DOCSIS® Best Practices and Guidelines

Cable Modem Buffer Control

CM-GL-Buffer-V01-110915

RELEASED

Notice

Document Status Sheet

Document Control Number:	CM-GL-Buffer-V01-110915			
Document Title:	Cable Modem Buffer Control			
Revision History:	V01 – Released 9/15/11			
Date:	September 15, 2011			
Status:	~~Work in Progress~~	~~Draft~~	Released	~~Closed~~
Distribution Restrictions:	~~Authors Only~~	~~CL/Member~~	~~CL/ Member/ Vendor~~	Public

Trademarks:

Contents

Figures

Tables

1 SCOPE

1.1 Introduction and Overview

Buffering of data traffic is a critical function implemented by network elements (e.g., hosts, switches, routers) that serves to minimize packet loss and maximize efficient use of the next-hop link. It is generally thought that for the Transmission Control Protocol (TCP) to work most efficiently, each element in a network needs to support sufficient buffering to allow the TCP window to open up to a value greater than or equal to the product of the next hop link bandwidth and the total round trip time for the TCP session (the bandwidth delay product). In many residential broadband scenarios, the cable modem is the head of the bottleneck link in the upstream direction, and as a result its buffering implementation can have a significant impact on performance - either imposing limits on upstream data rates, or increasing delay for latency-sensitive applications.

Historically, upstream buffer implementations in CMs have been sized (statically) by the modem vendor, and the network operator has had no control or visibility as to the size of the buffer implementation. In order to ensure optimal performance across the wide range of potential upstream configured data rates, modem suppliers have sized their upstream buffers so that there is sufficient buffering for TCP to work well at the highest possible upstream data rate, and at the greatest expected round trip time (i.e., the greatest possible bandwidth delay product). In the majority of deployment scenarios, the configured upstream data rate is significantly less than the highest possible rate, and the average TCP round trip time is significantly less than the greatest expected. The result is that in the majority of cases, the cable modem has an upstream buffer that greatly exceeds what is necessary to ensure optimal TCP performance.

In today's mix of upstream TCP and UDP traffic, some of which is latency-sensitive, some of which is not, the impact of a vastly oversized upstream buffer is that the upstream TCP sessions attempt to keep the buffer as full as possible, and the latency-sensitive traffic can suffer enormously. The result is poorer than expected application performance for VoIP, gaming, web surfing, and other applications. Studies have shown upstream buffering in deployed cable modems on the order of multiple seconds.

1.2 Purpose of Document

Recent work has established a standardized procedure for a cable operator to set the upstream buffer size at the cable modem via the modem's configuration file (in conjunction with other service defining parameters, such as the maximum traffic rate). This paper provides an introduction to this new capability, investigates the performance impact of buffer size on a range of traffic scenarios, and seeks to provide guidance for operators to properly configure buffer size for their service offerings.

2 REFERENCES

2.1 Informative References

This guideline uses the following informative references.

[1] V. Jacobson, Congestion Avoidance and Control, ACM SIGCOMM '88, August 1988.

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

[3] Braden, B., et al., Recommendations on Queue Management and Congestion Avoidance in the Internet, RFC 2309, April 1998.

[4] S. Floyd, Congestion Control Principles, BCP 41, RFC 2914, September 2000.

[5] K. Ramakrishnan, S. Floyd and D. Black, The Addition of Explicit Congestion Notification (ECN) to IP, RFC 3168, September 2001.

[6] M. Allman, V. Paxson and E. Blanton, TCP Congestion Control, RFC 5681, September 2009.

[7] C. Villamizar and C. Song, High Performance TCP in ANSNet. ACM CCR, 24(5):45-60, 1994.

[8] G. Appenzeller, I. Keslassy, and N. McKeown, Sizing Router Buffers, ACM SIGCOMM, USA, 2004.

[9] M. Dischinger, et al., Characterizing Residential Broadband Networks, Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement, Oct. 24-26, 2007, San Diego, CA, USA.

[10] A. Vishwanath, V. Sivaraman, M. Thottan, Perspectives on Router Buffer Sizing: Recent Results and Open Problems, ACM CCR, 39(2):34-39, 2009.

[11] ITU-T Recommendation G.114, One-way Transmission Time, http://www.itu.int

[12] ITU-T Recommendation G.107, The E-model, a computational model for use in transmission planning, http://www.itu.int

[13] ITU-T Recommendation G.108, Application of the E-model: A planning guide, http://www.itu.int

[14] ITU-T Recommendation G.109, Definition of categories of speech transmission quality, http://www.itu.int

[15] R. G. Cole and J. H. Rosenbluth, Voice over IP performance monitoring, ACM CCR, 31(2), 2001.

[16] Internetworking lectures from La Trobe University, http://ironbark.bendigo.latrobe.edu.au/subjects/INW/lectures/19/, 2011.

[17] J. Gettys, Bufferbloat: Dark Buffers in the Internet, http://mirrors.bufferbloat.net/Talks/PragueIETF/IETFBloat7.pdf, April 2011.

[18] L. Rizzo, Dummynet: a simple approach to the evaluation of network protocols, ACM CCR, 27(1), pp. 31-41, January 1997.

[19] S. Sundaresan, Broadband Internet Performance: A View From the Gateway, ACM SIGCOMM, August 2011.

[20] HTTP Archive, http://www.httparchive.org

[21] HttpWatch, http://www.httpwatch.com/download/

[22] Firebug, http://getfirebug.com

2.2 Reference Acquisition

- Cable Television Laboratories, Inc., 858 Coal Creek Circle, Louisville, CO 80027;
  Phone 303-661-9100; Fax 303-661-9199; Internet: http://www.cablelabs.com /
- Internet Engineering Task Force (IETF) Secretariat, 48377 Fremont Blvd., Suite 117, Fremont, California 94538, USA, Phone: +1-510-492-4080, Fax: +1-510-492-4001, http://www.ietf.org
- ITU-T Recommendations: www.itu.int/ITU-T/publications/recs.html

3 TERMS AND DEFINITIONS

This document uses the following terms:

Buffer	Temporary storage in a network element that holds data awaiting processing or forwarding
Buffer Control	Quality of Service (QoS) Parameter of a DOCSIS Service Flow used to control the amount of buffer storage allocated to the service flow
Dummynet	A network emulation software tool
Latency	A measure of time delay for transmission of packets
Latency Sensitive	Applications whose performance is dependent on the amount of latency between client and server
OTT VoIP	Over-the-top Voice over Internet Protocol
Ping Latency	Round-trip network transit time from one host to another and back, as measured using ICMP Echo Request/Response
Queue	Data temporarily held in a buffer
Wget	A software package for retrieving files using HTTP, HTTPS, and FTP

4 ABBREVIATIONS AND ACRONYMS

This document uses the following abbreviations:

3G	Third-Generation (mobile wireless)
ATA	Analog Terminal Adaptor
BDP	Bandwidth Delay Product
CM	Cable Modem
CMTS	Cable Modem Termination System
DNS	Domain Name Service
DSL	Digital Subscriber Line
FTP	File Transfer Protocol
GiB	gibibyte, 1,073,741,824 bytes
HTTP	Hypertext Transfer Protocol
I.E.	Microsoft Internet Explorer
ICMP	Internet Control Message Protocol
ITU	International Telecommunications Union
KiB	kibibyte, 1024 bytes
MB	Megabyte, 1,000,000 bytes
Mb/s	Megabits per second, 1,000,000 bits per second
Mbps	Megabits per second, 1,000,000 bits per second
MOS	Mean Opinion Score
PCMM	PacketCable Multimedia
QoS	Quality of Service
RTT	Round-Trip Time
TCP	Transport Control Protocol
TLV	Type-Length-Value tuple
VoIP	Voice over Internet Protocol
XP	Microsoft Windows XP

5 CM BUFFERING - BACKGROUND

5.1 TCP Best Practices - External Research

The Transport Control Protocol (TCP) [2] is a connection-oriented, end-to-end protocol that enables the reliable transmission of data across the Internet. The TCP is designed to recover data that is damaged, lost, duplicated, or delivered out of order by the network.

The original TCP specification provides reliability and end-to-end flow control through the use of sequence numbers and the use of a "receive window". All transmitted data is tracked by sequence number, which enables the TCP receiver to detect missing or duplicate data. The receive window enables flow control by identifying the range of sequence numbers that the receiver is prepared to receive, to prevent overflow of the receiver's data buffer space.

However, these controls were insufficient to avoid "congestion collapse" in the Internet in the mid-1980s [4]. Van Jacobson [1] developed the initial mechanisms for TCP congestion control, which many researchers have continued to extend and to refine [6]. In particular, TCP congestion control utilizes a "congestion window" to limit the transmission rate of the sender in response to network congestion indications.

Two key congestion control algorithms are "slow start" and "congestion avoidance". In slow start, the sender increases the congestion window by one segment (packet) for each TCP acknowledgement (ACK) received; this action doubles the transmission rate for each roundtrip time, leading to exponential growth. Once in congestion avoidance, the sender increases the congestion window by one segment per roundtrip time; this action increases the transmission rate linearly. If a segment is dropped due to a timeout, the threshold for entering congestion avoidance is set to one-half the current congestion window, and the congestion window itself is reset to one segment. This "sawtooth" behavior is illustrated in Figure 1 [16].

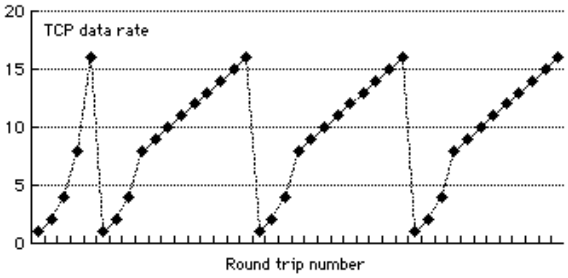


Figure 1 - TCP Congestion Control Algorithm

Modern TCP implementations also implement "fast retransmit" and "fast recovery" algorithms. According to fast retransmit, when the sender receives three duplicate ACKs, the sender assumes a segment has been lost and re-transmits that segment. If that segment is subsequently transmitted successfully, then under fast recovery the sender cuts its congestion window in half (rather than resetting to one segment).

Using modern congestion control, the TCP sender is constantly attempting to increase its transmission rate to the maximum possible, and it decreases the transmission rate in the presence of segment loss. As a result of this behavior, the TCP sender will send packets at a continuously increasing rate, filling the buffer of the network device at the head of the bottleneck link, until one or more packets is dropped. (This discussion ignores the presence of Active Queue Management [3] or Explicit Congestion Notification [5], neither of which are commonly implemented in broadband modems.)

Since the TCP sender's behavior is to keep the buffer full at the bottleneck link, it is helpful to review the research recommendations for the size of network buffers.

Back in 1994, Villamizar and Song [7] provided performance test results using up to eight TCP flows over a national backbone infrastructure. Their results led to the rule-of-thumb that bottleneck links need a buffer size equal to the bandwidth delay product (BDP) - that is, equal to the available bandwidth at the bottleneck multiplied by the roundtrip time of the TCP sessions - for the TCP flows to saturate the bottleneck link. A decade later, Appenzeller,

Keslassy and McKeown [8] confirmed the BDP rule-of-thumb for a single long-lived TCP flow, which may apply best to a bottleneck link at the edge of the network, such as a broadband modem's upstream link.

Since that time, the topic of buffering at network hops has become an active one, with some beginning to look at the impact that the rule-of-thumb has on non-TCP traffic, as well as whether the rule-of-thumb holds up under more real-world conditions. Vishwanath et al. [10] provide a good survey of recent work in this area.

If the optimal buffer size for TCP performance is presumed to be equal to the bandwidth delay product, then one might inquire how that compares to the buffer size implemented in current broadband modems. In 2007, Dischinger et al. [9] measured the performance of a number of DSL and cable operators in Europe and in North America. While the measured broadband modem minimum delay was typically under 15 milliseconds, the upstream buffer sizes were found to be quite large - resulting in 600 milliseconds of queuing delay for most DSL links, and several seconds for many cable modems. These results suggest that broadband buffer sizes are much larger than the BDP rule-of-thumb.

Gettys [17] shows that such large buffer sizes, or "bufferbloat", is an issue today with broadband modems both for wireline and for wireless networks. As one example, some have observed six seconds of latency in 3G wireless networks related to queuing. The issue of bufferbloat has also been observed in other consumer equipment, such as home routers and laptops. Gettys points to excessive buffering at the head of the bottleneck link as being disruptive to many user applications.

5.2 Cable Modem Upstream Buffering And Impacts on Application Performance/User Experience

In the upstream traffic direction, the cable modem is generally the entry point of the bottleneck link, both topologically and in terms of link capacity. As a result of this, its buffer size can have a significant impact on application performance. As noted previously, TCP upload performance can be sensitive to buffer size on the bottleneck link, and furthermore will seek to keep the bottleneck link buffer full. When latency-sensitive or loss-sensitive traffic is mixed with TCP traffic in the same cable modem service flow, issues can arise as a result of the modem buffer being kept full by the TCP sessions.

One impact will be that applications will see an upstream latency that can be predicted by the buffer size divided by the upstream configured rate. For example, in the case of a 256 KiB buffer and a configured upstream rate of 2 Mb/s, the upstream latency would be expected to be on the order of 1 second. More buffering will result in a proportional increase in latency.

Another impact will be that upstream applications will experience packet loss due to buffer overflow. Bulk TCP sessions are immune to packet loss resulting from buffer overflow, and in fact their congestion avoidance algorithm relies on it. However, other applications may be more sensitive. In contrast to the latency impact, the packet loss rate can increase if the buffer is undersized.

