UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Universidad Técnica Federico Santa María
Departamento de Informática

Testing, Detection and Possible Solutions for the Bufferbloat Phenomenon on Networks.

Thesis Degree

JUAN S. CATALAN OLMOS

Definición de Tema de Memoria
para optar al Título de:
INGENIERO CIVIL INFORMATICO

Referent Professor: Horst H. von Brand Skopnik
Coreferent Professor: Raúl Monge Anwandter

June 3, 2014
Valparaíso, Chile

Contents

1 Introduction **3**

2 The Transport Control Protocol **4**
2.1 Fundamentals of TCP . . . 4
2.2 TCP's Phases . . . 6
2.2.1 Slow start and Congestion avoidance . . . 7
2.2.2 Congestion control algorithms . . . 8
2.2.3 Fast retransmit and fast recovery . . . 10

3 Router's Buffers effects **12**
3.1 Full Buffers . . . 12
3.2 High Latency . . . 13
3.3 Sizing Router's Buffers . . . 14

4 Active Queue Management **16**
4.1 RED . . . 17
4.2 BLUE . . . 19
4.3 CoDel . . . 20

5 Hidden Flaws and Bufferbloat **22**

6 Experimental Work **24**
6.1 The Test Setup . . . 24
6.1.1 Hardware Characterization . . . 24
6.2 Tools Definition . . . 26
6.2.1 Speedtest . . . 27
6.2.2 Netalyzer . . . 27
6.2.3 Iperf . . . 28
6.2.4 Page Benchmarker . . . 29
6.2.5 Smokeping . . . 29
6.3 Test Description . . . 30
6.3.1 Speed test . . . 30
6.3.2 Signs of trouble . . . 31
6.3.3 Collapse test . . . 31
6.3.4 Load benchmark test . . . 32
6.3.5 Smoke the path . . . 33

7 Results **34**
7.1 Speed test . . . 34
7.2 Netalyzr test . . . 34
7.3 Iperf test . . . 34
7.4 Benchmark test . . . 35
7.5 Smokeping Test . . . 35
7.6 Summarising Results . . . 36

8 Conclusions and Further Work **37**

A Definitions **38**

B Speed Test results **39**

List of Figures

1 The Chronology of a Slow-Start.[14] 7
2 The window growth function of CUBIC 9
3 CUBIC's Congestion Window.[i] 9
4 Compound TCP sending window.[ii] 10
5 RED's Flowchart 18
6 BLUE algorithm 19
7 CoDel droptime interval, n=iteration 20
8 Result Summary example from Netalyzr. Source: `www.dslreports.com` 28
9 Speed and Pings based on Appendix B, Table 4 34
10 Variation ratio of offered vs measured speed based on Appendix B, Table 5 35

List of Tables

1 Page Benchmark Test: Total load mean times 35
2 Page Benchmark Test: Variation ratio over own iteration 36
3 Page Benchmark Test: Variation ratio over all 36
4 Speed Test: Speeds and Ping measured 39
5 Speed Test: Variation ratio of offered vs measured speed 40

1 Introduction

2 The Transport Control Protocol

The Transport Control Protocol or TCP is main transport protocol of the internet, providing reliable host to host communication over unreliable transport media[21]. The advantages of its connectionless design, flexibility and robustness provides reliable, ordered delivery of a stream of bytes from a program on one computer to another program on another computer, but for that, the cost are: the needed of careful design so it provides a good service under heavy loads. Many modifications has been done since it was originally defined in 1981.

The root idea of any algorithm related to transport connections must be based into the "*packet conservation principle*". This principle claims:

Definition 2.1. A new packet isn't put into the network until an old packet leaves.

2.1 Fundamentals of TCP

TCP operates basically making two hosts exchange segments of data. The connection between these two hosts is identified uniquely at each end by the host's network address and a 16 bit port number. The communication between the hosts is initiated by a three way handshake between them. This is where the sequence number is synchronized between the participants. The sequence number is a 32-bit number and it is the basis for a reliable data transport, working as a mechanism to assure the exchange of data through an unreliable network medium counting the bytes transmitted making possible be checked by the receiver and be acknowledged. This is, starting from the initial sequence number, each data byte sent as part of the connection has a corresponding sequence number; and only after having being acknowledged by the receiver is the data considered to be transmitted successfully.

TCP made use of the idea of pipe size and the knowledge there was reasonable but not excessive buffering along the data path to send a window of packets at a time. To control the amount of data that flows through the network path, the receiver sent the information of how much data can receive at once, so the network resource is utilized in a more efficiently way. This window of data means how many bytes the receiver can work with, given the available buffering and processing constraints. That is, window size represents how much data a device can handle from its peer at one time before it is passed to the application process. All the extra data in excess will be dropped. This window is also a constrain to the sender, because the sender is not allowed to transmits no more data than the window before waiting for the response of the host as acknowledgment of the sent data.

To manage the data sent and the acknowledgment received for those packets, TCP uses a cumulative scheme. After a packet is in flow from the server with its corresponding sequence number, sent data goes to a retransmission queue where it is held until the corresponding acknowledge from the other end has come in or, to be resent if not acknowledgment within a timeout. When the acknowledgment of a sequence number is received, the sender discard all data with sequence numbers bellow the sequence number in the acknowledgment that has arrive. For retransmission, TCP uses an adaptive scheme. The timeout is automatically set from the measured round trip time of the connection, taking into account the variance of the measured values[14]. This helps avoid retransmitting potentially lost segments too quickly or too slowly.

Because today's networks are dynamic and in different configurations, both topologically speaking as a bandwidth, TCP must handle these changes and still be able to maintain communication between the two hosts. The lost of communications can be an issue that TCP needs to take into account. In case of lost of one packet means subsequent packets cannot be acknowledged until the lost packet is retransmitted. This can lad to excessive retransmission and unnecessary load. TCP extension has been developed that allows the receiver to send selective acknowledgments of block of received data with sequence numbers that are not cumulative with the data acknowledged in the traditional way[18].

Since networks are shared and conditions change along the path, the algorithms continually probe the network and adapt the number of packets in flight. It is not hard also (but it is often the case) to find along the way decreases in the bandwidth. This spots are the bottlenecks and they are important because the performance or capacity of the entire connection (connection as a state between the two hosts) is limited by the resources that this trace has. With this, controlling the optimal rate of data transmission is a hard work, and the receiver window as communicated by TCP is not a necessarily a very good indicator. To fix this issue, TCP received an addition to its specification: *congestion window.* The congestion window plays a crucial role in estimating the available bandwidth between the hosts. After this modification, the minimum between the receiver window and the congestion window is used as the transmit limit. All the TCP additions attempt to keep the network operating near the inflection point where throughput is maximized, delay is minimized, and little loss occurs.

2.2 TCP's Phases

To provide a good service, it's needed that TCP flows respond to orders given by the hosts machines that controls the connection during congestion. This characteristic of flows is called "*responsiveness*" and tells when flows must back-off during congestion. In the second half of the 80's, TCP began to experience a strange phenomenon that manifested itself through a dramatically diminish throughput. After a deep analysis, Van Jacobson showed that TCP needed a mechanism to limit transmission speed in the face of congestion[14]. This lead to the development of a better congestion control mechanism for TCP, which was added as a requirement for hosts connected to Internet[3]. Since then, the congestion controls methods implemented in TCP has been updated several times, adding two new algorithms, the *slow start* and *congestion avoidance* were designed to keep in "equilibrium" the data that is in exchange. Later a third where added, the *fast retransmit and fast recovery.*

