# Active Queue Management in DOCSIS® 3.1 Networks

*Greg White*

## ABSTRACT

An important new feature in the DOCSIS 3.1 specification is active queue management (AQM). AQM provides a solution to the problem of providing good application layer quality of experience when multiple applications share a network connection. The need for AQM arises due to the presence of packet buffering in network elements and the mechanics of the TCP congestion avoidance algorithm. Based on simulated performance and implementation considerations, a variant of the Proportional Integral Controller Enhanced (PIE) algorithm, called DOCSIS-PIE, is now included in DOCSIS 3.1 and has recently been added to the DOCSIS 3.0 specification as well. Implementation of DOCSIS-PIE is mandatory for implementation in DOCSIS 3.1 cable modems, and recommended for implementation in DOCSIS 3.0 cable modems. In addition to the mandatory/recommended algorithm, DOCSIS 3.1 and 3.0 cable modem vendors are free to support additional AQM algorithms of their choosing. For managing downstream traffic, the DOCSIS 3.1 specification mandates that the CMTS support an AQM technology, but does not require a specific algorithm. This article discusses the underlying problem that is addressed by AQM, the selection of DOCSIS-PIE as the algorithm of choice for DOCSIS 3.1 cable modems, and the expected performance of the algorithm in simulated network conditions.

## INTRODUCTION

### WHY IS LATENCY IMPORTANT?

Packet forwarding latency can have a large impact on the user experience for a variety of network applications. The applications most commonly considered as latency-sensitive are real-time interactive applications such as voice over Internet protocol (VoIP), video conferencing, and networked "twitch" games such as first-person shooter titles. However, other applications are sensitive as well; for example, web browsing is surprisingly sensitive to latencies on the order of hundreds of milliseconds.

There are established models for the degradation in user experience for VoIP caused by latency. In the model used for estimating VoIP quality in this article [1], latency has a minimal impact on the VoIP mean opinion score (MOS) (approximately 0.005 MOS points per 20 ms of latency) as long as one-way latency remains below 177 ms. Beyond the threshold of 177 ms, each additional 20 ms of latency reduces VoIP quality MOS score by a more significant 0.13 MOS points.

While online games do not have similar well vetted models for the impact that network parameters have on user experience, a number of researchers have studied the topic, and some data exists to indicate that access network latencies should be kept below 20 ms in order to provide a good user experience.

Loading a web page involves an initial HTTP GET method to request the download of an HTML file, which then triggers the download of dozens or sometimes hundreds of resources that are then used to render the page. While many servers may be involved in providing the page contents, generally speaking, the majority of the resources are served from a small number of servers (four or five). Web browsers will typically fetch the resources from each server by opening up multiple (typically six) TCP connections to the server and requesting a single resource via each connection. Once each individual resource is received, the browser will close the TCP connection and open a new one to request the next resource, thus keeping the same number of connections open at a time. The result of this hybrid parallel-serial download is that the page load time is in some cases driven by the serial aspect, that is, the number of sequential downloads (one completing before the next can start), of which there may be a dozen or more. Round-trip latency can impact the page load time due to the fact that completion of each resource download is delayed by any additional round-trip time (RTT) in the network. Thus, increases in the effective RTT between the client and the server(s) can increase page load time by 10×–20× that amount.

## MEGABITS MYTH?

Contrast the above with the sensitivity of page load time to link bandwidth. At the rates cable modem customers are getting today, web page

*The author is with Cable Television Laboratories, Inc. (CableLabs).*


0163-6804/15/$25.00 © 2015 IEEE

load times have reached the point of diminishing returns. In fact, any improvement in link rate beyond about 6 Mb/s returns almost imperceptible improvements in page load time.

Similarly, many other network applications operate at data rates well below what is commonly provisioned for cable modem service.

There is a lot of focus on bandwidth: it is the top-line number that has been used to market high-speed data service. For the foreseeable future that will probably be the case, but when it comes down to the user experience for the actual applications broadband customers are using, improvements in latency may be more important at this point than improvements in bandwidth.

Some sources of latency are hard to address. There is the propagation delay from the user to the server or, for a VoIP session, between two users. There is not much that can be done about the speed of light, but routing paths can be made as short as possible, and content delivery networks can reduce the physical distance and number of hops for some content.

On the other hand, there is a significant issue that has gained a lot of attention in technical circles in the past few years pointing to the fact that a lot of network elements have more buffering memory in them than is really good for application performance.