5.2.1 Performance of Real-Time Applications

Commonly used VoIP and video conferencing applications, such as Vonage, Skype, iChat, FaceTime, and umi, provide a better user experience when end-to-end latency is kept low. The ITU has developed models and recommendations for end-to-end delay for voice and other interactive services. The ITU-T Recommendation G.114 [11] suggests one-way latency be kept below 150ms in order to achieve "essentially transparent interactivity", and that delays greater than 400 ms are unacceptable for interactive applications. Furthermore, it has defined the "E-model" (ITU-T G.107 [12], G.108 [13] & G.109 [14]) used for predicting digital voice quality based on system parameters. The E-model has been reduced to a simpler form for VoIP applications by Cole & Rosenbluth, which points to significant degradation in voice quality beginning when the one-way latency exceeds ~177 ms. Packet loss also degrades quality, with losses of less than 2.5% being necessary to achieve "Best" quality. Included in the packet loss statistic are any packets that exceed the ability of the receiver de-jitter buffer to deliver them isochronously.

Multiplayer online games can also be sensitive to latency and packet loss. Fast-paced, interactive games are typically the most sensitive to latency, where some games require that players have less than 130 ms round-trip time between their host and the game server in order to be allowed to play, and lower latency is generally considered better.

Even more latency-sensitive are games that are run on a cloud server, with a video stream sent to the player, and a control stream sent from the player back to the server. Services such as these may require a maximum of 20-50 ms round-trip time across the cable operator's network.

While not typically considered to be a latency-sensitive application, web-browsing activities can also be impacted by upstream latency. Consider that a webpage load consists of a cascade of perhaps a dozen or more round trips to do DNS lookups and HTTP object fetches. An upstream latency on the order of 500 ms or more is likely to cause the web browsing experience to be much slower than a user might find acceptable, particularly if they are paying for a downstream connection speed measured in tens of megabits per second.

In addition, any downstream-centric TCP application (such as web browsing) has the potential to be throttled due to the latency experienced by the upstream TCP-ACK stream from the receiver. In practice this is not an issue, because many cable modems support proprietary means to accelerate TCP-ACKs by queuing them separately from bulk upstream data.

5.2.2 Current Cable Modem Buffering Implementation

Historically, cable modems have implemented static buffer sizes regardless of upstream data rate. Evidence suggests that cable modems in the field today may have buffers that are sized (using the BDP rule-of-thumb) for the maximum possible upstream data rate (~25 Mb/s for DOCSIS 2.0 CMs and ~100 Mb/s for current DOCSIS 3.0 CMs) and the maximum coast-to-coast Round-Trip Time (RTT) that might be experienced (100 ms or more).

Our observations indicate that most conventional cable modems are built with buffer size between 60 KiB and 300 KiB.

Since the majority of modems (particularly those in residential service) are currently operated with upstream rates in the range of 1 to 2 Mb/s, the result (as corroborated by Dischinger [9] and Sundaresan [19]) is that the modems have buffering latencies on the order of several seconds, which may be 1 or 2 orders of magnitude greater than would be ideal.

6 BUFFER CONTROL FEATURE

6.1 Definition of feature

A recent addition to the DOCSIS 3.0 specifications provides, for the first time, the ability for cable operators to tune the transmit buffers for cable modems and CMTSs in their networks. This new feature gives the operator control of the configured buffer size for each DOCSIS Service Flow. The feature controls upstream buffering in the cable modem, and downstream buffering in the CMTS.

The new feature is referred to as Buffer Control, and is defined as a Quality of Service (QoS) Parameter of a DOCSIS Service Flow. Specifically, the Buffer Control parameter is a set of Type-Length-Value (TLV) tuples that define a range of acceptable buffer sizes for the service flow, as well as a target size. This allows the CM/CMTS implementer some flexibility on the resolution of its buffer configuration. For example, a CM implementer might choose to manage buffering in blocks of 1024 bytes, if that offered some implementation advantages. When presented with a Buffer Control TLV, such a CM would then choose a number of blocks that results in a buffer size that is within the range of acceptable sizes, and is as close to the target buffer size as possible.

The Buffer Control TLV is composed of three parameters:

- **Minimum Buffer** - defines the lower limit of acceptable buffer sizes. If the device cannot provide at least this amount of buffering, it will reject the service flow. If this parameter is omitted, there is no minimum buffer size.
- **Target Buffer** - defines the desired size of the buffer. The device will select a buffer size that is as close to this value as possible, given its implementation. If this parameter is omitted, the device selects any buffer size within the allowed range, via a supplier-specific algorithm.
- **Maximum Buffer** - defines the upper limit of acceptable buffer sizes. If the device cannot provide a buffer size that is less than or equal to this size, it will reject the service flow. If this parameter is omitted, there is no maximum buffer size.

Each of these parameters is defined in bytes, with an allowed range of 0 - 4,294,967,295 (4 GiB).

As noted, inclusion of each of the three parameters in the service flow definition is optional. If all three parameters are omitted, the interpretation by the device is that there is no limitation (minimum or maximum) on allowed buffer size for this service flow, and that the device should select a buffer size via a vendor-specific algorithm. This is exactly the situation that existed prior to the introduction of this feature. As a result, if the operator does not change its modem configurations to include Buffer Control, the operator should see no difference in behavior between a modem that supports this new feature and one that does not.

In many configuration cases it will be most appropriate to omit the Minimum Buffer and Maximum Buffer limits, and to simply set the Target Buffer. The result is that the modem will not reject the service flow due to buffering configuration, and will provide a buffer as close as the implementation allows to the Target Buffer value.

In certain cases however, the operator may wish to be assured that the buffer is within certain bounds, and so would prefer an explicit signal (i.e., a rejection of the configuration) if the modem cannot provide a buffer within those bounds. Hard limits are provided for these cases.

Since they are part of the QoS Parameter Set, the Buffer Control TLVs can be set directly in the cable modem's configuration boot file; they can be set indirectly via a named Service Class defined at the CMTS; and they can be set and/or modified via the PacketCable™ Multimedia (PCMM) interface to the CMTS.

In order to use this new feature to control upstream buffering in DOCSIS 3.0 cable modems, it is necessary that the CM software be updated to a version that supports it. Furthermore, CMTS software updates are necessary in order to ensure that the CMTS properly sends the TLVs to the CM, regardless of which of the above methods is utilized.

6.2 CM Support

As of April 2011, support for Buffer Control is mandatory for DOCSIS 3.0 Cable Modems submitted for certification at CableLabs.

While the Buffer Control TLVs can encode values up to 4 GiB, cable modems are only required to support configurations up to 24 KiB. Support for significantly larger buffers is expected, but the 24 KiB value provides a "lowest common denominator" that is guaranteed. If the Minimum Buffer parameter is set to a value that is greater than 24576, then there is some risk that a modem implementation will reject the configuration. Operators choosing to use such values should test their configurations prior to deployment.

6.3 CMTS Support

6.3.1 CMTS Support for CM Buffer Control - Upstream Service Flows

In order to utilize Buffer Control for upstream service flows, it is necessary for the CMTS to properly support passing the Buffer Control parameters to the CM. CMTS support for CM Buffer Control can be characterized by three levels:

- **Minimal Implementation:** Support for CM Buffer Control via Configuration File. The CMTS simply copies the Buffer Control parameters from the Registration Request message to the Registration Response message. This support is mandatory for CMTS implementations in order to be considered compliant with DOCSIS 3.0, and is necessary for the CM Buffer Control feature to be utilized.
- **Scalable Implementation:** Support for CM Buffer Control via Service Class Name. The CMTS supports configuration of Buffer Control parameters as part of a named Service Class, which is then referenced in the CM configuration file. This support is mandatory for CMTS implementations in order to be considered compliant with DOCSIS 3.0.
- **Dynamic Implementation:** Support for PCMM-based dynamic creation and modification of upstream service flows having a specified CM Buffer Control setting. This support is mandatory for CMTS implementations in order to be considered compliant with PCMM.

6.3.2 CMTS Support for CMTS Buffer Control - Downstream Service Flows

Support for configuration of Buffer Control on downstream service flows is optional on the CMTS.

6.3.3 Current CMTS Support

At the time of publication of this document, support for Buffer Control by CMTS vendors is seriously lacking. While all vendors have plans to implement it, currently only one CMTS vendor supports CM (upstream) Buffer Control (via the Minimal Implementation only). No CMTS vendors support CMTS (downstream) Buffer Control.

7 RESULTS OF APPLICATION PERFORMANCE TESTING

7.1 General Testing Notes

The overall goals of the testing were to understand the impact of Buffer Control settings on the performance of various applications in various scenarios.

While all testing was performed in a laboratory environment in order to control the conditions of the experiments, certain test cases did involve transit across the Internet and as a result were subject to some variability. Even the tests that were entirely contained within the laboratory exhibited some degree of variability due to the complex dynamics at play between the multi-layered protocol stacks and multiple network hops. In all cases, experiments were repeated multiple times, and we report the calculated sample mean and/or median value.

Due to the constrained physical size and small number of network elements in the test network, artificial latency was added to emulate TCP sessions that traverse the Internet. In most cases, the Dummynet tool was used to insert a fixed amount of latency to selected traffic as appropriate for the test. Latency values of 40 milliseconds and 100 milliseconds were chosen for this testing. This latency is in addition to the inherent latency of the test network, which was approximately 17 milliseconds round-trip, so the total round-trip times for these tests were approximately 57 ms and 117 ms. The 57 ms value is representative of the round-trip time of a connection between an Internet user in North America accessing a server in North America, whereas the 117 ms value is at the upper end of what might be experienced in that scenario (see Appendix I). Both values are below what would be experienced in accessing a server overseas.

7.2 CM Buffer Control - Upstream Service Flows

Experiments were conducted using representative cable modems that implemented the Buffer Control feature, and the CMTS that implemented the Minimal Implementation of support for CM Buffer Control (this was sufficient for our purposes).

The applications tested were: upstream TCP file transfer, general low-bandwidth latency-sensitive upstream applications, over-the-top VoIP, and web page download.

7.2.1 Upstream TCP File Transfer

7.2.1.1 Sustained Transfer

In this test, the CM was configured with an upstream rate limit of 35 Mb/s (effectively no rate-limiting), and we investigated the achievable TCP data rate for a range of buffer sizes, both for the 57 ms RTT case and for the 117 ms RTT case. Figure 2 shows the resulting achieved data rate for 10 simultaneous upstream TCP sessions, along with the data rate that would be predicted based on the BDP rule-of-thumb.

The maximum data rate that a single DOCSIS upstream channel could support is approximately 25 Mb/s, so the achieved rates for 256 KiB, 128 KiB and perhaps 64 KiB should be considered to be artificially limited by this constraint.

It is expected that the single TCP session throughput would be limited by the CM Buffer Size as well as by the Maximum Receive Window of the TCP stack. So, for this case, the x-axis should be interpreted as the minimum of those two parameters. In other words, for Windows XP machines (for which the Maximum Receive Window is 64 KiB), single TCP session performance would max out at 4.5 Mbps for the 117 ms RTT case and 9.2 Mbps for the 57 ms RTT case.

It is interesting to note that in nearly all cases, when multiple TCP sessions were utilized, we were able to achieve a higher data rate (and in some cases a much higher data rate) than would be predicted based on the BDP rule-of-thumb. Our conjecture is that the multiple TCP sessions were desynchronized (that is, the TCP sawtooths were not synchronized across sessions), and Appenzeller, Keslassy and McKeown [8] demonstrate that significantly less buffering than the BDP is needed in this scenario. Further, in the multiple TCP scenario, we do not see a significant difference in throughput resulting from the TCP RTT differences.

This test was not intended to provide guidance for MSOs who don't use rate limiting, or who set the upstream rate limit beyond the channel rate, and in fact we noted different behavior between different modem implementations in this scenario. Rather, this test was intended to predict the buffer size that allows a single TCP session to completely utilize the provisioned upstream rate of the CM. As a result, this chart should be read by using the modem's provisioned upstream rate as the goal for Achievable Throughput, and then reading the resulting buffer size for the scenario of interest.