2.2.1 Slow start and Congestion avoidance

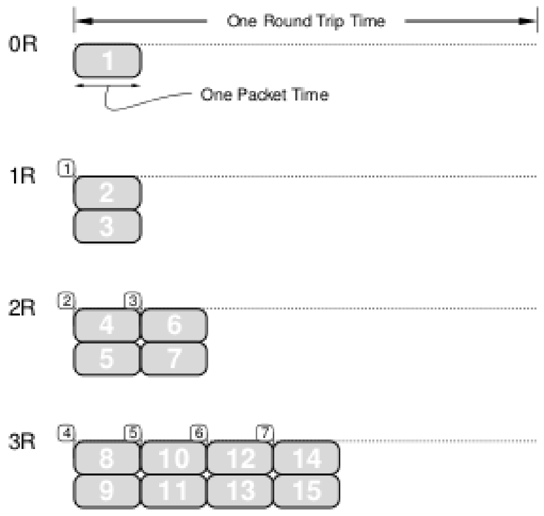


Figure 1: The Chronology of a Slow-Start.[14]

The idea behind bandwidth occupancy in TCP is to use as much as it is allow to use. With this in mind, slow start proves the network by increasing exponentially the rate of how many packets sends; the congestion window is increases by one packet and adds another for each packet acknowledged. This behavior continues until the slow start threshold (ssthrsh) is reached or congestion is signaled. If congestion is detected, the slow start threshold is reset to half of the amount of congestions window, at the time that congestion window is reset to a size of one segment, and slow start is restarted. In case that the slow start threshold is reached, the sender switches to congestion avoidance mode, which is employed to maintain the transmission rate, increasing by one segment each RTT[iii].

TCP must be smart enough to signal to the hosts when path is experiencing congestion, so it must have policies that decrease the network utilization or increase it if congestion signals are received or not. Also, based on the afore mentioned principle ("*packet conservation principle*"), congestion collapse would become the exception rather than a rule. Thus congestion control involves finding place that violate conservation and fixing them. It is important detect early this misbehavior, since new packets are added exponentially also the congestion grows exponentially, so if is detected early, only small adjustments will cure it.

Typically, when the sender is signaled that packets are been dropped, TCP/IP networks understand that as congestion, but typical effects include queuing delay, packet

[iii]Each ack increases the congestion window by $ss * ss/cwnd$. This results in a linear increase of the congestion window; that means in 1 RTT, congestion window will be increased by 1 [14] [22] [5]

loss or the blocking of new connections. New implementations define a explicit congestion notification methods [23], that allows notification of congestion between the ends without any packet loss. TCP specification mandates initially setting the congestion window to between 2 and 4 segments of data depending on the segment size[1].

2.2.2 Congestion control algorithms

One of the biggest problems that TCP had in the past years for its development has been the achieve of optimal utilization preventing congestion associated with the differences of bandwidth existing in the medium through which communication is propagated. That is why lately there have been various implementations for proper handling of the original algorithm originally deployed in TCP for attaining prevent or lessen the effects of this problem. Among the algorithms that currently exist, the two predominant in modern operating systems will be mentioned: CUBIC and Compound TCP.

2.2.2.1 CUBIC TCP is the default implementation for congestion control in the Linux kernel since 2.6.19. *Scaling the window growth rate to match large BDP*[iv] *is rather straightforward, tackling the fairness issues of new protocols has remained as a major challenge. Although BDP implies the network capacity by packet count(or window size), packet count is not adequate to characterize TCP performance because growth rate of TCP depends on RTT*[12].

Because the basis of their growth is a cubic function (hence its name), has a less aggressive behavior than their predecessors, while retaining excellent scalability, fairness and stability. As can be seen in figure 2, the cubic function is compound of three parts: a concave part where the window grows rapidly over small time values, a plateau set at the previous maximal congestion window size, and the end portion where the increasing bandwidth above the maximum which exhibits a convex shape.

As can be better seen in the formula 3, the congestion window ain't dependent of its previous acknowledged packets; instead, the congestion window is computed at each

[iv]View definition in appendix A

step from a calibrated function. With this calibration, the plateau is at the previous maximal congestion window size. This feature allow the algorithm to respond faster to changes in available bandwidth. Since in previous algorithms throughput is defined by the packet loss rate as well as RTT, the throughput in CUBIC is defined by only the packet loss rate.

Another feature that CUBIC has with respect to its predecessor in the Linux kernel, it also changes the behavior of the Slow Start algorithm, replacing it with

$$W_{cubic} = C(t-K)^3 + W_{max}$$

Figure 3: CUBIC's Congestion Window.[v]

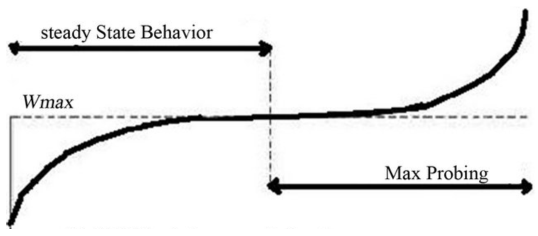


Figure 2: The window growth function of CUBIC

one called Hybrid Slow Start (or HyStart)[11]. The idea behind this algorithm is to prevent the extra drop of packets caused by the overshoot of bandwidth done by the regular slow start. This extra amount of data cause unnecessary on both ends hosts and network, and taking long time to recover from, that can be worst if the packet drop is made at a late stage of the slow start[vi].

2.2.2.2 Compound TCP is the modification to TCP stack algorithm promoted for Microsoft and implemented in their systems since Windows Vista and Server 2008. This algorithm aims to achieve a significant improvement in resource utilization by TCP, presenting an fair mix between delay-based approach and loss-based approach.

CTCP is based in the idea that if the link is under-utilized, the protocol should

[v] C is a scaling factor, textitt is the elapsed time from the last window reduction. W_{max} is the window size before the last window reduction. $K = sqrt[3]W_{max}beta/C$. And *beta* is a constant multiplication decrease factor applied for window reduction at the time of loss event

[vi] since slow start adds one extra packet for each acknowledged

incraese the sending rate in a agressibly manner so it can obtain available bandwidth more quickly. But when the link is nearly full utilization, keep that aggressiveness is unproductive because at this stage it is expected that the protocol is more accurate in achieving complete the remaining bandwidth. So CTCP incorporates a scalable delay-based component into the standard TCP congestion avoidance algorithm[32].

As can be seen in figure 4, the congestion window or cwnd which manages the loss-based component is untouched, and a delay window - dwnd - is added to manage the delay-based component. The awnd is the advertised window from the receiver. This algorithm has shown that in periods of under- utilization, it has an aggressive and scalable increase rule, and gradually it reduce the sending rate accordingly when the network is sensed to be fully utilized. But thanks to the packet loss component, it also reacts to packet loss reducing the sending rate to keep the conservative property of TCP. Also, Tan et al[31] has reveal that CTCP has a good performance over current high BDP networks too.

$$win = min(cwnd + dwnd, awnd)$$

Figure 4: Compound TCP sending window.[vii]

2.2.3 Fast retransmit and fast recovery

Timers are an important part of this self-regulated protocol. TCP maintains four timers[viii]. One of them, the retransmission timer, causes the sender to retransmit the segment over which at the expiry of this timer(which is a function of the estimated RTT), has not yet received their respective acknowledge, assuming the segment is lost in the network. When this action is unleashed, the server will interpret it as a sign of congestion, but data still flowing.

As already seen in **2.2.2.1**, slow start can do a lot of damage to our networks, but

[viii]The timers that TCP has for each connection are: (1)Persist Timer: Ensures that window size information is transmitted even if no data is transmitted. (2)Keepalive Timer: Detects crashes on the other end of the connection. (3)2MSL Timer: Measures the time that a connection has been in the TIME_WAIT state. (4)Retransmission Timer: The timer is started during a transmission. A timeout causes a retransmission

fast retransmit helps to recover from lost without hurting the network that much. It is triggered after a specified number of acknowledgments, usually is at the third, with the same acknowledge number are received by the sender. This gives makes the sender reasonably confident that the next higher sequence number was dropped and needs to be retransmitted; all this without the need for the action triggered by the timeout.

Fast recovery manages the continuous transmitting of new data on subsequent duplicate acknowledgments, until the retransmitted packet is received. and after a non-duplicated acknowledge is received, the congestion window is reset back to the value it had before the fast retransmit was initiated, and the fast recovery mode is ended, going back to congestion avoidance mode.

