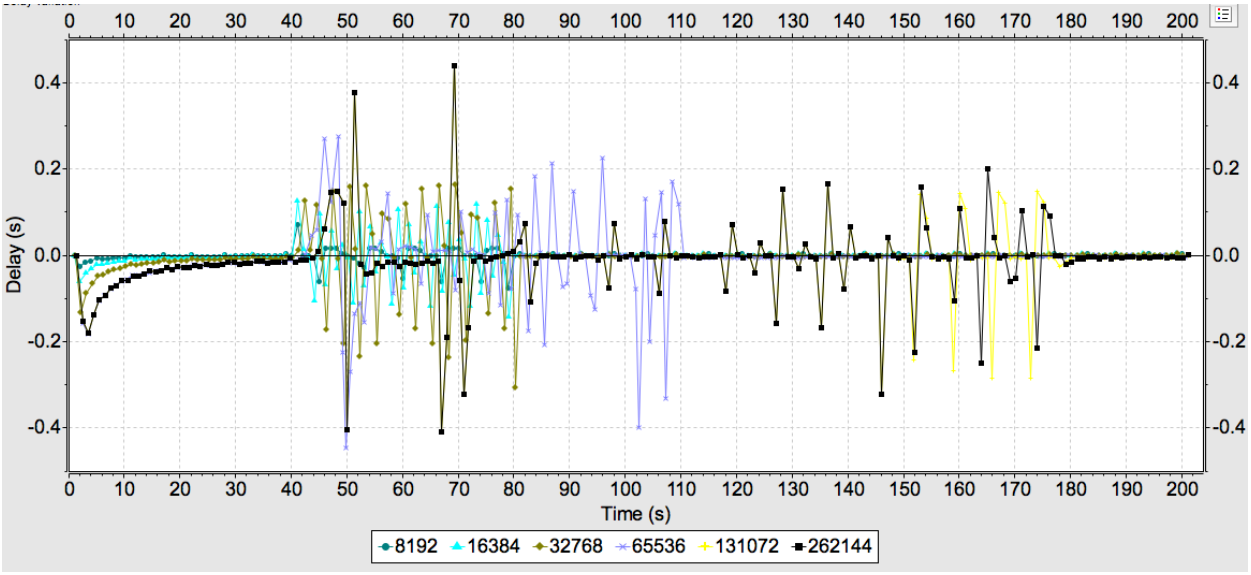

Figure 4: Packet Delay Variation for varying Advertised Window sizes

2.

The recorded Packet Delay Variation of Event 3 is shown in Figure 4. Per arrived packet the delay since the last packet is calculated. We know that a packet was sent every second. The delay is then compared with the mean of the delay, and the difference is shown here per packet.

3.2 Tuning Buffer Size

The next parameter we will experiment with is the buffer size of the outgoing queue on router1. Based on the maximum length the queue grows in the previous experiments we run the experiment with 5 different buffer sizes.

On a glance Figure 5 tells us that there the most frequent outliers are for the small buffers.

Figure 6 shows us a zoom-in on the smaller delay variance (the x-axis is similar). Here the picture tells us something different then what we just described. Although the real outliers are for the smaller buffer sizes, most of the frequent delay variation happens for the larger buffer sizes.

In Figure 7 we can see how hard certain thresholds are. We plot the build up queue length for the earlier mentioned configured buffer sizes. If we for example compare the buffers with size 400 and 500, we can conclude that although the protocol would only need a queue just a bit over 400, the TCP protocol causes a totally different behavior. We can see the typical TCP congestion control behavior in the scenario when the buffer in the router is small, it drops packets, and TCP adjusts to this.

4 Discussion

In the previous section we have described our experiments and commented on the gathered re-

beide hebben ze een stream tussen een andere stream gezet om de impact te meten

dit is moeilijk te interpreteren... sommige kleuren lijken op elkaar...