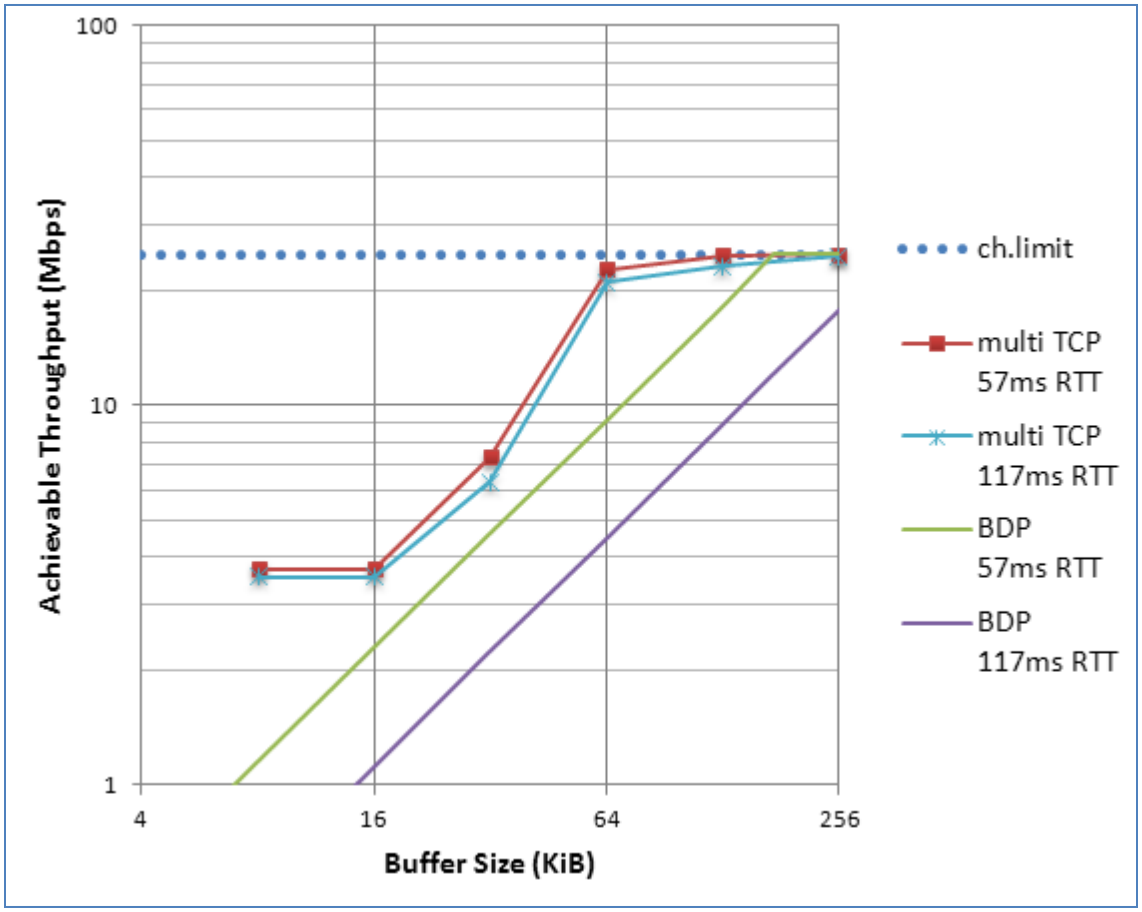


Figure 2 - Upstream TCP Data Rate

7.2.1.2 Token Bucket Burst settings

The impact of CM buffer control on performance for configurations that use a large Token Bucket Burst (i.e., power boost) is not fully understood at this time. MSOs using such features should test their configurations in order to determine the best setting.

7.2.2 Latency-Sensitive Applications - General

7.2.2.1 Goals of Test

In these tests we gauge the impact of CM Buffer Controls on latency for upstream traffic. We test a variety of scenarios, corresponding to 1 Mbps, 2 Mbps, 5 Mbps and 10 Mbps upstream rates in the presence of a single upstream TCP session as background traffic.

7.2.2.2 Network Configuration and Test Methodology

In this test, a single computer was connected behind the cable modem to generate a long term upstream TCP session (using wget). Dummynet was used to introduce 40 ms or 100 ms or additional latency for this traffic (for a total of ~57 ms and ~117 ms, respectively).

Simultaneous to the upstream TCP traffic, a series of ICMP ping messages was sent to the computer on the NSI side of the CMTS. The mean ping round-trip time is reported.

7.2.2.3 Test Results (ping latency)

We saw very little variability in ping latency between tests with 57 ms TCP RTT and 117 ms TCP RTT, which indicates that the single background TCP session was able to keep the CM's buffer equally full regardless of the RTT for the TCP session. As a result, we averaged the results for the 57 ms and 117 ms tests, and simply report ping latency vs. CM Buffer Size and Upstream Rate.

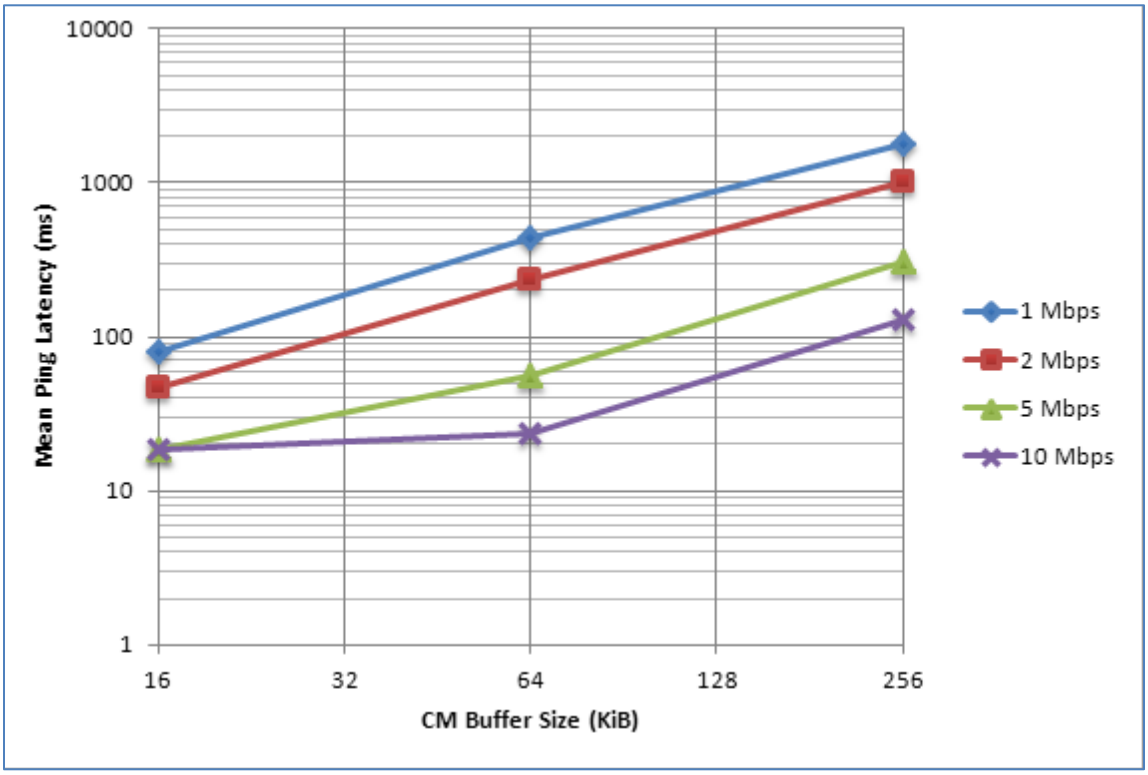


Figure 3 - Ping Latency vs. Buffer Size and Upstream Rate

For all upstream services, the pattern is that the bigger the buffer, the higher the ping latency. The expected pattern of the test results is that the ping time is composed of a constant term due to the round-trip time of the network (~17 ms) plus a buffering latency that is directly proportional to the buffer size and inversely proportional to the upstream rate (assuming that the single TCP session is keeping the buffer equally full in all test cases). Figure 4 (below) investigates how well the data fit this pattern. Plotted are the mean ping time vs. the predicted buffer latency (buffer size divided by upstream rate) for each of the test conditions. Also shown is an affine regression line constrained to reflect the expected 17 ms network RTT.

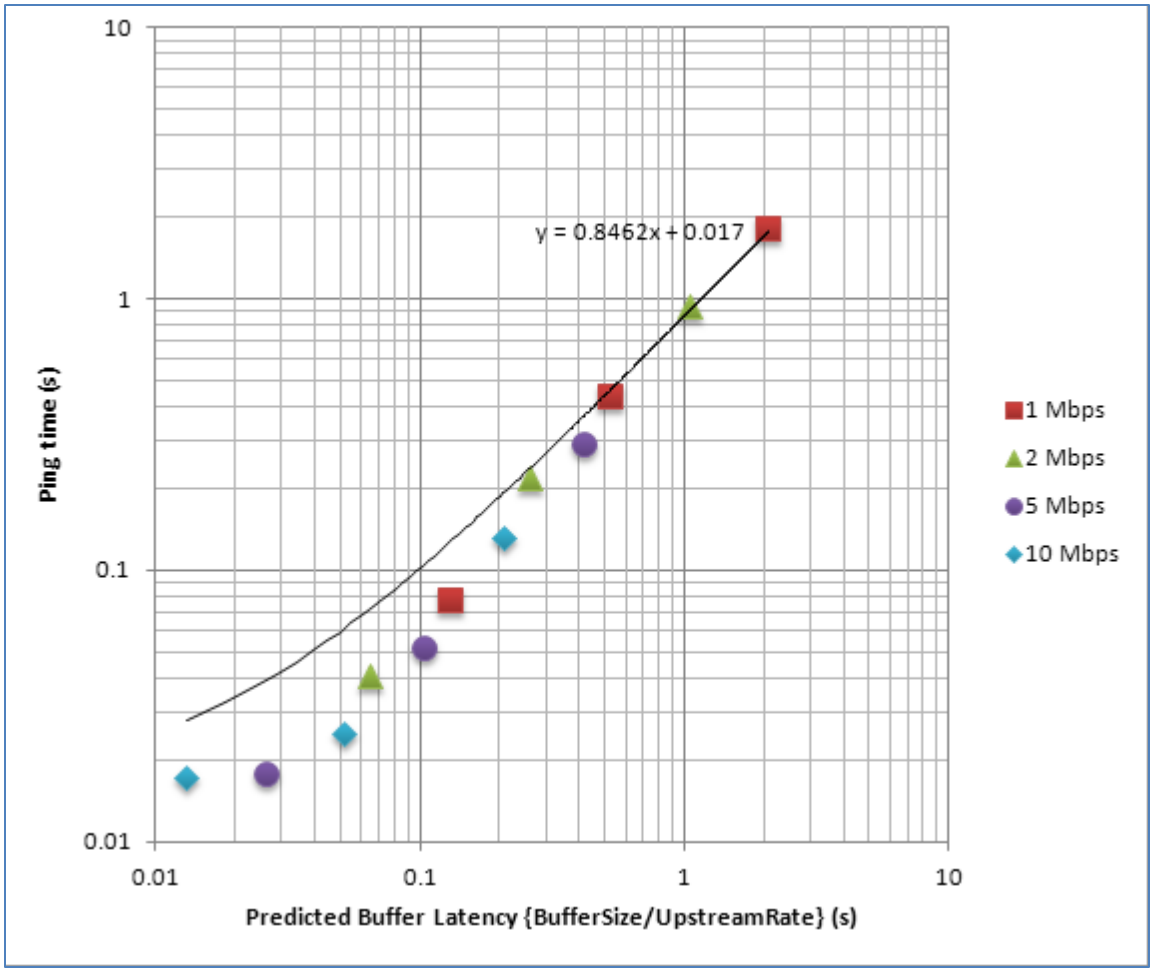


Figure 4 - Ping Time vs. Predicted Buffer Latency

The results show that for the larger values of predicted buffer latency, the single TCP session appears to keep the CM buffer about 85% full on average (regression line slope = 0.8462). However, we see a lower than predicted ping time for the smaller values of predicted buffer latency. This is due to the buffer being kept significantly less than 85% full by the single TCP session in these test cases. Recall that the TCP session was seeing either 57 ms or 117 ms of RTT. When the Predicted Buffer Latency is less than the TCP RTT, the TCP will not be able to achieve the provisioned upstream rate, and the buffer will periodically (at a period based on the RTT) be briefly filled then quickly drained and will sit empty. If multiple TCP sessions were utilized, we would expect the average buffer occupancy to be closer to 100% for all test cases, and thus to see the ping times track slightly above the regression line in Figure 4.

7.2.3 OTT VoIP

7.2.3.1 Goals of Test

The goals of this test are to investigate the impact of CM Buffer Controls on the voice quality of an "over-the-top" voice service in the presence of competing upstream TCP traffic. For this experiment we used the Vonage residential VoIP telephony service.

In order to characterize voice service quality, we utilized the traditional Mean Opinion Score (MOS) metric. The MOS is a continuous value ranging from 1 to 5 that represents the arithmetic mean of individual opinions across a group of listeners using the following subjective interpretation.

Table 1 - Mean Opinion Score (MOS) Interpretation

5	Excellent
4	Good
3	Fair
2	Poor
1	Bad

"Toll quality" is generally considered as meaning a MOS score of at least 4.

For our experiments, rather than using a listening panel and generating MOS in the traditional sense, we utilized a test tool that predicts MOS using objective measures.

7.2.3.2 Network Configuration and Test Methodology

Figure 5 depicts the experimental setup, which emulates a typical Vonage deployment. The home network consisted of PCs running test applications and other hardware for measuring network characteristics and performance. Typical home networks would include a home gateway between the CM and the rest of the home network equipment. To eliminate any bias that might be introduced by the packet forwarding performance of a home gateway, we instead used a Gigabit switch to connect the home network equipment to the CM. We then configured the CM to allow multiple devices connecting to it.

It should be noted that an alternative arrangement is for the additional home network traffic to be routed through the Vonage ATA in order that the voice traffic can be prioritized ahead of competing data traffic. While this arrangement may reduce the impact of buffer controls on VoIP service, it also reduces upstream throughput (and eliminates any power boost advantages). This installation is not recommended by Vonage except in the case that the user has a single PC.

