A Principled Look at the Utility of Feedback in Congestion Control

Mohit P. Tahiliani
National Institute of Technology
Karnataka, Surathkal, India
tahiliani@nitk.edu.in

Vishal Misra
Columbia University
New York, NY, USA
vishal.misra@columbia.edu

K. K. Ramakrishnan
University of California, Riverside
Riverside, CA, USA
kk@cs.ucr.edu

ABSTRACT

Networked applications are ubiquitous and their performance requirements are becoming increasingly stringent. Network congestion can seriously impact performance contributing to increased latency, packet loss and poor throughput. To address these problems, the networking community has come up with a large number of congestion control algorithms. Congestion control schemes developed over the past few decades can be classified into two broad classes: one based on an end-system's perception of network congestion and the other based on the network providing feedback to flows that pass through it.

In this paper, we make the observation that the pure end-system based congestion control schemes are faced with the significant challenge of receiving ambiguous signals that make it difficult to infer where the congestion is occurring and if this flow is even the cause of that congestion. This ambiguity makes it difficult for pure end-system based control schemes to achieve fairness across different flows. Modern routers and switches in the meantime, have grown in computing capability and can generate fine grained feedback at line speeds for flows traversing them. We show that even relatively simple feedback generated in-network at the point of congestion eliminates the ambiguities faced by pure end-system based congestion control mechanisms, thus ensuring the network functions at the right fair and efficient operating point. We provide the theoretical underpinnings establishing the need for in-network feedback to enable the network to operate at a unique fixed point at the intersection of the desired fair and efficient operation regimes, and demonstrate through emulation experiments that our use of the well-established and studied PI-control for Active Queue Management and Explicit Congestion Notification meets the goals of low latency, high throughput and fine granularity control of the queue while achieving fairness.

CCS CONCEPTS

• **Networks → Transport protocols**; **Network experimentation**; **Network performance analysis**;

KEYWORDS

Congestion control, Queue management, ECN

BS'19, December 2-3, Stanford, CA

ACM Reference Format:
Mohit P. Tahiliani, Vishal Misra, and K. K. Ramakrishnan, 2019. A Principled Look at the Utility of Feedback in Congestion Control. In *Proceedings of Buffer Sizing Workshop (BS'19).* ACM, New York, NY, USA, 5 pages.

1 INTRODUCTION

Networked applications have diverse requirements ranging from ultra low latency to very high throughput. The performance of the network is becoming an increasingly important component. Congested networks can significantly impact application performance due to increase in latency, packet loss and poor throughput. The networking community has worked on the problem of congestion control for decades [1]. Typically, congestion control mechanisms can be classified into two broad classes: end-to-end mechanisms and network assisted mechanisms. The traditional packet loss based end-to-end congestion control mechanisms in TCP implicitly couple congestion signals to packet loss, and thus also implicitly cause increased delays as part of the congestion detection mechanism. Explicit Congestion Notification (ECN) [2] based network assisted congestion control decouples the congestion signal from packet loss or significant buffer occupancy. Environment specific congestion control schemes, such as for data centers are being developed to take advantage of the low feedback delay and the desire to keep latency very small. Some of these approaches have challenges when being used in more diverse environments, such as when there are multiple congested points or when the competing flows have very diverse feedback delays.

While a number of the approaches designed work well when flows go through a single, dominant bottleneck, there are still challenges when flows traverse through multiple congested points that experience significant queueing. Pure end-system based congestion control schemes are faced with the significant challenge of receiving ambiguous signals that make it difficult to infer where the congestion is occurring and if this flow is even the cause of that congestion. This ambiguity makes it difficult for pure end-system based control schemes (e.g., BBRv1 [3]) to achieve fairness across different flows. Methods that seek to interpret feedback information as a precise indication of the level of a congestion (e.g., BBRv2[1]) at a given bottleneck also have difficulty in the ambiguity that arises when multiple points in the network are congested. This has the potential for unfair treatment across flows, with some flows receiving an unfair share for significant time periods when congestion persists. While short-term fairness is often viewed as not being critical, unfair behavior even over short periods can significantly

[1] https://datatracker.ietf.org/meeting/104/materials/slides-104-iccrg-an-update-on-bbr-00

degrade application performance (user QoE, search performance, etc.) with increased queueing and packet loss.

