International Journal of Current Engineering and Technology
E-ISSN 2277 – 4106, P-ISSN 2347 - 5161

Available at http://inpressco.com/category/ijcet

Research Article

Controlling Queue Delay (CoDel) to counter the Bufferbloat Problem in Internet

Tanvi SharmaÀ*

Department of Information Technology, U.I.E.T., Panjab University, Chandigarh, India-160014

Accepted 05 June 2014, Available online 20 June 2014, **Vol.4, No.3 (June 2014)**

Abstract

Network management and control is a complex problem, which is becoming even more challenging with the increased demand to use the internet for time/delay-sensitive applications with differing QoS requirements. The existing TCP congestion avoidance and control mechanisms are insufficient to provide good service in all conditions. The TCP has successfully governed the Internet congestion control for two decades. It is by now, however, recognized that TCP has started to reach its limits and that new congestion control protocols are needed in the near future. This has spurred an intensive research effort searching for new congestion control designs that meet the demands of a future Internet. Despite massive awareness and research, the full buffer problem has not gone away, but on the contrary has become worse. Therefore, there has been an increased interest in exploring AQM in Internet routers so as to reduce the queue latency. This research describes a recently developed AQM, Controlled Delay (CoDel) algorithm, designed to work in modern network settings and can be deployed as a major part of the solution to bufferbloat. Effectiveness of CoDel is tested by carrying out simulations in ns-2 and comparing its performance with two other promising network queue management algorithms, RED and DropTail. The CoDel AQM displays extremely good performance in most bufferbloat traffic scenarios.

Keywords: *Active Queue Management, Bufferbloat, CoDel, Congestion control, Random Early Detection, DropTail, Queues.*

1. Introduction

Wireless networks are playing a major role in the area of communication. Nowadays we are using wireless networks in military applications, industrial applications and even in personal area networks. With the advancement in technologies and relatively low cost, there is a rapid rise in the use of personal communication devices. These devices easily get access to network through wireless interfaces. The demand for more bandwidth with the advent of network intensive internet applications has made router buffer management a high priority. Some buffering is needed for proper functioning of the network, to absorb bursts of traffic and ensure links are utilized to their full capacity. Consequently, the routers have to be designed in such a way that they can withstand large queues in their buffers in order to accommodate for transient congestion.

At present, the Transport Control Protocol (TCP) detects congestion only after a packet has been dropped. Because of the increasing high speed networks it is important to have high throughput and keep the average queue sizes low. The term bufferbloat has been coined to describe the problem that occurs when computer network buffers (queues) misbehave, inducing unnecessary latency. Many algorithms of Active Queue Management tend to reduce the effects of bufferbloat. The CoDel mechanism discussed in this research, proposed by Kathleen Nichols and Van Jacobson, is designed to provide a no-knobs approach to queue management to overcome bufferbloat (Nichols *et al*, 2012). CoDel has implementation advantages over other AQMs as it does nearly all of its work at dequeue stage (when packets are transmitted).

2. Related Work

Y. Gong *et al*. (2014) suggested that some researchers indicate active queue management (AQM) as the solution to the problem of bufferbloat and others suggest end-to-end low-priority congestion control (LPCC) techniques. CoDel is an example of AQM and LEDBAT for LPCC. In this paper, AQM and LPCC techniques were combined. An extended set of experiments were conducted on both controlled testbeds and on the Internet to show the problem to hold in the real world for any tested combination of AQM policies and LPCC protocols.

Dipesh M. Raghuvanshi *et al*. (2013) indicated that Routers presently using Passive Queue Management (PQM) mechanisms merely have any control over the queue occupancy which increases the queue latency. The authors studied the effectiveness of CoDel by carrying out simulations in ns-2 and comparing its performance with existing AQM mechanisms in variety of Internet setups.

*Corresponding author: **Tanvi Sharma**

Based on the simulation results, the advantages and shortcomings of CoDel were discussed.

