Revisiting Old Friends: Is CoDel Really Achieving What RED Cannot?

Nicolas Kuhn
IMT Telecom Bretagne
2 Rue de la Châtaigneraie
35510 Cesson-Sévigné, France
nicolas.kuhn@telecom-bretagne.eu

Emmanuel Lochin
Université de Toulouse, ISAE, TéSA
10 Avenue Édouard Belin
31400 Toulouse, France
emmanuel.lochin@isae.fr

Olivier Mehani
National ICT Australia
Level 5, 13 Garden St
Eveleigh NSW 2015, Australia
olivier.mehani@nicta.com.au

ABSTRACT

We use *ns*-2 simulations to compare RED's `gentle_` mode to CoDel in terms of their ability to reduce the latency for various TCP variants. We use a common dumbbell topology with Pareto background traffic, and measure the packet delays and transmission time of a 10 MB FTP transfer.

In our scenarios, we find that CoDel reduces the latency by 87 %, but RED still manages to reduce it by 75 %. However, the use of CoDel results in a transmission time 42 % longer than when using RED. In light of its maturity, we therefore argue that RED could be considered as a good candidate to tackle Bufferbloat.

Categories and Subject Descriptors

C.2.6 [**COMPUTER-COMMUNICATION NETWORKS**]: Internetworking—*Routers*

General Terms

Performance

Keywords

AQM; Bufferbloat; CoDel; *ns*-2; RED

1. INTRODUCTION

Bufferbloat occurs due to oversized buffers at routers where packets arrive at a faster rate than they can be forwarded [11, 5]. As a result, packets are queued until they are either transmitted, or dropped when the buffers are full. As the queues fill, it takes longer for each packet to be transmitted to its destination. Moreover, most congestion control mechanisms do not recognise an increase of the delay as a congestion event and thus, keep increasing their rate on an already-saturated path.

While some contention exists as to whether or not Bufferbloat really is a widespread problem [1], it has been observed in mobile networks [16]. The use of Active Queue Management (AQM) has been proposed as a way to alleviate Bufferbloat [*e.g.*, 13, 11]: there has been a renewed interest at IETF is evaluating AQM; a working group has been created in 2013 and produced recommandations for AQM [4]. Most notably, CoDel [23] has been proposed specifically to counter Bufferbloat by ensuring that packets do not remain "too long" in the managed queues. CoDel is said to be parameter-less, yet it is arbitrarily configured with two hard coded parameters (target delay and interval).

While CoDel has been specifically designed to tackle Bufferbloat, other AQMs such as RED [10] could also be used to the same effect. RED has the advantage of having been well both studied and tested. Although past studies used to show that RED remains hard to tune [25, 21], RED parameters found to be fairly well understood [8] and can be now set or managed automatically [15, 20]. Moreover, the authors of [17] mention that there are lessons to be learned from the old AQMs as their performance can compete with new AQMs proposals. In this article, we perform *ns*-2 simulations to compare the trade-off between latency and throughput proposed by CoDel and RED, by introducing specific traffic patterns, metrics and transport protocols.

As AQM reduces queues by dropping packets, an evaluation of the effectiveness of AQMs against Bufferbloat should study the trade-off between a reduced packet delay, and an increased data transmission time (due to losses and retransmissions). Moreover, not all congestion control (CC) mechanisms react the same to changed loss rates or delays. These AQMs should therefore be studied in light of their impact on the various transport protocols they are expected to manage.

This paper systematically studies this trade-off. We use *ns*-2 to simulate a loaded network where Bufferbloat occurs. We then introduce various AQMs, and observe their impact on flows transported using various TCP's CC mechanisms. We find that both AQMs give the same advantages for all CC algorithms: be they loss-based or delay-based, they all experience relatively similar conditions. Moreover, we show that RED performs quite well, reducing the packet delay by a sizeable amount—yet not as much as CoDel. More interestingly, we also find that RED allows for shorter transmission times than CoDel does, which suggests that RED might

be closer to achieving the trade-off of efficiency versus Bufferbloat.

