On the Incompatibility of Scalable Congestion Controls over the Internet (Extended version)

Ferenc Fejes, Gergő Gombos, Sándor Laki,
Department of Information Systems
ELTE Eötvös Loránd University
Budapest, Hungary
{fejes, ggombos, lakis}@inf.elte.hu

Szilveszter Nádas
Traffic Analysis and Network Performance Laboratory
Ericsson Research
Budapest, Hungary
szilveszter.nadas@ericsson.com

***Abstract*—In addition to classic ones, a new family of congestion controls regarded as "scalable" has recently emerged. Scalable congestion control exploits ECN to provide much finer control as rate scales, enabling reactions proportional to the congestion level. Nowadays, different initiatives propose the use of Scalable congestion control over the public Internet, with the separation of Scalable and Classic flows. Similar to classic approaches, the evolution of Scalable congestion control has also begun as BBRv2 implemented a "DCTCP-inspired" ECN mechanism. In this paper, we investigate the compatibility of two available Scalable congestion controls, Data Center TCP, and Google's recent proposal called BBRv2, and evaluate them with different ECN marking strategies. Our evaluation relies on numerous measurements carried out in our testbed with 1 to 10Gbps bottleneck capacities and heterogeneous round trip times. At the bottleneck, the ECN marking of two traditional AQMs (STEP and PI2) most commonly proposed for Scalable flows and a non-traditional core-stateless AQM (CSAQM) using in-network resource sharing control is examined. After showing compatibility issues, we conclude that as long as the end-to-end flows control resource sharing, it will be tough to create a new, evolved congestion control that is fair to legacy solutions, and its deployment does not harm the Internet.**

***Index Terms*—Scalable Congestion Control, AQM, Fairness, BBR, DCTCP**

I. Introduction

Jacobson and Karels proposed congestion avoidance and control in their seminal paper in 1988 [1] and saved the Internet from congestion collapse. Since then the Internet has gone through an enormous evolution and became one of the most complex systems ever built by humans. Today's Internet faces lots of challenging problems to satisfy the various requirements and expectations of end users. As the technology evolved, changing NIC capacities from Mbps to Gbps and memory from KBs to GBs, buffers in the routers have become larger and larger. In addition to loss-based congestion detection, approaches based on Explicit Congestion Notification (ECN) have also emerged.

Deployment of new Congestion Control (CC) algorithms has accelerated in the past few years, with the aim of allowing users to exploit the benefits of high-speed links with acceptable end-to-end delay. The scalable Data Center TCP (DCTCP) CC was originally designed for Data Centers, but in recent years it is considered to be used over the Internet [2]. Low Latency, Low Loss, Scalable Throughput (L4S) Internet Service [3] also aims to allow classic and scalable TCP flows to coexist over the same Internet bottleneck. DCTCP was not designed for the Internet, it assumes extremely low end-to-end latency and thus fast feedback loop. To make its scalable CC mechanism suitable for usage over the Internet, TCP Prague [4] has recently been proposed.

Among the different new CC proposals, one of the most prevalent is BBR (Bottleneck Bandwidth and Round-trip propagation time) [5] that applies a complex congestion model instead of simple packet losses. In contrast to its initial version, BBRv2 also implements a "DCTCP-inspired ECN" mechanism [6]. The compatibility of BBRv2 and Cubic CCs has been demonstrated in [6] and analyzed in more detail in [7]. However, we are unaware of any published results when it comes to the compatibility of BBRv2 and DCTCP. A harm-based threshold is introduced in [8] to quantify the deployability of new CC algorithms in the Internet. However, the practical applicability of this insight is still challenging.

With the introduction of 5G for mobile and Fiber To The Home (FTTH) for fixed Internet access, the capacity of the last mile has been significantly increased, resulting in much higher load on the Access-Aggregation Network (AAN) than before and thus moving bottlenecks to routers in the AAN from the edge. In such a network, the CCs used by the flows and the (propagation) round trip times (RTTs) are much more heterogeneous than in data centers and other closed enterprise networks where the whole infrastructure is controlled by a single entity. To handle the increased load on AANs and serve these high-speed bottlenecks, new router equipment is needed where reasonable fairness among traffic aggregates (e.g., users, TCP flows) is again an important factor in the design. The physical size of the buffer is also a question to minimize both cost and latency.