## "Bufferbloat"

Every network element supports buffering of some amount of packets that are destined to be forwarded on the next link. This buffering is important to ensure good utilization of the network link, especially in cases where the incoming traffic rate exceeds the outgoing link rate. In these bottleneck situations, the buffer serves to absorb high-rate traffic bursts so that they can then be played out on the slower outgoing link. Without buffering, most of the packets in the high-rate burst would simply be dropped.

From the perspective of egress link utilization, larger buffers reduce the chance that the egress link will go idle. For bulk TCP traffic (file transfers), user experience is driven by how quickly the file transfer can complete, which is directly related to how effectively the protocol can utilize the network links, again supporting the view that more buffering is better.

But the downside to large buffers is that they result in excessive latency. While this is not an issue for bulk file transfers, it is clearly an issue for other traffic, and the issue is exacerbated by TCP itself.

The majority of TCP implementations use loss-based congestion control, which means the TCP ramps up its congestion window (effectively ramping up its sending rate) until it sees packet loss, cuts its congestion window in half, and then starts ramping back up again until it sees the next packet loss, and that saw-tooth continues. In a lot of networks, especially wired networks, packet loss does not come from noise on the wire. It comes from buffers being full, and when a packet arrives at a full buffer it has to be discarded. This is how TCP automatically adjusts its transmission rate to match the available capacity of the bottleneck link.

The result of this saw-tooth behavior being driven by buffer exhaustion is that the buffer at the head of the bottleneck link is going to saw-tooth between partially full and totally full. Depending on the particular flavor of TCP congestion control (Reno, New Reno, CUBIC, etc.) the portion of time spent in the full (or nearly full) state will vary, and if there are multiple TCP sessions sharing that bottleneck link, the average buffer occupancy will increase. Furthermore, if the buffer is oversized, its average occupancy will be higher as well.

In DOCSIS networks, the cable modem is generally at the head of the bottleneck link for upstream traffic. Historically, and still typically today, cable modems have had a much bigger buffer than is needed to keep TCP working smoothly. Dischinger *et al.* [2] measured buffering latencies on the order of 2–3 s in deployed broadband modems.

Those two factors together — the modem being at the head of the bottleneck link and having an oversized buffer — plus the fact that TCP is going to try to keep that buffer full, results in high upstream latency through the modem whenever there is an upstream TCP session.

The term "Bufferbloat" [3] has been coined to refer to the practice (sometimes inadvertent) of sizing network buffers to be significantly greater than needed to ensure good link utilization, and the resulting significant degradation of interactive applications in the presence of concurrent TCP traffic.

The result of Bufferbloat is that applications other than upstream TCP suffer. Even though the other applications might be low bandwidth, and TCP will back off to accommodate them on the link, their packets arrive to a full or nearly full buffer that may take hundreds of milliseconds or even seconds to play out. This can make web browsing perform poorly, and make VoIP, video chat, or online games unusable. In addition, this could potentially affect downstream TCP performance as well, since the upstream TCP acknowledgments would experience similar latencies. However, this last effect was identified some time ago, and as a result all cable modems have for years supported some kind of TCP acknowledgment prioritization scheme that allows upstream TCP acknowledgments to bypass the large queue.

One reason this situation has persisted in DOCSIS cable modems is that modems, going back to those compliant with the DOCSIS 1.1 specification, have supported multiple service flows. The presumption on the part of modem developers has been that if operators are concerned about latency for certain traffic flows, they can create a separate service flow (which provides a separate buffer) to carry that traffic. Unfortunately, this is not a feasible solution in the vast majority of cases.

## Active Queue Management

As awareness of the topic of Bufferbloat has risen, so too has interest in methods to resolve it. Active queue management (AQM) appears to be the most promising approach because significant network-wide benefits can be derived by

*The result of Bufferbloat is that applications other than upstream TCP suffer. Even though the other applications might be low bandwidth, and TCP will back off to accommodate them on the link, their packets arrive to a full or nearly full buffer that may take hundreds of milliseconds or even seconds to play out.*

| N | F1 | F2 | Fs | W | VG | C | T |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | — | 1 | 1 | 0 | 0 |
| 2 | 1 | 1 | infinite | 1 | 1 | 0 | 0 |
| 3 | 3 | 3 | 5MB | 1 | 1 | 0 | 0 |
| 4 | 3 | 3 | 5MB | 1 | 1 | 6 | 0 |
| 5 | 1 | 0 | 27MB* | 1 | 1 | 0 | 0 |
| 6 | 3 | 3 | 5MB | 4 | 4 | 0 | 0 |
| 7 | 3 | 3 | 5MB | 4 | 4 | 6 | 0 |
| 8 | 5 | 5 | 0.25MB | 4 | 4 | 6 | 0 |
| 9 | 3 | 3 | 5MB | 4 | 1 | 0 | 100 |