The rest of this paper is organised as follows. In the next section we present and discuss the various elements of our simulations: AQMs, CCs, topology, and parameters causing Bufferbloat. Section 3 presents the impact of AQM on various congestion control mechanisms while Section 4 summarises our results on transmission and packet delays. We discuss these results in Section 5, before offering a short concluding summary in Section 6.

2. SIMULATING BUFFERBLOAT IN *NS*-2

This section presents the AQM schemes, congestion control, topology, traffic patterns and parameters that trigger Bufferbloat.

2.1 Active Queue Management Mechanisms

This article compares the impacts of various AQM and transport layer protocols. We consider the following AQM.

2.1.1 DropTail

DropTail is a default queuing mechanism that **drops incoming packets** when the **queue is full**. This default queue management is the baseline for our evaluations.

2.1.2 RED

RED [10] drops packets depending on the **number of packets in the queue**. In order to reduce the occupancy in the gateway and manage the size of the queue when the average queue size exceeds a threshold, packets are randomly dropped to reduce the throughput of different non cooperating sources.

We consider the default implementation of RED in *ns*-2 with `gentle_` mode activated. This mode consists of a basic improvement that *must* be considered,[1] as it makes RED more suitable for large scale deployment.

Exhaustive performance comparisons between RED versions is beyond the scope of this paper. As a result, we do not evaluate Adaptive RED (ARED) [9]. Gentle RED is already adaptive, but differs in that `max_p` is not adjusted to better control the average queue size, as it is in ARED.

We use the default values for the parameters of RED, such as the `targetdelay` (not to be confused with CoDel's target value for the queue delay), set to 5 ms.

2.1.3 CoDel

CoDel [23] drops packets depending on the **sojourn time** of each packet in the queue. CoDel switches between its "dropping mode" and "non dropping mode" depending on the measured and maximum authorised sojourn times (set by default to 5 ms). The CeroWRT project [7] implements CoDel as a way to solve the different problems encountered at a home router, particularly Bufferbloat.

A variant of CoDel, Flow-Queuing CoDel (FQ-CoDel) [14], which combines Stochastic Fairness Queueing (SFQ) [22] and CoDel, could not be evaluated in this article, as the description of this scheduling has just been released as an Informational IETF draft. Moreover, it is not fair to consider FQ-CoDel, while other tested AQMs do not integrate any kind of scheduling. Future work will evaluate the interactions between scheduling and AQM algorithms.

[1] `http://www.icir.org/floyd/red.html#parameters`

2.1.4 PIE

determines a **dropping probability** that is based on the **average queueing latency** and limit the queuing latency to a **target value**.

While it is very relevant to this study, at the time we ran the simulations, no implementation of PIE was available. We therefore focused on the comparison between RED and CoDel to evaluate if a new generation of AQM is needed. Future work will consider PIE.

2.2 Congestion Control Algorithms

Congestion control (CC) mechanisms may be based on loss events or measurements of latency. To better understand the impact of these AQM mechanisms on CC algorithms, we need to consider the following transport layer protocols:

TCP NewReno loss-based CC and baseline for our evaluations;

TCP Vegas delay-based CC. In the context of reducing Internet latency, there is a renew interest for delay-based CC, which are less aggressive but which introduce less delay in the network. Hybrid CC (both delay and loss based) are also of interest;

TCP Compound hybrid (loss/delay) CC which is implemented in Windows systems;

TCP CUBIC loss-based CC which is deployed on a large scale in Android/Linux systems.

We use the native Linux implementations for these protocols in *ns*-2.