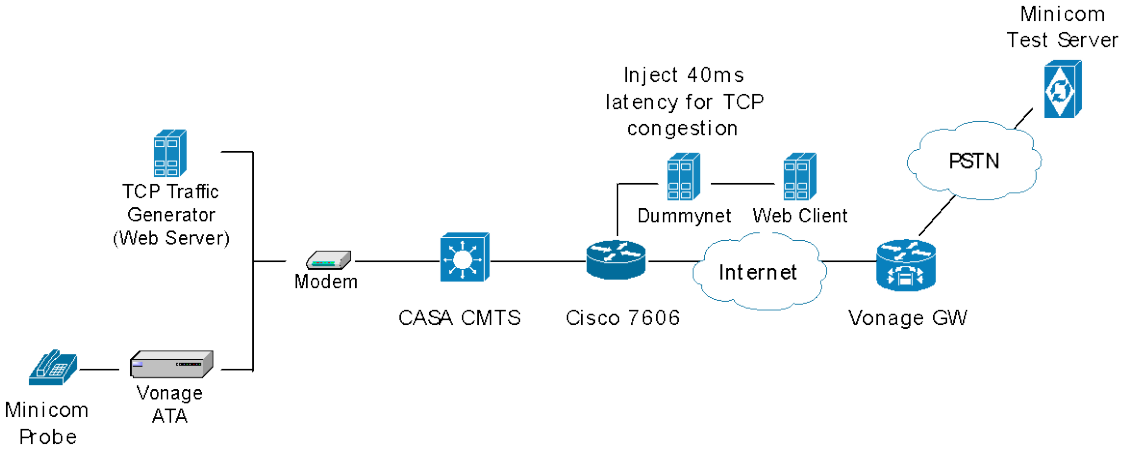


Figure 5 - Experimental Setup for VoIP Application Testing

The CM was installed with the new DOCSIS 3.0 firmware that provides the capability of using the Buffer Control feature so that specific buffer queue sizes might be set. A CMTS with knowledge of such settings was connected to the CM. Inside the home network, there are two applications: a VoIP Analog Terminal Adaptor (ATA), and a Web Server. A Minicom VoIP tester probe was connected to the ATA. This tester would calculate an estimate of the Mean Opinion Score (MOS) when the ATA made a phone call via the VoIP gateway in the Internet. The Web server was setup to generate packets over TCP by uploading a large file via HTTP to the client behind the CMTS. In order to simulate the network latency, Dummynet was used. Dummynet [18] is an internal packet filter for FreeBSD, which can inject artificial delay in the network. It was connected between the "headend" switch and the Web client. Delay was injected in both upstream and downstream. For example: we configured Dummynet to inject 20 ms in both directions for the HTTP upload test to simulate 40 ms additional RTT delay.

7.2.3.3 Test Results (Vonage MOS)

In this test, we wanted to understand how buffer size would impact the VoIP MOS under congestion. The VoIP ATA was connected into the simulated home network in our test environment. We started a large upstream file transfer (using a single TCP session) to simulate a self-congested upstream. The Dummynet test set injected either 40 ms or 100 ms additional round-trip latency to the TCP stream. For each bandwidth/buffer/delay permutation, we ran 10 tests and computed the average MOS. Note that the Vonage VoIP traffic traverses the Internet in order to reach the Vonage gateway. As a result, some variability in results is expected due to the uncontrolled nature of a portion of the network.

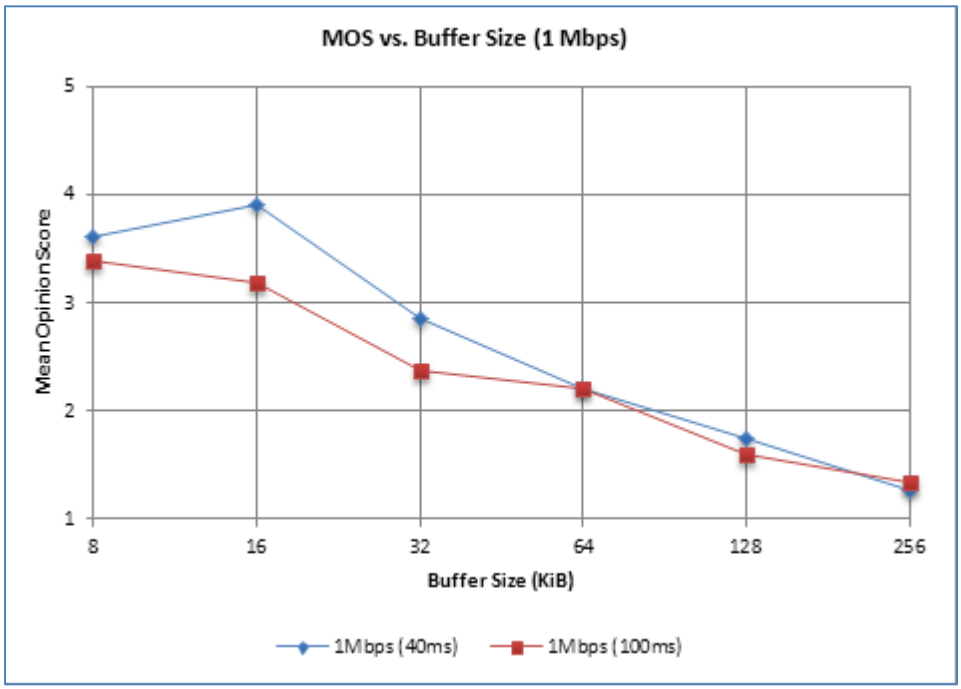


Figure 6 - Results with 1 Mb/s Upstream Configuration

Figure 6 shows the results when the upstream was configured to 1 Mb/s. In this case, the best MOS result was achieved with 8 KiB or 16 KiB buffer size, depending on the TCP RTT. When the buffer size increased to 32 KiB, the MOS dropped significantly. In fact, when the buffer size was set to 256 KiB with 100 ms additional delay, many tests couldn't be completed. The test set reported a timeout. The timeout could be caused by long delay introduced by the large buffer.

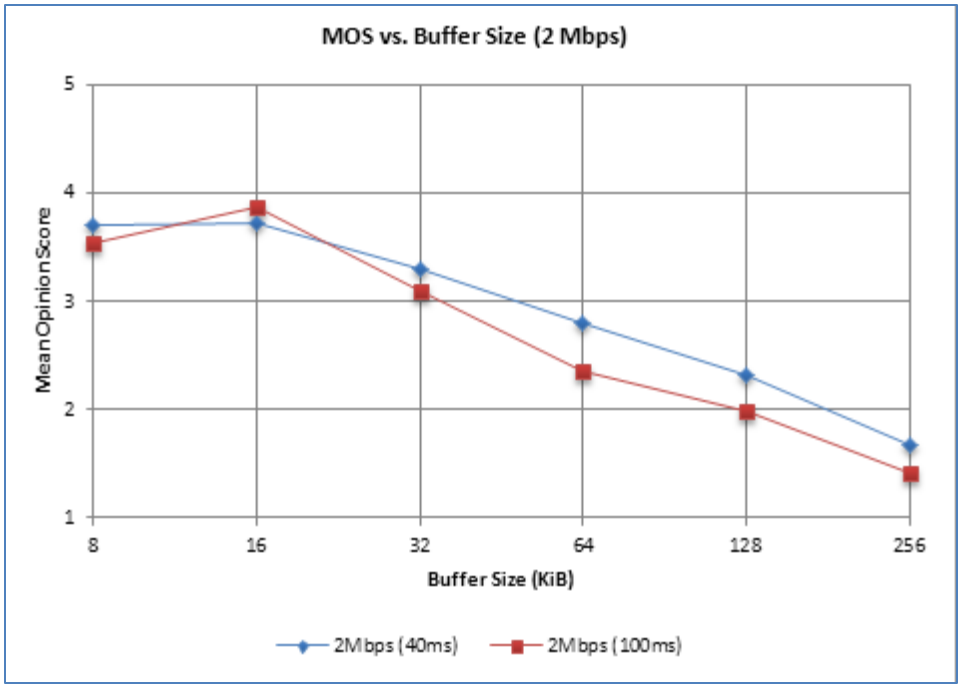


Figure 7 - Results with 2 Mb/s Upstream Configuration

Similar results were obtained for 2 Mb/s, as shown in Figure 7, although we see a more distinct peak in MOS at 16 KiB.

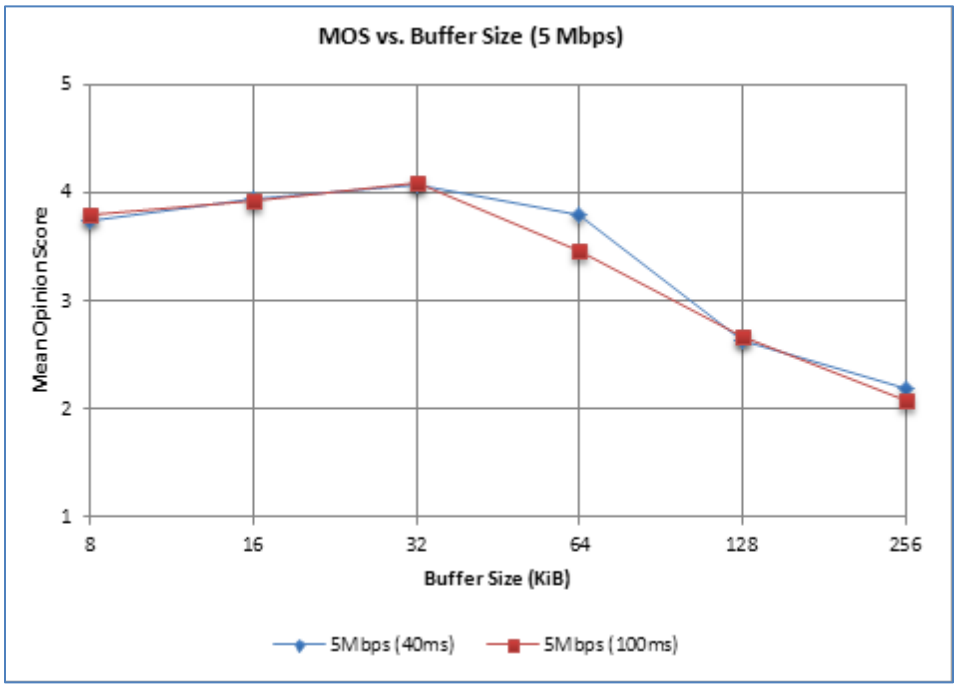


Figure 8 - Results with 5 Mb/s Upstream Configuration

Figure 8 shows the results using the 5 Mbps upstream rate. For this case, the best MOS results were recorded when the buffer size was set to 32 KiB. When the buffer was set below 32 KiB, there should not be any congestion

because the buffer size was too small for the competing TCP session to utilize the full upstream bandwidth. Also, when the buffer size was set to 8 KiB or 16 KiB, the VoIP traffic should see very low buffering latency (between 6 ms and 26 ms). Both of these factors should result in high MOS scores. However, the MOS score was degraded when the buffer size was set to 8 KiB or 16 KiB, and this low MOS is likely to be caused by VoIP packet loss due to the periodic saturation of the small CM buffer by the competing TCP congestion window.

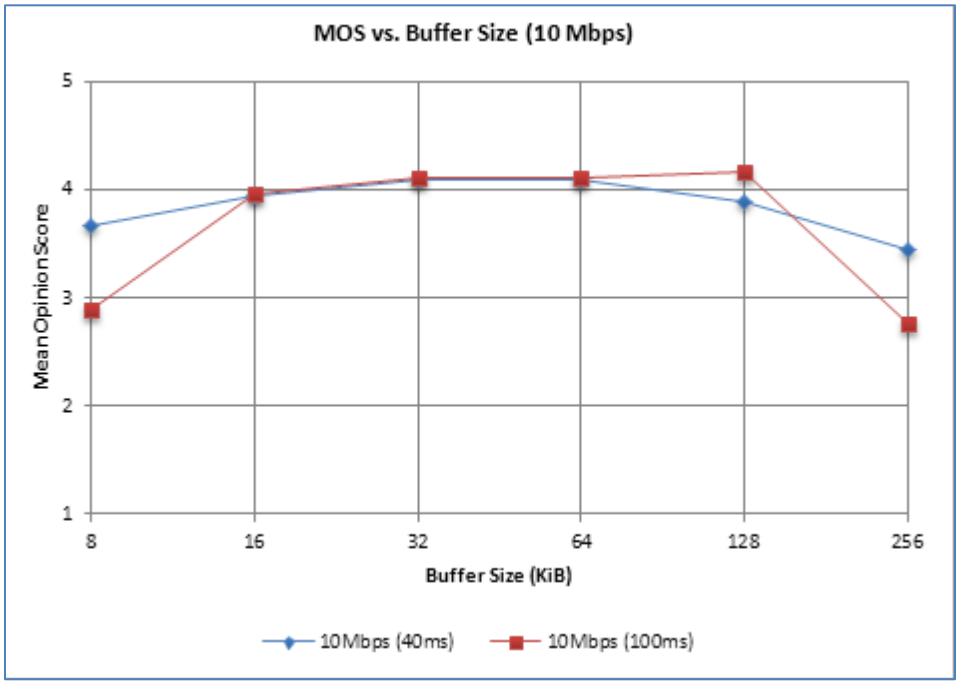


Figure 9 - Results with 10 Mb/s Upstream Configuration

Figure 9 shows the results for the 10 rate. At this configured data rate, we saw consistently good MOS across a range of buffer sizes from 16 KiB to 128 KiB, with the peak occurring between 32 KiB and 64 KiB. When we increased the buffer size above 128 KiB, the MOS started to deteriorate. In fact, the test set failed to complete some tests and required us to re-run the test in a number of cases. This could be caused by the long delay of the VoIP signaling packets for the call setup.

Another interesting observation is that, while adding latency to the TCP session did in some cases affect the VoIP MOS, in most cases it didn't point to a different optimal buffer size. Additionally, TCP RTT is not a controllable attribute for an MSO; users will select the sites that they wish to interact without regard to RTT. As a result, we have summarized the findings of this test by averaging the 40 ms and 100 ms additional RTT data points, as shown in Figure 10.