We believe it is useful to re-examine the effectiveness of the current purely end-system based congestion control algorithms and compare them against those that take advantage of in-network feedback. For this, we consider an environment that is sufficiently general to encompass a range of network conditions that congestion control algorithms have to be effective in: multiple bottlenecks; vastly different round-trip times; a reasonably significant number of flows; with implementations that are representative of what is being used in practice (i.e., at least Linux-based environments). For this, we consider a topology that spans both a data center network and a wide-area network links with flows that are both just within the data center and go beyond the data center to the wide-area network (WAN).

We examine the performance of representative end-system based congestion control method that employ three different kind of feedback signals, as detailed in Section 3. The congestion control protocols that we use in conjunction with these signals are BBRv2, an approach that seeks to take advantage of both in-network feedback and end-system based feedback information and CUBIC TCP [4].

2 BACKGROUND AND RELATED WORK

Data Center TCP (DCTCP) [5] uses explicit network feedback to achieve ultra low latency and high throughput in Data Center Networks (DCN). DCTCP sender estimates the amount of network congestion by using ECN and employs an adaptive-multiplicative decrease technique to reduce the congestion window (*cwnd*) proportionately. However, ECN feedback in DCTCP is based on a threshold based mechanism, as opposed to prior AQM approaches which treated the ECN signal as a "drop-in" replacement from loss feedback, and the presence of a **single** ECN mark in an ACK packet results in a constant multiplicative decrease. DCTCP style ECN on the other hand counts the number of marks coming back and has a controller running at the sender which interprets these marks to decide the level of congestion and do a variable factor multiplicative decrease based on this interpretation.

BBRv2 congestion control algorithm uses an estimate of Bandwidth Delay Product (BDP) to minimize the queue delays. It responds to packet losses differently than ECN marks. BBRv2 sender follows DCTCP style adaptive rate decrease for ECN marks and follows CUBIC style fixed rate decrease (30%) for packet losses.

In [6] the authors compared the performance of ECN versus delay as the congestion feedback in the context of congestion control protocols for RDMA based transport, and proved a general impossibility results for delay based protocols which said that a pure delay feedback based protocol can achieve either fairness (via a unique fixed point) or a fixed operating delay but not both simultaneously. In this paper we go beyond that result and identify more limitations with a pure end-to-end feedback (like delay).

3 WHAT IS THE RIGHT FEEDBACK?

In this paper we compare the performance of three different feedback mechanisms. The feedback mechanisms that we compare are

(1) End-to-end feedback via delay
(2) In network feedback via ECN
  (a) ECN signal generated via a threshold mechanism as detailed in the design of DCTCP, denoted by ECN_t [5].
  (b) ECN signal generated via an Active Queue Mechanism as detailed in [2], which we denote as ECN_a.

We first present an analysis of the behavior of the three feedback signals along two dimensions, and then backup our analysis via experimental results.

3.1 Ability to reach a unique fixed point

If a combination of a protocol and feedback signal is able to reach a unique fixed point, then that translates into fair and predictable behavior for the congestion control protocol. For instance if there is a bottleneck link of capacity C shared by n flows, a unique fixed point would ensure that each flow receives a rate r_i of C/n. While fairness in of itself may not be a driving concern, having a unique fixed point ensures predictability in performance. If, on the other hand, a protocol leads to infinite fixed points as predicted in [6], then it leads to unpredictable performance. If a protocol has an infinite number of fixed points, the only thing that is guaranteed is that $\sum_i r_i = C$ (i.e., operates 'efficiently') and individual r_i's can take any value, leading to unpredictability.