N: traffic load index
F1: number of simultaneous FTP uploads with 20ms RTT
F2: number of simultaneous FTP uploads with 100ms RTT
Fs: FTP file size
W: number of simultaneous web users
VG: number of simultaneous VoIP/gaming sessions C: CBR data rate (Mb/s)
T: number of torrent (LEDBAT) connections
*Filesize and repetition pattern chosen to exercise DOCSIS "powerboost" feature

**Table 1.** Traffic scenarios.

implementing it in a relatively small number of bottleneck network elements (e.g., broadband modems). Current AQM approaches seek to detect the "standing queue" created by TCP and, once detected, send TCP a congestion signal (by dropping a packet). The modern algorithms do this without the need to be tuned for network conditions.

## DOCSIS 3.1 Active Queue Management

From June 2013 through January 2014, CableLabs worked with the developers of DOCSIS equipment to define an AQM algorithm that would be mandatory for implementation of a cable modem compliant with the DOCSIS 3.1 specification. The DOCSIS 3.1 AQM Working Group evaluated several existing candidate algorithms (extending one of these to improve performance) and two new algorithms developed by CableLabs [6].

Among the candidate algorithms, a version of the Proportional Integral Controller Enhanced (PIE) algorithm [4] optimized for implementation in DOCSIS cable modems was the most attractive candidate due to implementation complexity and alignment with the DOCSIS 3.0/3.1 medium access control layer.

PIE has a distinct advantage over the other algorithms in that the most important parts of the algorithm lend themselves to implementation in software in DOCSIS 3.1 cable modems. This has a couple of advantages. One advantage is that it reduces the development risk for each DOCSIS 3.1 cable modem silicon vendor, since the algorithm does not need to be extensively tested prior to taping out the system on a chip. Another is that it reduces risk for the DOCSIS 3.1 platform in that it allows the algorithm to be modified in the future in devices that are in the field. An additional benefit of PIE is that the algorithm has the potential to be implemented in existing DOCSIS 3.0 cable modems.

CableLabs' DOCSIS 3.1 specification [7] mandates that cable modems implement a specific variant of the PIE AQM algorithm. This specific variant is referred to as DOCSIS-PIE [5]. CableLabs' DOCSIS 3.0 specification has been amended to recommend that cable modems implement the same algorithm. Both specifications allow that cable modems can optionally implement additional algorithms that can then be selected for use by the operator via the modem's configuration file. These requirements on the cable modem apply to upstream transmissions.

Both specifications also include requirements (mandatory in the DOCSIS 3.1 specification and recommended in the DOCSIS 3.0 specification) that the cable modem termination system (CMTS) implement active queue management for downstream traffic; however, no specific algorithm is defined for downstream use.

## Performance Simulation

To illustrate the performance benefits expected from AQM in DOCSIS 3.1 networks, simulations have been performed using the Network Simulator (ns2). These simulations focus on scenarios that are anticipated to be particularly relevant for DOCSIS 3.1 deployments in the 2016 timeframe.

### Service Model

The configured data rates for the service are extrapolated for 2016 as follows (parameters defined in [7]).

Upstream:
- Maximum sustained traffic rate (MSR): 20 Mb/s
- Maximum traffic burst: 3 MB
- Peak traffic rate (burst rate): 25 Mb/s

Downstream:
- MSR: 100 Mb/s
- Maximum traffic burst: 33 MB
- Peak traffic rate (burst rate): 150 Mb/s

## Traffic Models

### Broadband Usage Scenarios

Broadband user activity is modeled using nine traffic loads as shown in Table 1. These traffic loads are skewed toward representing peak upstream usage times for a user, rather than being representative of the broadest range of activity levels.

