



To switch or not to switch to TCP Prague? Incentives for adoption in a partial L4S deployment

Fatih Berkay Sarpkaya

fbs6417@nyu.edu

New York University

Brooklyn, NY, USA

Fraida Fund

ffund@nyu.edu

New York University

Brooklyn, NY, USA

Ashutosh Srivastava

as12738@nyu.edu

New York University

Brooklyn, NY, USA

Shivendra Panwar

panwar@nyu.edu

New York University

Brooklyn, NY, USA

ABSTRACT

The Low Latency, Low Loss, Scalable Throughput (L4S) architecture has the potential to reduce queuing delay when it is deployed at endpoints and routers throughout the Internet. However, it is not clear how TCP Prague, a prototype scalable congestion control for L4S, behaves when L4S is not yet universally deployed. Specifically, we consider the question: in a partial L4S deployment, will a user benefit by unilaterally switching from the status quo TCP to TCP Prague? To address this question, we evaluate the performance of a TCP Prague flow when sharing an L4S or non-L4S bottleneck queue with a non-L4S flow. Our findings suggest that the L4S congestion control, TCP Prague, has less favorable throughput or fairness properties than TCP Cubic or BBR in some coexistence scenarios, which may hinder adoption.

CCS CONCEPTS

• **Networks** → **Transport protocols**.

KEYWORDS

TCP, Congestion Control, Low Latency, L4S, AQM

ACM Reference Format:

Fatih Berkay Sarpkaya, Ashutosh Srivastava, Fraida Fund, and Shivendra Panwar. 2024. To switch or not to switch to TCP Prague?

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org. ANRW '24, July 23, 2024, Vancouver, AA, Canada
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 979-8-4007-0723-0/24/07

<https://doi.org/10.1145/3673422.3674896>

Incentives for adoption in a partial L4S deployment. In *Applied Networking Research Workshop (ANRW '24)*, July 23, 2024, Vancouver, AA, Canada. ACM, New York, NY, USA, 8 pages. <https://doi.org/10.1145/3673422.3674896>

1 INTRODUCTION

Low Latency, Low Loss, Scalable Throughput (L4S) [3] is an architecture that allows low-delay and classic (non-scalable) data flows to coexist in the same network with reduced latency. It achieves this primarily through three mechanisms: scalable congestion control [1, 4], more accurate Explicit Congestion Notification (AccECN) [5, 6], and a dual queue Active Queue Management (AQM) [7–9]. When all three of these components are in place at the sender, receiver, and bottleneck router, an L4S flow can achieve high throughput with very low latency. However, like any new Internet technology, the deployment of L4S will be incremental. In the initial stages of deployment, L4S flows will coexist with non-L4S flows at L4S or non-L4S bottleneck routers.

This partial L4S deployment scenario is the primary focus of our work. In particular, we are interested in the perspective of a sender that has not yet switched to TCP Prague, the scalable congestion control protocol that has been proposed as part of L4S [1, 4]. Given that the bottleneck router may or may not have a dual queue AQM, and given that the other flows sharing the same bottleneck may not be TCP Prague flows, what benefit can a sender expect from unilaterally switching its own congestion control (CC) to TCP Prague? This is a key consideration for the eventual deployment of L4S on the Internet. The benefit to individuals deciding whether or not to unilaterally adopt a new technology determines whether or not it will reach a “tipping point” and achieve a stable non-zero equilibrium deployment [10].

To address this question, we conduct a series of experiments on the FABRIC [11] testbed and measure the throughput and latency of a TCP Prague flow in various partial

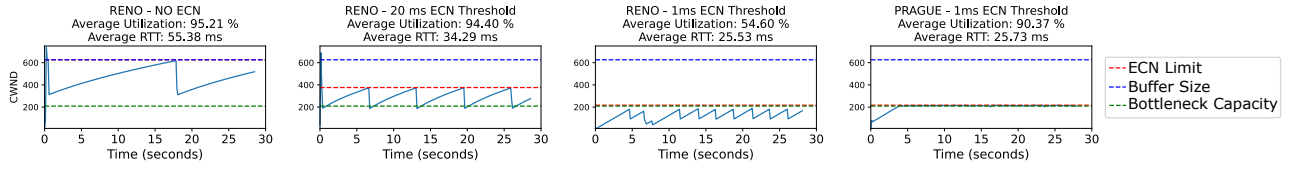


Figure 1: Fundamental Problem of Classic Congestion Control. Adapted from [1], we conduct a FABRIC experiment using a line network with 100 Mbps bottleneck capacity, a base RTT of 25 ms, and a buffer size of 2 BDP. The bottleneck AQM is FIFO with ECN. The artifacts to reproduce this experiment are available in [2].

deployment scenarios. The results may inform further development of L4S, especially with regard to its behavior in an incremental deployment. While the IETF Transport and Services Working Group (TSVWG) members have been active in evaluating L4S in various scenarios, the academic literature does not cover it extensively. Our work aims to address this gap in the literature, and we will elaborate on this in Sec. 2. We share our experiment artifacts for an open access testbed (FABRIC) [11] so anyone can build on and validate our research¹. The rest of this paper is organized as follows. Section 2 provides background information on the L4S architecture and its key components. This section also discusses the principles of L4S coexistence and the proposed strategies for its incremental deployment. Section 3 describes the experiment methodology with which we evaluated the coexistence of L4S flows with classic flows and bottleneck routers. Section 4 presents the results of our experiments and offers a detailed analysis of these findings. Section 5 concludes with a summary and directions for future work.