In this paper, we explore a heterogeneous environment where different CC and AQM solutions coexist under various network conditions (bottleneck capacity and RTTs). We also examine the efficiency of recent AQMs and show potential issues: DCTCP which is designed with Data Center conditions in mind, has fairness issues when the RTTs are higher or

ISBN 978-3-903176-28-7

heterogeneous. BBRv2 has similar but less serious problems. When mixing BBRv2 and DCTCP, even having equal RTTs results in unfairness and with heterogeneous RTTs this problem escalates. We also show that a recent Core-Stateless AQM proposal called CSAQM that applies resource sharing control can solve these issues in general by harmonizing CC behavior, but it requires assistance from both end-hosts and network routers, resulting in strong constraints on the Internet-wide deployment. Interestingly in some cases (e.g., large RTT) even CSAQM cannot help to achieve good fairness. Another issue we have found is that the recently introduced loss mode of DCTCP makes buffer sizing for scalable flows harder.

We believe that while the idea of harm-based fairness [8] is good, the expectation that a novel CC should never harm existing CCs, unnecessarily slows down CC evolution. Instead, we highlight that if the network can enforce fairness among traffic aggregates, much faster CC innovation can be enabled.

II. Congestion Control Algorithms

In this paper, we aim to investigate the compatibility of different Scalable CCs, selecting two such CC algorithms: DCTCP is the de-facto standard scalable CC, while the recent BBRv2 also supports scalable response to ECN signals.

DCTCP [9]. It was originally designed for data center networks. For practical consideration, it assumes ECN marking with STEP AQM in the routers. [10] shows that the target delay in STEP AQM shall be at $K \times RTT, K \geq 0.17$ to achieve 100% utilization, but the utilization remains higher than 94% even for K as small as 0.01. During its design the maximum buffer length was not studied or optimized.

In the absence of ECN Congestion Encountered (CE) signal DCTCP reuses the Additive Increase of Reno. When ECN CE is experienced, the decrease of the congestion window (*cwnd*) is proportional to the averaged fraction of marked received packets (α), $\alpha = (1-g) \times \alpha + g \times F$, where F is the fraction of marked packets in the last RTT and $g = 1/16$ by default. The reduced *cwnd* is calculated as $cwnd = cwnd \times (1 - \alpha/2)$. Important to note that since Linux kernel 5.1 DCTCP also reacts to real packet loss by halving the *cwnd*.

BBRv2 [6]. Rather than using events such as loss or buffer occupancy, which are only weakly correlated with congestion, BBR starts from Kleinrock's formal model of congestion and its associated optimal operating point [5]. That operation point is reached when the link utilization is high while it avoids creating (long) queues at the bottleneck. BBR version 1 does not treat packet loss or ECN CE as a congestion signal and it has known compatibility problems with Cubic [11]. BBR version 2 aims to provide fair throughput sharing among BBRv2 and Cubic connections; and it also implements a Scalable "DCTCP-inspired ECN" mechanism [6].

Though it was never claimed that BBRv2 is DCTCP compatible, we are not aware of any publication investigating fairness between BBR and DCTCP. BBR's α is determined similarly to that of DCTCP (using the same default g). However, its effect on the *cwnd* is smaller, $cwnd = cwnd \times (1 - f_{ECN} \cdot \alpha)$, where $f_{ECN} = 1/3$ by default. In addition, BBRv2 increases the *cwnd* in a very different way from DCTCP. Considering these differences, it is quite unlikely that DCTCP and BBRv2 would share a bottleneck fairly.

BBRv2 is likely not the final CC in the current CC evolution. Furthermore, with more and more user space CC implementations one can expect flows with highly heterogeneous CCs present in the AAN.

Note on TCP Prague [4]. DCTCP implementation in the Linux Kernel has undergone a significant transformation since its original version in Kernel 3.X. However, it was not updated for compatibility with the Internet, keeping the applicability of DCTCP limited to data centers or similar low-latency environments. TCP Prague is a recent CC proposal that extends DCTCP to be compatible with the Internet. Though the implementation of TCP Prague is in an experimental, early-stage and currently cannot be deployed over the Internet, TCP Prague may solve several issues of DCTCP, including the fairness problems among flows with different RTTs and faster increase of congestion window.

III. AQM Algorithms

Four queue management strategies are considered in this paper: STEP, the one used during DCTCP design; PI2, which was designed to be compatible with both classic and scalable CCs (including DCTCP); a recent core-stateless AQM proposal (CSAQM), whose in-network control of resource sharing enables the co-existence of different CC approaches including both scalable and classic CCs; and tail dropping as reference in some scenarios (TailDrop). The first three AQMs are ECN capable, and have a delay target parameter, which is set to $D_{target} = K \cdot RTT$, where K is a parameter we change. We translate it to queue length target by $L_{target} = D_{target} \cdot C$, where C is the capacity of the outgoing link.

STEP AQM marks all incoming packets when the queue size is above L_{target}. It is the reference AQM considered during the design of DCTCP [9]. STEP AQM behavior is achieved by configuring the widely deployed RED (Random Early Detection) AQM as proposed in [9].