3.1.1 End-to-end delay. In [6] the authors proved the impossibility result that any congestion control protocol which utilizes only delay as the feedback signal and tries to converge to a *fixed* delay at the bottleneck can either have that fixed delay or have fairness (i.e. a unique fixed point) but not both simultaneously. While the proof is in the paper, a high level explanation is that when a flow is sharing a bottleneck link with other flows and the only feedback is end-to-end delay, then at convergence (to a fixed delay) the feedback signal contains no additional information on how many other flows it is sharing the link with since the protocol is designed to converge to that fixed bottleneck delay regardless of the number of flows. Thus, concurrently converging to a fair operating point is not possible because there is no additional information to cause the system to deviate from the point of convergence to a fixed delay. This general impossibility result applies to all protocols that use delay as the (only) congestion signal, e.g., TCP-Vegas [7], Timely [8], BBRv1 [3], etc.

3.1.2 ECN_t. A threshold-based ECN feedback that was introduced in DCTCP [5], is attractive because of it being extremely simple. At the bottleneck link, the buffer is monitored, if the buffer size goes above a pre-defined threshold, then all packets above the threshold in the buffer are ECN marked. The ACKs reflect those ECN marks back to the sender. The sender in turn tracks the *average number* of marks received, and interprets the level of congestion in the network from that number. While the classical AIMD based mechanism [9] has a fixed multiplicative decrease constant on receiving of an ECN mark (or detection of a packet loss), DCTCP modifies this behavior by having a "level of congestion" dependent decrease in the sending rate. In the DCTCP paper the authors analyze the behavior of the protocol via a fluid model and show that it does not converge to a fixed point, but rather moves about in decreasing limit cycles. However, even with these limit cycles, DCTCP can suffer from unfairness. Consider a scenario where the flows are transmitting at rates that sum up to the link capacity, and the buffer is below the threshold. Now, when a new flow starts up the buffer will start growing, and go beyond the threshold. This growth will be happen at a rate proportional to the rate of the newly arrived flow, and eventually ECN marks will begin to be generated. However, existing flows will interpret these new marks as a lower level of congestion, since the "average" number of marks that it observes will include a period of no congestion. The newly arrived flow on the other hand will see a higher average, and consequently its multiplicative decrease factor will be higher than that of the older flows, which already have a higher rate. This is the opposite of what is intended to happen in classical AIMD - flows with a higher rate have a higher reduction during times of congestion. In general, ECN_t can lead to a scenario similar to end-to-end delay, in which the sum of the rates r_i is equal to the bottleneck link capacity, without a guarantee that the r_is are all the same.

3.1.3 ECN_a. With a classical AQM-based ECN (e.g., [10], [11]), the ECN mark is interpreted as a probability. Depending on the number of flows sharing the bottleneck, this probability is different and hence ECN_a implicitly conveys the number of flows sharing the bottleneck link to each flow in a distributed protocol, and hence protocols based on ECN_a can converge to a unique and fair fixed point, as shown in [6]. The throughput for each flow is given by a function of the round trip time and "loss" probability experienced by the flow, where the ECN_a marking probability is interpreted as the loss probability, as described in [12] for instance.

3.2 Ability to disambiguate congestion location

In this section we analyze the scenario where there are multiple possible bottlenecks on a flows path, and the protocol has to correctly identify the location and severity of the congestion, and take appropriate action to mitigate it.

3.2.1 End-to-end delay. Since end-to-end delay reports the aggregate of propagation delays and queueing delays on a flow's path back to the sender, it is not possible to break out the component queueing delays without additional information (e.g., as in [13]). In a scenario where the link capacities on a flow's path vary significantly, this can be a problem. Consider a scenario where a flow goes through both a 10 Gbps and a 100 Gbps link. If the flow experiences a delay increase of 1 ms, then the severity of congestion is 10 times worse than if that delay was because of queueing on the 100 Gig link rather than on the 10 Gig link. Consequently the nature of the reaction should be different for the two scenarios. However, since it is not possible to isolate the source of that increased delay, the congestion control protocol has to make a choice on the severity of the reaction. It can either react *too much*, or *too little* (thereby resulting in fast/slow reaction to congestion) if the choice of the reaction does not match the location of the congestion. This issue coupled with the previous problem of infinite fixed points can lead to severe unfairness in the behavior of these protocols.