2 BACKGROUND

The L4S architecture, detailed in [3], is designed to reduce network queuing delay using three critical components: scalable congestion control, AccECN, and dual queue AQM. In this section, we describe the key components of the L4S architecture, the state-of-the-art regarding the coexistence of L4S with classic TCP, and the potential hurdles to its deployment from a content provider’s perspective.

Classic congestion control, such as TCP Reno or TCP Cubic, responds to network congestion signals such as dropped packets or ECN markings by multiplicatively decreasing its congestion window (*cwnd*), e.g., by a factor of 2 in TCP Reno. A small ECN threshold would be ideal for maintaining low queuing delays. However, a significant drop in the *cwnd* at every ECN mark will lead to under-utilization of the link capacity. As depicted in Figure 1, for TCP Reno, a high ECN marking threshold prevents under-utilization, but the latency remains high. Setting a low marking threshold (e.g., 1 ms) causes under-utilization. This highlights a fundamental limitation of traditional ECN-based congestion control

mechanisms: the inability to achieve extremely low queuing delays without under-utilizing network capacity.

Scalable congestion controls, like DCTCP, [12] and TCP Prague, address this issue along with an enhancement to ECN known as AccECN. Using AccECN, a scalable sender can calculate the fraction of ECN-marked packets in the last round and reduce *cwnd* in proportion to a moving average of this fraction. The scalable approach tries to react to the extent of congestion and not only its presence. This behavior is also illustrated in Figure 1, where TCP Prague achieves high utilization despite a very low ECN threshold.

When a scalable TCP shares a single bottleneck queue (not dual queue or multiple queues) with a classic TCP flow, the difference in their response to ECN marks can cause fairness issues [8]. To address this concern, TCP Prague includes an optional ECN fallback heuristic [13] to detect the presence of a single queue, non-L4S, ECN-capable AQM, primarily through RTT variation measurement. On detecting this type of queue, TCP Prague should revert to Reno-like behavior.

While the ECN fallback mechanism prevents TCP Prague from starving classic traffic, we cannot realize the low latency benefits of the L4S solution without upgrading routers. A non-L4S AQM cannot set an extremely low ECN threshold as this is detrimental to classic TCP traffic (Fig. 1). Per-flow queuing enables flow isolation, but it would need to enable marking at two different thresholds: a shallow threshold for L4S traffic and a higher threshold for classic traffic. However, as pointed out in [3], per-flow AQMs rely on packet inspection and thus may not be compatible with full end-to-end encryption of transport layer identifiers for privacy and confidentiality, such as IPsec or encrypted VPN tunnels. Per-flow queuing approaches may also not be scalable to core network bottlenecks with thousands of competing flows, a common occurrence in peering links between ISPs [14, 15].

To address this, the Dual-Queue Coupled AQM [9] separates L4S and non-L4S flows into different queues with different ECN marking thresholds. The marking/drop response of the classic queue is coupled with that of the L4S queue in order to ensure a fair share of available capacity between the classic and low-latency traffic. Hence, the dual queue solution can provide low latency for L4S traffic and achieve

¹Artifacts are available at: <https://github.com/fatihsarpkaya/L4S>

fair coexistence without the need for per-flow queuing. The DualPI2 AQM introduced in [7] implements this idea.

Like any new Internet protocol, L4S will be deployed in an incremental manner. Since L4S involves changes at TCP senders (scalable congestion control), TCP receivers (AccECN), and routers (dual queue AQM), a key area of focus in L4S design is the behavior of the protocol when some of these elements are not yet in place. Ideally, to encourage widespread deployment, an L4S flow should have throughput and delay characteristics at least as favorable as a classic flow, even if some elements of the full architecture are missing. Also, an L4S flow should not harm classic flows.

With respect to incremental deployment, the L4S architecture design [16] envisions that the L4S AQM will first be deployed on access network bottlenecks in the downstream direction (e.g., as part of low latency DOCSIS [17]). Then, L4S flows that are part of highly controlled trials will demonstrate the benefit of the architecture. Following this, scalable congestion control and AccECN will be deployed on endpoints to enable more general use of L4S.