PI2 (Proportional Integral improved with a square [12]) was designed to support the coexistence of Classic and Scalable CCs. It has been shown to work in environments with Scalable CC flows only [12], though its response might be too slow to achieve low latency in dynamic scenarios. It uses a PI controller to determine a base probability (p_{base}) periodically to maintain the target delay. This base probability is then applied to derive the ECN-marking/dropping probabilities for scalable ($k \cdot p_{base}$) and classic (p_{base}^2) TCP sources. We set the update period to 10 ms, otherwise, we use the default controller parameters proposed in [13] ($\alpha = 0.16$ and $\beta = 3.2$). We repeated several tests with different update periods (including 16 ms as in [13]) α and β values, but the results did not change significantly.

CSAQM (Core-Stateless AQM [14]) was developed to support controlled resource sharing for flows with arbitrary CC. In contrast to previous solutions, it requires two key mechanisms inside the network: 1) packets are marked with a packet

value (PV) before the bottleneck (e.g. at the edge of the network) where the value is derived from the per connection resource sharing policy to be applied. We configured equal resource sharing for all flows in our tests. 2) At the bottleneck packet handling is solely based on the carried PV, connection identification and policy knowledge is not needed. When the queue length exceeds the delay target, packets with the lowest PVs are ECN CE marked until the amount of non-CE marked packets is less than the delay target. This behavior is approximated by calculating a Congestion Threshold Value (CTV) periodically (every 5 ms), and marking all outgoing packets, which have $PV < CTV$ (see details in [14]). CSAQM by design does not mark packets from connections with throughput below their fair share. Thus even if the packet marking rate is high, none of those are affecting new connections in slow start, see e.g. Figure 5 in [14].

TailDrop is used as reference in our analysis. TailDrop simply drops the incoming packet once the queue is full. It is not ECN capable. Buffer sizing of TailDrop queues has widely been studied in the past decades [15].

Note on DualPI2 We do not evaluate DualQ AQMs [3], because all the investigated CCs are scalable, thus all the traffic would go to the low delay, scalable queue. DualPI2 [13] uses STEP AQM in its scalable queue and PI2 as the coupled AQM. Consequently, the results with STEP AQM are more relevant to understand the performance of DualPI2.

IV. Testbed description

Our testbed consists of 3 servers, all of them being Xeon E5-1660 v4@3.2GHz, 64Gb RAM with Intel XL710 40GbE Ethernet NIC. They run Ubuntu Server 18.04 and are connected in a chain topology. The two ends, the sender and the receiver, use BBRv2 alpha kernel (5.4.0-rc6) including BBRv2 ECN support. We aim to keep Linux default settings, when possible: Generic Segmentation Offload (GSO) and Generic Receive Offload (GRO) was enabled and pacing for DCTCP was disabled (BBR has internal pacing). We repeated several tests with GRO and GSO disabled, but the results did not change. The middle machine acts as a router and uses Linux kernel 4.15 (Ubuntu 18.04 default) and it executes our DPDK-based AQM and bottleneck emulator. We implemented the AQM algorithms of Section III on the top of DPDK 19.11. We use iperf2 to generate traffic, we modified it in a way that each flow starts its transmission after a random delay selected uniformly from 0 to 1 seconds. The propagation RTT is emulated with Linux tc netem [16] by delaying ACKs on the receiver side (queue limit $= 10^6$ packets). We set the tc netem queue limit $= 10^6$ packets making sure that no ACK are dropped due to the overflow of the delay emulation queue. We perform 3 minutes long experiments, and measure the total transmitted bytes; the ECN marked bytes; and the dropped bytes per flow in the AQM implementation. We also create a histogram of the packet sojourn times (1 ms resolution), and measure the bottleneck utilization there.

V. Experimental Analysis

To analyze the behavior of DCTCP and BBRv2 with the different ECN-marking strategies of selected AQM methods, we have carried out experiments with various network conditions (bottleneck capacities: 1 Gbps and 10 Gbps; and (propagation) RTTs: 5 and 50 ms). The delay target is defined as a delay factor (K) times the base RTT where the examined K values are 0.05, 0.1, 0.17, 0.25, 0.5, 1 and 2 (e.g., $K = 0.5$ means that the delay target is half of the RTT). By default the buffer length is set to 500 ms, mimicking almost infinite buffers to avoid packet losses. In the small buffer cases the buffer length is set to 1.33 times the delay target. The number of TCP connections (N) in different test cases are 2, 10, 20 and 100. With respect to the applied CCs, we introduce two test cases: 1) *mono-CC tests* where all the connections use the same CC and 2) *multi-CC tests* in which half of the connections apply BBRv2 while the other half uses DCTCP. In *Multi-RTT* setups connections with different propagation RTTs (5 ms and 50 ms) are present in the evaluation.