In the web user model the client fetches a single file (representing the html file) and then upon completion of this file transfer proceeds to download 100 resources (of log-normally distributed size) that are spread evenly across four servers. The client maintains six active TCP

connections to each server until all 25 resources have been requested from that server. Extrapolating from http://httparchive.org/trends.php for 2016, the total page size (sum of all 101 resources) was configured to be 3.8 MB. The web user downloads the web page, waits 5 s, and then repeats. The metric of interest here is page load time, calculated from the initiation of the TCP connection to download the initial file, to the completion of the TCP session that downloads the final resource. Notably, the web model does not include the Domain Name Service (DNS) lookups that would be present in many real-world page loads. While the number of DNS lookup packets is extremely small relative to the total number of packets involved in loading a page, they are very sensitive to packet loss. The retry timeout for DNS clients is commonly 5 s, so a single DNS packet loss would increase the page load time by approximately that amount.

A single traffic type is used to model both VoIP and online gaming. This traffic type consists of UDP packets of 218 bytes at 50 pkt/s. Packet loss and per-packet latency is monitored, and from these a VoIP MOS score is estimated using a derivation of the International Telecommunication Union (ITU) E-Model for G.711 [1], under the assumption of a 60 ms de-jitter buffer at the receiver and 20 ms of latency outside of the access network.

The upstream torrent traffic is rate-limited as an aggregate to 50 percent of the upstream MSR as an approximation of typical client behavior.

### Model Used for TCP Performance Metrics

To assess performance of TCP applications, a model somewhat inspired by Ookla Speedtest.net was used. This scenario is an interesting one because it both represents a commonly utilized methodology for assessing TCP performance in real networks, and is a common scenario in which the user's experience is directly driven by the average TCP upload throughput. In many other upload cases (email, cloud storage and cloud backup, etc.) uploads happen more or less in the background, with the user's interaction (when there is direct interaction) ending with the initiation of the upload task (e.g. clicking "send" on the email message) rather than on its completion.

In the Ookla test, upstream TCP throughput is measured via the use of two simultaneous upstream TCP sessions that transfer data for a total of approximately 10 s. The closest server is chosen by default, but the user can select to run a test to any server in the world. Correspondingly, the simulation uses two TCP sessions, but they are not terminated at 10 s; instead, they are allowed to continue for up to 100 s. Four values of RTT are simulated (20 ms, 50 ms, 100 ms, and 200 ms). The data point from the simulation set that most closely represents typical Ookla throughput results is the average throughput for a 10 s transfer using an RTT of 20 ms, but the other results provide interesting insights into other file transfer conditions.

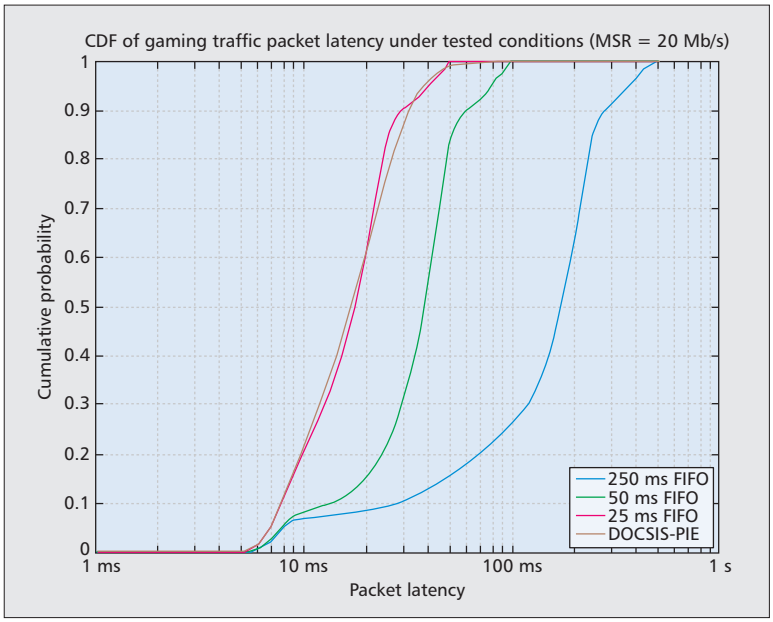


**Figure 1.** Packet latency statistics for VoIP/gaming traffic.

## Shared Channel Congestion Models

Congestion of the shared DOCSIS channel by other users is modeled in order to examine the ability of AQM to respond to changes in available link capacity:

- No congestion: Channel capacity exceeds the peak traffic rate of 25 Mb/s.
- Light congestion: Channel capacity varies among 16.5, 20, 22.5, and 25 Mb/s.
- Moderate congestion: Channel capacity varies among 18.5, 19, 20, and 22.5 Mb/s.
- Heavy congestion: Channel capacity varies among 10, 12, 18, and 20 Mb/s.