2.3 Topology and traffic

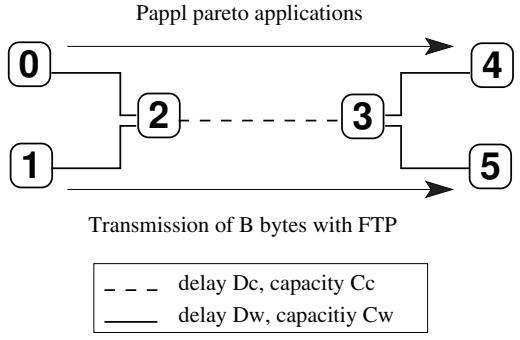


Figure 1: Dumbbell topology used for the simulations. Background Pareto traffic goes from node 0 to 4, while the flow under study is sent from 1 to 5.

Figure 1 presents the dumbbell topology and corresponding notations. This network topology is common to evaluate the performance of TCP [2]. Nodes 0 and 1 (resp. 4 and 5) are connected to node 2 (resp. 3) by a link of D_w (ms) one way delay and C_w (Mbps) capacity. Nodes 2 and 3 are connected by a link of D_c (ms) one way delay and C_c (Mbps) capacity. We integrate the AQM mechanism under evaluation in node 2.

Two classes of traffic are injected inside this network:

- From node 0 to node 4, we introduce P_{appl} applications; each of them transmit a file whose size is generated according to a **Pareto law**: 80% of short flows and 20% of long flows corresponding to the distribution of flow size measured in the Internet [6, 19]. We use the Pareto traffic generator in *ns*-2. This traffic is injected to **dynamically load the network**.

- From node 1 to node 5, we introduce an **FTP transmission** of B bytes. This traffic is **studied** to understand the protocols and cross-algorithm impacts.

2.4 Parametrization and Bufferbloat

In this section, we present the combination of traffic loads, delays and capacities where the Bufferbloat occurs that will be considered in the rest of this article.

2.4.1 Load and Bottleneck Capacity

For the rest of this article, we consider: $P_{appl} = 80$, $D_w = D_c = 25\,\text{ms}$, $C_w = 10\,\text{Mb}$, $C_c = 1\,\text{Mb}$, $B = 10\,\text{MB}$, which we found to lead to a stable, non null, queueing delay, *i.e.*, Bufferbloat occurs. A larger set of parameters may need to be considered to extend the validity of the results presented in this article, this will be improved in a future work.

2.4.2 Buffer Sizes

In [3], the authors justify why setting the size of buffers to the product $RTT \times C$ is outdated. Due to a potentially large number of TCP connections that transmit data through the router, the queuing delay introduced cannot be neglected. Due to the asymmetric architecture of our topology ($C_w = 10 \times C_c$), we could not respect those specifications, otherwise there would have been too much buffer overflow.

We size the buffer of the central node (node 2) based on the characteristics of the wing links (from node 0 to node 2), $RTT \times C_w = 41$ packets. We also present the results with a buffer: (1) smaller than the $RTT \times C_w$ product (buffer of 10 packets), (2) sized by the $RTT \times C_w$ product (buffer of 45 packets), (3) larger than the $RTT \times C_w$ product (buffer of 125 packets and without limitation).

3. IMPACT OF AQM WITH CUBIC AND VEGAS

In this section, we focus on the file transmission of $B = 10\,\text{MB}$. We evaluate the throughput (measured at node 5), queuing delay (measured at node 2) and drop ratio for this particular flow. These metrics have been chosen following the guidelines presented in [2]. These metrics are presented depending on the choice of AQM (DropTail, RED, CoDel), CC mechanism and the size of the queues (as explained above). Due to the lack of space, we only present the results with TCP CUBIC (loss-based CC) and TCP Vegas (delay-based CC). The performance of other loss-based congestion controls shows the same behaviour as TCP CUBIC.

In Figure 2, we plot the drop ratio depending on the queuing delay for TCP CUBIC (results for TCP Vegas are qualitatively similar). In Figure 3 (resp. Figure 4), we plot the throughput depending on the queuing delay with TCP Vegas (resp. TCP CUBIC). Each point represents the average metric measured during one second of the simulation. The throughput may drop to 0. Depending on drop events, no packet may be received in one second.