We analyze various metrics: relative throughput of the connection classes; system utilization; average and maximum queueing delays; ECN-marking and packet-drop ratios; and the Jain's fairness index (among all connections). A *connection class* contains all TCP flows with the same CC and the same base RTT. *Relative throughput* is defined as the ratio of the average throughput within a connection class, and the ideal per-connection fair-share (the bottleneck capacity per the total number of TCP connections N). For example, relative goodput 1.0 means that the connections of the given class experience the fair-share of the total capacity in average. The average delay presented in the figures is relative to the delay target applied in the given scenario.

Due to space limitation we only present some selected results, but all further cases are available at [17].

A. Mono-CC experiments with equal RTT

We first consider DCTCP flows with base RTT of 5 ms and 1 Gbps bottleneck capacity. One can observe in Figure 1 and 2, that though the fairness among the connections are quite good in most cases, the utilization is affected by the number of DCTCP flows and the applied delay target. The worst utilization can be seen for small target delays with limited number of flows (2 and 10). $K = 0.17$ with STEP AQM reflects the recommended ECN-marking setup for DCTCP, resulting in utilization below 60% for two flows. Surprisingly, the STEP AQM used during the DCTCP design provides the smallest utilization. Figure 3 indicates that all the investigated AQMs keep the queueing delay below the target value (1 in the figure) in most of the cases. STEP looks somewhat conservative while PI2 cannot properly control the delay for 2 flows when the target value is small. Figure 4 shows that the utilization in the small buffer case is even worse. We believe this is due to the halving the *cwnd* in case of packet loss. Strangely, the buffer size requirement for decent utilization is larger for DCTCP than for Cubic (see [7]).

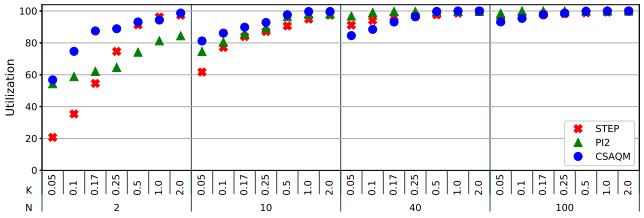

Figure 1. Mono-DCTCP utilization (1Gbps, 5ms).

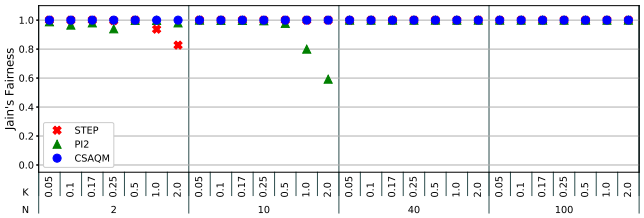

Figure 2. Mono-DCTCP fairness (1Gbps, 5ms).

Note that we have also examined the case when the propagation RTT is increased to 50 ms. Though the utilization becomes higher, the fairness is significantly degraded, except for CSAQM (see [17]).

Figure 5 shows the (1 Gbps, 5 ms) setup with BBR CC. One can observe that the utilization is higher than for DCTCP, even for small target delays and small number of TCP connections. Note that we have got acceptable fairness for all the investigated AQMs (see [17]). However, when the RTT is increased to 50 ms, as depicted in Figure 6, STEP and PI2 AQMs result in significant unfairness for some settings. With STEP, Jain's fairness index only starts reducing as the delay target increases. With PI2, it is not so clear; when the number of connections is small (2 and 10), the observed behavior is similar to the one of STEP, but for 40 and 100 connections, the obtained unfairness is significant and seems independent of the target delay parameter. Note, the target delay is properly kept by STEP and CSAQM, but PI2 exceeds delay targets for small Ks and leads to underutilization in these cases.

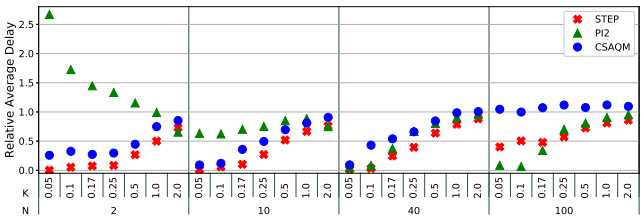

Figure 3. Mono-DCTCP avg. delay (1Gbps, 5ms).

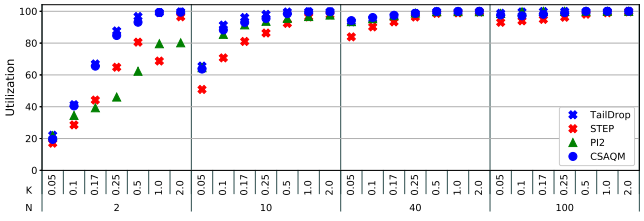

Figure 4. Mono-DCTCP utilization (1Gbps, 5ms, small buff.).