Naeem Khademi *et al*. (2013) indicated that delays on the order of seconds have become common due to the placement of unreasonably-sized FIFO/Drop-Tail buffers at the edge of many networks. CoDel and PIE, in spite of recently been presented and discussed at the IRTF and the IETF have not yet been thoroughly evaluated or compared against each other except by simulation. The authors, in this study have performed an experimental evaluation using real world implementations, in both wired and wireless test beds and compared them with an old variant of RED -Adaptive RED.

Eduard Grigorescu *et al*. in (2013) explored the performance of PIE and CoDel in simulated rural broadband networks where capacity is limited. They compared the new algorithms using Adaptive RED as a reference and observed that to achieve a small queuing delay PIE and CoDel both escalate packet loss. Loss-sensitive unreliable multimedia applications such as real-time and near-real-time video were explored and the results showed that PIE performs better than CoDel in terms of packet loss rates affecting video quality and performance of ARED is comparable to that of PIE and CoDel in constant capacity links.

3. The bufferbloat problem

Every piece of equipment in the network requires to have some amount of buffering in order to handle bursts of packets on an entry link, and then to play them out on an outgoing link. It's particularly important in cases where there is a mismatch between the rate into the device and the rate out. Buffering provides space to queue packets waiting for transmission, thus reducing data loss. Bufferbloat can be defined as excess and poorly managed, buffering in network equipment, resulting in high latency and reduced throughput. It can be understood as the buffering of too many packets in flight between two network end points, resulting in unnecessary delays and confusion of TCP's flow control algorithms. Bufferbloat allows the queues to grow too long before any packets are dropped.

Considering it a simple problem, and offering a simple solution of making the buffers smaller, only complicates it. An accurate solution to bufferbloat requires a deeper understanding of what is going on, and improvement in the software across the net. Cheap memory has led to a bigger-is-better approach among networked systems designers, and excess buffering has resulted both in huge latencies-sometimes, of the order of seconds. Bottlenecks (bandwidth reduction) are most common at the edge of the Internet and therefore, a lot of care must be taken to avoid queuing delays of all sorts.

4. Active Queue Management

AQM refers to algorithms and methods to control the amount of data stored in network node buffers. The basic principle of AQM is that of preventively dropping packets depending on some metric of local congestion or queue utilization. They can also serve as the basis of explicit congestion notification, by e.g. marking packets instead of dropping them.

Why are AQM algorithms essential? Buffers cannot be effective for their intended purpose of handling bursts of packets if they fill up, and TCP cannot function properly in the face of congestion unless it is signaled to slow down by packet drop or ECN in a well-timed fashion. TCP's responsiveness to sharing a link is quadratic: 10 times the delay means it will respond 100 times more slowly to competing traffic. Some researchers point towards local active queue management (AQM) techniques (i.e., affecting the scheduling and dropping of packets in the buffer differently from a traditional FIFO discipline) as the ultimate solution of reducing queuing delay.

Active queue management algorithms are also needed to tackle the latency issue. Buffers are needed to cope with traffic bursts, so bufferbloat cannot be solved simply by reducing buffer sizes to very low values. AQM's are not only needed in Internet routers. They are needed anywhere there is a possibility of a queue growing, including in our home routers, laptops and smart phones, where AQM's are not currently present. Since existing deployed algorithms are typically RED (Floyd *et al*, 1993) variants, they require tuning, and are often not configured even on routers where they would prove to be of great benefit.

CoDel Controlling Queue Delay by Kathie Nichols and Van Jacobson (Nichols *et al*, 2012) has some implementation advantages over other AQMs and may solve the above discussed problems of adaptive AQMs. It does nearly all of its work at dequeue stage (when packets are transmitted). CoDel does require adding a timestamp to each individual packet as it is received, but even if this can't be done by the network hardware, timing information is directly available from a CPU register in modern CPUs.

5. Controlling Queue Delay

5.1 Overview

Controlling Delay (CoDel) is a newly proposed Active Queue Management (AQM) scheme to address the Bufferbloat problem in the Internet. Recently, there have been lot discussions on delays in the Internet that can grow up to several second or even minutes. Those delays impede the usage of delay-sensitive services like VoIP/Skype. Part of the problem is large buffers in Internet routers which will get filled up by long-lasting TCP connections as e.g. download. The CoDel algorithm was published into the public domain on May 2012 [CoDel], and the authors make an open source implementation available. Random Early Detection (RED) already existed for a long time but has a higher complexity in parameterization.