However, there are likely to be scenarios in which L4S flows will traverse non-L4S AQMs. First, multiple studies have shown that bottlenecks at peering points between ISPs are also common [14, 15]. Also, non-cable access links such as 4G/5G cellular, Wi-Fi, and satellite networks can also be potential bottlenecks. While major wireless network vendors and device manufacturers like Apple are showing interest in L4S [18], widespread deployment is still some time away. Upgrading legacy access network routers, e.g., Wi-Fi routers or 3G/4G base stations worldwide, to support L4S will also be a challenge. Thus, the behavior of an L4S flow over a non-L4S AQM is of great interest to content providers who might consider switching to scalable congestion control.

Field trials and prototype demonstrations validate the low latency benefits of the L4S architecture in controlled environments [19, 20]. However, some early work evaluating L4S [21–23], has raised concerns about partial deployment. Although most of this work has not been published in the academic literature, these results have been used in IETF meetings and discussions [24, 25] and to inform further protocol development. It is shown in [22] and validated in [23] that L4S flows dominate both Cubic and Reno flows in a single-queue bottleneck with ECN. The ECN Fallback heuristic [13] has been proposed to address this issue, but previous work on low latency congestion control suggests that the delay variance measure used in this heuristic is not a reliable signal in all settings, and that in fact, there is no universally effective signal of sharing a bottleneck with classic flow [26–28]. Additionally, [22] mentions that DualPI2 consistently provides a throughput advantage to Cubic flows, and [29] finds that the fairness of the DualPI2 AQM is not robust to small variations in protocol implementations.

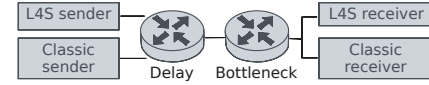


Figure 2: Experiment Topology

In this work, we seek to validate these early findings, evaluate the effectiveness of the ECN Fallback heuristic, and to also consider scenarios where a TCP Prague flow shares a bottleneck with a TCP BBR flow (as a substantial portion of Internet traffic uses BBR [30]).

3 EXPERIMENT METHODOLOGY

In this section, we describe our experiment setup for evaluating the performance of TCP Prague when sharing a bottleneck link with TCP Cubic or TCP BBR.

Experiment Platform: We conduct experiments on FABRIC [11], a national scale programmable experimental networking testbed. Each node in our experiment is a virtual machine running Ubuntu 22.04, with 4 cores and 32 GB RAM.

Topology: We use a line topology comprising two senders and two receivers (L4S and classic), connected via a delay node and a bottleneck router, as illustrated in Figure 2. At the delay node, we use netem to emulate a base RTT, with half of the delay added in each direction. At the bottleneck router, we use the token bucket filter implemented in tc-htb to configure the bottleneck bandwidth and buffer size.

Network Settings: On the Internet, a flow is most likely to encounter a bottleneck either at a peering point, or at the access link. For these experiments, we emulate network conditions that are representative of an access link: 10 ms base RTT, 100 Mbps bottleneck link capacity.

Queue: Our experiments consider the following queue configurations at the bottleneck router:

- **FIFO:** a single drop tail queue without ECN support, realized with tc-bfifo.
- **FIFO + ECN:** a single drop tail queue with ECN support using a static 5 ms marking threshold, realized with tc-fq. (Although tc-fq is multi-queue, by setting the orphan_mask option to 0, we enforce that all flows are hashed to a single queue.)
- **CoDel:** a single queue with CoDel AQM [31], which uses the local minimum queue size within a monitoring window as a measure of the standing queue, and marks packets if there is a standing queue exceeding a target value. We use tc-codel with a 5 ms target and the ECN option enabled, so that it marks packets for flows with ECN support and drops packets otherwise.
- **FQ:** a fair queue with flow isolation and ECN (using a static 5 ms marking threshold), realized with tc-fq.
- **FQ-CoDel:** combines fair queuing with the CoDel AQM. We realize this queue with tc_fq_codel, with a 5 ms target and the ECN option enabled.

- **DualPI2**: a dual queue coupled AQM designed for L4S [9], realized using `tc-dualpi2` from the L4S repository [32] (commit 4579ffb). The target parameter for the Proportional Integral (PI) controller is 5 ms and the `step_thresh` parameter for the L4S queue is 1 ms.

For each type of queue, we consider bottleneck buffer sizes that are shallow and deep, including the following multiples of the link bandwidth delay product (BDP): 0.5, 1, 2, 4, and 8.

Congestion control and AccECN at endpoints: In our experiments, L4S flows run TCP Prague, using the implementation in the L4S repository [32] (commit 4579ffb). The L4S sender and receiver support AccECN. We also evaluate TCP Prague with and without the ECN Fallback heuristic [13], which is an optional setting that needs to be turned on explicitly. For the classic flow, we consider two CCs that a TCP Prague flow is likely to encounter at a shared bottleneck:

- **TCP Cubic** [33] - according to a recent estimate, this is the predominant TCP variant on the Internet [30]. We consider a classic Cubic flow (implementation in Linux kernel 5.13.12) with and without ECN support.