3 Router's Buffers effects

The main function of routers is to absorb bursts of traffic coming from the routers, and ensure that links are used to their maximum capacity. A network that does not have buffers, do not have the ability to store packets that are waiting to be transmitted, so any package that is over the capacity of the link will be dropped. In order to operate without buffers, arrivals must be constant and predictable; synchronization of the entire network would be necessary to avoid any loss. Such networks are complex in design, expensive to implement and particularly restrictive.

The throughput of each network is limited by the capacity of the slowest link causing a bottleneck; no matter how much more packets are introduced to the network, the transfer will not be faster than that determined by this link. That is why the major location of buffers in areas where bottlenecks occur is essential for proper functioning of the network, but this transfer between fast-to-slow networks is different on each route, even for the reverse path.

The presence of buffers is necessary to help to reduce data loss. Also because in the past the high cost of memory kept the buffers quite small, causing it to fill pretty quickly after the network became saturated, signaling to the communications protocol the presence of congestion and thus the need for compensating adjustments.

Unfortunately the network transport protocols, in order to operate at full capacity, require that the hosts are notified in a timely manner when they should back up and thus able to adapt its transmission rate to the available capacity. The absence of such timely notification triggers the presence of full buffers and increased communication latency.

3.1 Full Buffers

Packet networks require absorb bursty arrivals that TCP performs in order to communicate two hosts. When received, a packet is immediately validated, but not necessarily immediately processed and transferred out without kept in a buffer. That means that if the rate at which packets are received is greater than delay that takes to process and remove a package from the buffer, it tends to fill up[ix] and stay congested, contributing to excessive traffic delay and losing the ability to perfom their intended function of absorbing bursts.

This standing queue is the essence of Bufferbloat and is the result of the difference between the window and the bandwidth of the link, only creating long delays in communication, but no benefit in overall throughput. *It is not a phenomenon threated by queue or traffic theory, which, unfortunately, results in it being almost universally misclassified as congestion(a completely different and much rared pathology)*[24].

When the packet reach a bottleneck queue, is squeezed down in bandwidth but must strech out in time since its size stays constant.

3.2 High Latency

The latency a packet experiences in a network is made up of transmission delay(the time it takes to send it across communications links), processing delay(the time each network element spends handling the packet) and queuing delay(the time spent waiting to be processed or retransmitted). But large buffers only increases latency, and this only causes conflict with the needs of now days applications.

Once packets in-fly reach a bottleneck, they begin to spooling. Because the characteristics already explained, more and more packets are coming, and this queue continues to increase, which leads that each new arriving packet spends more time through the queue than the predecessor packet, which means an increase in the latency. Then, eventually, packets start to be dropped, notifying the hosts of the presence of congestion on the path.

As stated in [6], it is quite common to find these high- latency queues in the last mile. The causes can be many, but among the most common are both link quality at homes, as different implementations of traffic shaping methods implemented by ISPs,

[ix]When this occurs, the receiving device may need to adjust window size to prevent the buffer from being overloaded.

or massive buffers set by the latter to avoid loss of data, which can add a delay of up to several milliseconds. Bottlenecks at the Internet's edge can easily move between the wireless access(when its bandwidth is slow) and the provider's up-link, both of which can have highly variable bandwidths.

For networks that use coaxial cable, multiple clients concatenated their upward flows in a single transmission, resulting in a burst with a large volume of data, which can led to a high fluctuation in latency. This concatenation can also generate jitter time,which can be produce a miss interpretation for some protocols of incipient congestion and cause to enter into congestion control avoidance too early. The other type of highly used networks are the DSL, which the more distance is between the supplier reduces its transmission rate. That is why it is needed advanced signal processing and error correction algorithms which can lead to high packet propagation delays.

3.3 Sizing Router's Buffers

Unmanaged buffers are more critical today since buffers sizes are larger, delay-sensitive applications are more prevalent, and large downloads common but overly large, unmanaged, and uncoordinated buffers create excessive delays that frustrate and baffle end users.

Correct buffer sizing is not an easy problem. Undersizing - making buffers smaller than the traditional BDP- is fraught with problems. Today's links vary in bandwidth, and individual connections vary in RTT. This makes it impossible to properly pick a static size for most edge links. A simple, robust algorithm that can manage buffer delay regardless of buffer size and link bandwidth without a negative impact on utilization can make over-sized buffers irrelevant.

A widely used rule-of-thumb recommends the buffers of size $B = \overline{RTT}xC$, where $\overline{RTT}$ is the average round trip time experienced by connections utilizing the link, and C is the data rate of the link. Many buffers have been inserted without sufficient care due to the complexity to establish the appropriate size and test on real environments[38], helping to determine where further complicate occurs Bufferbloat.

While the rule-of-thumb comes from the idea of maintaining the medium as busy as possible, and to maximize the throughput of the medium. But due to the characteristics of TCP, no matter how large the buffer is, it will always be saturated.

The main condition to choose the proper size of a buffer is the ability to keep sending data in the periods in which the sender is stopped, preventing possible downtimes in order to maximize utilization of the link and kept high throughput at all times. The buffer will avoid to idle if the first packet from the sender shows up at the buffer just as it hits empty.

After a lost is detected, the cwnd is set to half of is last value, so if we denote as $(W_{max}/2)/C$ the amount of time that packets are sent in congestion phase, and as B/C the time that takes a buffer with size B to drain, the size of a buffer B needed is $B \leq (W_{max}/2)$ [2].

The problem occurs when the required size buffers are implemented, and the size is exceeded producing overbuffering. Overbuffering hurts anytime a link is saturated causing extra delay, and destroying many uses of moder Internet. As stated in [10], *a TCP connection must react quickly to changes in demand at the bottleneck link, but TCP reaction time is quadratic to the amount of overbuffering.*

Unfortunately this error comes from the fact that manufacturers began to think that adding more buffer to their products should bring positive consequences, especially considering the current volumes of data, in addition to providing more equitable access to available bandwidth handled.

4 Active Queue Management

Ensure communication only checking the endpoints has revealed that it is not enough and that it is also necessary to check the status of in-flight data transmission. With the passage of time and with increasingly high speed networks it has become more and more important to have mechanisms that keep throughput high but average queue size low. The changes implemented in TCP for congestion control have proven not to maximize the capacity of the medium along with a performance degradation. Among the most common problems are: connections still experience multiple packet loss, low link utilization and congestion collapse. It is important to keep in mind that when a packet is dropped before it reaches its destination, all of the resources it has consumed in transit are wasted.

This is when the role of routers extends beyond joining and interconnect local networks and the Internet together. The role of routers becomes important because they are able to better manage congestion that may occur in networks due the knowledge they has about other routers on the network and can choose the most efficient path for the data to follow, or as already seen, send message to the ends to fall back or even drop data. Also they must allocate the available bandwidth fairly between all flows.

Typically the queues that routers maintains are designed to smooth and accommodate the burst of data delivered by the hosts and transient congestion, but as the queues start to fill, the routers drops packets using a FIFO based drop-tail management. This discipline has two mayor drawbacks:

1. Lock Out: a small number of flows tends to monopolize the usage of the buffer capacity[4] .
2. Full Queue: these buffers tends to be always full, leading to experience high latency.

Also, as explained in [8], when a router use tail-drop, *the more bursty the traffic from a particular connection, the more likely it is that the gateway queue will overflow* *when packets from that connection arrive at the gateway/router.*

All this led to develop and implement more aggressive or active strategies to congestion control on the Internet called Active Queue Management or AQM for their acronym. AQM is a group of FIFO based queue management mechanisms to support end-to-end congestion control in the Internet. The goals behind AQM are to reduce the average queue length and with that decreasing the end-to-end delay. Also to reduce packet losses that reflect as a more efficient resource allocation.