For each congestion case, the channel capacity remains at each value for 10 s, and changes in a repeating pattern that exercises all 12 possible rate transitions.

## Downstream Queuing

The new CMTS requirements include mandatory support for AQM for DOCSIS 3.1 CMTS and recommended support of AQM for DOCSIS 3.0 CMTS, but no specific algorithm is required. For purposes of this simulation, downstream traffic sees a drop-tail queue at the CMTS with 100 ms of buffering at the MSR (1.25 MB of buffering).

## Upstream Queuing

To illustrate the performance of the DOCSIS-PIE AQM algorithm, it is compared against three simple first-in first-out (FIFO) queue configurations. The FIFO queue is representative of earlier generations of DOCSIS modems (e.g., DOCSIS 3.0) or of a situation in which the operator has disabled AQM on a particular service flow. In all, four queuing scenarios are modeled.

**250 ms FIFO**: In the first of the three FIFO configurations, the cable modem supports a

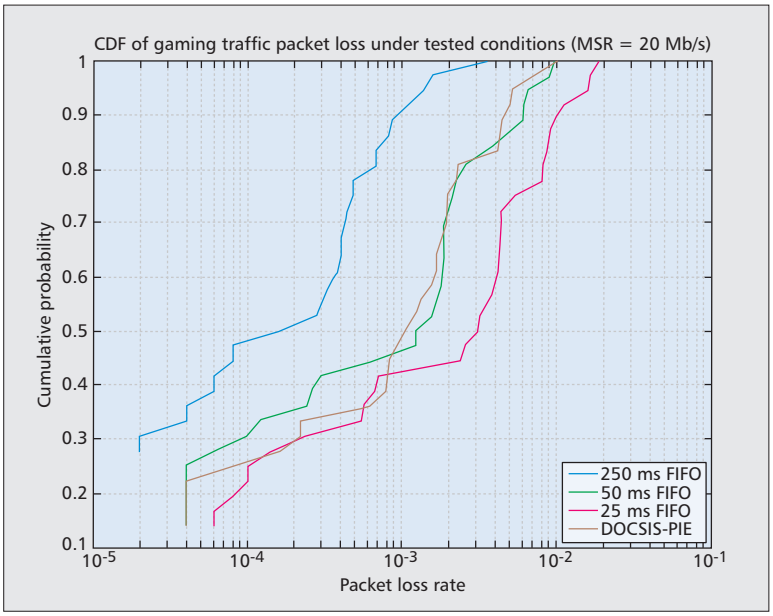


**Figure 2.** Packet loss statistics for VoIP/gaming traffic.

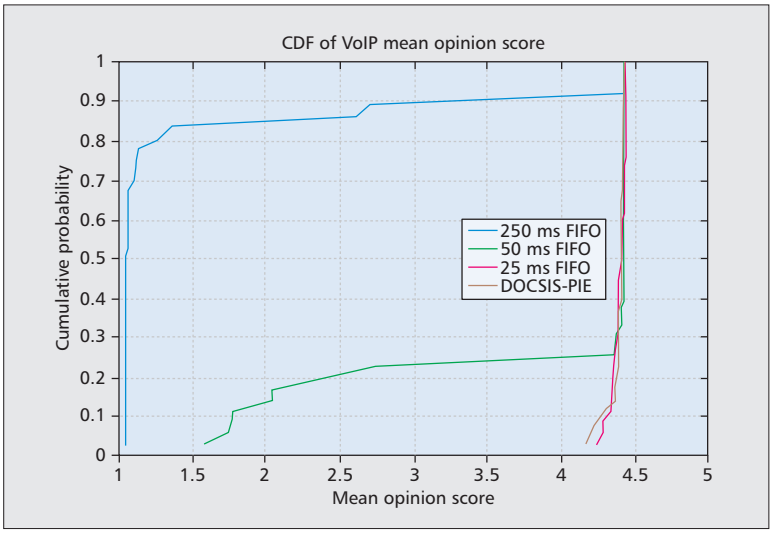


**Figure 3.** VoIP MOS statistics.

buffer size that is equivalent to 250 ms at the MSR, or 625 kB. This buffer size approximates (or perhaps even understates) the default buffering condition present in a DOCSIS 3.0 modem.