In Figure 2, we can see that the introduction of RED and CoDel results in the occurrence of drop events. With DropTail, the queuing delay is maximised by the size of the queue, whereas with RED and CoDel the maximum queuing delay is reduced to 300 ms and to 50 ms. The queuing delay ranges between 0.01 s and 0.1 s with CoDel, whereas between 0.1 s and 0.5 s with RED. With CoDel, the size of the queue has no impact. CoDel is based on the sojourn time of each packet. On the contrary, with RED, there is a tiny impact of the buffer size on the queuing delay as the dropping policy of RED is based on the number of packets in the queue.

Firstly, we compare the performance of TCP Vegas and TCP CUBIC with the queuing mechanism DropTail. In Figure 3, we can see that with DropTail and TCP Vegas, the throughput decreases as the queue size increases. Indeed, when the queue is unlimited, TCP Vegas CC reacts as expected to queuing delay increases. The larger the queue is, the larger the queuing delay is and the more TCP Vegas decreases the congestion window. On the contrary, in Figure 4, with DropTail and TCP CUBIC, the throughput increases with larger queues. The larger the queue, the bigger the queueing delay, but fewer congestion losses events occur.

Secondly, we compare the performance of RED and CoDel with TCP Vegas as a CC protocol. Apart from the queuing delay (the queuing delay is between 0.01 s and 0.1 s with CoDel, whereas it is between 0.1 s and 0.5 s with RED), the throughput is the same whatever the choice of AQM.

Finally, we compare the performance of RED and CoDel with TCP CUBIC. Apart from the queuing delay (the queuing delay is between 0.01 s and 0.1 s with CoDel, whereas it is between 0.1 ms and 0.5 ms with RED), the throughput is larger with RED (up to 0.75 Mbps) than with CoDel (up to 0.45 Mbps).

The early conclusions that we can derive from this section is that CoDel is a good candidate to reduce latency. However, we also showed that RED reduces the latency as well and still transmits more traffic and better exploits the capacity of the bottleneck. This observation suggests that a better trade-off might exist between latency reduction and more efficient capacity use.

4. TRANSMISSION AND PACKET DELAY

In this section, we focus on the end-to-end performance and cross-impact of the choice of AQM and transport layer protocols in presence of Bufferbloat. We evaluate the packet transmission time (for the packets that are successfully transmitted) and the time needed to transmit 10 MB, while the network is loaded with the *P_appl* applications. The packet transmission time gives an overview of the latency inside the network whereas the time needed to transmit 10 MB provides an end-to-end viewpoint on the throughput.

In Figure 5, we present the average packet transmission times and in Figure 6 the time needed to transmit 10 MB both averaged over a large number of iterations. For each metric, we present the minimum and maximum values (bottom and top of the candle), the 95 percentile (full box), the median value, and an estimate of its 95 % confidence interval (dashed box). We measured these metrics with queues of 125 packets, corresponding to the bandwidth×delay product.

Figure 5 illustrates that RED and CoDel enable a better latency reduction than DropTail. With TCP CUBIC the packet transmission time is reduced by 87 % with CoDel and by 75 % with RED. The average packet transmission

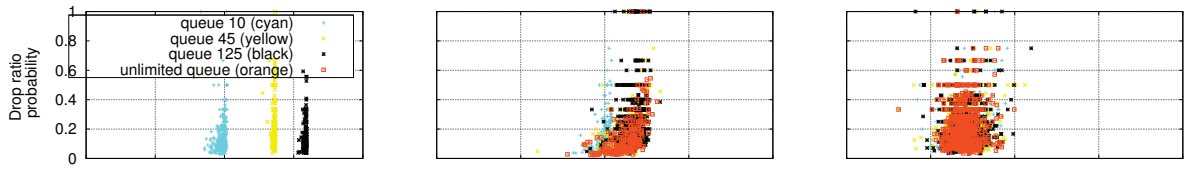


Figure 2: TCP CUBIC: Drop ratio versus queuing delay (TCP Vegas shows the same qualitative behaviour)