AQM maintains the network in a region of low delay and high throughput by dropping packets before queues become full and can reliably distinguish between propagation delay and persistent queuing delay. Also, because the router can monitor the size of the queue over time, the gateway is the appropriate agent to detect incipient congestion. The most common AQM algorithms are: RED, SRED, BLUE, SFB, CoDel.

4.1 RED

Random Early Detection or RED is the algorithm introduced in [8]. RED monitors the average queue size and when it exceeds a preset threshold, the router drops the arriving packet with a certain probability, where the exact probability is a function of the average queue size. If ECN is active, the packets are choose randomly and marked. When a connection uses a larger share of available bandwidth, it is more likely to be marked. The idle periods (periods when the queue is empty) are also take into account for the calculations of the average queue size.

The objectives behind the implementation of RED are:

1. Minimize packet loss and queue delay.
2. Avoid global synchronization of sources by randomize its marking decision and by spacing out its marking.
3. Maintain high link utilization.
4. remove biases against bursty sources by limiting the queue occupancy so that

there is always room left in the queue to buffer transient.

Because RED can control the average queue size while accommodating transient congestion, implementing RED in routers can provide high throughput and low average delay in high-speed networks with TCP connections that have large windows. Also, in routers where RED and ECN are implemented the bursty traffic is well handled and the global synchronization between flows is avoided by decreasing their window at the same time.

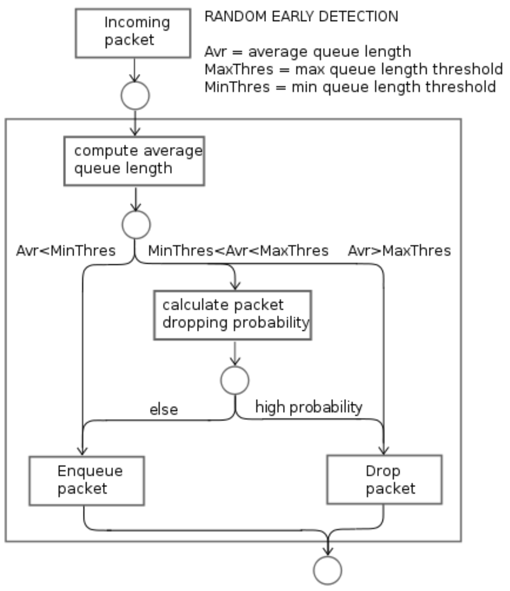

Incoming packet
RANDOM EARLY DETECTION
Avr = average queue length
MaxThres = max queue length threshold
MinThres = min queue length threshold
compute average queue length
Avr<MinThres
MinThres<Avr<MaxThres
Avr>MaxThres
calculate packet dropping probability
else
high probability
Enqueue packet
Drop packet

Figure 5: RED's Flowchart

The most common critics to RED is its complexity to properly configure its parameters. It is because of this that after their official publication, it was necessary to perform the second publication [15], which aims to explain a few more details on how to configure it properly. Since RED rely on queue length as index of congestion, it can reflect the presence of congestion but not the severity nor the numbers of competing connections sharing the link. Since long ago networks ceased have a static configuration, RED require a wide range of parameters to operate correctly under different congestion scenarios. Unfortunately, this failure leads to deterministically mark packets and suffer from poor link utilization. *While ECN timeouts allows RED to avoid packet loss, the deterministic marking eventually causes underutilization at the bottleneck link*[7].

As stated in [20], *RED was simple and can be effective at reducing persistent queues, little guidance was available to set its configuration parameters and it functioned poorly in a number of cases. This led to a general reluctance to use it.*

Also to archive the ideal operating point, it can only do so when it has a sufficient amount of buffer space, which leads to default induce an extra latency.

4.2 BLUE

Introduced by Wu-chang Feng et al in [7], BLUE aims to manage congestion by using packet loss and link utilization history. The concept behind BLUE development is to avoid drawbacks of RED like parameter tunning problems or to determine the fluctuations of actual queue length. The key idea is to perform queue management based directly on packet loss and link utilization rather than on the instantaneous or average queue lengths.

```
Upon packet loss
    if (now - last_update >freeze_t)
        Pm = pm + d1
        last_update = now
Upon link idle
    if (now - last_update >freeze_t)$
        Pm = pm - d2
        last_update = now
```

Figure 6: BLUE algorithm

Through the use of a single probabilistic indicator called p_m, it simplifies the process of dropping or marking packets when they are enqueued. Against a continuous dropping of packets due to overflow, the probability increases, increasing the rate at which it sends back congestion notification. Also, to control how this parameter changes over time, the *freeze_time* is used to determine the minimum time interval between updates. The value on which to increase or decrease, if the link is idle, is determined by $\delta_1 and \delta_2$.

The effects of BLUE, as proved in [7], are the reduction of the buffer size, which also reduces the end-to-end delay and helps to improve the effectiveness of the congestion control algorithm. BLUE was also shown that it has a better performance when marking incoming packets, leading to congestion notifications causes periods of sustained packet loss nor periods of continual underutilization.

4.3 CoDel

After several years using based on the average queue size, Nichols and Van Jacobson algorithms realized that this approach although they have information of congestion, no hard data on how serious is this. Jacobson et al, determined that it may submit two types of queues: one that smooths bursts of arriving packets turning it into a stable and continuous sent and tends to last less than one RTT (called good queues); and those that only tend to create excessive delays and lasting several RTTs (bad call queues). With this they were able to determine that the core of bufferbloat detection problem is separating good from bad queues. Bearing in mind the above is that the algorithm is controlled delay or CoDel by its acronym were developed.

Presented in [20], CoDel is based on the idea that it is sufficient to maintain a threshold above which the tail can not exceed rather than keeping a window of values to compute the minimum. Rather than measuring queue size in bytes or packets, it is used the packet-sojourn time through the queue. The time each packet spends in the queue is independent of the transmission rate, the sojourn is more valuable information about the behavior of the buffer, in addition to being much more related to the effects you may experience. The sojourn is based on a timestamp corresponding to when the packet arrive to the queue, at which this mark is added to the package information. The minimum packet sojourn can be decreased only when a packet is dequeued.

Another difference that occurs is that CoDel accepts the existence of queues, but it is unacceptable to drop packets when you have a buffer with less than MTU's bytes.

$$f(n) = \frac{100}{\sqrt{n}} * f(n-1)$$

$$\text{f}(0) = 1$$

Figure 7: CoDel droptime interval, n=iteration

But when the queue delay has exceeded target for at least interval, a packet is dropped and a control law sets the next drop time accordingly to 7. The ways in which CoDel stops packet drop is either when the delay queue is less than the target value or when the buffer contains less than MTU's bytes. Furthermore the algorithm has additional logic that controls reenter to dropping state too soon.

Beside the minimum packet sojourn, the other two parameters that CoDel needs are the target and interval, which are almost self-explanatory. Target is the acceptable standing queue delay and interval works as the time on the order of a worst-case RTT of connections through the bottleneck. The recommended target value is 5ms since tests reveal that with this value, the utilization of the links is the optimal. Interval is loosely related to RTT since it is chosen to give endpoints time to react without being so long that response time suffers. A setting of 100ms works well. *CoDel's efficient implementation and lack of configuration are unique features that make it suitable for managing modern packet buffers*[20].

5 Hidden Flaws and Bufferbloat

Because much of the basis for how TCP was designed in the early days of the Internet, where speeds were low, the burden on the networks were only a few kbps, and the networks interconnecting were separated by a couple milliseconds, the use was ideal. But it is not hard to see that in the past few decades, Internet growth has exceeded the expectations of even the most experienced analysts and dreamers.