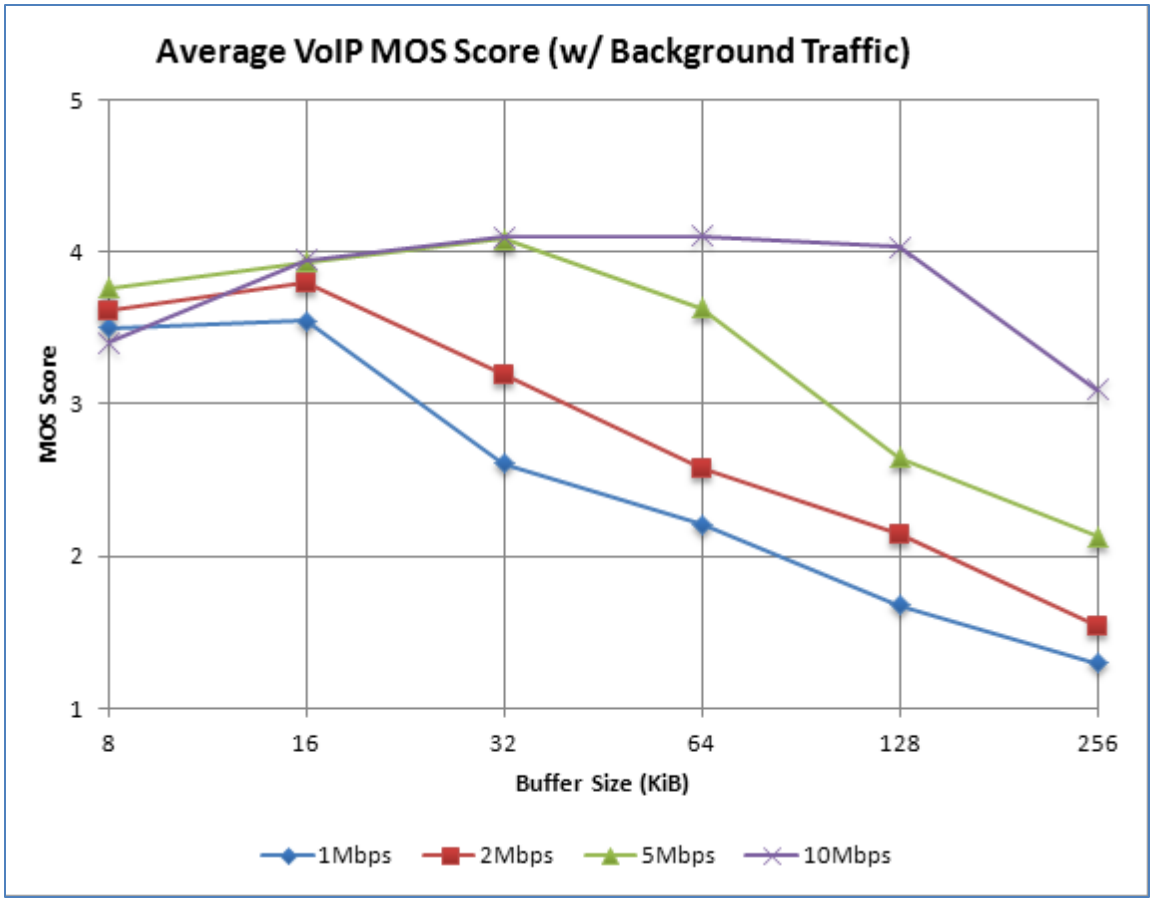


Figure 10 - Average VoIP MOS Score

7.2.4 Web Page Download

Web browsing application performance was tested by way of measuring the time to download a single web page. While web browsing is largely considered to be a downstream-centric activity, and in terms of bytes-transferred it is heavily weighted toward downstream, it has an upstream component, and is (perhaps surprisingly) very sensitive to upstream latency.

Current (August 15, 2011) data from HTTPArchive [20] indicates that, on average, a webpage in the top 1000 webpages contains 86 resources (and in fact some content-rich web pages can have upward of 300 resources), and this value has been trending upwards over time. As a web browser downloads a web page, it requests for each resource individually via an HTTP GET method. Browsers differ in terms of how they manage these requests, but in all cases the requests are handled serially to some extent. The result is that the total webpage download time is governed by several, perhaps dozens, of round-trips. The upstream buffering latency in the cable modem is thus multiplied several times over for each webpage download.

In this test we utilized a webpage that consisted of 102 resources: a single HTML file (20 KiB), a javascript library (94.5 KiB), and 100 image files (average size 3343 bytes). The total webpage size was ~441 KiB. The page was hosted locally on a server behind the CMTS, and 40 ms of additional round-trip latency (for a total of ~57 ms) was added to emulate a typical web browsing experience. The webpage was constructed to add randomized query strings on each image file request so that the files would be downloaded across the network each time, rather than being served by local cache. The javascript library was not subject to the same treatment, and so was retrieved from cache by the browser.

To test performance under upstream load, TCP upstream traffic was introduced. For this we utilized FTP with a RTT of 117 ms (100 ms latency added by Dummynet). Testing was performed with no background traffic, with a single background TCP session, and with 10 simultaneous TCP background sessions. The plots in the forthcoming subsections show results labeled "0TCP", "1TCP", and "10TCP" to indicate the three scenarios, respectively.

Testing was performed with upstream rates configured as 1 Mbps, 2 Mbps, 5 Mbps and 10 Mbps. In all cases, an upstream Maximum Burst (token bucket depth) of 40 KiB was used. No downstream rate limiting was configured.

For all test cases, the webpage was downloaded 10 times in succession, and the total load time for each download was recorded. We present the median load time.

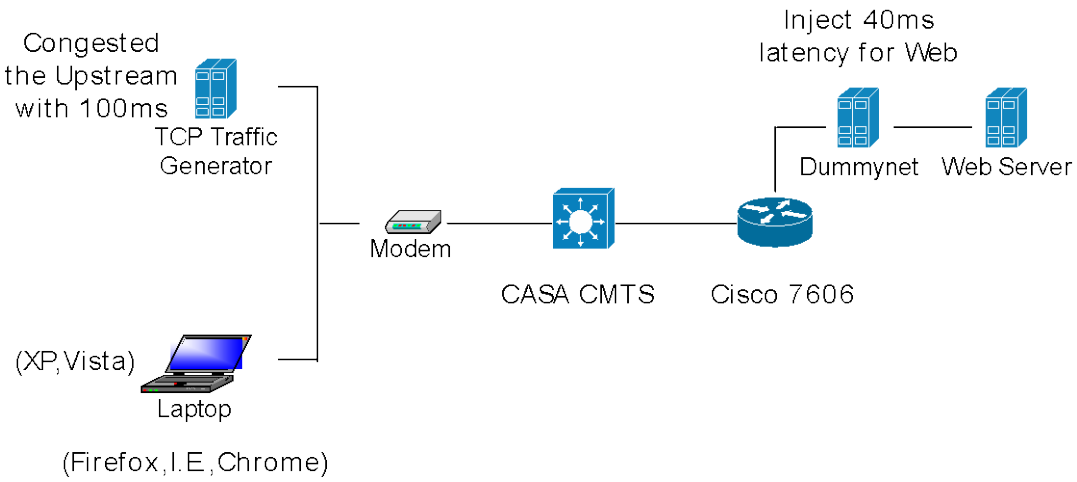


Figure 11 - Web Page Download Testing Setup

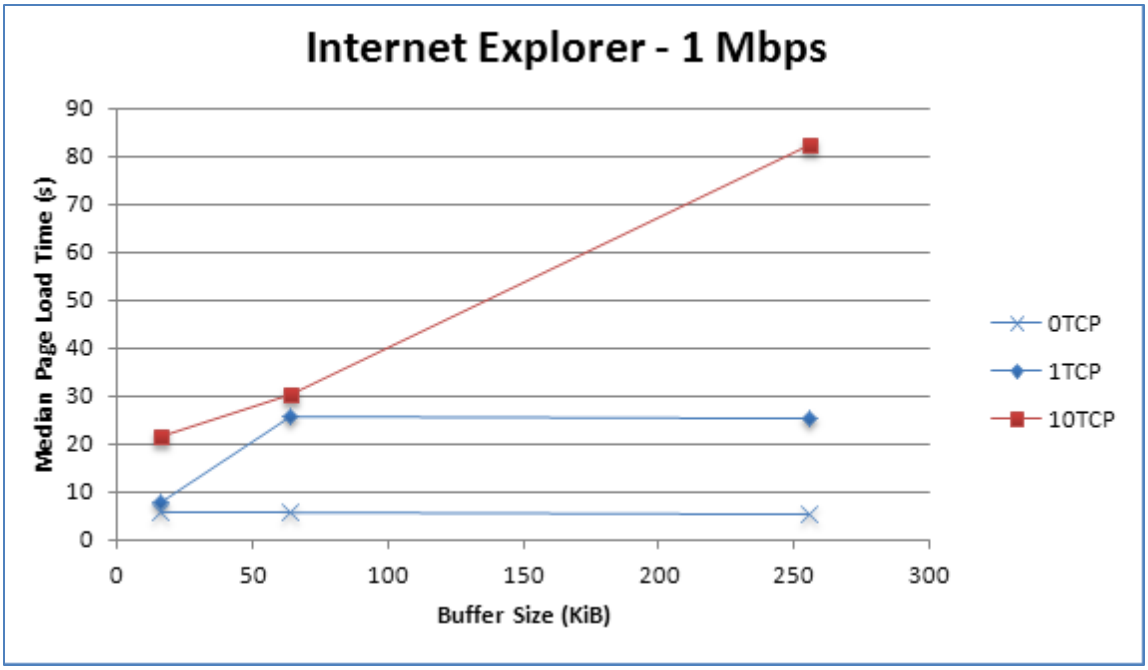
7.2.4.1 Internet Explorer results

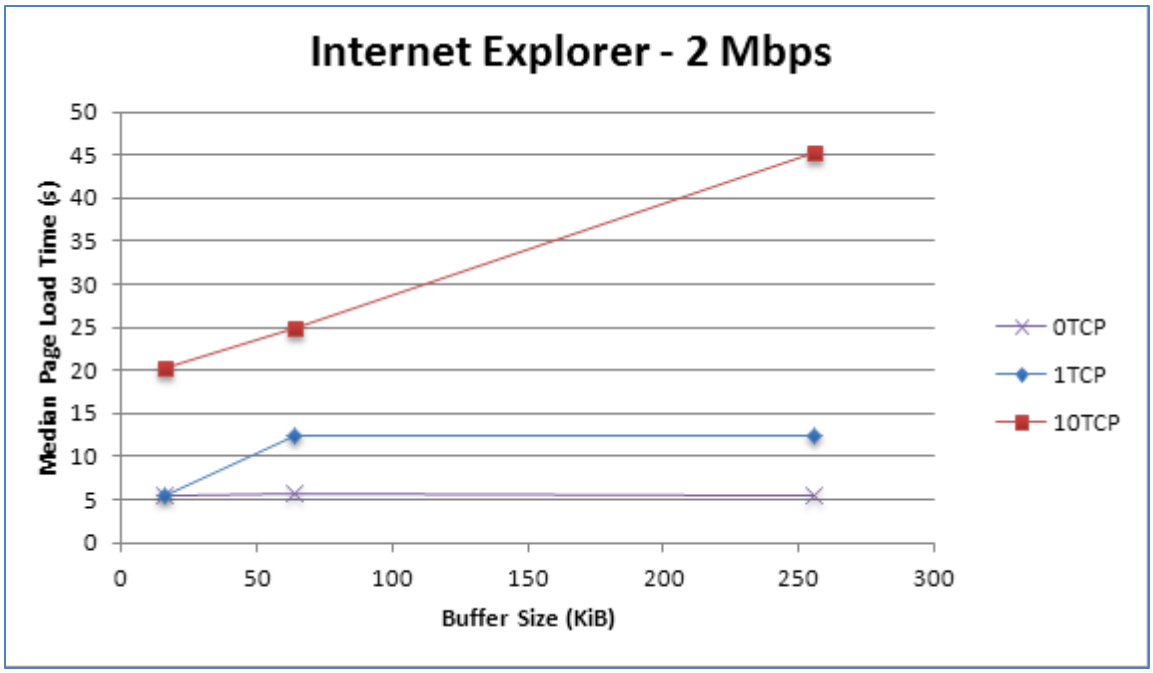
Testing was performed using Microsoft Internet Explorer 7.0 running on a Windows XP machine. The HTTPWatch [21] browser plugin (v.7.2.4) was used to monitor web page load time.

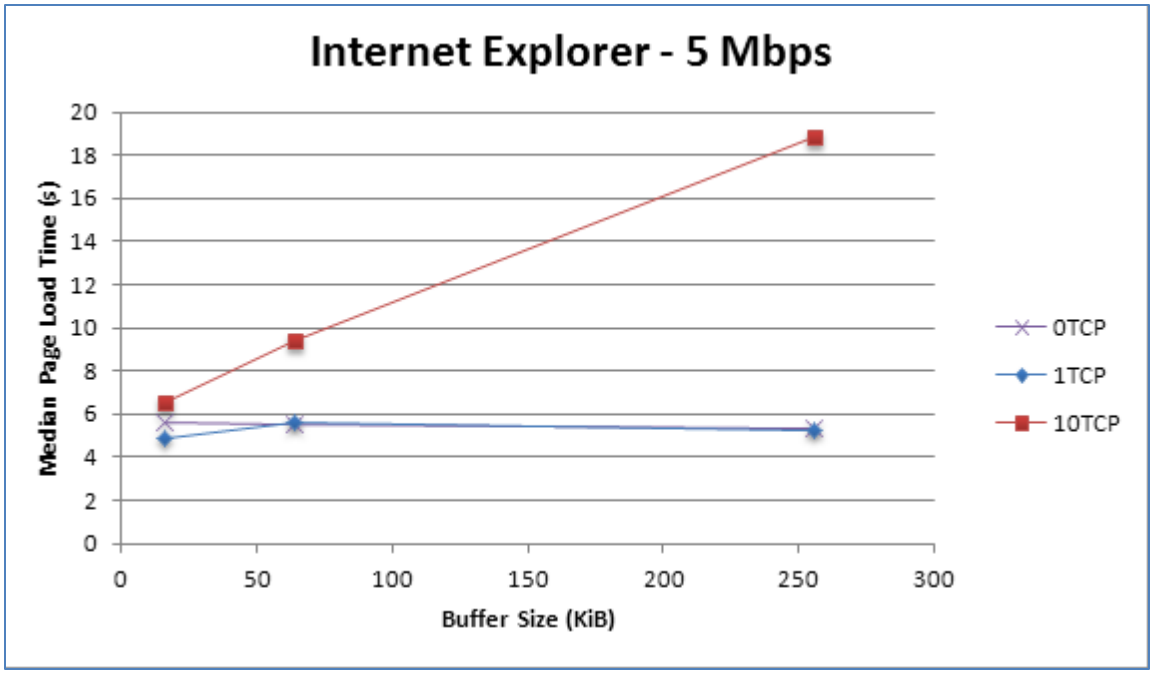
The testing covered CM Buffer Sizes of 16 KiB, 64 KiB and 256 KiB.