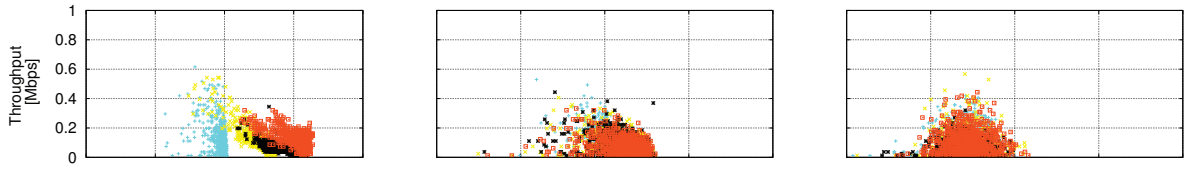
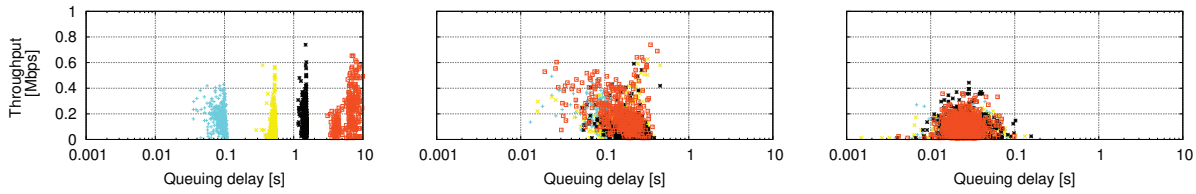


Figure 3: TCP Vegas: Throughput versus queuing delay



(a) DropTail (b) RED (c) CoDel

Figure 4: TCP CUBIC: Throughput versus queuing delay

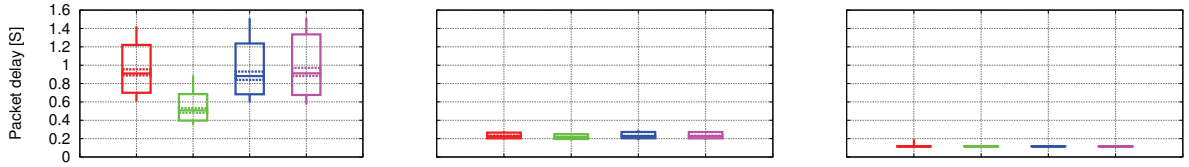
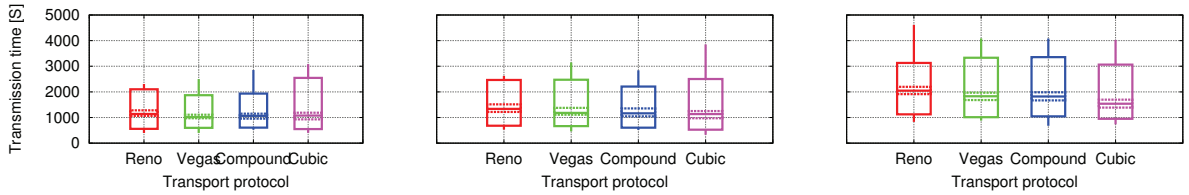


Figure 5: Packet transmission times



(a) DropTail (b) RED (c) CoDel

Figure 6: Time needed to transmit 10 MB

time with TCP CUBIC and CoDel is 115 ms compared to 226 ms with RED. We confirm that, with DropTail as an AQM mechanism, TCP Vegas introduces less latency than any other congestion control protocols which legitimizes the trend of considering delay-based CC to reduce Internet la-

tency: the latency is reduced by 44 % when the CC is TCP Vegas rather than TCP CUBIC.

Figure 6 shows the time needed to transmit 10 MB depending on the choice of AQM and CC protocol. One result that can be seen in this figure is that the dropping events generated by RED do not impact this transmission time much, whatever the choice of the CC. With TCP CUBIC, introducing RED increases the average transmission time for 10 MB by 5 % compared to DropTail. However, introducing CoDel results in an increase of 42 % of this transmission time.