Intensive-consumption applications like on-line games, audio and video streaming for music or on-line calls, torrent or P2P applications and other sites that have a high level of resource utilization like Youtube, Facebook or Netflix, have made networks of today are quite different from those that formerly existed, but still running the same protocols[x]. If that was not enough, the creation and incorporation of new devices such as smart phones, tablet and even smart-watch with the ability to transmit data over the Internet, the more necessary efficient and properly handling the wealth of information is becoming

TCP relies on timely notification to adjust its transmission rate to the available bandwidth, which is commonly signaled by the packet loss. But since couple of years, and helped by lower prices on hardware manufacturing along with the thought that more is always better, the manufacturers had decide to prevent as much as possible the loss with the addition of larger buffers on their devices. This simple action is slowly bringing a new collapse not only TCP but data transmission in general, as can be seen in [24]

With the increase in the size of the buffers, the packages spent more time “on the fly” in this big buffers instead of dropped, signaling to TCP to reduce the sending rate. When this large amount of data is dropped, causes TCP sharply drop in transmission rate, freeing up bandwidth. Unfortunately, due to the size of the buffers is static, when the new TCP slow-start phase start, again due to the lack of signals out of time, TCP will work with diminish performance. This problem is cyclical, resulting in exponential

[x]While many have been updated, the basics remain the same

TCP back-off, throughput degradation and very high latency

But the path on Internet are shared by multiple TCP streams so the buffers, and this back-off behavior has a tendency to synchronize flows [2]. This cause all the flows to throttle back their transmission rates simultaneously, amplifying the effect. This decrease in latency and reduced throughput are the effects of Bufferbloat phenomenon.

As stated in [10], *long delay from Bufferbloat are frequently misattributed to insufficient bandwidth and this misinterpretation of the problem leads to the wrong solutions being proposed.* As seen in the past sections, the packet loss for TCP is not in itself a problem, conversely it is essential for its functioning. The real problem is the excessive and consecutive packet losses that occurs when the buffers kept persistently fulls. This effects may cause that modern applications timeout their connections or in others users may experience some delay. This is the real effects and problems of Bufferbloat.

6 Experimental Work

The goal of the experiments outlined in this section is to examine different residential and public networks with the objective to proof the existence of the phenomenon under an uncontrolled scenario. This will be done by running a set of tests with different tools, first to define and characterize the network, then to measure and compare how the network behave with and without load. More specifically, the factor to be tested is the latency under load and analyzed to identify if the latency that occurs is due to an excess of buffers or due to some other problem.

This section aims to explain the setup that will be used to perform the tests, the selected tools for each of these tests and what is to be achieved and expected from each one of these tests.

6.1 The Test Setup

The tests will be ran under a pseudo controlled environment, using one physical machine and a second virtual machine hosted in first machine. These systems runs under a regular OS without any modification beside the ones that the own OS as by default. Also, for some tests two other devices will be added, one acting as an Iperf Server and a regular Android Tablet that will be used to add some extra load to the network when the Ethernet cable is used as medium.

All of these tests will be carried out in a real-world scenario, where no packet prioritization is done by the server against our flows, the routes can vary between each iteration of the same test, and many different flows will collide with other flows from different sizes and types. Nor there is more information about how the flows are treated by the queue manager algorithms or about how they are configured.

6.1.1 Hardware Characterization

Physical Machine :

The physical machine runs as host OS, Windows 7 SP1, that works with an Intel(R) Core(TM) i7-2670QM CPU 2.20GHz with 8GB of available RAM. This machine will always be connected through its wireless adapter, a *Broadcom Corp. BCM4313 802.11b/g/n Wireless LAN Controller (rev 01)*. The Ethernet controller is a *Realtek Semiconductor Co., Ltd. RTL8111/8168 PCI Express Gigabit Ethernet controller (rev 06)* adapter, and will be bridged to the virtual machine for some tests.

Virtual Machine :

The virtual machine is hosted using VMWare Player 6.0.1, with 4 processors assigned for use plus 4GB of RAM, and with the Ethernet adapter connected only for certain tests. The OS selected is a Debian based OS called Kali Linux, and using the kernel release identified as Debian 3.12.6-2kali.

The wireless adapter is a AIR-802 USB adapter with Zydas chipset and a TP-link 8dbi antenna. This adapter is directly connected to the virtual machine and hooked to the physical machine without the USB extension, this way any extra signal loss is avoided.

Iperf Server :

This machine will be used as a server for the Iperf test. This is a VPS hosted by Digital Ocean [xi] with 512MB Ram, 20GB SSD Disk, and located in New York data center. This machine runs Ubuntu 12.04.3 x64 under KVM software using as a processor an Intel Hex-Core 3 GHz.

The idea of using an external device to connect the virtual machine to the network and not using a bridged configuration provided by software, is mainly because with an USB device the machine will take care of all the management and administration of the device, avoiding any possibility that the host machine modify or manages any flow. Also, by using a second machine to overload the uplink, causes to avoid overflow the queue on the machine that is performing the tests. With this, it is expected to minimize

[xi] `https://digitalocean.com`

the possibility that our testing machine is causing extra latency, either by the saturation of the wireless channel, or by the queue in the traffic control subsystem into the kernel.

For the tests, the Bufferbloat community[35] has created a set of best practices[28] to follow so the results are consistent and repeatable, but in this case, computers and routers will not be modified as it will attempt to analyze what an every-day-user experience. Those practices will be taken in consideration if it is the QOS present in routers and deactivate it.

6.2 Tools Definition

The tools used for the benchmark were selected by the capability to determinate the presence of the Bufferbloat phenomenon in the network of study by the analysis of its indicator, the latency or the round trip time[xii]. So, with this as a main consideration, the tools selected for the benchmark will be chosen by the complexity and accuracy to measure and determine the RTT into an IP/TCP connection, and how hard is to consistently replicate the results under similar contexts. Other tools were selected to help determine, characterize, and proof the capability of the network to cause Bufferbloat.

[xii]To SUBTEL, organization responsible for control and supervision in the performance of telecommunications in Chile, RTT and Latency correspond to the same thing [25]

The tools selected to be used in this tests are the following:

- Speedtest test by Ookla
- Netalyzr by ICSI
- Iperf Tool with Tcptrace/Xplot.org
- Page Benchmarker extension for Google Chrome
- Smokeping Latency Tool

6.2.1 Speedtest

Developed by Ookla[xiii], this tool is used for most of the ISPs and many users in Chile to test their broadband's connections globally. It can be used not only in their website *www.speedtest.net* but also in Android, iOS or Windows Phone. A command line interface developed in Python can be used too for testing Internet bandwidth.

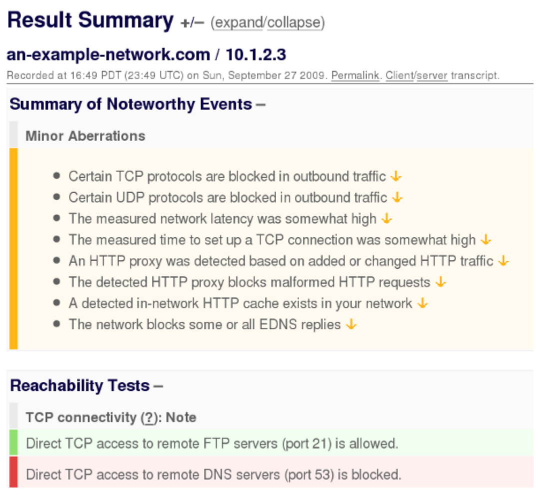
The server that will be selected to perform every test, either has or not the fastest ping, is the one hosted by the Pontificia Universidad Católica de Valparaíso (this host is the default selected most of times by the site also).

6.2.2 Netalyzer

For end-users, little is revealed about how ISPs manage their networks. That is why since 2009 ICSI has developed Netalyzr, a "two click " tool developed to test networks that runs in web browser as a Java applet. Once downloaded, the applet contacts the back-end server using a range of protocols and mechanisms employed as part of the testing, and then conducts a series of tests. Once completed, it uploads its findings to the back-end where they are distilled into a detailed report breaking down the findings into correctly operating aspects, those that show potential signs of trouble, and those that are downright broken. Figure 8 is an example of the output.