3.2.2 ECN_t. In the same scenario as above, if ECN_t is used as the feedback signal, it can successfully disambiguate the location of the congestion. Let's assume the 10 Gbps link generates marks at an average fraction p_1 and the 100 Gbps link generates marks at the average fraction p_2. The fraction of ECN marks received at the sender would then be $1 - (1 - p_1)(1 - p_2) \approx p_1 + p_2$ if the average number of marks is small enough. If the additional 1ms delay is happening at the 10 Gbps link, then both p_1 and p_2 would be low and the reaction of the protocol would to be to make a minor adjustment in the sending rate. However, if the delay is being caused by the queue on the 100 Gbps link, then p_2 is likely to be high and the protocol would react by cutting the sending rate by a significant amount. Thus, ECN_t is able to disambiguate the location of congestion implicitly.

3.2.3 ECN_a. The exact same analysis that we presented for ECN_t works for ECN_a as well and it is also able to successfully disambiguate the nature and severity of congestion. Note that for ECN_a, typical values of p_i, the marking probability, in well configured systems is very low, and hence the approximation $1-(1-p_1)(1-p_2) \approx p_1+p_2$ is likely to be valid more than for ECN_t. Hence, ECN_a can do a better job of disambiguating the congestion location and severity than ECN_t.

In the next sections, we present experimental results which clearly demonstrate the behavior that our analysis predicts for the three protocols. We use BBRv2 without ECN as our pure end-to-end delay based protocol, BBRv2 with ECN turned on as the protocol with ECN_t as the feedback signal, and CUBIC with ECN_a generated at the bottleneck routers by a PI controller [10] as our example of a protocol which uses ECN_a as the feedback signal. We pick PI as the AQM system since it is able to decouple the queue length from the congestion feedback, and is the AQM of choice in the DOCSIS 3.1 standard [11] and is widely deployed on cable modems across the world.

4 MODEL AND EVALUATION FRAMEWORK

To evaluate the effectiveness of the different approaches, we use the Flent [14] tool with a testbed of emulated data sources, sinks and routers that are setup using network namespaces. The implementation of both TCP and the congestion control component of the router use the current Linux kernel-based distribution. The testbed emulates a multiple bottleneck environment, using a hybrid Data Center-WAN configuration as shown in Fig. 1. One bottleneck link is in the DCN (G1-G2) and the other in the WAN (G3-G4). All links in DCN have a capacity of 1Gbps, while the WAN link has a capacity of 50 Mbps. The propagation delay for the DCN link is 2μs, and 500μs for the WAN link. All the DCN end-point to switch links have a propagation delay of 1μs, while the end-point to switch links in the WAN are 10μs. Google's Github repository[2] with the Linux kernel with the alpha release of BBRv2 (and CUBIC TCP) is used for all our experiments.

The DCN has four senders (S1 to S4) with three of the flows remaining entirely in the DC (going to three receivers R2 to R4). There are 3 flows that are solely in the WAN (senders S5 to S7 to receivers R5 to R7). One flow spans both the DC and the WAN and this is a long lasting TCP flow from S1 to R1. This flow crosses both the bottleneck links. To show the responsiveness of the congestion control algorithms, we make most of the TCP flows – starting from sources S2 to S7 flows to be short lived, having an ON and OFF duration of 12 seconds each. The S2 to S4 flows start and end in

[2] https://github.com/google/bbr/tree/v2alpha

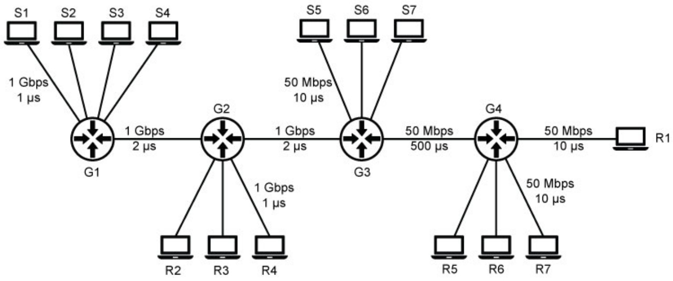
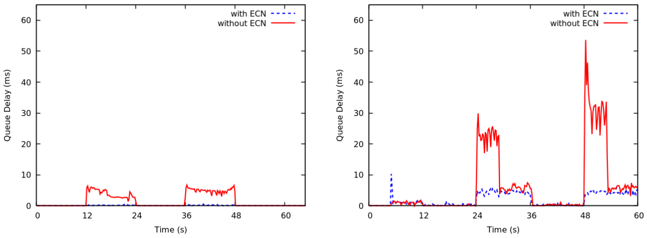


Figure 1: Hybrid DC-WAN testbed

DCN, and S5 to S7 flows start and end in WAN, only traversing a single bottleneck link.