- **TCP BBR** - BBR and its variants account for 22% of the top websites and approximately 40% of Internet traffic by volume [30]. We conduct experiments with BBRv1 (implementation in Linux kernel 5.13.12), which does not support ECN. We also consider BBRv2, with and without ECN, using the implementation in the `v2alpha` branch of the official BBR repository [34]. Finally, we also run experiments with an L4S-compatible BBRv2 flow, using the BBRv2 with AccECN support from the L4S repository [32] (commit 4579ffb).

Flow Generation: For each network configuration, we generate a single TCP Prague flow from the L4S sender and a single TCP Cubic or TCP BBRv1/v2 flow from the classic sender using the `iperf3` utility, for a duration of 60 seconds. We then record the average throughput and RTT values for each flow. The results presented are the average of 10 trials.

4 EXPERIMENT RESULTS

In this section, we present and evaluate the throughput and queuing latency of a TCP Prague flow when it shares a single bottleneck with a competing flow (Cubic or BBR) under various previously described network conditions. The main question we would like to answer is: **should content originators use L4S (TCP Prague) as their congestion control protocol?** Our results are presented in Figures 3, 4, 5 and 6. Here, we discuss some of the major findings.

Prague throughput is degraded when sharing a single, ECN-enabled queue without AQM alongside a BBRv2 or Cubic flow that *does not respond to ECN* signals. This is shown in Fig. 3 (Cubic - No ECN) and Fig. 4 (BBRv2 - No ECN) for the **FIFO + ECN** AQM. This outcome is expected because if the classic sender does not respond to ECN, it fills

the bottleneck buffer. However, we want to highlight this as a concern for L4S deployment because such a scenario can occur when an endpoint does not support ECN or an Internet path encounters “ECN bleaching”, i.e., an intermediate network device clearing the ECN flags. Both of these remain problematic according to a recent measurement study [35].

Although not shown here, the same trend is observed when competing with a BBRv1 flow¹. Prague is starved by BBRv1 in shallow buffer settings for most non-L4S queues (except FQ-Codel). BBRv1’s domination over TCP Cubic in shallow buffers has also been observed in previous work [36].

We observe that Prague is dominated by a BBRv2 flow whose endpoints do not support ECN when sharing an L4S AQM queue (**Fig. 4, DualPI2**). The DualPI2 coupling design assumes the classic queue carries loss-based TCP traffic (Cubic or Reno) but the same coupling parameters do not work with BBR flows in the classic queue. This raises a broader concern that the dual queue coupled AQM strategy may not generalize well when an L4S CC competes with non-L4S traffic that is not loss-based TCP. We further observe that even when competing with Cubic, Prague achieves slightly lower throughput (around 40%) under DualPI2 AQM regardless of ECN support, contradicting the observation in [22] that DualPI2 provides a throughput advantage to Prague flows.

Prague takes more than its fair share of throughput when competing with a Cubic or BBRv2 flow that *responds to ECN* while sharing a single, ECN-enabled queue (FIFO + ECN or Codel + ECN). This arises from differences in TCP Prague’s and other TCP’s response to ECN signals, as explained in section 2. Our results validate the observations in [22] for single queue with classic ECN AQMs. This scenario motivated the implementation of the ECN fallback heuristic in TCP Prague, which we also evaluated. With ECN fallback turned ON for TCP Prague, we see improved coexistence between Prague and ECN-capable classic flows over single queue ECN AQMs. However, we observed other problems (to be discussed shortly) with ECN fallback.

An AQM like CoDel drops packets from classic flows that *do not react to ECN* to maintain queuing delay close to its target. Prague again dominates the classic flows in this case (**Fig. 3 & 4, CoDel**) because of its scalable `cwnd` drop strategy.

When there is a non-ECN bottleneck, there is no ECN signal, and Prague falls back to operating like TCP Reno to maintain friendliness to classic TCP traffic. In our experiment conditions, Prague then achieves a fair share of throughput when competing with TCP Cubic (**Fig. 3, FIFO**). This finding disagrees with [22] where Cubic flows dominate Prague flows in FIFO bottlenecks without ECN. However, the buffer size used in this work was much higher than the maximum buffer of 8 BDP used in our experiments.

On the other hand, Prague dominates the BBRv2 flow over a FIFO queue (**Fig. 4, FIFO**). A similar result was observed for