[xiii] `https://www.ookla.com/about`

The primary goal in developing Netalyzr's tests was to provide a new kind of diagnostic tool, *one that particularly illuminates under what sort of restrictions a user's Internet connection operates, like both forms of filtering (blocking) and proxying imposed by the user's ISP, and performance issues that arise from the nature of the user's Internet access setup*[17]. Among the performance considerations, Netalyzr's measures are packet loss, latency, bandwidth. Also, it compute in-path forwarding device buffer size by comparing small-packet latencies under idle and loaded states in networks (the perfect time to occur Bufferbloat). Other tests like general TCP and UDP service reachability.


Result Summary +/– (expand/collapse)

an-example-network.com / 10.1.2.3

Recorded at 16:49 PDT (23:49 UTC) on Sun, September 27 2009. Permalink. Client/server transcript.

Summary of Noteworthy Events –

Minor Aberrations

- Certain TCP protocols are blocked in outbound traffic
- Certain UDP protocols are blocked in outbound traffic
- The measured network latency was somewhat high
- The measured time to set up a TCP connection was somewhat high
- An HTTP proxy was detected based on added or changed HTTP traffic
- The detected HTTP proxy blocks malformed HTTP requests
- A detected in-network HTTP cache exists in your network
- The network blocks some or all EDNS replies

Reachability Tests –

TCP connectivity (?): Note

Direct TCP access to remote FTP servers (port 21) is allowed.

Direct TCP access to remote DNS servers (port 53) is blocked.

Figure 8: Result Summary example from Netalyzr. Source: `www.dslreports.com`

6.2.3 Iperf

Iperf is a well known and commonly used network testing tool. It can create a TCP and UDP data streams and measure the bandwidth and the quality of a network link. It can perform multiple tests like Latency, Jitter or Datagram Loss.

Iperf basically tries to send as much information down a connection as quickly as possible reporting on the throughput achieved. This tool is especially useful in determining the volume of data that links between two machines can supply. This two machines define the network, one acting like a server and the second as the client. For this scenario, the server will be the VPS that only will receive the Iperf connections (also will be running ssh but without further interaction). The VM Linux machine will work as an Iperf Client.

As mentioned in Iperf users mailing list " *When one runs TCP tests, there are 2* *things that block Iperf from having clear view of real throughput: buffering on sender's side (TCP/IP stack) and TCP behavior itself (acking). What Iperf can measure is the pace with which it sends data to TCP/IP stack; TCP/IP stack will only accept data from application when buffers are not full. If the buffer is huge, Iperf will see high throughput initially, then it will drop. If there's congestion or retrasmission going on, Iperf will see it as lower throughput*"[16], but the data generated by Iperf won't be further analyzed because the idea behind using this tool is a TCP's packet generator. This means that the packets generated by Iperf will captured and analyzed with tcpdump, tcptrace and xplot.org.

6.2.4 Page Benchmarker

Page Benchmarker is a Google Chrome extension that intent to test page load time performance within Chrome. Measures time-to-first-paint, overall page load time KB read/written, and several other metrics, and with its capability to clear the cache and existing connections between each page load, makes this tool one of the main sources of income to analysis.

6.2.5 Smokeping

SmokePing is a latency logging and graphing tool that consists of a running daemon which organizes the latency measurements and a CGI which presents the graphs. SmokePing give us the ability to measure latency and packet loss in the current network, and with RRDtool, is capable to maintain a long term data store and to draw different graphs with the giving up-to-the-minute information on the state of each network connection.

Smokeping can be configured to perform a wide range of latency measurement probes each one directed to an independent target or over a set of targets selected for each proof.

6.3 Test Description

To test the existence of the Bufferbloat phenomenon, five tests are conducted as described below. Each test will be repeated under the following contexts:

1. Twice on the same day in one network to determine if does exists a considerable variance in latency for different times in a day.
2. Select different public and private networks with different *"speeds"*.
3. Use the Ethernet cable in order to compare the results with those previously obtained using Wireless.

Because it is intended to prove the existence of the phenomenon under circumstances experienced by everyday users, no kernel parameters will be amended nor changed, and only the ability to perform QOS on routers that have this feature is disabled. Furthermore, the overall question these tests seek to answer is the following:

Theorem. *"The networks that we use every day, have the necessary characteristics to generate the Bufferbloat phenomenon whether under low loads and if does exists, the how serious are the effects ?"*

6.3.1 Speed test

The idea under this test, is to set a baseline by comparing the speed offered by the ISP and the one at the moment of testing. To find the speed provided, the tool used is the Speedtest in the web site of Ookla.

The benefits of using this web sites is that not only it will reveal not only the available the national uplink and downlink, also it will set the baseline for the ping. The expected pings, independently of the connection bandwidth, should be around $\sim 12ms$, based on the data presented in [37] and [34] by the two most used ISP in Chile[xiv].

As data, it is estimated that the ratio of the average speed as compared to the rated speed of the service offered for domestic bonds has a variation of 85%, while for

[xiv] Telefónica has a 39.2% market share while VTR owns 38.8% respectively. [27]

international links decreases drastically to 35%[26]. Due this factor, none of the tests will be performed on servers located outside our country, as it is believed that the data will not very representative regarding actual service.

6.3.2 Signs of trouble

After characterize our network based on the speed, it is needed to try to define the state of the service based on the quality in which our network operates, in example: what kind of traffic is passing through our network, and give also get a little more detail on the average rates of delay and buffers with which the traffic can find along the path. To collect all this information will be used Netalyzr.

While laws present in Chile assure consumers networks free of traffic shaping barriers and filters, ISPs apply traffic management measures[36][33] to deliver an optimum experience of the service's use and thus, achieve an efficient use of the network, thus further to protect the safety of users while maintaining network stability.

This is why it is expected that some ports or services are partially or completely blocked, but the most important is that this test will first light on the existence of the phenomenon under study.

6.3.3 Collapse test

Already having a clear idea of the stat of the studied network, the next will be try to check empirically that the results obtained by the previous test above described are valid. For this, Iperf will be used to generate as much traffic as possible for 5 minutes between servers (both national and international). But the main goal is not to measure the throughput of the network; instead to capture the packets that Iperf produces and study them. So before run Iperf, tcpdump will capture this information. Subsequently, the tcp's trace is taken and extracted with tcptrace tool and generate the RTT graph.

This exercise will be conducted three times as part of the test, running the first time as the sole source of network load. For the second and third time, after 50 seconds after Iperf was started, from the Windows machine will be performed an upload to a Dropbox account with a file big enough to the upload take more than the time defined to this test. This upload intends to saturate the upstream link, leading to an overload of existing buffer (indistinct whether the server is national or not, there are several levels where the route is common). The upload will be stopped 50 seconds before the time limit.

The time gap established before loading and the end of the test is to let Iperf Iperf frames reach a steady state flow and thus to generate a basis on which to analyze the time when the traffic is maximum.

The expected RTT are around $\sim 12ms$ and $\sim 100ms$ without load, for national and international servers respectively. About the expected value under load, it is hard to estimated, further that their expected behavior is to be stable round a certain value. In case of no stable behavior, the possible causes will be analyzed and tracked.

6.3.4 Load benchmark test

With the effects of excess buffers already determined, this tests seeks to show how this phenomenon affects users who surfs through a web browser. This is why the extension of Google Chrome Page Benchmarker comes in and it will be used to do 5 iterations of 10 loads to a website and analyze the behavior of these.

As in the previous experiment, the first will be without any load on the network. Then, again a file will be uploaded to Dropbox from the second machine and after 30 seconds from the start of the load, and with the load active, will proceed to a second iteration. The third will begin the after a minute, canceling the file upload and after waiting 30 seconds after cancel the upload, the benchmark will be measured again. The fourth and fifth will be a minute and a half since the previous iteration have finished. The site chosen is `http://www.usm.cl`.

As like Jim Gettys showed in his video demonstration[9], the benchmark increases between 10 to 15 times with load against the original times. For this test, the expected proportions are lower, this mainly because the kernel machine that will run the tests has implemented already some improvements to mitigate this problem.