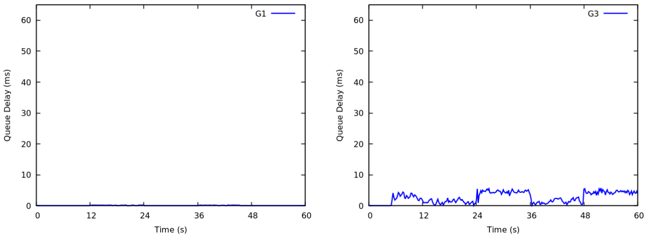
We compare the performance of BBRv2 (with and without ECN) vs CUBIC (with PI+ECN) in terms of controlling the queue on bottleneck links and the proportional fairness across the competing flows in terms of throughput. The reference queue length for ECN marking is set to 20 packets for all experiments, with packet size set to 1500 bytes (inline with the default marking threshold in BBRv2.)



(a) DC router

(b) WAN router

Figure 2: BBRv2 with and without ECN: Queue Delay



(a) DC router

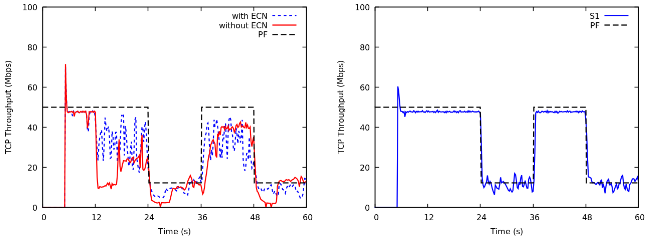
(b) WAN router

Figure 3: CUBIC with PI+ECN: Queue Delay

5 RESULTS

5.1 Queueing Delays

We first show the queueing delays at the two bottleneck routers, the Data Center (DC) router and WAN router in Figures 2 and 3. As can be seen, environments where ECN is used the queue delay is much better controlled. The queue with an AQM such as PI (Fig. 3) is consistently below or very close to the desired set-point (20 packets, 5 ms on the WAN router). The end-to-end approach (BBRv2 without ECN) has a much more difficult time with keeping the queues small at the bottleneck routers.