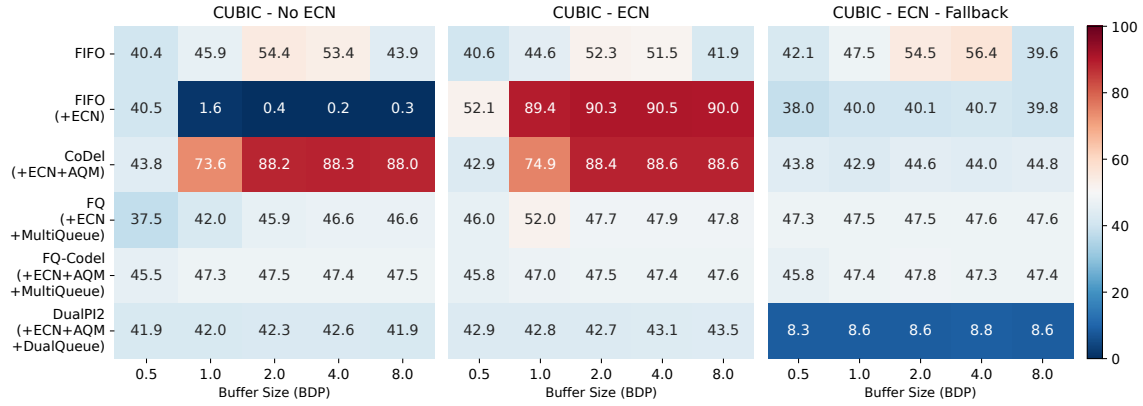


Figure 3: Prague throughput (Mbps) when sharing 100 Mbps bottleneck with Cubic flow.

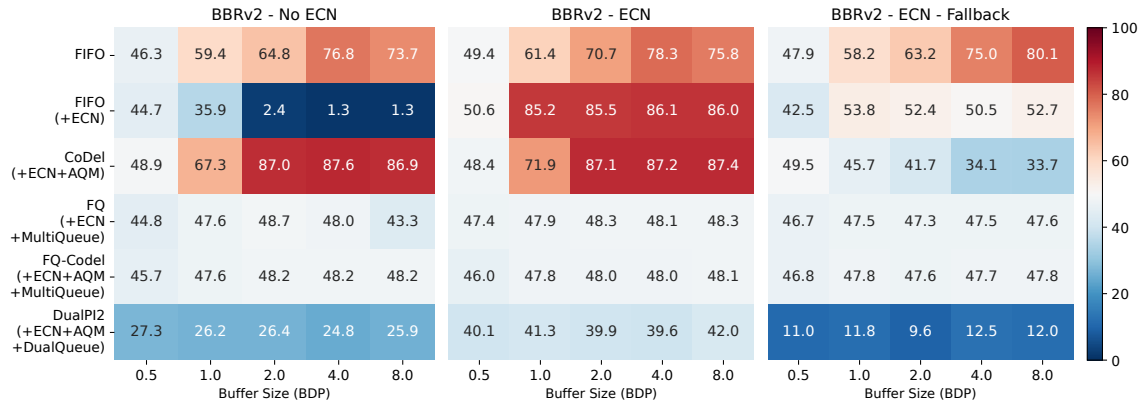


Figure 4: Prague throughput (Mbps) when sharing 100 Mbps bottleneck with BBRv2 flow.

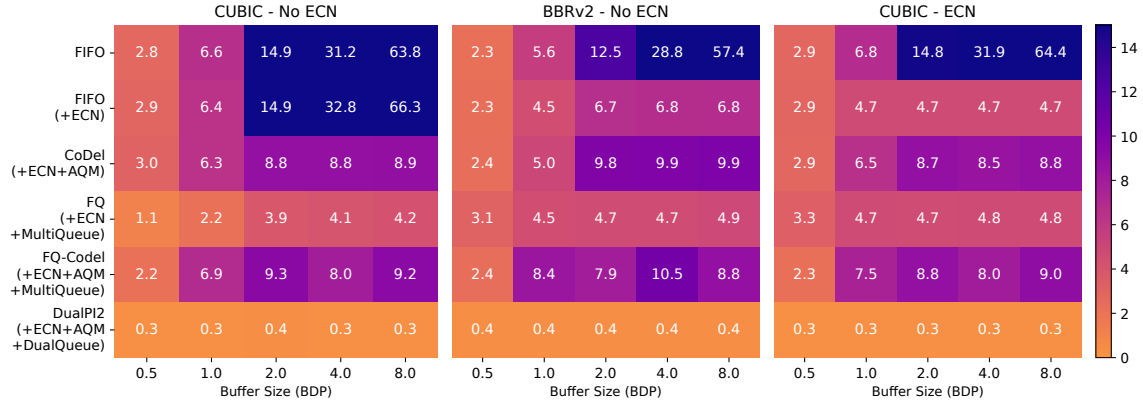
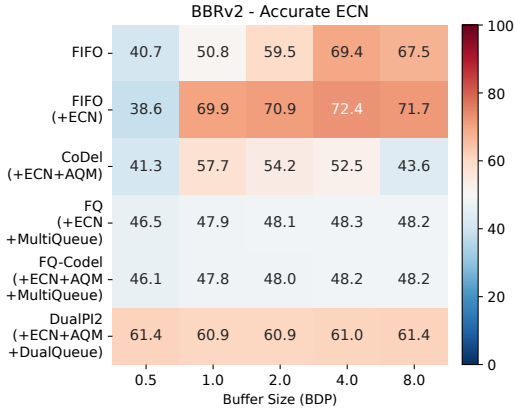


Figure 5: Prague queuing delay (ms) when sharing bottleneck with classic flow. (ECN threshold is 5 ms, where applicable. For DualPI2, L4S queue has 1 ms threshold.)