**50 ms FIFO**: The second FIFO configuration is one in which the cable modem implements a buffer size equivalent to 50ms at the MSR (125 kB). This models a case where the operator has explicitly configured the DOCSIS 3.0 upstream buffer size to a more appropriate size for the service offering.

**25 ms FIFO**: The third FIFO configuration is one in which the cable modem implements a buffer sized to be 25 ms at the MSR (62.5 kB). This value was chosen to give the closest match to the gaming latency performance of the DOCSIS-PIE algorithm, and could represent a DOCSIS 3.0 service that is optimized for online gaming.

**DOCSIS-PIE**: The fourth option represents the default condition for a DOCSIS 3.1 cable modem, in which the DOCSIS-PIE AQM algorithm is enabled.

## Application Performance

### VoIP/Gaming Traffic Performance

Online rapid action games, such as first-person shooter or sports games, require low latency in order to provide a good user experience. In addition, excessive packet loss can impact performance as well.

Figure 1 shows aggregated packet buffering latency results for simulations of the four queuing options in the various network and traffic load scenarios. As can be seen, both the 25 ms FIFO and DOCSIS-PIE options give good (and nearly identical) latency performance for gaming traffic, achieving a median latency of less than 20 ms and a 90th percentile packet latency of approximately 30 ms. The other FIFO options (50 ms buffer and 250 ms buffer) would provide a marginal or poor user experience for games that require low latency.

Figure 2 shows the statistics of packet loss for gaming traffic. While none of these options appear to show excessive packet loss, the DOCSIS-PIE algorithm does excel relative to the 25 ms FIFO option by dropping approximately 1/3 as many packets.

In terms of VoIP user experience, recall that MOS is a 5-point scale with the following values: 5–excellent; 4–good; 3–fair; 2–poor; 1–bad. The MOS estimator used for this analysis assumes a G.711 codec with a maximum MOS of approximately 4.4, and takes into account the latency, jitter, and packet loss experienced by the stream.

The MOS estimator shows that DOCSIS-PIE AQM and 25 ms FIFO both provide excellent performance, whereas the longer FIFO queues result in unacceptable audio quality in certain cases. In fact, the 250 ms FIFO only results in good audio quality in traffic scenario 1 (no TCP upload traffic).

## Web Performance

Figure 4 shows the statistics of web page load time. Both 25 ms FIFO and DOCSIS-PIE provide very good web performance, with 90th percentile page load times of approximately 3 s. The 50 ms FIFO also performs fairly well, with a 90th percentile page load time of less than 4 s.

## TCP Throughput Performance

In terms of TCP throughput, all of the queuing approaches provide good performance when the session has a short RTT. As the RTT increases, performance degradation can be seen, particularly with the 25 ms FIFO scenario.

Figure 5 shows the averaged TCP performance over moderately short timescales for the four values of RTT. In many locales, and in particular in the United States and other developed regions, RTTs in the 10–50 ms range are likely the most common for upload RTT (due in part

to the proliferation of geographically distributed data centers).

As mentioned above, the data point that most closely represents the result one would expect from using speedtest.net is the average throughput at 10 s in the 20 ms RTT case. At that data point, all of the queuing algorithms perform nearly identically. In the longer RTT cases (50ms, 100ms, 200ms) the 25 ms FIFO results show markedly degraded performance relative to the other queuing options.

It is worth noting that the 200ms RTT case has a high bandwidth-delay product (BDP) of 500 kB, equivalent to over 330 packets. When in congestion avoidance, a single Reno-based TCP session (e.g. Windows or Mac OS X) will take a long time to recover from a congestion window decrease. For example, in a case where 50 ms of buffering results in a packet drop, the congestion window will drop from ~400 packets to ~200 packets, and then will take 40 s to recover. A small number of unlucky packet drops can thus have a significant effect on throughput in this case. More advanced TCP implementations such as cubic should recover much more quickly in this situation.