(a) BBRv2 with and without ECN

(b) CUBIC with PI+ECN

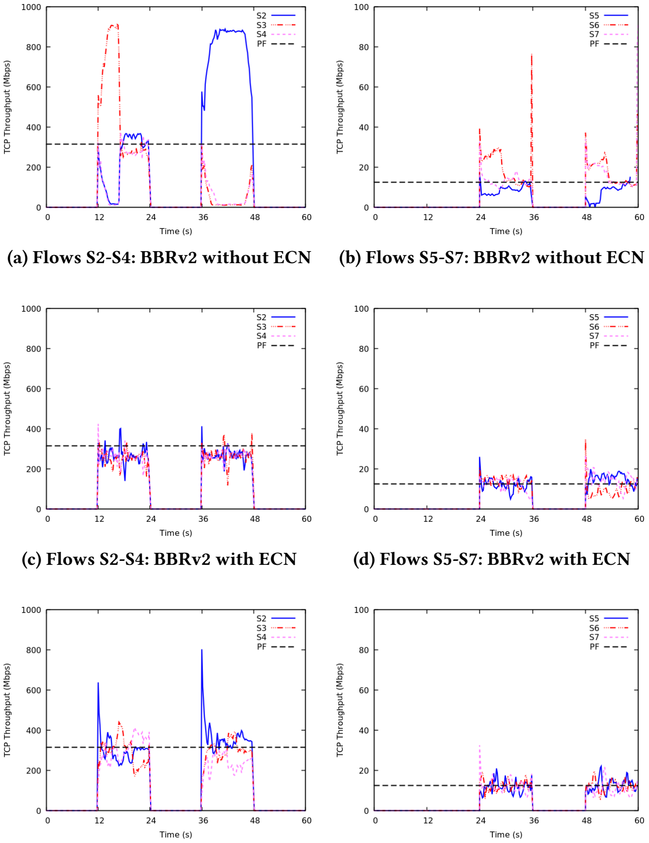
Figure 4: Flow S1: Throughput

5.2 Throughput of Flows: end-to-end delay and ECN_t as feedback signals

Now we investigate the throughput of the different flows in our experiment. Recall that S1 is a flow which is a hybrid one, i.e. it spans both the data center as well as the WAN and it starts at 5 seconds and lasts during the duration of the experiment subsequently. Flows S2, S3 and S4 are data center flows which share a link with S1, and they go on and off as described in Section 4, and flows S5, S6 and S7 are WAN flows which again share a (different) bottleneck link with S1. The idea of the experiment is to generate congestion at different locations and observe if the protocol is able to react appropriately. In all the throughput figures we mark the Proportional Fair (PF) share of the flows and observe if the protocol is able to achieve it. In Figures 4 and 5 we plot the throughputs of the different flows looking at the BBR variants. We can see in Figures 5(a) and (b) that when end-to-end delay is the only feedback signal, neither the data center flows and the WAN flows achieve their PF share, nor do they converge to a unique fixed point. In Figures 5(c) and (d) we see that the introduction of the ECN_t feedback signal lets the flows converge to a unique (average) fixed point around the PF share. The data center flows obtain a throughput slightly higher than their PF share which is not obvious from their throughput graph, but can be inferred from the throughput graph of the hybrid flow S1 in Figure 4(a). We can observe that when there is no other flow competing with it, then S1 achieves close to its PF share. It also achieves close to its PF share when it is competing with the WAN flows. However when it is competing with the data center flows it achieves *less* than its PF share, which is predicted by our analysis as a problem with both end-to-end as well as ECN_t signals, which is the unpredictability in performance due to multiple fixed points.

5.3 Throughput of Flows: ECN_a as feedback signal

In this experiment we use CUBIC with an ECN_a signal generated by the PI controller. As we can see from all the throughput related plots in Figure 4 and Figure 5, the protocol is able to react appropriately in all scenarios and all flows achieve their PF share at all times. Note that the DC flows show a somewhat high variance in their throughput, but that is a limitation of our experimental



(a) Flows S2-S4: BBRv2 without ECN

(b) Flows S5-S7: BBRv2 without ECN

(c) Flows S2-S4: BBRv2 with ECN

(d) Flows S5-S7: BBRv2 with ECN

(e) Flows S2-S4: CUBIC with PI+ECN

(f) Flows S5-S7: CUBIC with PI+ECN

Figure 5: Flows S2-S7: Throughput

framework where the kernel is not able to update the PI controller probability at the required frequency for the data center network. This results in a controller which has somewhat slower dynamics than what is appropriate for data center environments. However, this is not a problem with modern switching hardware, e.g. based on the Barefoot Tofino-2 ASIC. Another point in favor of ECN_a over ECN_t is that ECN_t is a static and stateless signal and reacts to the instantaneous queue. ECN_a generated from controllers like PI are stateful and can *anticipate* the onset or conclusion of congestion events by tracking derivatives of queue lengths and hence are able to react faster than ECN_t.

6 CONCLUSIONS

In this paper we took a principled look at three different kinds of feedback signals that are used in networks: end-to-end delay, ECN_t, and ECN_a. Our analysis, backed up by experimental results show that ECN_a is the most desirable form of congestion signal for networks, as it is able to both deliver fairness as well as disambiguate the location and severity of congestion in the network. The next signal in order of preference is the ECN_t which is able to provide the disambiguation, however it can suffer from the problem of multiple fixed points and hence unfair performance. Pure end-to-end delay is the most inadequate signal in terms of performance. Unsurprisingly, the computational requirements for these signals on routers/switches are in the exact reverse order of desirability with end-to-end delay requiring no computations at all and ECN_a requiring the most. However, modern hardware is capable of performing the needed calculations at line rate and protocols should be designed with ECN_a as a feedback signal.