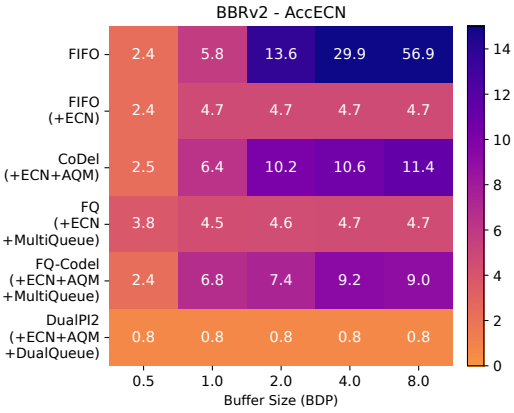
BBRv1 in our experiments and is a well-documented issue for BBR's coexistence with loss-based TCP in deep buffers [36].

Prague gets its fair share of throughput when sharing an FQ bottleneck, regardless of whether the competing flow is TCP Cubic or BBRv2, or whether the Fallback algorithm is

ON or OFF (Fig. 3 & 4, FQ + ECN & FQ-CoDel + ECN). This confirms the claim from [22] that the CoDelAF AQM allows for fair sharing between Cubic and Prague flows, as verified by our results with per-flow queuing (FQ) bottlenecks.



(a) Prague throughput (Mbps)



(b) Prague queuing delay (ms)

Figure 6: Prague throughput and queuing delay, sharing a 100 Mbps bottleneck with an AccECN BBRv2 flow.

For latency-sensitive applications, the only case where we observe TCP Prague consistently getting sub-1 ms queuing delays is with an L4S AQM that can mark L4S packets at a shallow threshold (like DualPI2) (Fig. 5). The ECN fallback heuristic, aimed at preventing Prague from dominating classic traffic in classic AQMs, may not benefit content originators. It may detect a classic queue even in cases where the bottleneck is DualPI2. When this happens, TCP Prague uses classic TCP behavior, and is starved because of the 1 ms threshold of the L4S queue (Fig. 3, 4, **Fallback, DualPI2**).

If the BBRv2 flow supports accurate ECN, it can also experience the low latency benefits of L4S. With a DualPI2 bottleneck, both the Prague and AccECN enabled BBRv2 are classified into the low latency queue, leading to sub-1 ms queuing delays (Fig. 6b). The throughput share is 60 : 40 in Prague's favor (Fig. 6a).

Table 1 summarizes our findings into recommendations on using the TCP Prague CC. Since the TCP sender does not know what type of bottleneck it will encounter, the decision

Buffer Type	ECN Fallback OFF		ECN Fallback ON	
	Cubic	BBRv2	Cubic	BBRv2
SQ w/o ECN	✓	X	✓	X
SQ + ECN	X	X	✓	✓
FQ + ECN	✓	✓	✓	✓
DualPI2	✓	✓	X	X

Table 1: Is it okay to turn on TCP Prague (✓) or not (X)? (SQ: single queue, FQ: fair queuing)

of whether or not to turn on TCP Prague at the sender depends on the expected types of queues likely to be deployed at bottlenecks and the types of flows likely to traverse these queues. If the bottleneck uses fair queuing (FQ), TCP Prague can be safely turned ON. While the ECN fallback algorithm improves fairness in single queue AQMs, it is prone to misdetection of dual queue AQMs, which can lead to Prague being starved even when AQMs compatible with the L4S architecture are deployed, e.g., DualPI2. This finding motivates the need to engineer better solutions for TCP Prague coexistence in non-L4S bottlenecks (for encouraging incremental deployment), to the extent possible [37], or at least to ensure that Prague will not be harmful [38]. Additionally, it is important that these solutions do not degrade Prague's performance in an L4S-enabled network.

5 CONCLUSION

In this work, we have investigated the conditions under which switching to TCP Prague is beneficial for a content originator and/or safe for other flows. Our findings suggest that if the content originator of TCP Prague cannot be sure what type of queue is at the bottleneck router, a range of outcomes for throughput and latency are possible, some of which are unfavorable. It is demonstrated that bottlenecks with per-flow isolation are the only type that ensures fairness, while dual queue AQM bottlenecks are the only type that guarantees low latency. In certain scenarios, TCP Prague may consume much more than its fair share of the link capacity when competing with Cubic, BBRv1, or BBRv2, and in other scenarios, TCP Prague gets much less than its fair share.

For future work, we hope to extend this analysis to more diverse network environments, including multiple flows, multiple bottlenecks, and more realistic traffic patterns. We will also consider other scalable congestion control algorithms that may be relevant to a content provider adopting L4S.

ACKNOWLEDGMENTS

This research was supported by the New York State Center for Advanced Technology in Telecommunications and Distributed Systems (CATT), NYU WIRELESS, and the National Science Foundation (NSF) under Grant No. CNS-2148309 and OAC-2226408.