In network routing, CoDel (pronounced coddle) for controlled delay is a scheduling algorithm for the network scheduler developed by Van Jacobson and Kathleen Nichols. It is designed to overcome bufferbloat in network links (such as routers) by setting limits on the delay network packets suffer due to passing through the buffer being managed by CoDel. CoDel aims at improving on the overall performance of

the RED algorithm by addressing some fundamental misconceptions in the algorithm (as perceived by Jacobson) and by being easier to manage (since, unlike RED, CoDel does not require manual configuration). An implementation of CoDel was written by Michael D. Täht and Eric Dumazet for the Linux kernel and dual licensed under the GNU General Public License and the 3-clause BSD license.

Authors in (Gong *et al*, 2014) highlight an interesting similarity between the most recent approaches of either class, namely the CoDel AQM and the LEDBAT CC, as both aim at explicitly controlling queuing delay. They both employ a target delay parameter upon which dropping decisions or congestion window modifications are based respectively. Figure 1 illustrates a timeline of the evolution of AQM and LPCC (Low Priority Congestion Control) algorithms.

Fig. 1 Timeline of LPCC and AQM algorithms

Some important characteristics of CoDel are numbered below:

- It is parameterless--no knobs/handles are required for operators, users, or implementers to adjust.
- It treats good queue and bad queue differently - that is, it keeps the delays low while permitting bursts of traffic.
- It controls delay, while insensitive to round-trip delays, link rates, and traffic loads.

It adapts to dynamically changing link rates with no negative impact on utilization. Authors while working with CoDel differentiate between two types of queues used in the concept of this algorithm. The queues are discussed in the next section.

5.2 Queues in CoDel

Good queue: Defined as a queue that exhibits no bufferbloat i.e. it quickly drains as packets are transmitted. Such queues are necessary to smooth over the inherent bursts of arriving packets at bottlenecks in the network. Such bottlenecks can be between a fast LAN and a slower Internet connection, between a wired and a wireless network,

Bad queue: Defined as a queue that is filled up at the same rate as packets are transmitted, so the queue never empties. In its steady state, the TCP protocol will release more packets as the reception of earlier packets is acknowledged. At this point, a queue at the bottleneck between the sender and the receiver will become a standing queue—filling up and draining at the same rate to stay the same size.

In order to be effective against bufferbloat, a solution in form of an Active queue management (AQM) algorithm must be able to recognize an occurrence of bufferbloat in a queue and react with deploying effective countermeasures.

The question remains that *what is the correct length of a queue?* In the face of steady traffic, the correct length is nearly empty, just enough to keep utilization of a link at an acceptable level. More than that and all that is being added is unneeded delay. In the next section working of the CoDel is explained and how it manages its queues to extract the minimum delay out of it.

5.3 CoDel: Working

CoDel works by adding a timestamp to each packet as it is received and queued. When the packet reaches the head of the queue, the time spent in the queue is calculated; it is a simple calculation of a single value, with no locking required, so it will be fast. If the time spent by a packet within the queue is higher than a defined threshold, the algorithm sets a timer to drop a packet at dequeue. This dropping is only done when the queuing delay within a time window is above the threshold value and the queue holds at least one MTU's worth of bytes.