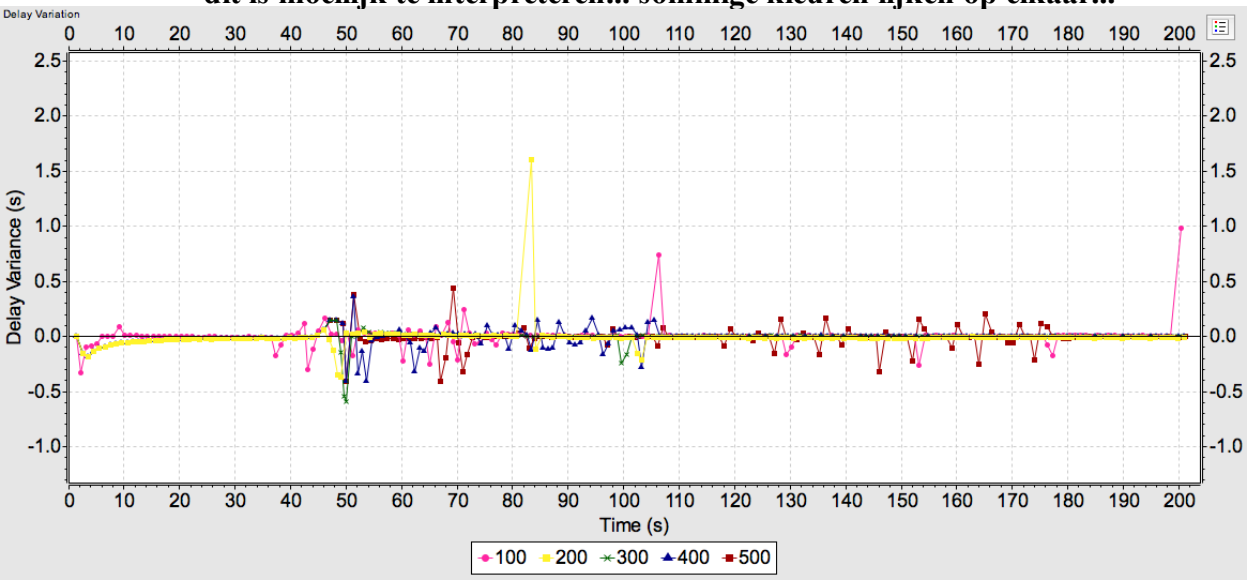

Figure 5: Packet Delay Variance for varying Buffer Size

dit figuur illustreert niet echt de frequentie van delay variantie... histrogram was beter geweest

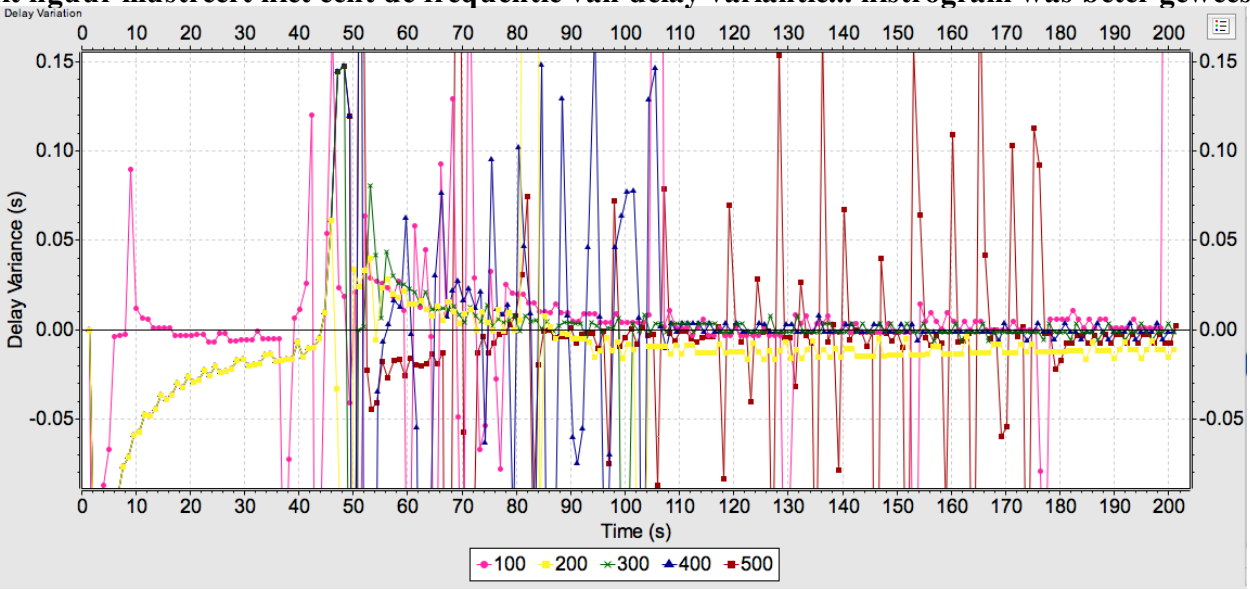

Figure 6: Packet Delay Variance for varying Buffer Size (zoomed)