REFERENCES

- [1] B. Briscoe, K. De Schepper, O. Tilmans, M. Kühlewind, J. Misund, O. Albisser, and A. S. Ahmed, "Implementing the 'Prague Requirements' for Low Latency Low Loss Scalable Throughput (L4S)," *Netdev 0x13*, 2019.
- [2] F. F. Fatih Berkay Sarpkaya, "Reproducing "Scalable Congestion Control Resolves the Delay Utilization Dilemma",", <https://github.com/fatiharpkaya/TCP-ECN>, 2024.
- [3] B. Briscoe, K. D. Schepper, M. Bagnulo, and G. White, "Low Latency, Low Loss, and Scalable Throughput (L4S) Internet Service: Architecture," RFC 9330, Jan. 2023. [Online]. Available: <https://datatracker.ietf.org/doc/html/rfc9330>
- [4] K. D. Schepper, O. Tilmans, B. Briscoe, and V. Goel, "Prague Congestion Control," Internet Engineering Task Force, Internet-Draft draft-briscoe-icrg-prague-congestion-control-03, Oct. 2023, work in Progress. [Online]. Available: <https://datatracker.ietf.org/doc/draft-briscoe-icrg-prague-congestion-control-03/>
- [5] B. Briscoe, M. Kühlewind, and R. Scheffenegger, "More Accurate Explicit Congestion Notification (ECN) Feedback in TCP," Internet Engineering Task Force, Internet-Draft draft-ietf-tcpm-accurate-ecn-28, Nov. 2023, work in Progress. [Online]. Available: <https://datatracker.ietf.org/doc/draft-ietf-tcpm-accurate-ecn-28/>
- [6] M. Kühlewind, R. Scheffenegger, and B. Briscoe, "Problem Statement and Requirements for Increased Accuracy in Explicit Congestion Notification (ECN) Feedback," RFC 7560, Aug. 2015. [Online]. Available: <https://datatracker.ietf.org/doc/html/rfc7560>
- [7] O. Albisser, K. De Schepper, B. Briscoe, O. Tilmans, and H. Steen, "DUALPI2—Low Latency, Low Loss and Scalable (L4S) AQM," *NetDev 0x13*, Prague, 2019.
- [8] K. D. Schepper, O. Albisser, O. Tilmans, and B. Briscoe, "Dual Queue Coupled AQM: Deployable Very Low Queuing Delay for All," 2022.
- [9] K. D. Schepper, B. Briscoe, and G. White, "Dual-Queue Coupled Active Queue Management (AQM) for Low Latency, Low Loss, and Scalable Throughput (L4S)," RFC 9332, Jan. 2023. [Online]. Available: <https://datatracker.ietf.org/doc/html/rfc9332>
- [10] N. Economides, "The Economics of Networks," *Intl. Journal of Industrial Organization*, vol. 14, no. 6, pp. 673–699, 1996.
- [11] I. Baldin, A. Nikolich, J. Griffioen, I. I. S. Monga, K.-C. Wang, T. Lehman, and P. Ruth, "FABRIC: A National-Scale Programmable Experimental Network Infrastructure," *IEEE Internet Computing*, vol. 23, no. 6, pp. 38–47, 2019.
- [12] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data Center TCP (DCTCP)," *SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 4, p. 63–74, aug 2010. [Online]. Available: <https://doi.org/10.1145/1851275.1851192>
- [13] B. Briscoe and A. S. Ahmed, "TCP Prague Fall-back on Detection of a Classic ECN AQM," 2021. [Online]. Available: <https://arxiv.org/abs/1911.00710>
- [14] A. Dhamdhere, D. D. Clark, A. Gamero-Garrido, M. Luckie, R. K. P. Mok, G. Akiwate, K. Gogia, V. Bajpai, A. C. Snoeren, and K. Claffy, "Inferring persistent interdomain congestion," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 1–15. [Online]. Available: <https://doi.org/10.1145/3230543.3230549>
- [15] A. Akella, S. Seshan, and A. Shaikh, "An empirical evaluation of wide-area internet bottlenecks," in *Proceedings of the 2003 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, ser. SIGMETRICS '03. New York, NY, USA: Association for Computing Machinery, 2003, p. 316–317. [Online]. Available: <https://doi.org/10.1145/781027.781075>
- [16] K. D. Schepper and B. Briscoe, "The Explicit Congestion Notification (ECN) Protocol for Low Latency, Low Loss, and Scalable Throughput (L4S)," RFC 9331, Jan. 2023. [Online]. Available: <https://datatracker.ietf.org/doc/rfc9331/>
- [17] J. Livingood, "Comcast Kicks Off Industry's First Low Latency DOCSIS Field Trials," <https://corporate.comcast.com/press/releases/comcast-multi-gig-symmetrical-speeds-world-first-docsis-4-deployment>, 2023.
- [18] G. White, "L4S Interop Lays Groundwork for 10G Metaverse," <https://www.cablelabs.com/blog/l4s-interop-lays-groundwork-for-10g-metaverse>, 2022.
- [19] N. Corporation, "Nokia collaborates with Hololight to deliver reliable immersive XR experiences with latency-improving technology L4S," <https://www.nokia.com/about-us/news/releases/2023/11/02/nokia-collaborates-with-hololight-to-deliver-reliable-immersive-xr-experiences-with-latency-improving-technology-l4s/>, 2023.
- [20] J. Livingood, "Comcast L4S Field Trial Update," <https://datatracker.ietf.org/meeting/118/materials/slides-118-tsvwg-sessa-61-l4s-experience-01>, 2023.
- [21] P. Heist, "sce-l4s-bakeoff," <https://github.com/heistp/sce-l4s-bakeoff>, 2019.
- [22] —, "L4S Tests," <https://github.com/heistp/l4s-tests>, 2021.
- [23] T. Henderson, O. Tilmans, and G. White, "Testbed and Simulation Results for TSVWG Scenarios," 2019, accessed: 2024-06-12. [Online]. Available: https://l4s.cablelabs.com/l4s_issues.html
- [24] G. W. Bob Briscoe, Koen De Schepper, "L4S Status Update," Presented at IETF 112, Online, 2021, accessed: 2024-06-12. [Online]. Available: <https://datatracker.ietf.org/meeting/112/materials/slides-112-tsvwg-sessa-32-l4s-ecn-drafts-01.pdf>
- [25] O. T. G. W. Bob Briscoe, Koen De Schepper, "Low Latency Low Loss Scalable Throughput (L4S)," <https://www.ietf.org/proceedings/interim-2020-tsvwg-01/slides/slides-interim-2020-tsvwg-01-sessa-l4s-tcp-prague-update-00.pdf>, February 2020, interim 2020 TSVWG Meeting.
- [26] A. Srivastava, F. Fund, and S. S. Panwar, "Coexistence of delay-based TCP congestion control: Challenges and opportunities," in *2022 IEEE International Workshop Technical Committee on Communications Quality and Reliability (CQR)*. IEEE, 2022, pp. 43–48.
- [27] Ł. Budzisz, R. Stanojevic, A. Schlote, F. Baker, and R. Shorten, "On the fair coexistence of loss-and delay-based TCP," *IEEE/ACM transactions on networking*, vol. 19, no. 6, pp. 1811–1824, 2011.
- [28] Y. Zhu, M. Ghobadi, V. Misra, and J. 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*, ser. CoNEXT '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 313–327. [Online]. Available: <https://doi.org/10.1145/2999572.2999593>
- [29] D. BoruOljira, K.-J. Grinnemo, A. Brunstrom, and J. Taheri, "Validating the sharing behavior and latency characteristics of the L4S architecture," *ACM SIGCOMM Computer Communication Review*, vol. 50, no. 2, pp. 37–44, 2020.
- [30] A. Mishra, X. Sun, A. Jain, S. Pande, R. Joshi, and B. Leong, "The Great Internet TCP Congestion Control Census," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 3, no. 3, dec 2019. [Online]. Available: <https://doi.org/10.1145/3366693>
- [31] K. Nichols and V. Jacobson, "Controlling Queue Delay," *Commun. ACM*, vol. 55, no. 7, p. 42–50, jul 2012. [Online]. Available: <https://doi.org/10.1145/2209249.2209264>
- [32] L4S development hub, "Linux kernel tree with L4S patches," <https://github.com/L4STeam/linux>, 2024.
- [33] S. Ha, I. Rhee, and L. Xu, "CUBIC: a new TCP-friendly high-speed TCP variant," *SIGOPS Oper. Syst. Rev.*, vol. 42, no. 5, p. 64–74, jul 2008. [Online]. Available: <https://doi.org/10.1145/1400097.1400105>