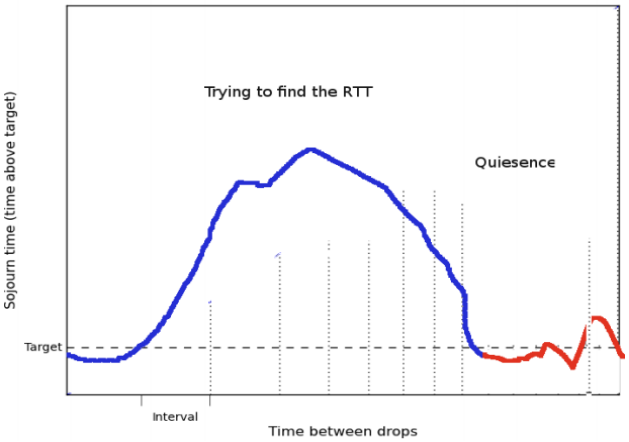


Fig. 2 CoDel drop scheduler behavior

From the illustration

- CoDel assumes that a standing queue of target is acceptable (as determined by the local minimum) and that it is unacceptable to drop when there are fewer than an MTU worth of bytes in the buffer
- To ensure that the minimum value is not stale, it has to have been experienced in the most recent interval (sliding window)
- When the sojourn time has exceeded target for at least an interval, a packet is dropped and a control law used to set the next drop time
- The next drop time is decreased in inverse proportion of the square root of the number of drops since the dropping state was entered (to get a linear change in throughput)
- When the sojourn time goes below target the controller stops dropping.
- Prevented from re-entering dropping state too soon after exiting it and resumes the dropping state at a recent control level if one exists.

CoDel has been tested in simulation tests by Nichols and Jacobson, at different MTUs and link rates and other variations of conditions. In general, results indicate:

- In comparison to RED, CoDel keeps the packet delay closer to the target value across the full range of bandwidths (from 3 to 100 Mbit/s). This seems to result in good queue, since the measured link utilizations are consistently near 100% of link bandwidth.
- At lower MTU, packet delays are lower than at higher MTU. Higher MTU results in good link utilization, lower MTU results in good link utilizations at lower bandwidth, degrading to fair utilization at high bandwidth. Certain simulations with DropTail were also performed by Greg White and Joey Padden at CableLabs (Greg White *et al*, 2013). Following is a flowchart depicting the CoDel algorithm in a simplified form.

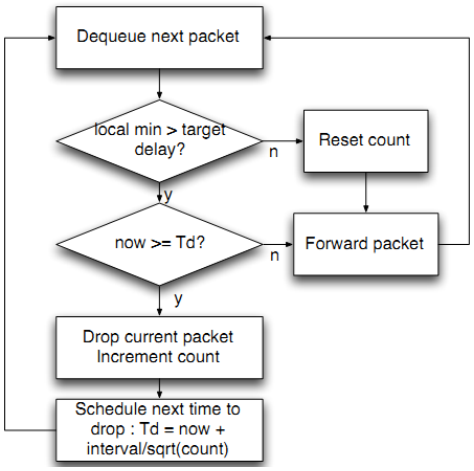


Fig. 3 Simplified CoDel algorithm flowchart

6. Performance Evaluation

6.1 Performance Metrics

To evaluate the performance of the Controlled Delay active queue management mechanism for Internet, the following criteria can be used:

- Average end-to-end Delay

It is the average elapsed time to deliver a packet from source to destination. This metric includes all possible delays caused by queuing at the interface queue, propagation and transfer times, and retransmission delays. Delay indicates the queue delay in our simulation results.

- Packet Loss Rate

Packet loss rate indicates the number of packets that are dropped due to router buffer overflow and for congestion notifications (in AQM techniques) and includes the data that the source or intermediate nodes drop during the simulation time.

- Throughput

Throughput is defined as the average data rate of a source sending packets and received by the receiver.

6.2 Simulation Model: Overview

The Network Simulator 2 (ns-2.35) is used to conduct preliminary simulations and to demonstrate the performance of our proposal. NS2 is a discrete event simulator for networking research and provides simulation support for transport protocols, routing and multicast protocols over wired and wireless networks. The basic structure of an ns-2 topology is composed of nodes that are connected by links. A node represents a network element such as a host, switch or router. A link represents the physical and MAC layers of the connection between two nodes. Traffic is generated via source and sink agent that are bound to individual nodes. Buffering and any buffer management are performed by the link.

In this simulation, data are sent from the one wireless node to another wireless node (0 and 1). The wireless nodes are stationary in the simulation. All simulations in the thesis are conducted multiple times over multiple scenarios to verify the validity of the protocol in talk. The goal of simulations in this Chapter is to compare performance of the CoDel algorithm to existing queue management algorithms. Average end-to-end Queuing delay is the main evaluation criteria in the comparison. More specifically, latency under load is tested with queuing handled by each of CoDel, RED and DropTail. Throughput and Packet loss ratio are considered only as an indication of the possible better packet delivery or bandwidth utilization by in the simulations.