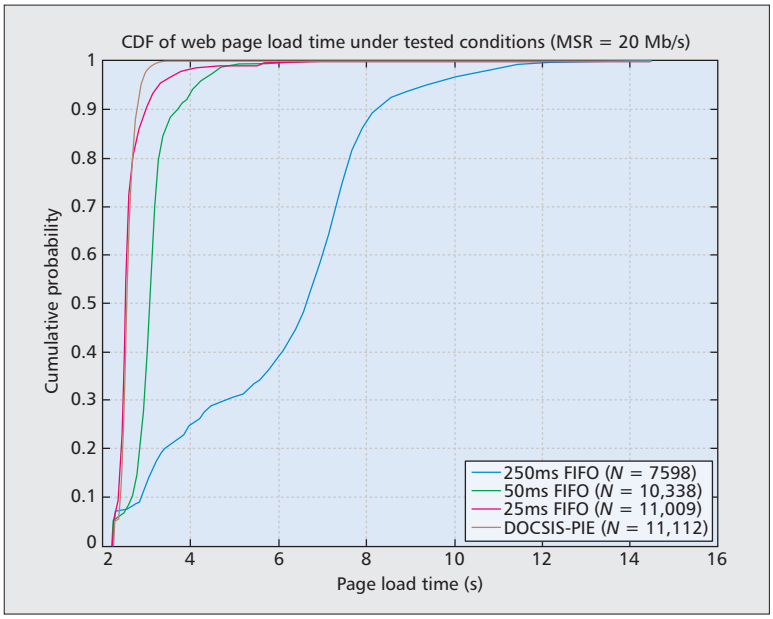


**Figure 4.** Web page load time statistics.

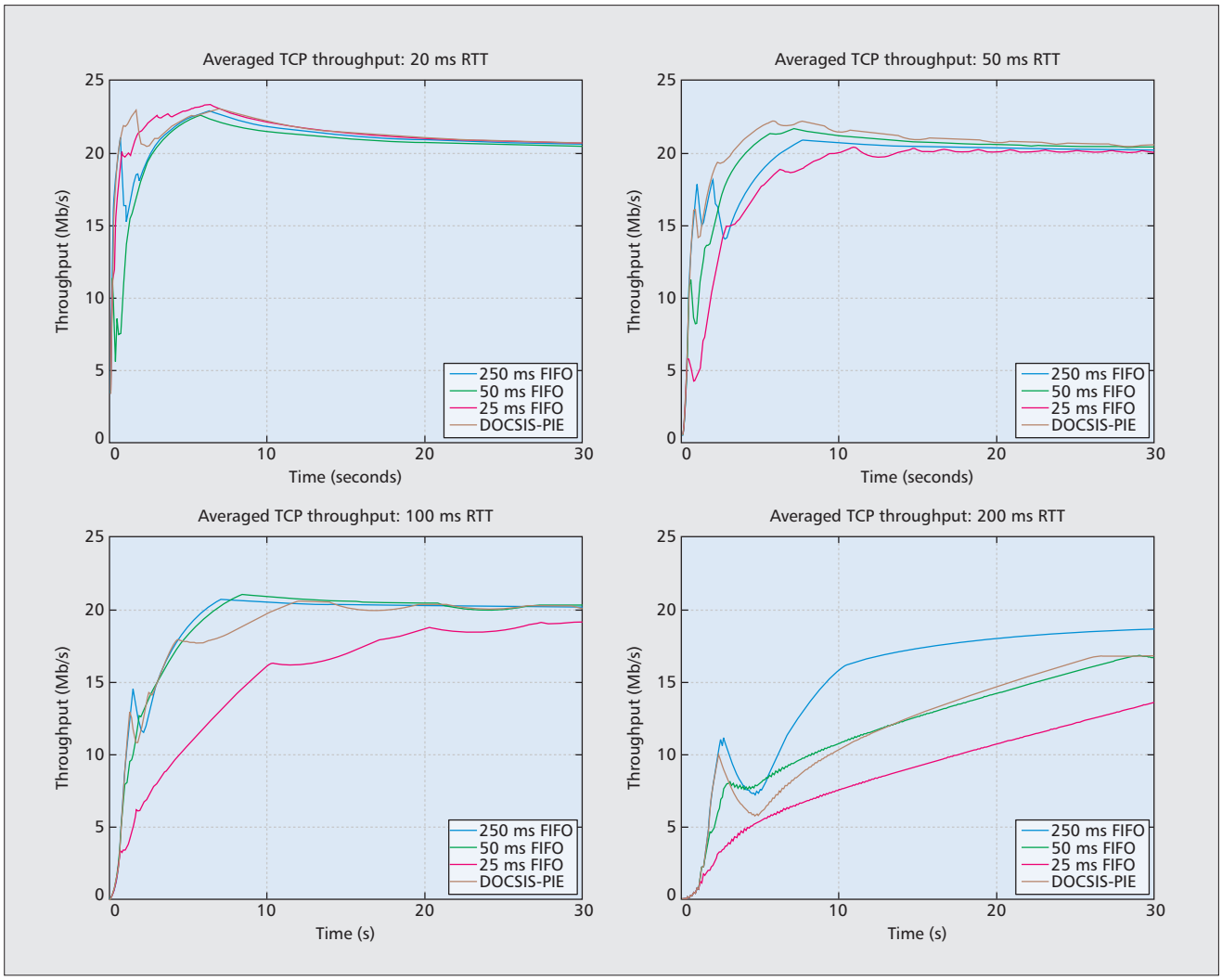


**Figure 5.** TCP performance

*The AQM algorithm implemented on DOCSIS 3.1 cable modems will reduce upstream latency (in loaded conditions) from hundreds or thousands of milliseconds down to tens of milliseconds. In addition, AQM algorithms on the CMTS will do the same for downstream traffic.*

## Performance Summary

DOCSIS-PIE is shown to provide VoIP quality, gaming latency, and web page load time performance that meet or exceed those provided by 25 ms FIFO, but with approximately 1/3 of the packet loss rate, and without the severe degradation of TCP performance caused by the short FIFO. Moreover, the DOCSIS 3.1 default configuration significantly outperforms the default DOCSIS 3.0 configuration on nearly every measure.

For other protocols and applications beyond those tested, extrapolations from this data can be made. Applications that support loss-based rate control similar to TCP (e.g. TCP-friendly rate control or Stream Control Transmission Protocol) would be expected to have similar average throughput performance with AQM as without.

## Conclusion

The DOCSIS 3.1 specification brings an assortment of new technologies and capabilities to cable data networks that will serve to bring major improvements in the performance and reliability of broadband cable Internet service.

By improving the latency performance of broadband connections, the active queue management functionalities in DOCSIS 3.1 equipment will have the potential to substantially improve the user experience for interactive applications.

The AQM algorithm implemented on DOCSIS 3.1 cable modems will reduce upstream latency (in loaded conditions) from hundreds or thousands of milliseconds down to tens of milliseconds. In addition, AQM algorithms on the CMTS will do the same for downstream traffic. The result will be a significant reduction in sluggish web browsing performance, and much enhanced reliability for online games and audio and video telephony applications.