- [34] Google, “BBR - Source code,” <https://github.com/google/bbr>, 2024.
- [35] H. Lim, S. Kim, J. Sippe, J. Kim, G. White, C.-H. Lee, E. Wustrow, K. Lee, D. Grunwald, and S. Ha, “A Fresh Look at ECN Traversal in the Wild,” 2022.
- [36] R. Ware, M. K. Mukerjee, S. Seshan, and J. Sherry, “Modeling BBR’s Interactions with Loss-Based Congestion Control,” in *Proceedings of the Internet Measurement Conference*, ser. IMC ’19. New York, NY, USA: Association for Computing Machinery, 2019, p. 137–143. [Online]. Available: <https://doi.org/10.1145/3355369.3355604>
- [37] V. Arun, M. Alizadeh, and H. Balakrishnan, “Starvation in end-to-end congestion control,” in *Proceedings of the ACM SIGCOMM 2022 Conference*, ser. SIGCOMM ’22. New York, NY, USA: Association for Computing Machinery, 2022, p. 177–192. [Online]. Available: <https://doi.org/10.1145/3544216.3544223>
- [38] R. Ware, M. K. Mukerjee, S. Seshan, and J. Sherry, “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*, ser. HotNets ’19. New York, NY, USA: Association for Computing Machinery, 2019, p. 17–24. [Online]. Available: <https://doi.org/10.1145/3365609.3365855>