For comparison, we ran our test scenarios with the ns-2 RED and DropTail AQM module substituted for CoDel. The most recent settings and code for ns-2 RED, were used which reads the initial link bandwidth and delay to adjust its settings, rather than the original RED of Floyd and Jacobson (Floyd *et al*, 1993).

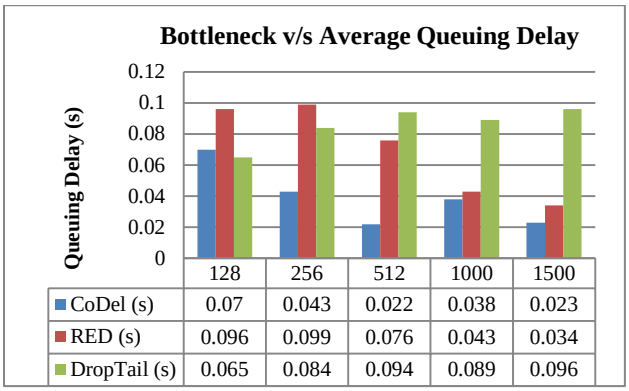
Table 1 Simulation Environment

Parameter	**Value**
Simulator	Ns2 (v-2.35)
Algorithms	CoDel, RED, DropTail
Topology	Single bottleneck topology with two way traffic
File transfer application	Linux TCP suite
Value of *interval*	100ms (constant)
Value of *target delay*	5ms (constant)
Buffersize	~16xBDP (actual size doesn't matter)

6.3 Results & Discussion

The bottleneck bandwidth is varied from 128 kbps to 100 Mbps. The following figures show the results for bottleneck average queuing transmission delay, packet drop rate at the bottleneck queue and bottleneck throughput respectively. The desirable properties of an optimal AQM mechanism are: minimum queuing delay, least packet drop rate and high throughput. It can be observed from the above mentioned graphs that CoDel satisfies all the desirable properties. When bandwidth is less, delay resulting from bursts of packets is expected to be more and hence, the queue occupancy is also expected to be more. When bandwidth is more, delay resulting from bursts of packets is expected to be less and hence, the queue occupancy is also expected to be less.

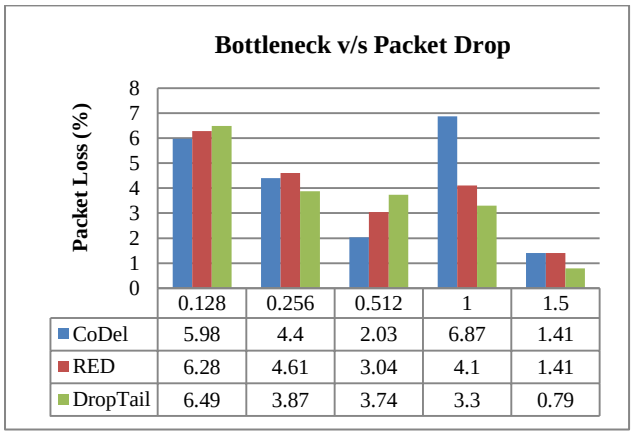
Figure 4 shows the variation of queuing delay with the changing bottleneck bandwidths. We varied the range of bottleneck from 128 kbps to 1.5 Mbps and the tests showed promising results as expected i.e. in the case of sending a packet from one end to another, CoDel showed minimum delay as compared RED and DropTail. Packets cannot arrive at the destination any faster than the time it takes to send a packet at the bottleneck rate. So a packet has to suffer at least a minimum delay in the face of bottleneck bandwidth.