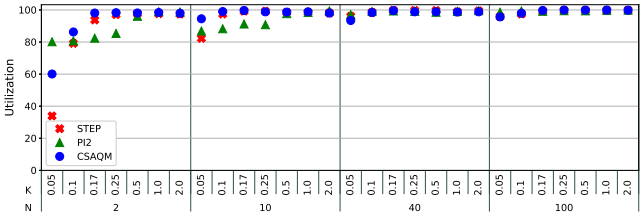

Figure 5. Mono-BBR utilization (1Gbps, 5ms).

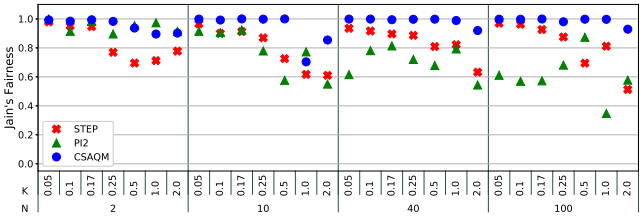

Figure 6. Mono-BBR fairness (1Gbps, 50ms).

B. *Multi-CC experiments with equal RTT*

Figures 7-9 depict Jain's fairness index, relative throughput and utilization for the multi-CC test in which multiple CCs coexist in the same system with 1 Gbps bottleneck capacity and 5 ms RTT settings. Similarly to the mono-BBR case, the utilization is quite good for all the AQMs, significant deviation from full utilization can only be seen for small number of connections with small delay targets. Except CSAQM, significant unfairness can be observed. Figure 8 shows that the different ECN-marking strategies of STEP and PI2 lead to different behavior; while STEP favors DCTCP flows in most cases, in PI2 DCTCP flows are suppressed by the BBR traffic, esp. for small number of flows. Increased target delay ($K = 2$) can help PI2 to achieve an acceptable 1:3 DCTCP-BBR fairness ratio. Through its in-network resource sharing mechanisms, CSAQM can apply different empirical marking probabilities for the dissonant flows, taking their specific behavior into account and thus resulting in good fairness even among BBR and DCTCP traffic. Figure 10 shows this phenomenon in

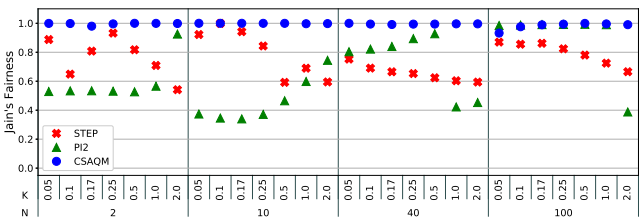

Figure 7. Multi-CC fairness (1Gbps, 5ms).

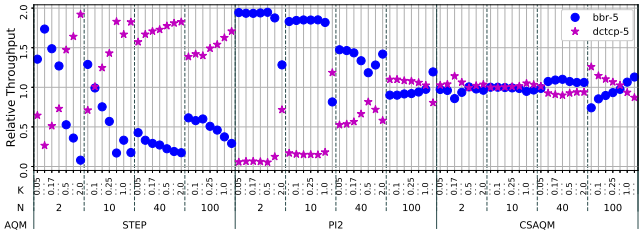

Figure 8. Multi-CC relative throughput (1Gbps, 5ms).

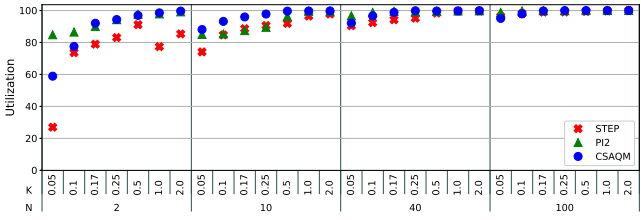

Figure 9. Multi-CC utilization (1Gbps, 5ms).

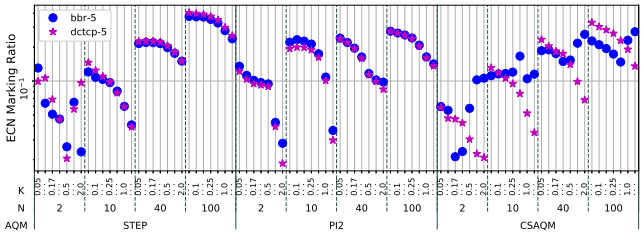

Figure 10. Multi-CC ECN marking ratio (1Gbps, 5ms).

more detail. While PI2 and BBR result in the same ECN-marking ratio for both BBR and DCTCP, CSAQM can find the appropriate marking ratio for the different classes. One can also observe that the marking ratios required for the two CCs rely on both N and K, indicating fundamental differences on how ECN-CE is handled by them.