ACKNOWLEDGMENTS

This work was supported in part by NSF Grants CNS-1763929 and CNS-1618911.

REFERENCES

[1] Van Jacobson. Congestion Avoidance and Control. In *ACM SIGCOMM Computer Communication Review*, volume 18, pages 314–329. ACM, 1988.

[2] K Ramakrishnan, S Floyd, and D Black. RFC 3168. *The addition of Explicit Congestion Notification (ECN) to IP. The Internet Society*, 2001.

[3] Neal Cardwell, Yuchung Cheng, Soheil Yeganeh, and Van Jacobson. BBR Congestion Control. *Working Draft, IETF Secretariat, Internet-Draft draft-cardwell-iccrg-bbr-congestion-control-00*, 2017.

[4] Sangtae Ha, Injong Rhee, and Lisong Xu. CUBIC: a new TCP-friendly High-Speed TCP Variant. *ACM SIGOPS Operating Systems Review*, 42(5):64–74, 2008.

[5] Mohammad Alizadeh, Albert Greenberg, David A Maltz, Jitendra Padhye, Parveen Patel, Balaji Prabhakar, Sudipta Sengupta, and Murari Sridharan. Data Center TCP (DCTCP). *ACM SIGCOMM Computer Communication Review*, 41(4):63–74, 2011.

[6] Yibo Zhu, Monia Ghobadi, Vishal Misra, and Jitendra Padhye. ECN or Delay: Lessons Learnt from Analysis of DCQCN and TIMELY. In *Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies*, pages 313–327. ACM, 2016.

[7] Lawrence S. Brakmo and Larry L. Peterson. TCP Vegas: End to End Congestion Avoidance on a Global Internet. *IEEE Journal on Selected Areas in Communications*, 13(8):1465–1480, 1995.

[8] Radhika Mittal, Nandita Dukkipati, Emily Blem, Hassan Wassel, Monia Ghobadi, Amin Vahdat, Yaogong Wang, David Wetherall, David Zats, et al. TIMELY: RTT-based Congestion Control for the Datacenter. In *ACM SIGCOMM Computer Communication Review*, volume 45, pages 537–550. ACM, 2015.

[9] K. K. Ramakrishnan and Raj Jain. A Binary Feedback Scheme for Congestion Avoidance in Computer Networks with a Connectionless Network Layer. In *ACM SIGCOMM Computer Communication Review*, volume 18, pages 303–313. ACM, 1988.

[10] Chris V Hollot, Vishal Misra, Don Towsley, and Wei-Bo Gong. On designing improved controllers for AQM routers supporting TCP flows. In *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No. 01CH37213)*, volume 3, pages 1726–1734. IEEE, 2001.

[11] Rong Pan, Preethi Natarajan, Chiara Piglione, Mythili Suryanarayana Prabhu, Vijay Subramanian, Fred Baker, and Bill VerSteeg. PIE: A Lightweight Control Scheme to address the Bufferbloat Problem. In *IEEE 14th International Conference on High Performance Switching and Routing, HPSR 2013, Taipei, Taiwan, July 8-11, 2013*, pages 148–155, 2013.

[12] Jitendra Padhye, Victor Firoiu, Don Towsley, and Jim Kurose. Modeling TCP Throughput: A Simple Model and Its Empirical Validation. *SIGCOMM Comput. Commun. Rev.*, 28(4):303–314, October 1998.

[13] Yuliang Li, Rui Miao, Hongqiang Harry Liu, Yan Zhuang, Fei Feng, Lingbo Tang, Zheng Cao, Ming Zhang, Frank Kelly, Mohammad Alizadeh, et al. HPCC: High Precision Congestion Control. In *Proceedings of the ACM Special Interest Group on Data Communication*, pages 44–58. ACM, 2019.

[14] Toke Høiland-Jørgensen, Carlo Augusto Grazia, Per Hurtig, and Anna Brunstrom. Flent: The Flexible Network Tester. In *Proceedings of the 11th EAI International Conference on Performance Evaluation Methodologies and Tools*, pages 120–125. ACM, 2017.