beetje chaotisch plaatje... wat halen we hier nou uit... er is teveel te zien

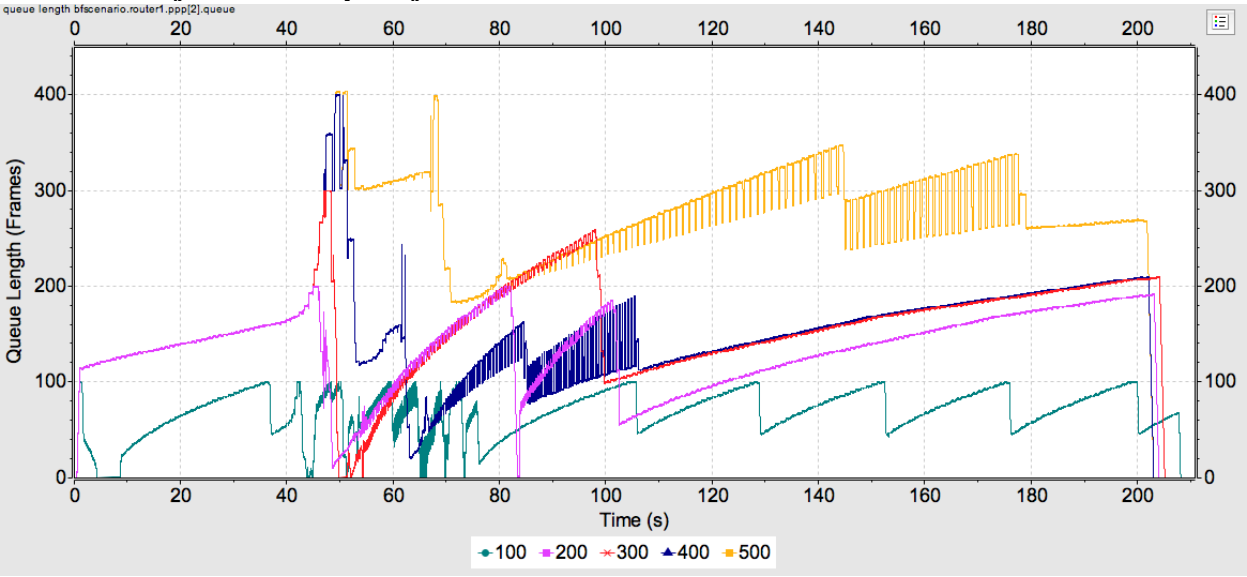

Figure 7: Queue Lengths for varying Buffer Size

sults. In this section we will put the results into context.

4.1 Buffer usage on intermediate router

The result of excessive buffering shows increase in round trip time, starting at 40ms peaking at 2s. Ever increasing windows size calculation due to increasing RTT. As it is in the congestion avoidance phase, there is a linear growth of the congestion window, this growth never stops.

4.2 TCP Advertised Window

As the default AW of the simulator is 14*MSS (7504 bytes) we had to raise this to get reasonable and realistic throughput and not have the receiver be the bottleneck.

the window size wordt aangepast aan de rtt. Die groeit door de buffers. Dan kunnen packets ook langer in transit zijn

4.3 Increased Packet Delay Variation

What does this mean for application experience of the end users? We can say that applications that are not affected by high round trip times and packet delay variance will not have a problem. This can be applications such as video streaming, file downloading or uploading. For other applications such as VoIP, where low latency is a critical factor, will be heavily affected.

good

4.4 Realism

Simulations of a network can never fully represent real world situations. They do however present meaningful data which can be related to different scenarios. In this test there was precaution when tuning the different scenarios to leave all other TCP settings at their default values. This is a completely isolated and focused test.

good

5 Conclusions and recommendations

In our simulations we were able to reproduce the effects of bufferbloat as reported by [3]. The primary tuning parameter for that was, as expected, the size of the buffer on the outgoing interface before a network bottleneck. The simulations also show that TCP was designed to be controlled by packet loss, and that the lack of loss packets and increased RTT hurts its operations. As most router equipment is designed with single FIFO queues, the aggressive buffering of TCP has large impact on other traffic. The fact that the RTT is heavily influenced by the queuing leads us to conclude that either the RTT calculation should be changed (e.g. take the minimal RTT) or it should have less impact on the window size growth. Other possible solutions include per flow queues for more fair scheduling of packets.

they did not try a lot of flows

References

[1] RFC 5681, http://tools.ietf.org/html/rfc5681

[2] RFC 1323, http://tools.ietf.org/html/rfc1323

[3] Gettys, The criminal mastermind: bufferbloat! http://gettys.wordpress.com/2010/12/03/introducing-the-criminal-mastermind-bufferbloat/

[4] Omnet++ Simulator, http://www.omnetpp.org/, 23 feb 2011

[5] INET Framework, http://inet.omnetpp.org/, 23 feb 2011