For this test, the upstream background traffic sessions utilized a 64 KiB maximum receive window, meaning that each background TCP would never have more than 64 KiB "in-flight" at any time. Thus, in the tests with a single TCP session, there would never be more than 64 KiB in the CM buffer, and as a result we see no further degradation of performance above 64 KiB CM buffer size.

For each webpage download, we observed the Internet Explorer browser to open up a single TCP session to fetch the HTML file and then open 9 subsequent TCP sessions, and use these 10 resulting TCP sessions to download the 100 image files (one at a time per session). Once all resources had been received, the browser would close the 10 TCP sessions.


Internet Explorer - 1 Mbps
Median Page Load Time (s)
90
80
70
60
50
40
30
20
10
0
0
50
100
150
200
250
300
Buffer Size (KiB)
0TCP
1TCP
10TCP


Internet Explorer - 2 Mbps
Median Page Load Time (s)
50
45
40
35
30
25
20
15
10
5
0
0
50
100
150
200
250
300
Buffer Size (KiB)
0TCP
1TCP
10TCP



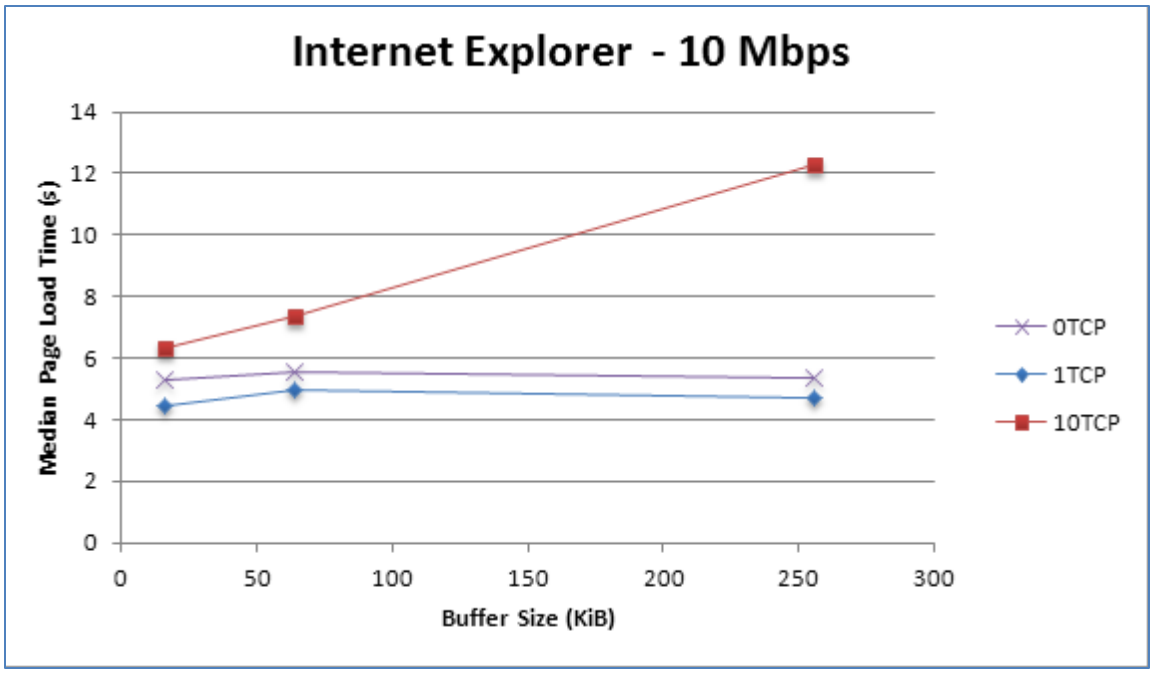


Figure 12 - Internet Explorer Results with 1, 2, 5 and 10 Mbps

7.2.4.2 Chrome Results

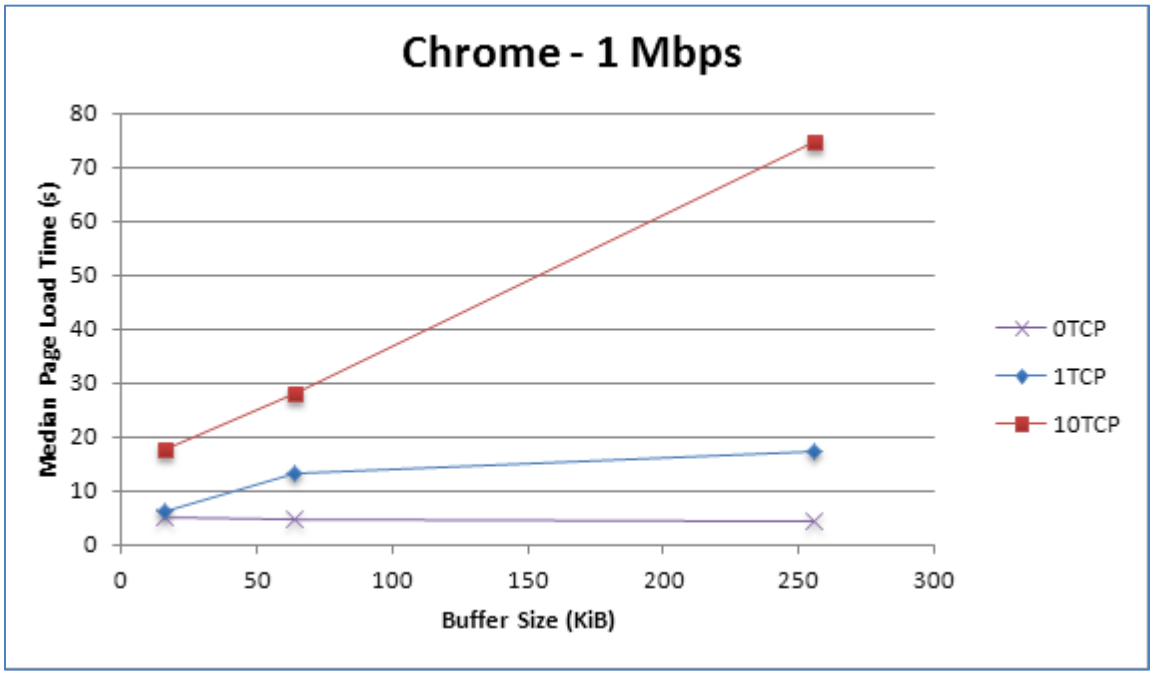
Testing was performed using Google Chrome v13.0 running on a Windows XP machine. The Chrome Developer Tools were used to monitor page load time.

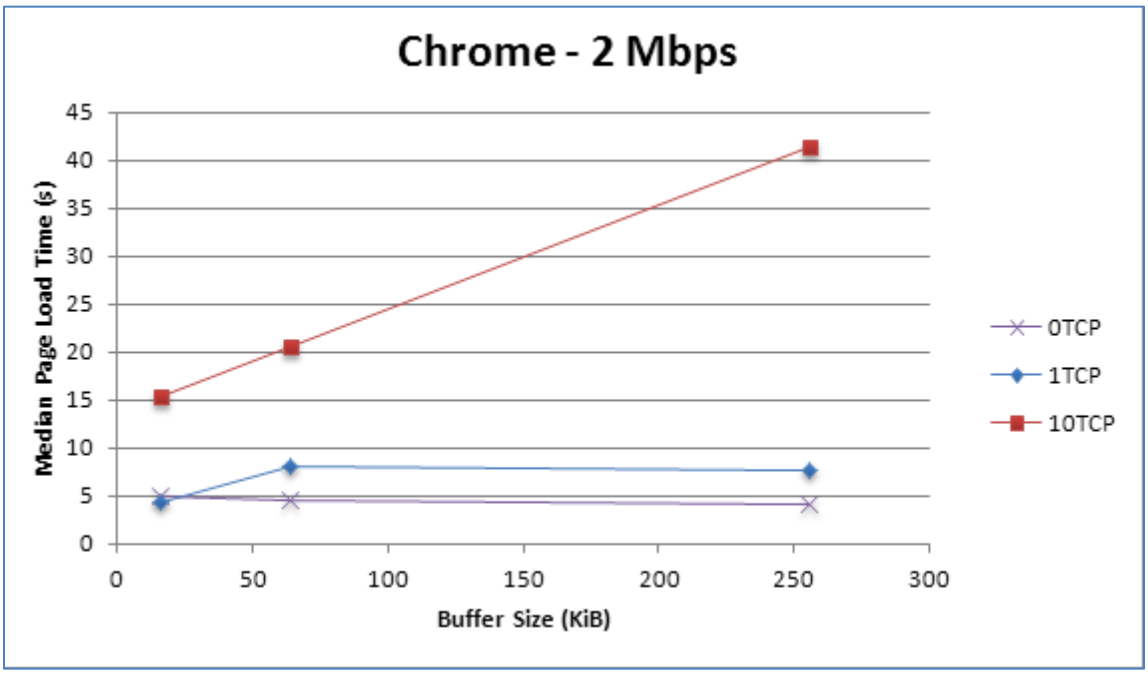
The testing covered CM Buffer Sizes of 16 KiB, 64 KiB and 256 KiB.

For this test, the upstream background traffic sessions utilized a 64 KiB maximum receive window, meaning that each background TCP would never have more than 64 KiB "in-flight" at any time. Thus, in the tests with a single

TCP session, there would never be more than 64 KiB in the CM buffer, and as a result we see no further degradation of performance above 64 KiB CM buffer size.

For each webpage download, we observed the Chrome browser to open a new TCP session for each resource downloaded. The browser would limit itself to 6 simultaneous sessions.