5. DISCUSSION

The results presented in this article support the interest for re-considering AQM. Section 2.1 illustrated that the DropTail performance is directly related to queue size. However, sizing a sending buffer depends on the available capacity at the physical layer and is not a convenient way to tackle Bufferbloat, such as in the context of Wi-Fi where bandwidth fluctuates. Moreover large buffers may be needed when an application transmits large bursts of packets. An AQM scheme should accept incoming bursts and fight Bufferbloat by limiting the persistent occupation of the buffer. Conversely, both Sections 2.1 and 4 highlighted that with either RED or CoDel as AQM, sizing of the queues can be neglected without any substantially negative impact. Having large buffers is important to absorb large bursts of packets but AQM is needed so that Bufferbloat does not occur.

We observed that both RED and CoDel reduce the latency. In our specific simulations, CoDel reduced latency by 87 % and RED by 75 %. However, a trade-off must be found between reducing latency and degrading end-to-end goodput. CoDel increased the time needed to transmit 10 MB by 42 %, while RED only introduced a 5 % increase. This significant difference suggests that RED is a legitimate candidate to tackle Bufferbloat.

Moreover, we believe RED is a worthy solution because its deployment issues are known. RED has been improved over the years and Adaptive RED has better performance than Gentle RED (used in this article) in terms of adaptability and deployment. The idea behind CoDel is to consider delay as a key element to manage dropping events whereas RED deals with the current size of the queue.

CoDel is said to be parameter-less, but we believe that, before large scale deployment, this point should be evaluated in light of the metrics proposed by [18]. As an example, in a document published by CableLabs [13], the authors explain that they had to adjust CoDel's target value to account for MAC/PHY delays even for packets reaching an empty queue. This justifies the need for studies evaluating the impact of the internal parameters of CoDel in contexts where delays matter, such as satellite communications or data-centers.

Finally, large scale deployment of AQMs should be mindful of the intended traffic to be carried, as it may impact the end-to-end properties of certain applications. As an example, LEDBAT [24] is a transport layer congestion control mechanism designed for background data. This kind of traffic has been pointed out as a root cause of Bufferbloat. However, the introduction of AQM degrades the Less-than-Best-Effort aspect of this protocol [12], by restoring fairness, as also suggested by our results with TCP Vegas.

Therefore, we believe that RED is a good candidate to reduce Internet latency, and has the additional advantage of being well studied. Where the performance of RED might be in small proportion linked to buffer size, CoDel considers fixed parameters making assumptions on the PHY/MAC layer latency. CoDel does not outperform RED. Resolving Bufferbloat with AQM strategies is (1) finding a trade-off between reducing latency and using the fully available capacity, and (2) considering deployment issues, which are known for RED.

6. CONCLUSION

We evaluated the performance of various AQMs (RED and CoDel), and their ability to limit Bufferbloat. Our simulations have shown that, while CoDel performs as expected, RED works similarly well. Moreover, we have found that while both reduce the per-packet queueing delay, RED did so with less degradation of the end-to-end transmission time. We also showed that the use of either AQM made the network fairer to delay-based congestion control mechanisms.

While this paper does not question CoDel's effectiveness against Bufferbloat, our results suggest that CoDel is not the only AQM which could be used to solve that issue. In that respect, RED appears to be a worthy contender. Moreover, RED's parametrisation has been widely studied and is fairly well understood. In contrast, CoDel's supposed parameter-lessness relies on a hard-coded static value for which an optimal value appears hard to find.

AQMs should be more widely studied before large scale deployment and the focus on CoDel, at the exclusion of any other, might be too intense. Our future work will focus on other AQM variations, and their parametrisation, as well as their interaction with different congestion control algorithms, such as less-than-best effort.

Acknowledgements

The author wish to thank Paul Amer for numerous comments on this study. Nicolas Kuhn is supported by the Reducing Internet Transport Latency (RITE) project (ICT-317700).

References

[1] M. Allman. "Comments on Bufferbloat". In: *SIGCOMM Computer Communication Review* (Jan. 2013).