Figures 11-13 depict similar multi-CC cases with different bottleneck capacities (1 Gbps and 10 Gbps) and RTT values (5 ms and 50 ms). One can observe that larger RTT helps both STEP and PI2 in achieving slightly better fairness, but at 10 Gbps the fairness among the two classes is generally worse than at a 1 Gbps bottleneck. PI2 only shows some improvements if delay target factor K is large. One can also notice that drawing a general rule of thumb for appropriate parameter settings seems hard or even impossible in case of STEP and PI2. However, CSAQM can still reach good fairness in most cases, and is less prone to its parameter settings. It is not perfect, but reasonable even for 50 ms RTT and low number of flows.

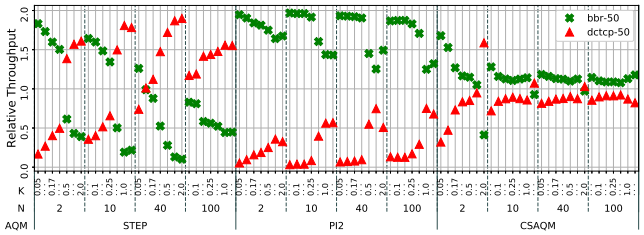

Figure 11. Multi-CC relative throughput (1Gbps, 50ms).

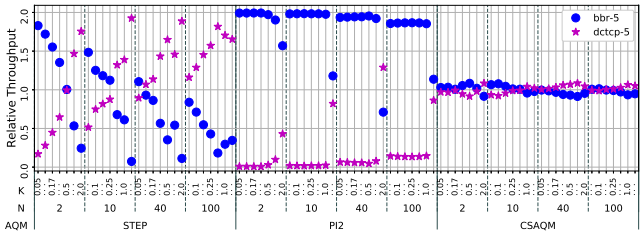

Figure 12. Multi-CC relative throughput (10Gbps, 5ms).

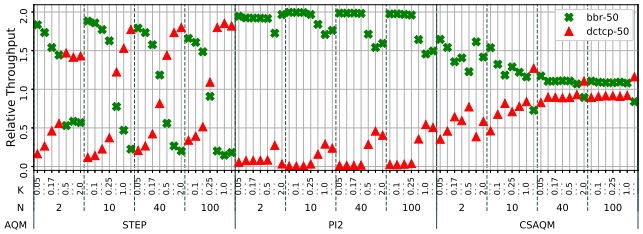

Figure 13. Multi-CC relative throughput (10Gbps, 50ms).

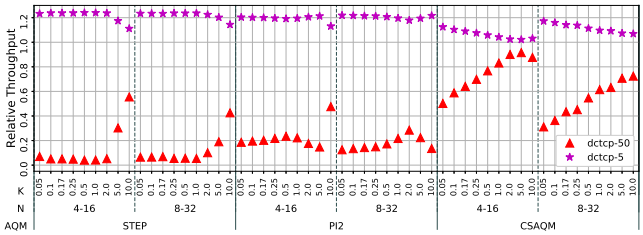

Figure 14. DCTCP relative throughput (1Gbps, 5&50ms).

C. Heterogeneous RTTs

In contrast to previous sections where equal RTT has been assumed, in this section we consider connections with multiple propagation RTTs: either 5 ms or 50 ms. N is 4-16 (4 with 50 ms RTT and 16 with 5 ms) or 8-32, respectively. Note that the target delay is set according to the smaller RTT (5 ms), and we also investigate the K values of 5 and 10. Figures 14 and 15 depict the relative throughput with mono-CC tests using either DCTCP or BBR CCs, respectively. One can observe that in case of DCTCP CC, connections with 5 ms RTT reach much higher throughput than with 50 ms as expected. Large delay targets in STEP and PI2 can slightly improve the fairness between the flow classes of different RTT. One can also see that even CSAQM can only handle large differences in the RTTs if the delay target is large (around the maximum RTT).

In the BBR case, both STEP and PI2 show much better fairness, the 50 ms RTT flows get somewhat higher share in general and as the target delay increases it becomes even worst. CSAQM can enforce good fairness as long as $K < 1$ (until the delay-based congestion detection of BBR is not turned on). Interestingly, BBR with CSAQM can reach very good fairness even for extremely small delay targets.

One can also observe that while DCTCP flows with smaller RTT suppresses the others with larger RTT, in BBR the opposite behavior can be experienced, also reflecting fundamental differences between these CCs.

Figures 16 and 17 show the results of multi-CC and multi-

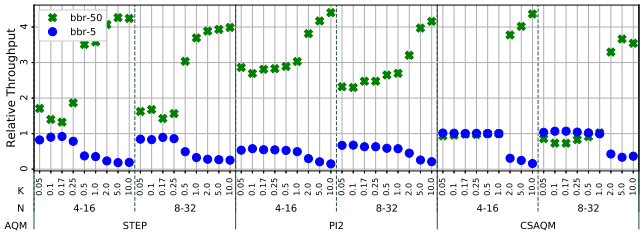

Figure 15. BBR relative throughput (1Gbps, 5&50ms).