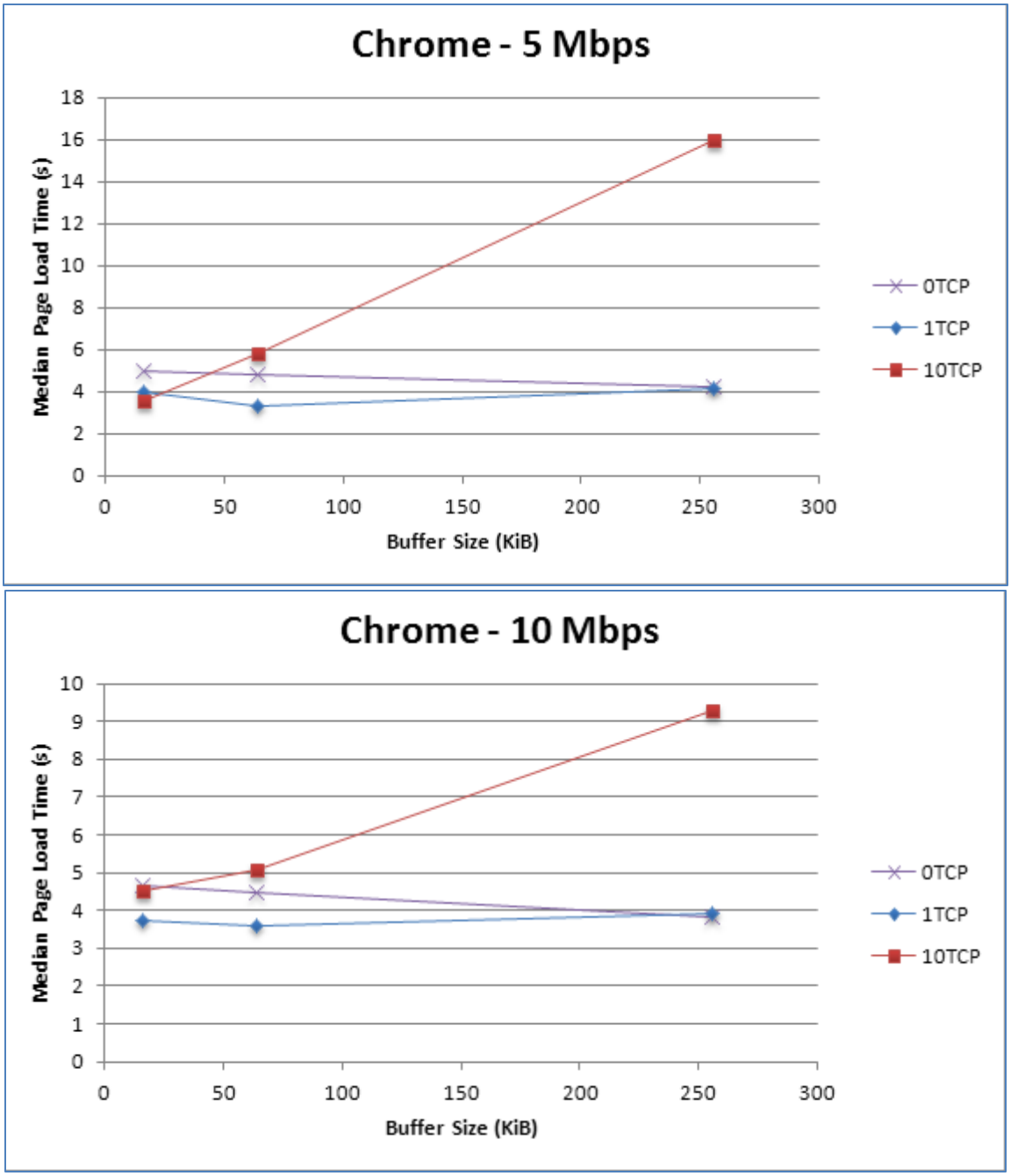


Figure 13 - Chrome Results with 1, 2, 5 and 10 Mbps

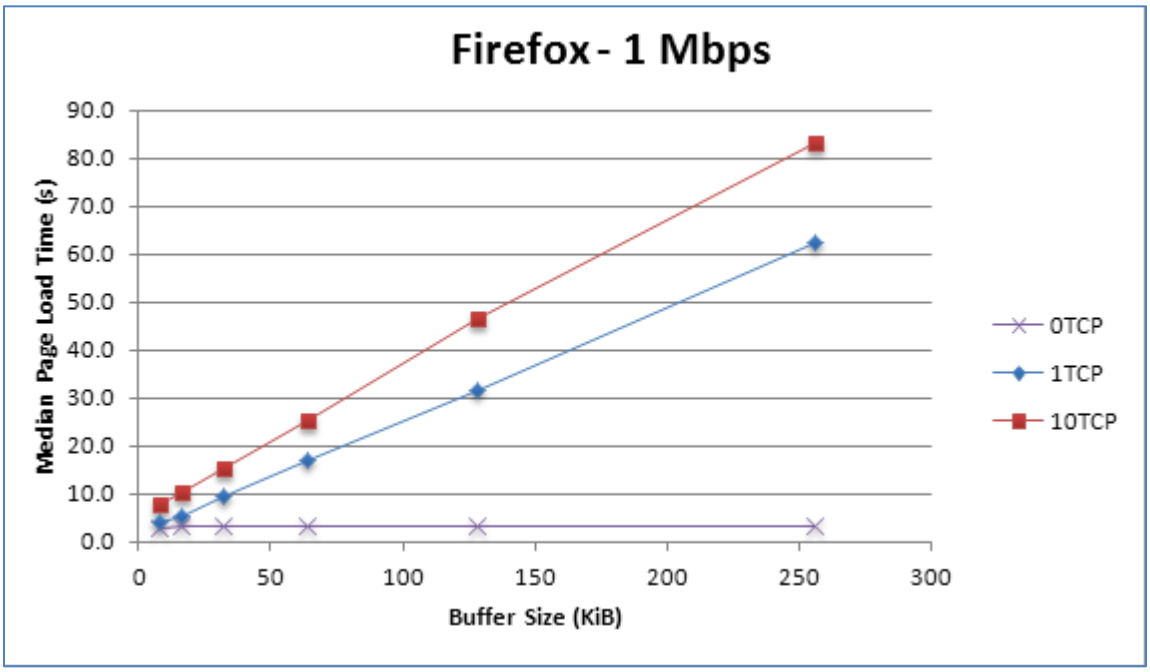
7.2.4.3 Firefox results

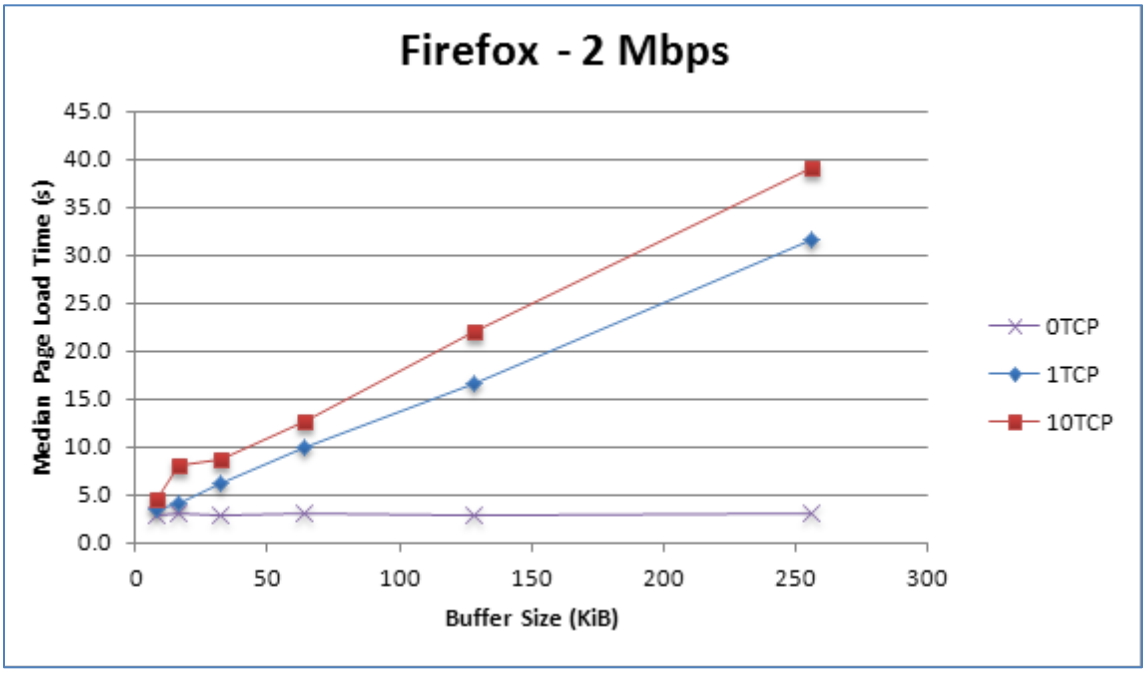
Testing was performed using Firefox 6.0 running on a Windows XP machine. The Firebug browser plugin (v.1.8.1) was used to monitor web page load time.

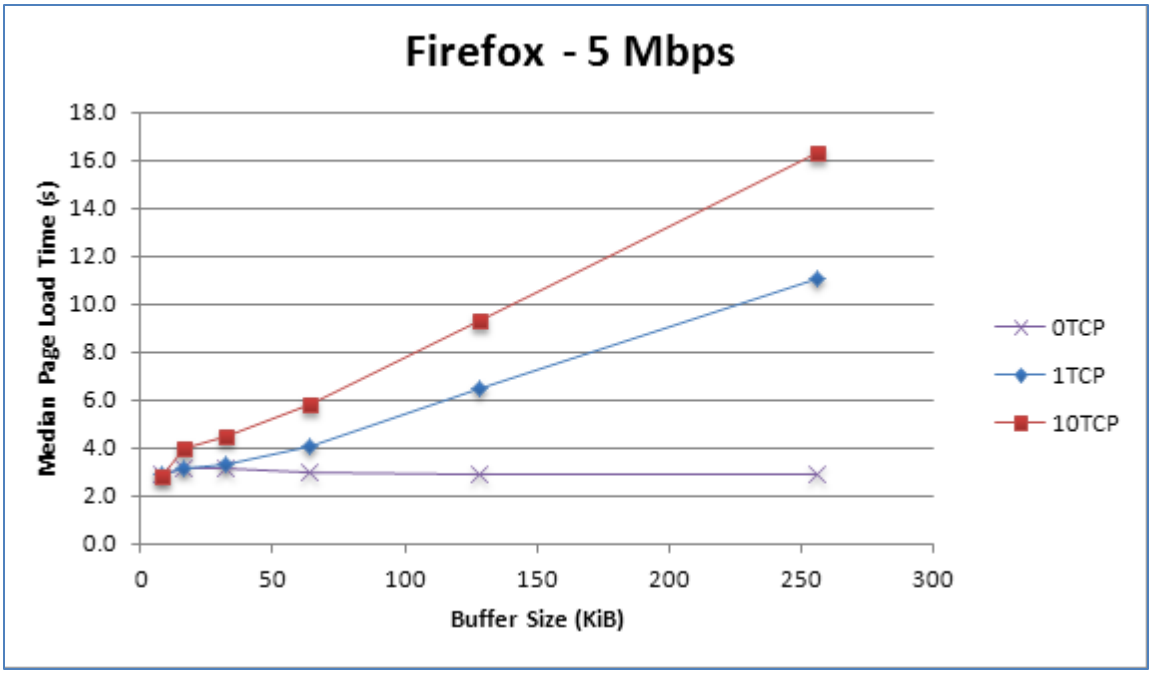
The testing covered CM Buffer Sizes of 8 KiB, 16 KiB, 32 KiB, 64 KiB, 128 KiB and 256 KiB.

For this test, the upstream background traffic sessions utilized a 256 KiB maximum receive window. This differs from the test environment used for the other browsers, where a 64 KiB maximum receive window was used. As a

result, we see the 1 TCP results tracking (slightly below) the 10 TCP results for all CM buffer sizes tested, as opposed to leveling off above 64 KiB.







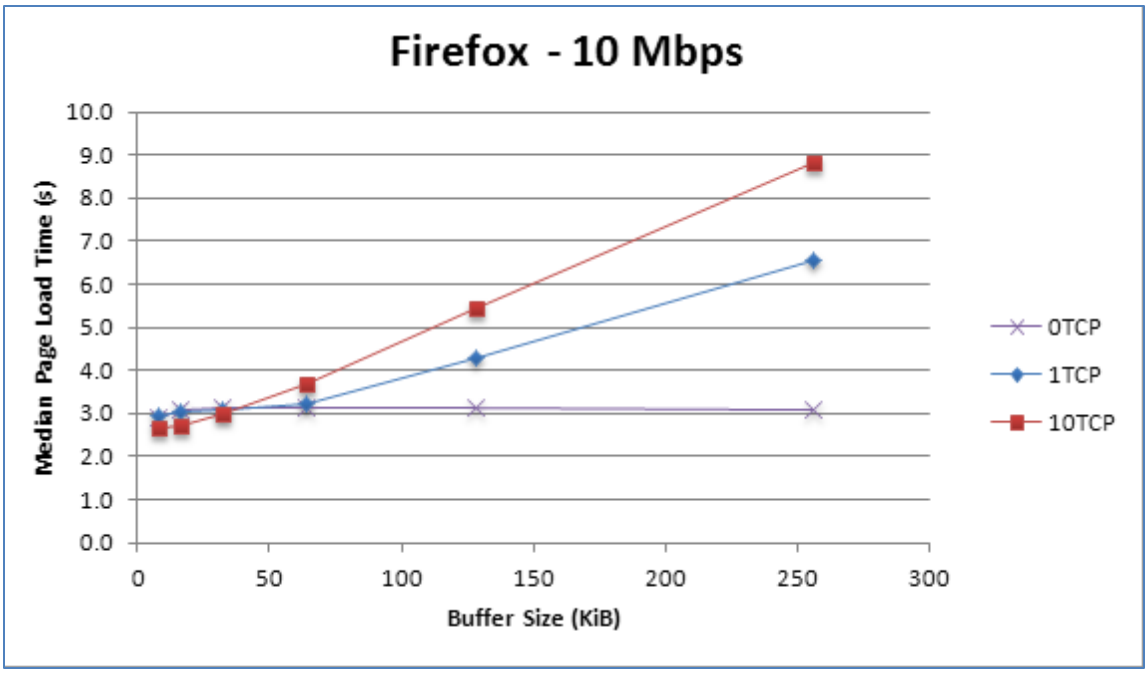


Figure 14 - Firefox Results with 1, 2, 5 and 10 Mbps

7.2.4.4 Summary of Webpage Download Testing

Figure 15 shows the results for the three tested browsers for each of the four upstream rates, in the presence of upstream background traffic (10 TCP).