[2] L. Andrew et al. "Towards a Common TCP Evaluation Suite". In: *PFLDNet*. 2008.

[3] G. Appenzeller, I. Keslassy, and N. McKeown. "Sizing Router Buffers". In: *SIGCOMM Computer Communication Review* 34.4 (Aug. 2004).

[4] F. Baker and G. Fairhurst. *IETF Recommendations Regarding Active Queue Management*. RFC In WG Last Call. RFC Editor, 2014.

[5] "BufferBloat: What's Wrong with the Internet?" In: *Communications of the ACM* 55.2 (Feb. 2012).

[6] D. Ciullo, M. Mellia, and M. Meo. "Two Schemes to Reduce Latency in Short Lived TCP Flows". In: *IEEE Communications Letters*. 2009.

[7] J. Edge. *CeroWRT: Bufferbloat, IPv6, and more*. 2012.

[8] S. Floyd. *RED: Discussions of Setting Parameters*. Nov. 1997. URL: http://www.icir.org/floyd/REDparameters.txt.

[9] S. Floyd, R. Gummadi, and S. Shenker. *Adaptive RED: An Algorithm for Increasing the Robustness of RED's Active Queue Management*. Tech. rep. 2001.

[10] S. Floyd and V. Jacobson. "Random Early Detection Gateways for Congestion Avoidance". In: *Networking, IEEE/ACM Transactions on* (1993).

[11] J. Gettys. "Bufferbloat: Dark Buffers in the Internet". In: *IEEE Internet Computing* (2011).

[12] Y. Gong et al. "Fighting the Bufferbloat: On the Coexistence of AQM and Low Priority Congestion Control". In: *INFOCOM Workshops*. 2013.

[13] C. Greg White. "Active Queue Management Algorithms for DOCSIS 3.0: A Simulation Study of CoDel, SFQ-CoDel and PIE in DOCSIS 3.0 Networks". In: *Cable Television Laboratories* (2013).

[14] T. Hoeiland-Joergensen et al. *FlowQueue-Codel*. IETF Informational. 2013. URL: http://tools.ietf.org/html/draft-hoeiland-joergensen-aqm-fq-codel-00.

[15] V. Jacobson, K. Nichols, and K. Poduri. *RED in a Different Light*. Tech. rep. Sept. 1999.

[16] H. Jiang et al. "Understanding Bufferbloat in Cellular Networks". In: *CellNet*. ACM, 2012.

[17] N. Khademi, D. Ros, and M. Welzl. "The New AQM Kids on the Block: Much Ado About Nothing?" In: *Global Internet Symposium Workshop* (2014).

[18] N. Khademi et al. *AQM Evaluation Criteria and Scenarios*. IETF presentation. 2013. URL: http://www.ietf.org/proceedings/88/slides/slides-88-aqm-5.pdf.

[19] C. Labovitz et al. "Atlas Internet Observatory Annual Report". In: *47th NANOG*. 2009.

[20] E. Lochin and B. Talavera. "Managing Internet Routers Congested Links with a Kohonen-RED Queue". In: *Elsevier Eng. Appl. Artif. Intell.* 24.1 (Feb. 2011), pp. 77–86. ISSN: 0952-1976.

[21] M. May et al. "Reasons Not to Deploy RED". In: *Proc. of 7th. International Workshop on Quality of Service (IWQoS'99), London*. June 1999, pp. 260–262.

[22] P. McKenney. "Stochastic Fairness Queueing". In: *INFOCOM 1990*. 1990.

[23] K. Nichols and V. Jacobson. "Controlling Queue Delay". In: *Queue* (2012).

[24] S. Shalunov et al. *Low Extra Delay Background Transport (LEDBAT)*. RFC 6817. RFC Editor, Dec. 2012.

[25] T. Ziegler, S. Fdida, and C. Brandauer. "Stability Criteria of RED with TCP Traffic". In: *IFIP ATM&IP Working Conference*. Budapest, June 2001.