	128	256	512	1000	1500
CoDel (s)	0.07	0.043	0.022	0.038	0.023
RED (s)	0.096	0.099	0.076	0.043	0.034
DropTail (s)	0.065	0.084	0.094	0.089	0.096

Fig. 4 Variation of average queuing delay with bottleneck bandwidth

CoDel does not allow a packet to stand in the queue for more than 5ms, so it drops the packet and prevents the congestion of the buffer. In RED on the other hand, packets are randomly dropped before the buffer is full, and the drop probability increases with the average queue size. RED controls the macroscopic behavior of the queue length (looking at the average fixed in the predetermined parameters) they often cause slow response and fluctuation in the instantaneous queue length. As a result, a large variation in end-to-end delays is observed in the case of RED as the bottleneck increases. DropTail queue has the highest average queue delay, since it only starts dropping when the buffer overflows. DropTail has no control on the time a packet remains inside the queue. So, it continues to fill its buffer as long as it is full and drops only at the tail and the packets at the end of the tail can be made to wait for the count of seconds.



	0.128	0.256	0.512	1	1.5
CoDel	5.98	4.4	2.03	6.87	1.41
RED	6.28	4.61	3.04	4.1	1.41
DropTail	6.49	3.87	3.74	3.3	0.79

Fig. 5 Variation of packet drop rate with bottleneck bandwidth

Figure 5 illustrates the variation of bottleneck bandwidth and its subsequent Packet losses in CoDel, RED and DropTail. CoDel has the highest drop rate at 1Mbps of all the simulated AQMs. This is because it tries to keep the queue small and the queuing delay below a target delay. So it drops as many packets it needs to. RED begins dropping packets when congestion occurs at a rate which was selected during configuration. Also, CoDel has slightly higher packet drop rate for more number of FTP-flows.

Packets in RED are randomly dropped before the buffer is full, and the drop probability increases with the average queue size. Looking at the Figure 4 and 5, it can be concluded that as the queue size is increasing in RED i.e. delay is increasing; the packets drops are also growing according to Figure 5. It uses early packet dropping in an attempt to control the congestion level, limit queuing delays, and avoid buffer overflows. Under DropTail, when queue overflow occurs, arriving packets are dropped. DropTail queue has the longest average output queue length, since it only starts dropping when the buffer overflows, whereas RED starts dropping packets before that. Hence, packet drops are also very low in the case of DropTail as it delivers the packet even in the face of causing a lot of delay.

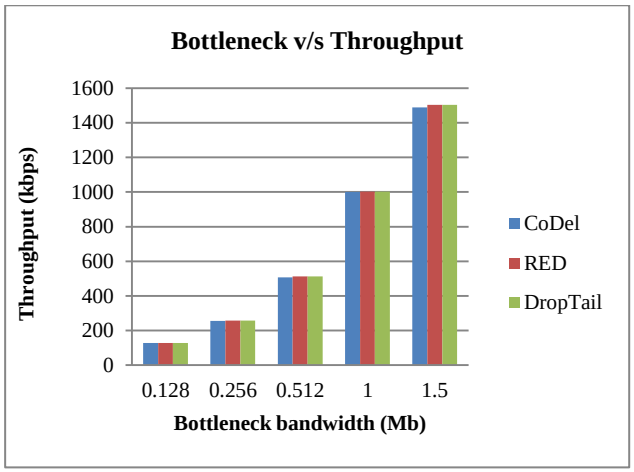


Fig. 6 Throughput variation with respect to bottleneck bandwidth

In Figure 6, we show how the performance of CoDel varies in the accordance to the bottleneck variation in terms of throughput. In this chapter Throughput and Packet loss ratio are considered only as an indication of the possible better packet delivery or bandwidth utilization by CoDel in the simulations because on the whole we are concerned about how well CoDel manages the delay in the internet. Throughput is the average data rate of a source sending packets and received by the receiver. CoDel performs comparatively to RED and DropTail in terms of throughput. A source cannot send a packet faster than the bottleneck bandwidth of line to the receiver.