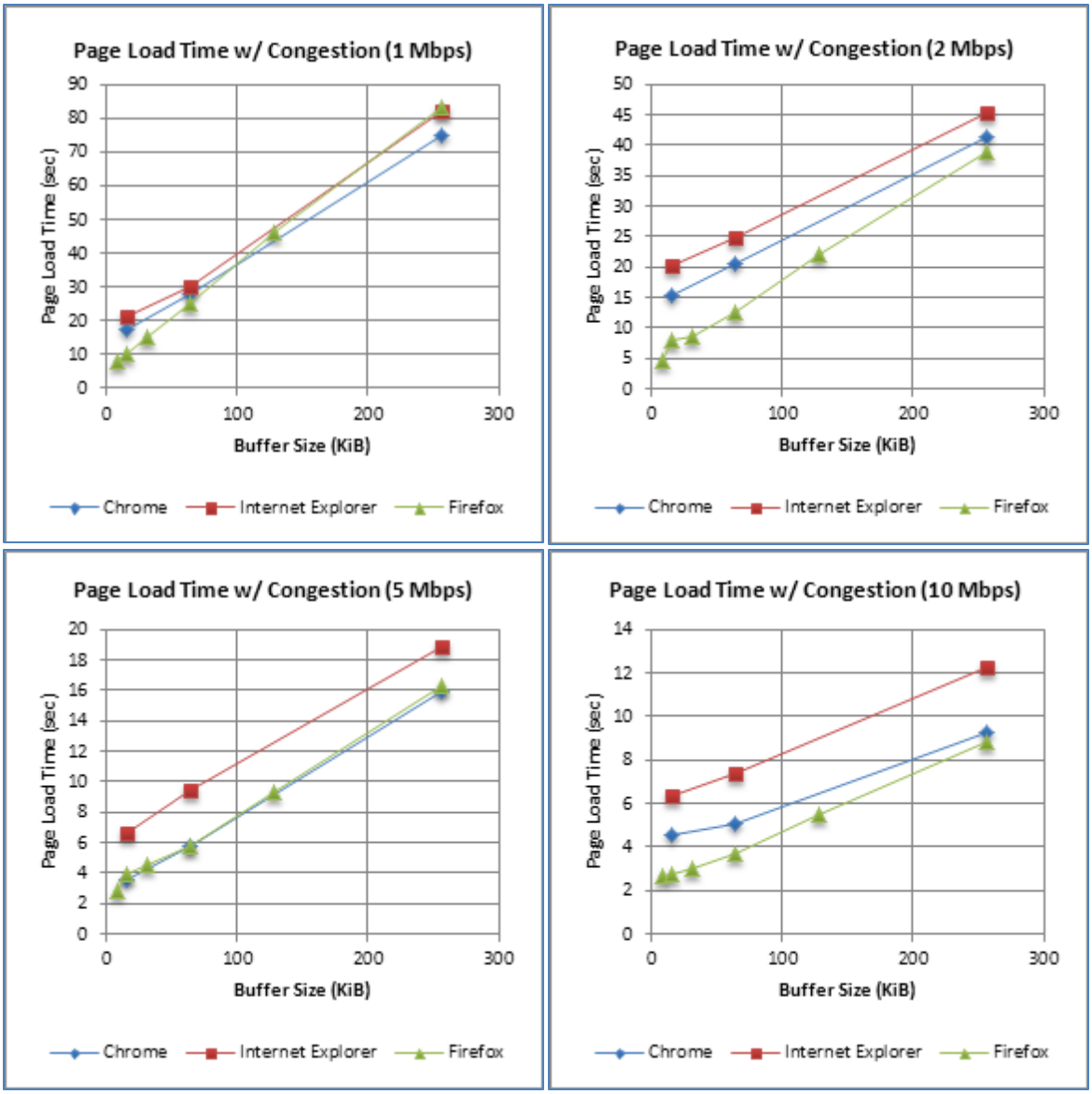


Figure 15 - Comparison of browser performance in presence of upstream congestion

The results of this test are striking. While there were differences observed between web browsers, it was universally true that a web page that would take on the order of 3-5 seconds to download when no background traffic was present could take substantially (in some cases more than 20x) longer when background traffic was present and the CM upstream buffer was set to 256 KiB. We observed the best web page download performance by using the smallest tested upstream buffer sizes. Figure 16 summarizes the performance achieved during upstream loading (the 10 TCP case) for the four different upstream rates.

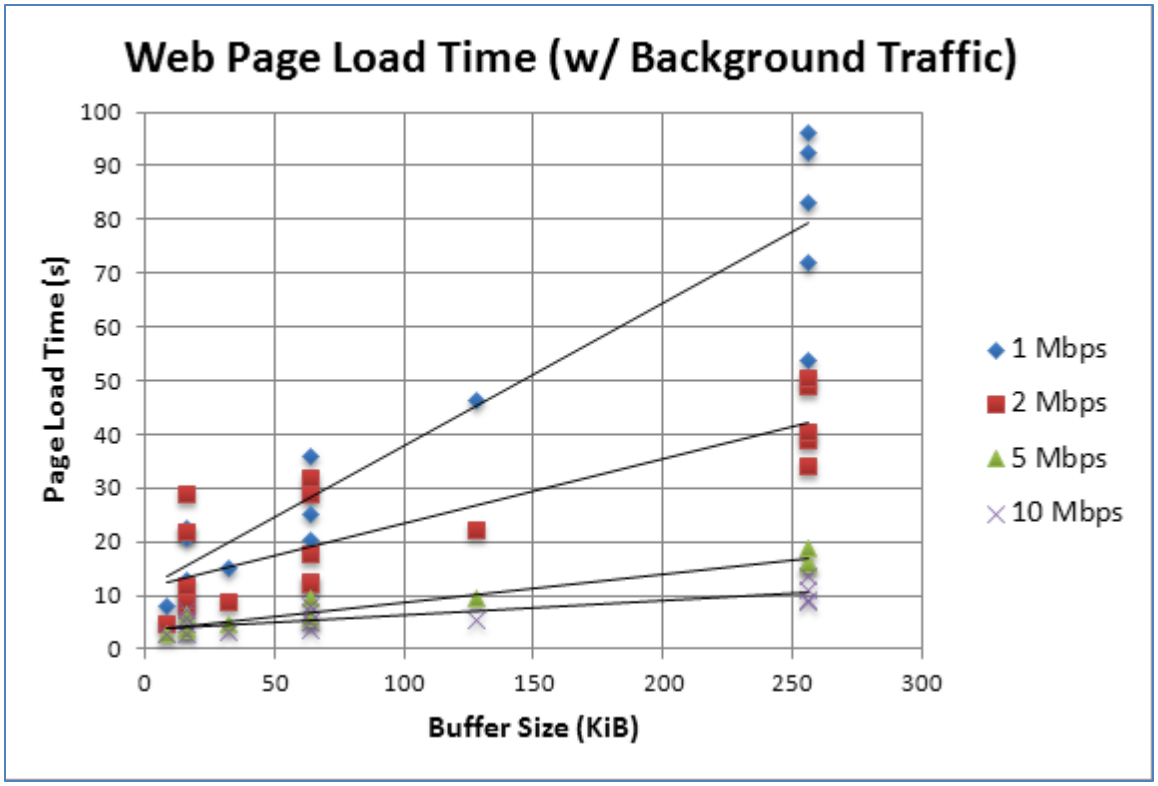


Figure 16 - Web Page Load Time (the 10 TCP case) for the Four Different Upstream Rates.

It should be noted that the results presented here are median values, and in many of the experimental runs there was significant variability in page load time. This is a complex, multilayered, dynamic system. The total page load time is driven by the browser dynamics in how it utilizes TCP sockets and how it responds to error conditions, the TCP stack in the client and in the server and their response to different packet loss situations, and the DOCSIS MAC layer and CM buffering process. There will be packet loss occurring in the system as a result of buffer exhaustion and the frequency of packet loss and, more importantly, the precise pattern of loss relative to the higher layer protocols can dramatically affect the page load time.

7.3 CMTS Buffer Control - Downstream Service Flows

At this time, no CMTS supports Buffer Control for downstream service flows. As a result, no testing could be performed.

8 RECOMMENDATIONS AND CONCLUSIONS

The overall recommendation on usage of CM Buffer Control is for the operator to independently validate performance with their own CM/CMTS equipment and their own configuration files. That said, the test results presented in this report could be used as starting points upon which to perform that validation.

Table 2 - Recommended Settings for CM Buffer Control (Upstream Service Flows)

Upstream Rate	Target Buffer Size
1 Mbps	8 KiB
2 Mbps	16 KiB
5 Mbps	32 KiB
10 Mbps	50 KiB

These values provide sufficient buffering for upstream TCP applications that use multiple simultaneous TCP connections to reach the provisioned upstream rate. Further, they provide sufficient buffering for upstream "single session" TCP applications to reach the provisioned rate when the RTT is less than 65 ms, 65 ms, 52 ms, or 40 ms for the four rates, respectively (calculated using BDP).

These values correspond to the peak OTT VoIP MOS scores achieved in our testing, and are as low as possible in order to maximize web browsing performance.

It should be noted that Service Flows that are intended to be optimized to carry very latency-sensitive traffic (such as interactive gaming traffic) will likely benefit from somewhat smaller buffer sizes. On the other hand, cable modems deployed in locations for which the average RTT to commonly used upload servers is significantly greater than the values mentioned above will likely benefit from somewhat larger buffer sizes.

For the majority of use cases, the MSO should not include the Minimum Buffer Size or Maximum Buffer Size TLVs in the configuration file. These TLVs are intended for cases where buffering needs to be strictly controlled, and where it would be preferable for the CM to reject the configuration rather than provide a buffer size outside of the acceptable range. One example of such a case might be a commercial account where the upstream buffer size is critical for meeting an established Service Level Agreement.

Appendix I Round Trip Times

The tables below show round-trip ping times (in milliseconds) from three locations served by residential cable modems to the top 15 US websites (as ranked by quantcast.com on 8/23/2011) along with 5 selected sites that are upload-centric.

Table 3 - Ping Round-trip times from Louisville, CO

rank	website	url	min	mean	max	std
1	google	www.l.google.com	59.418	66.438	95.109	11.577
2	facebook	www.facebook.com	76.679	84.106	109.776	12.423
3	youtube	youtube-ui.l.google.com	59.941	64.381	80.176	7.077
4	yahoo	any-fp3-real.wa1.b.yahoo.com	27.541	31.480	49.308	6.427
5	twitter	www.twitter.com	49.896	54.153	68.763	6.731
6	msn	*no response*	-	-	-	-
7	amazon	*no response*	-	-	-	-
8	wikipedia	text.pmtpa.wikimedia.org	96.023	102.138	113.864	5.706
9	ebay	*no response*	-	-	-	-
10	live.com	*no response*	-	-	-	-
11	microsoft	*no response*	-	-	-	-
12	blogspot	blogger.l.google.com	78.074	85.050	111.171	12.247
13	bing	a134.b.akamai.net	52.567	54.095	58.095	1.663
14	ask.com	a1230.b.akamai.net	48.803	57.630	105.847	17.178
15	blogger	blogger.l.google.com	78.492	80.977	94.627	4.591
	flikr	any-rc.a01.yahoodns.net	44.963	51.893	83.761	11.553
	picasa	www2.l.google.com	60.819	65.974	86.021	8.515
	carbonite	carbonite.com	65.278	66.609	67.667	0.785
	dropbox	v-www.sjc.dropbox.com	44.682	46.935	48.379	0.948
	mozy	mozy.com	52.311	61.992	92.663	14.519
All Sites from Louisville, CO			**27.541**	**64.923**	**113.864**	

Table 4 - Ping Round-trip times from Philadelphia, PA

rank	website	url	min	mean	max	std
1	google	www.l.google.com	25.506	29.832	39.677	3.651
2	facebook	www.facebook.com	104.828	111.788	151.402	13.298
3	youtube	youtube-ui.l.google.com	19.256	22.502	30.712	3.021
4	yahoo	any-fp3-real.wa1.b.yahoo.com	16.373	17.877	19.220	0.763
5	twitter	www.twitter.com	99.981	103.825	106.229	2.001
6	msn	*no response*	-	-	-	-
7	amazon	*no response*	-	-	-	-
8	wikipedia	text.pmtpa.wikimedia.org	40.476	45.755	48.909	2.505
9	ebay	*no response*	-	-	-	-
10	live.com	*no response*	-	-	-	-
11	microsoft	*no response*	-	-	-	-
12	blogspot	blogger.l.google.com	19.437	24.126	34.271	4.509

rank	website	url	min	mean	max	std
13	bing	a134.b.akamai.net	17.319	23.413	28.705	3.294
14	ask.com	a1230.b.akamai.net	16.790	21.340	25.292	2.620
15	blogger	blogger.l.google.com	18.428	22.739	28.641	3.632
	flikr	any-rc.a01.yahoodns.net	16.031	19.879	24.671	2.987
	picasa	www2.l.google.com	29.507	35.926	40.926	3.393
	carbonite	carbonite.com	41.583	47.811	51.499	2.920
	dropbox	v-www.sjc.dropbox.com	90.192	95.050	99.476	3.150
	mozy	mozy.com	89.233	100.644	134.534	12.585
All Sites from Philadelphia, PA			**16.031**	**48.167**	**151.402**	

Table 5 - Ping Round-trip times from Boston, MA

rank	website	url	min	mean	max	std
1	google	www.l.google.com	32.095	33.930	41.913	2.713
2	facebook	www.facebook.com	20.665	21.785	25.354	1.321
3	youtube	youtube-ui.l.google.com	15.564	17.523	23.248	2.183
4	yahoo	any-fp3-real.wa1.b.yahoo.com	20.583	22.075	26.173	1.491
5	twitter	www.twitter.com	112.189	114.093	119.203	2.052
6	msn	*no response*	-	-	-	-
7	amazon	*no response*	-	-	-	-
8	wikipedia	text.pmtpa.wikimedia.org	46.592	48.810	52.118	1.709
9	ebay	*no response*	-	-	-	-
10	live.com	*no response*	-	-	-	-
11	microsoft	*no response*	-	-	-	-
12	blogspot	blogger.l.google.com	22.111	23.276	26.561	1.211
13	bing	a134.b.akamai.net	21.534	23.239	26.544	1.756
14	ask.com	a1230.b.akamai.net	21.068	23.456	26.592	1.890
15	blogger	blogger.l.google.com	21.117	24.711	39.893	5.172
	flikr	any-rc.a01.yahoodns.net	21.741	22.439	24.095	0.636
	picasa	www2.l.google.com	15.148	17.998	24.711	3.231
	carbonite	carbonite.com	41.564	44.163	48.843	2.562
	dropbox	v-www.sjc.dropbox.com	97.185	99.011	107.236	2.777
	mozy	mozy.com	93.630	97.517	107.675	3.856
All Sites from Boston, MA			**15.148**	**42.268**	**119.203**	

Appendix II Acknowledgements

This DOCSIS Guidelines document was the result of a joint effort between CableLabs and Comcast. CableLabs wishes to heartily thank the following individuals and their organizations that contributed to drafting this document.

Richard Woundy, Yiu Lee, Carl Williams, Mehdi Nikkhah	Comcast
Joey Padden , Bart Brassell	CableLabs

Greg White, CableLabs