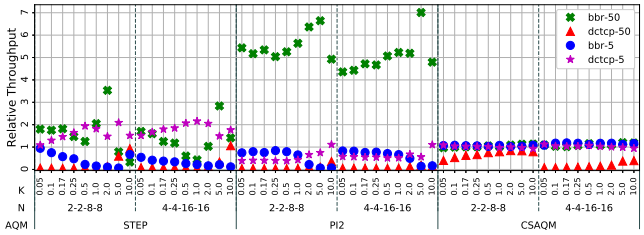

Figure 16. Multi-CC, multi-RTT rel. throughput (1Gbps).

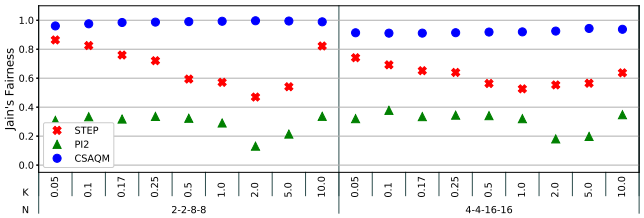

Figure 17. Multi-CC, multi-RTT fairness (1Gbps).

RTT cases. N is 2-2-8-8, indicating the number of connections in CC-RTT groups: BBR-50ms, DCTCP-50ms, BBR-5ms and DCTCP-5ms, respectively. One can observe that fairness with STEP and PI2 is quite bad in general. DCTCP-50ms gets very small throughput and in some cases BBR-5ms flows are also suppressed. CSAQM provides reasonable fairness for all connection groups, except for DCTCP-50ms when the number of flows is high (4-4-16-16). It is interesting to note that PI2 gives the BBR-50ms class unfairly high share, while STEP is free of this symptom. Figure 18 shows that the fairness in the small physical buffer case is better than in case of long buffers for STEP and PI2, while for CSAQM it consistently remains reasonable. This suggests that the loss mode of both DCTCP and BBRv2 CCs behaves more fairly than their scalable reaction to ECN CE.

VI. Speculation: How to save the Internet?

The utilization with DCTCP was much worse than expected, according to [10]. In general, the Linux 5.4 DCTCP implementation seems a poor choice for Internet RTTs, even when a separate buffer (e.g., DualQ) is available for Scalable CCs.

The Scalable mode of DCTCP and BBRv2 use different congestion window decreasing and increasing algorithms and parameters. It is not surprising that their compatibility is weak, though we hoped to have a "Beautiful Friendship" in the title. After analyzing the performance of DCTCP, we believe that it was a right choice not to mimic DCTCP behavior in BBRv2.

TCP Prague [4] addresses several issues of DCTCP and intends to be the "Internet DCTCP version". As BBRv2 has

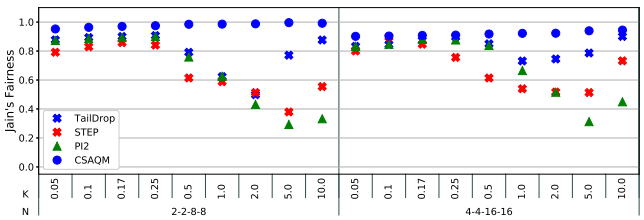

Figure 18. Multi-CC, multi-RTT fairness (1Gbps, small buff.).

several tunable parameter, it might be possible to tune these two scalable CCs to be compatible in specific scenarios. We are skeptical though that this compatibility can be generic, for several RTTs and traffic mixes.

Based on the results of this paper and [7], we believe that the requirement that a new CC should be fair to existing CCs is hard (if not impossible) to meet. Furthermore, the harm-based approach of [8] is softer than full fairness, but even this softer requirement is similarly hard to meet in practice. These requirements are obstacles blocking profound CC evolution and could be relaxed if fairness is ensured by other mechanisms (e.g., in-network control of resource sharing).

There exist different scheduling algorithms (e.g., by Hierarchical QoS or air interface scheduler) to ensure fairness between subscribers, while other approaches such as fq-YFA (fair queueing with Your Favorite AQM, e.g., fq-codel) can do the same among flows of the same user. These approaches have several drawbacks: 1) in some cases, it is not practical to provide user-fairness by scheduling, e.g., in AAN or interconnect. 2) Flow fairness and, in general, equal sharing is not optimal, not even for flows of the same user [18].

We believe that in-network approaches like CSAQM [14] provide flexible and efficient framework to control resource sharing, removing the burden of compatibility and fairness constraints from future CC developments. Its deployment is for future work, discussed in detail in [7].