Conclusions

The objective of this work is to investigate the effectiveness of CoDel in different congestion conditions

in the network. The performance of CoDel is compared with RED and DropTail. CoDel is a very general, efficient, parameterless AQM approach that can be applied to congested queues. It is a critical tool in solving bufferbloat. It achieves low end-to-end delay and comparable throughput to RED and DropTail. It depicts high drop rates at some levels of bottleneck bandwidths. This is because it tries to keep the queue small and the queuing delay below a target delay. So if we simulate it by increasing the target delay, a decrease in drop rates is expected. This indicates that CoDel performance can be tuned for special-purpose networking applications. The CoDel algorithm provides the missing link necessary for good TCP behavior and solving bufferbloat. But CoDel by itself is insufficient to solve and provide reliable, predictable low latency performance in today's Internet. On-going projects are creating a deployable CoDel in Linux routers and experimenting with applying CoDel to stochastic queuing with very promising results.

Future Scope

Further study could provide more guidance with respect to performance in a larger variety of scenarios, and provide guidance on CoDel performance relative to other active queue management approaches. While the results presented in this research are encouraging and show that it is possible to mitigate the bufferbloat problem quite effectively, much work remains to be done. Scheduling algorithms and queue management should be seen as complementary, not as replacements for each other. While these two AQM mechanisms are closely related, they address different performance issues.

Current implementations of CoDel do not support fixing of concatenated queues or multiple queues. Future studies can adopt the question of revising CoDel to be effective for multiple queues. All in all, though, while much work remains to be done, tackling the bufferbloat problem does appear to be amenable within the probable future.

References

Nichols, K., and V. Jacobson, (2012), Controlling Queue Delay, *Communication ACM, 55(7)*, pp. 42–50.

Gong, Y., D. Rossi, Claudio Testa, S. Valenti and M. D. Täht, (2014). Fighting the bufferbloat: on the coexistence of AQM and low priority congestion control. *Computer Networks*.

Raghuvanshi, Dipesh M., B. Annappa, and Mohit P. Tahiliani. (2013), On the Effectiveness of CoDel for Active Queue Management, *Advanced Computing and Communication Technologies (ACCT), IEEE 2013, Third International Conference*, pp. 107-114.

Khademi, Naeem, David Ros, and Michael Welzl. (2013). The New AQM Kids on the Block: Much Ado About Nothing?, *Research report, ISBN 82-7368-399-0, ISSN 0806-3036.*

Grigorescu, Eduard, Chamil Kulatunga, and Gorry Fairhurst. Evaluation of the Impact of Packet Drops due to AQM over Capacity Limited Paths. *CSWS* 2013.

White, Greg, and Joey Padden, (2012) Preliminary Study of CoDel AQM in a DOCSIS Network. *Technical Report, CableLabs*

Floyd, S.; Jacobson, V., (1993), Random Early Detection for Congestion Avoidance, *IEEE/ACM Transactions on Networking, Vol. 1, No. 4,* pp. 397–413.

B.C. Sreenivasa, G.C. BhanuPrakash and K.V.Ramakrishnan, (2011), Survey on congestion control techniques in ad-hoc network, *Elixir Adoc Net. 32*, pp. 2061-2067

Høiland-Jørgensen, Toke, (2012), Battling Bufferbloat: An experimental comparison of four approaches to queue management in Linux. *Master module project, Computer Science, Roskilde University*

Groenewegen, Danny, and Harald Kleppe. (2011). Detecting and quantifying bufferbloat in network paths. *Technical report, www.Bufferbloat.net.*

Gettys, Jim, and Kathleen Nichols. (2012), Bufferbloat: Dark buffers in the internet. www.*Queue.acm.org, Communications of the ACM, Vol. 55*.

Gong, YiXi, Dario Rossi, Claudio Testa, Silvio Valenti, and Dave Täht. (2012) Interaction or interference: can AQM and low priority congestion control successfully collaborate ? *Proceedings of the 2012 ACM conference on CoNEXT student workshop, ACM, 2012.* pp. 25-26.

White, Greg, and Dan Rice. (2013). Active Queue Management Algorithms for DOCSIS 3.0. *Technical report-Cable Television Laboratories, 2013.*

Koo, Jahwan, Seongjin Ahn, and Jinwook Chung. (2012). A comparative study of queue, delay, and loss characteristics of AQM schemes in QoS-enabled networks, *Computing and Informatics 23, no. 4*, pp. 317-335.

https://www.bufferbloat.net/projects/codel.

http://www.isi.edu/nsnam/ns.