6.3.5 Smoke the path

To verify that the effect of Bufferbloat, if it is the case, does not just happen on a single server or with a single TCP flow, the Smokeping tool will be configured to perform two types of tests: Fping and echopinghttp. These tests will be directed to different types of servers (ie: physical and VPS machines) which have different connection settings and/or type of services (some servers has dedicated link), and located in both Chile and the United States.

The tool will be left couple of minutes to track and save the state of the connection under no load(except the required for testing). After that, a upload of a big file will be performed from a second machine until it finish or the behavior has become stable and then re-enter to a phase without load for couple of more minutes and repeat the upload.

With this test, in addition to verifying the validity of the previously captured data is expected to determine the presence of other factors that can contribute to the degradation of the quality of service on the network, whether factors such as quality of service provided by the host, channel/path problems, problems related to the way handling packets by the router or the machine, and/or any other that may arise. ia

7 Results

Here goes a description of what will be shown in this section

7.1 Speed test

Here goes the information related to test speed getting the differences and results gathered by this tests.

Will be a table with the current speed and pings. Also will be difference table with the speed that must has.

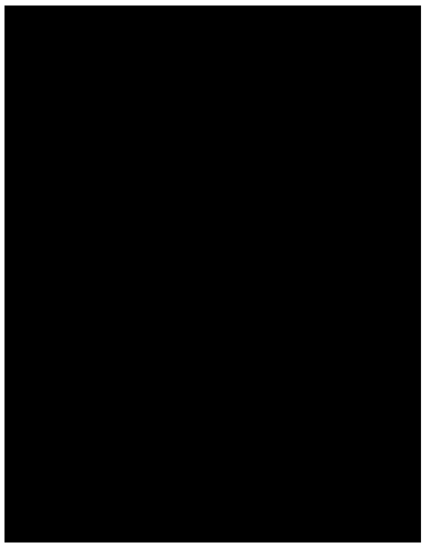


Figure 9: Speed and Pings based on Appendix B, Table 4

7.2 Netalyzr test

Here will be a resume of what netalyrz give us

7.3 Iperf test

Graphs and what happend with the rtts

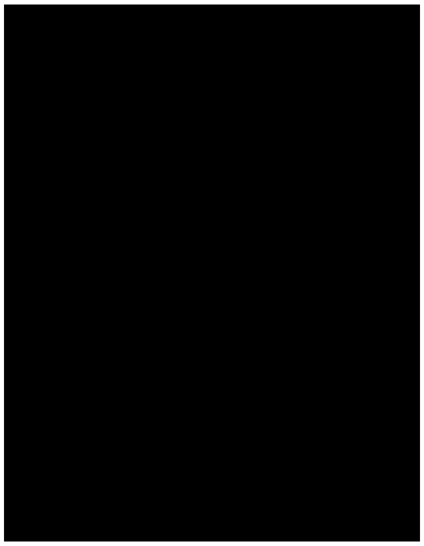


Figure 10: Variation ratio of offered vs measured speed based on Appendix B, Table 5

7.4 Benchmark test

Page benchmark results with tables with differences and mean results

Table 1: Total load mean times.

Lugar	1	2	3	4	5
4low	3415,1	20932,3	4490,5	4170,1	4087,1
casa	1654,8	3053,6	1718	1877,4	1666,8
mbahamon	2248,4	23868,5	2248	2182,6	2223,8
polmos	1623,4	3701,4	2420,6	1947	1659,3
polmos2	1912,4	4861,8	1738,5	2136,1	1827,5
nalucem	6421,4	26048,6	6280,8	6335,8	6306,4
casa2	1682,9	2354,4	1682,3	1667,5	1646,6
nalucem2	8199	32151,5	6479,1	6537,1	6349,4

7.5 Smokeping Test

Graph results

Table 2: Variation ratio over own iteration.

Lugar	1	2	3	4	5
4low	55,55	607,623	96,647	132,874	3202,323
casa	8,353	199,264	16,087	256,497	18,852
mbahamon	63,959	83,863	53,778	39,558	219,521
polmos	6,365	163,106	3802,472	8343,236	4050,064
polmos2	59,754	1891,192	8106,53	317,45	8126,291
nalucem	161,954	154,734	9,858	6,96	5,233
casa2	13,998	181,652	27,18	3,102	17,715
nalucem2	880,662	215,955	154,129	184,86	10,703

Table 3: Variation ratio over all iterations.

Lugar	min	max	%
4low	3415,1	20932,3	612,933
casa	1654,8	3053,6	184,529
mbahamon	2182,6	23868,5	1093,581
polmos	1623,4	3701,4	228,002
polmos2	1738,5	4861,8	279,654
nalucem	6280,8	26048,6	414,733
casa2	1646,6	2354,4	142,985
nalucem2	6349,4	32151,5	506,37

7.6 Summarising Results

all test

8 Conclusions and Further Work

conclusions

http://lalith.in/2012/02/15/fun-with-tcp-cubic/ Examples of iperf and wireless

[19]

further

analysis in specific of how the three different flows (Elephant, mice and ant)[11] [4] reacts to the bufferbloat

How the linux kernel now reacts to bufferbloat. [13]

The effects under laying the use of CTCP[31][32] in Windows and the recommendations of the Bufferbloat community[29]

ver notas en [19]

cerowrt project [30]

Also from [2] and [38], we can see that the rule-of-thumb doesn't longer apply to backbone routers, and a better estimator of the size of a buffer with n flows would be no more than $B = (\overline{RTT}xC)/\sqrt{n}$. With the assumption that short-flows plays a very small effect, and that the buffer size is dictated by the number of long flows, this factor will be proof that routers are much longer than they need to be, possible by two order of magnitude.

A Definitions

Bandwidth Delay Product (BDP) :

An amount of data measured in bits. It is equivalent to the maximum amount of data on the network circuit at any given time. Commonly measured by the RTT*bandwidth.

Bottleneck bandwidth :

The smallest bandwidth along the path. Packets cannot arrive at the destination any faster than the time it takes to send a packet at the bottleneck rate.

Bufferbloat :

Refers to excess buffering inside a network, resulting in high latency and reduced throughput.

Congestion Window (cwin) :

One of the factors that determines the number of bytes that can be outstanding at any time. It prevents the overload of the link with too much traffic. The size is calculated by estimating how much congestion there is between the two places.

Long Fat Network (LFN) :

A network with a large bandwidth- delay product. Often $>> 10^5$ bits (12500 bits).

Maximum Segment Size (mss) :

Largest amount of data (in octets) that a communication device can receive in a single TCP segment (single IP datagram). Established by pass on the syn packet.

Maximum Transmission Unit (MTU) :

MTU of a layer is the largest protocol data unit that the layer can pass onwards. In ethernet the MTU is 1500 bytes.

Slow start threshold (ssthrsh) :

It determine whether the TCP should do Slow Start or Congestion Avoidance. It is initialized to a large value, and after a congestion is signaled, cwin is divided in half and ssthrsh is set to cwin.

Soujourn :

A period of time when you stay in place as a traveler or guest. Example: Sojourned for a month at the mountains.

System Throughput :

Corresponds to the fastest rate at which the count of packets transmitted to the destination by the network is equal to the number of packets sent into the network.

Throughput :

Measure of the efficiency of a network expressed as the data(bits or packets) transfer rate of useful and non-redundant information. It depends on factors such as bandwidth, line congestion, error correction, etc.

B Speed Test results

Table 4: Speeds and Pings measured.

Location	Medium	Downlink (Mbps)	Uplink (Mbps)	Ping (ms)
4low	Wireless	0,6	8,03	29
mbahamon	Wireless	0,6	10,21	24
mhbrisas	Wireless	1,19	17,96	19
mgallard	Wireless	x	x	x
garriag	Wireless	x	x	x
casa	Wireless	4,95	15,48	18
casa2	Wireless	5,09	13,46	20
polmos	Wireless	2,32	19,61	26
polmos2	Wireless	2,34	18,72	22
nalucem	Wireless	0,57	3,97	35
nalucem2	Wireless	0,56	4,06	36
mcatala	Wireless	x	x	x
mcatala2	Wireless	x	x	x

Table 5: Variation ratio of offered vs measured speed.