VII. Summary

We have demonstrated that the two AQMs most commonly proposed for flows with Scalable Congestion Control cannot provide good fairness for scenarios where multiple CCs coexist. They perform poorly in both multi-CC equal-RTT; and mono-CC multi-RTT scenarios. For the most heterogeneous case (multi-CC multi-RTT), we experienced significant unfairness between the different flow groups. The design choice of BBRv2 was to implement a "DCTCP-inspired ECN" mechanism, and we believe it was never intended to be fair to DCTCP. At the same time, the performance of BBRv2 is generally superior to that of DCTCP, especially for small buffers. Based on this performance improvement, we argue that the developers of BBRv2 made the right choice. We also highlight that as long as the end-to-end CC controls resource sharing, it is extremely hard to create new, evolved CCs that are fair to legacy CCs. While providing a good resource sharing control out-of-CC is not a solved problem, we highlight the Core-Stateless AQM family as a possible solution worth further discussion and evaluation.

Acknowledgment

The authors thank the support of Ericsson Hungary Ltd. This research was supported by Thematic Excellence Programme, Industry and Digitization Subprogramme, NRDI Office, 2019. The research of S. Laki was supported by the János Bolyai Research Scholarship of the Hungarian Academy of

Sciences. The authors thank the anonymous reviewers for their suggestions on how to improve clarity of the paper.

REFERENCES

[1] V. Jacobson *et al.*, "Congestion avoidance and control.(1988)," in *Proceedings of the SIGCOMM*, vol. 88.

[2] M. Kühlewind *et al.*, "Using data center TCP (DCTCP) in the Internet," in *2014 IEEE Globecom Workshops*. IEEE, 2014, pp. 583–588.

[3] B. Briscoe *et al.*, "Low Latency, Low Loss, Scalable Throughput (L4S) Internet Service: Architecture," Internet Engineering Task Force, Internet-Draft draft-ietf-tsvwg-l4s-arch-04, Jul. 2019, work in Progress.

[4] B. Briscoe *et al.*, "Implementing the TCP Prague Requirements for Low Latency Low Loss Scalable Throughput (L4S)," *Proc. Linux Netdev 0x13, March*, 2019.

[5] N. Cardwell *et al.*, "BBR: Congestion-Based Congestion Control," *ACM Queue*, vol. 14, no. 5, pp. 50:20–50:53, Oct. 2016.

[6] N. Cardwell et. al, "An update on BBR," slides-104-iccrg-an-update-on-bbr-00, 2019-03.

[7] F. Fejes *et al.*, "Who will Save the Internet from the Congestion Control Revolution?" in *Workshop on Buffer Sizing*, 2019.

[8] R. Ware *et al.*, "Beyond Jain's Fairness Index: Setting the Bar For The Deployment of Congestion Control Algorithms," in *Proceedings of the 18th ACM Workshop on Hot Topics in Networks*, 2019, pp. 17–24.

[9] M. Alizadeh *et al.*, "Data Center TCP (DCTCP)," in *ACM SIGCOMM 2010*. ACM, 2010, pp. 63–74.

[10] M. Alizadeh *et al.*, "Analysis of DCTCP: stability, convergence, and fairness," in *ACM SIGMETRICS*. ACM, 2011, pp. 73–84.

[11] M. Hock *et al.*, "Experimental evaluation of BBR congestion control," in *IEEE ICNP*. IEEE, 2017, pp. 1–10.

[12] K. De Schepper *et al.*, "PI2: A Linearized AQM for Both Classic and Scalable TCP," ser. ACM CoNEXT '16. ACM, 2016, pp. 105–119.

[13] K. De Schepper *et al.*, "DualQ Coupled AQMs for Low Latency, Low Loss and Scalable Throughput (L4S)," IETF, Internet-Draft draft-ietf-tsvwg-aqm-dualq-coupled-11, Mar. 2020, work in Progress.

[14] S. Nádas *et al.*, "Towards a Congestion Control-Independent Core-Stateless AQM," in *ANRW '18*, 2018, pp. 84–90.

[15] A. Vishwanath *et al.*, "Perspectives on router buffer sizing: Recent results and open problems," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 2, pp. 34–39, 2009.

[16] S. Hemminger *et al.*, "Network emulation with NetEm," in *Linux conf au*, 2005, pp. 18–23.

[17] F. Fejes *et al.*, "Scalable CC compatibility: all measurement results," in *http://ppv.elte.hu/scalable-cc-comp/*, 2020-03.

[18] B. Briscoe, "Per-Flow Scheduling and the End-to-End Argument," Discussion Paper TR-BB-2019-001, bobbriscoe. net, Tech. Rep., 2019.