## References


[1] R. G. Cole and J. H. Rosenbluth, "Voice over IP Performance Monitoring," *ACM CCR*, vol. 31, no. 2, 2001.

[2] M. Dischinger *et al.*, "Characterizing Residential Broadband Networks," *Proc. 7th ACM SIGCOMM Conf. Internet Measurement*, Oct. 24–26, 2007, San Diego, CA, http://broadband.mpi-sws.org/residential/.

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

[4] R. Pan *et al.*, "PIE: A Lightweight Control Scheme to Address the Bufferbloat Problem," IETF Internet draft, https://datatracker.ietf.org/doc/draft-pan-aqm-pie/

[5] G. White and R. Pan, "A PIE-Based AQM for DOCSIS Cable Modems," IETF Internet draft, https://datatracker.ietf.org/doc/draft-white-aqm-docsis-pie/

[6] G. White, "Active Queue Management in DOCSIS 3.X Cable Modems," May 2014, http://www.cablelabs.com/wp-content/uploads/2014/06/DOCSIS-AQM_May2014.pdf.

[7] Cable Television Laboratories, Inc., Data Over Cable Service Interface Specification, MAC and Upper Layer Protocol Interface Specification v. 3.1, http://www.cablelabs.com/wp-content/uploads/specdocs/CM-SP-MULPIv3.1-I03-1406101.pdf.


## Biography


Greg White [M'91] (g.white@cablelabs.com) obtained his B.S. degree from Carnegie Mellon University in 1992 and his M.S. degree from the University of Wisconsin — Madison in 1994, both in electrical engineering. From 1995 to 1999, he held various positions at Motorola Land Mobile Products Sector Research and Motorola Labs. In 1999 he joined Cable Television Laboratories, Inc. (CableLabs) and served as the lead architect of the DOCSIS 2.0 and DOCSIS 3.0 specifications. He is presently a Distinguished Technologist at CableLabs, and his current area of interest is in developing advanced technology for improving broadband network performance and user experience.