Location	Uplink ratio	Downlink ratio
4low	120,00	80,30
mbahamon	120,00	102,10
mhbrisas	119,00	89,80
mgallard	0,00	0,00
garriag	0,00	0,00
casa	99,00	103,20
casa2	101,80	89,73
polmos	116,00	49,03
polmos2	117,00	46,80
nalucem	114,00	99,25
nalucem2	112,00	101,50

References

[1] Allman, M., Floyd, S., and Partridge, C. Increasing TCP's Initial Window. RFC 3390 (Proposed Standard), Oct. 2002.

[2] Appenzeller, G., Keslassy, I., and McKeown, N. Sizing router buffers. *SIGCOMM Comput. Commun. Rev. 34*, 4 (Aug. 2004), 281–292.

[3] Braden, R. Requirements for Internet Hosts - Communication Layers. RFC 1122 (INTERNET STANDARD), Oct. 1989. Updated by RFCs 1349, 4379, 5884, 6093, 6298, 6633, 6864.

[4] Burguers, B. Evolvements of tcp-, udp-, short-lived tcp- and long-lived tcp-flows. *6th Twente Student Conference on IT* (2006).

[5] Deering, S., and Hinden, R. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460 (Draft Standard), Dec. 1998. Updated by RFCs 5095, 5722, 5871, 6437.

[6] Dischinger, M., Haeberlen, A., Gummadi, K. P., and Saroiu, S. Characterizing residential broadband networks. In *Proceedings of the 7th ACM SIGCOMM*

conference on Internet measurement (New York, NY, USA, 2007), IMC '07, ACM, pp. 43–56.

[7] Feng, W.-c., Shin, K. G., Kandlur, D. D., and Saha, D. The blue active queue management algorithms. *IEEE/ACM Trans. Netw. 10*, 4 (Aug. 2002), 513–528.

[8] Floyd, S., and Jacobson, V. Random early detection gateways for congestion avoidance. *IEEE/ACM Trans. Netw. 1*, 4 (Aug. 1993), 397–413.

[9] Gettys, J. Bufferbloat: "dark" buffers in the internet - demonstrations only. `https://www.youtube.com/watch?v=npiG7EBzHOU`, January 2012.

[10] Gettys, J., and Nichols, K. Bufferbloat: Dark buffers in the internet. *Commun. ACM 55*, 1 (Jan. 2012), 57–65.

[11] Ha, S., and Rhee, I. Taming the elephants: New tcp slow start. *Comput. Netw. 55*, 9 (June 2011), 2092–2110.

[12] Ha, S., Rhee, I., and Xu, L. Cubic: A new tcp-friendly high-speed tcp variant. *SIGOPS Oper. Syst. Rev. 42*, 5 (July 2008), 64–74.

[13] Hoiland-Jorgensen, T. Battiling bufferbloat: An experimental comparision of four approaches to queue management in linux.

[14] Jacobson, V. Congestion avoidance and control. *SIGCOMM Comput. Commun. Rev. 18*, 4 (Aug. 1988), 314–329.

[15] Jacobson, V. Notes on using red for queue management and congestion avoidance. talk at NANOG 13, June 1998.

[16] Kozelj, M. Re: [iperf-users] how iperf works? `https://www.mail-archive.com/iperf-users\spacefactor\@mlists.sourceforge.net/msg00147.html`, nov 2009.

[17] Kreibich, C., Weaver, N., Nechaev, B., and Paxson, V. Netalyzr: Illuminating the edge network. In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement* (New York, NY, USA, 2010), IMC '10, ACM, pp. 246–259.

[18] Mathis, M., Mahdavi, J., Floyd, S., and Romanow, A. TCP Selective Acknowledgment Options. RFC 2018 (Proposed Standard), Oct. 1996.

[19] Mathis, M., Semke, J., Mahdavi, J., and Ott, T. The macroscopic behavior of the tcp congestion avoidance algorithm. *SIGCOMM Comput. Commun. Rev. 27*, 3 (July 1997), 67–82.

[20] Nichols, K., and Jacobson, V. Controlling queue delay. *Queue 10*, 5 (May 2012), 20:20–20:34.

[21] Postel, J. Transmission Control Protocol. RFC 793 (Standard), Sept. 1981. Updated by RFCs 1122, 3168, 6093, 6528.

[22] Postel, J. TCP maximum segment size and related topics. RFC 879, Nov. 1983.

[23] Ramakrishnan, K., Floyd, S., and Black, D. The Addition of Explicit Congestion Notification (ECN) to IP. RFC 3168 (Proposed Standard), Sept. 2001. Updated by RFCs 4301, 6040.

[24] Staff, C. Bufferbloat: what's wrong with the internet? *Commun. ACM 55*, 2 (Feb. 2012), 40–47.

[25] SUBTEL. Ref. fija indicadores de calidad de los enlaces de conexión para cursar el tráfico nacional de internet y sistema de publicidad de los mismos. `http://www.subtel.gob.cl/images/stories/articles/subtel/asocfile/res_698_trafico_internet.PDF`, June 2000.

[26] SUBTEL. Modelo de competencia por calidad de servicio. `http://www.subtel.gob.cl/images/stories/apoyo_articulos/notas_prensa/modelo_competencia_qos_subtel_23enero2012.pdf`, January 2013.

[27] SUBTEL. Información estadística, serie conexiones internet fija. `http://www.subtel.gob.cl/images/stories/apoyo_articulos/informacion_estadistica/series_estadisticas/06032014/1_SERIES_CONEXIONES_INTERNET_FIJA_DIC13_050214.xlsx`, March 2014.

[28] Taht, D., and Gettys, J. Best practices for benchmarking codel and fq codel. `https://www.bufferbloat.net/projects/codel/wiki/Best_practices_for_benchmarking_Codel_and_FQ_Codel`, March 2013. Wiki page on bufferbloat.net web site.

[29] Taht, D., and *BufferBloat Community*. Bloat: Windows tips. `http://www.bufferbloat.net/projects/bloat/wiki/Windows_Tips`, February 2011.

[30] Taht, D., and *BufferBloat Community*. Cerowrt project. `http://www.bufferbloat.net/projects/cerowrt`, 2014.

[31] Tan, K., and Song, J. Compound tcp: A scalable and tcp-friendly congestion control for high-speed networks. In *in 4th International workshop on Protocols for Fast Long-Distance Networks (PFLDNet), 2006* (2006).

[32] Tan, K., Song, J., Zhang, Q., and Sridharan, M. A compound tcp approach for high-speed and long distance networks. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings* (April 2006), pp. 1–12.

[33] Telefónica Chile, S. Medidas o acción para la gestión del tráfico y administración de red servicio banda ancha fijo. `http://www.movistar.cl/PortalMovistarWeb/ShowDoc/WLP+Repository/Portlets/P030_Generico/Recursivo/acordeones/acordeon_banda_ancha/pdf/RED_servicio_BandaAnchaFijo.pdf`, 2013.

[34] Telefónica Chile, S. Neutralidad en la red. `http://www.movistar.cl/PortalMovistarWeb/neutralidad-de-la-red`, 2013.

[35] *Bufferbloat.net*. Community web site. `http://www.bufferbloat.net/`, 2012.

[36] VTR Banda Ancha Chile, S. Políticas de administración de red. `http://vtr.com/neutralidad/b4.php`, 2013.

[37] VTR Banda Ancha Chile, S. Vtr - reglamento de neutralidad de red. `http://vtr.com/neutralidad/b3.php`, 2013.

[38] Vu-Brugier, G., Stanojevic, R. S., Leith, D. J., and Shorten, R. N. A critique of recently proposed buffer-sizing strategies. *SIGCOMM Comput. Commun. Rev. 37*, 1 (Jan. 2007), 43–48.