Congestion-Responsive Queuing for Internet Flows

Ike Kunze*, Constantin Sander*, Mike Kosek†, Lars Tissen*, Jan Pennekamp*, Klaus Wehrle*
*RWTH Aachen University, Aachen, Germany · {kunze, sander, tissen, pennekamp, wehrle}@comsys.rwth-aachen.de
†Technical University of Munich, Munich, Germany · kosek@in.tum.de

***Abstract*—Internet congestion management is once again undergoing radical change: QUIC has ignited a cambrian explosion in congestion control (CC) implementations while the many versions of BBR alone have increased the diversity in algorithms used with TCP, both making the congestion landscape more complex. At the same time, the interplay of CC and AQM is also evolving but congestion unresponsiveness remains a threat. In particular, L4S crucially requires a fine-grained CC and AQM interaction to provide its benefits and suffers from unresponsive traffic. Overall, we need more responsive traffic on the Internet as well as mechanisms that can cope with unresponsiveness.**

We present *Congestion-Responsive Queuing* (*CRQ*), our L4S-inspired system which is designed to promote responsive CC, manage unresponsive traffic, and handle QUIC and TCP flows alike. Similar to L4S, *CRQ* uses two queues for flow isolation. Yet, in contrast to L4S, we isolate flows based on their actual congestion responsiveness, moving responsive flows to one queue and leaving the remaining flows in the other. Our evaluation with an eBPF prototype highlights the efficacy of our design and shows that *CRQ* can provide effective incentives for responsive CC.

I. Introduction

The stability of the Internet relies on congestion control (CC) which helps end hosts in finding a suitable share of the limited bandwidth and avoid a congestion collapse [1]. Yet, despite over 30 years of research and standardization, the domain is still very dynamic: many new CC algorithms (CCAs) have been proposed over the years, experts frequently revise existing CCAs [2], [3], and set new guidelines for their specification [4]. As a result, there has always been a large diversity in practically deployed CCAs [5], [6] with a significant impact on performance [7], [8].

Recently, QUIC, with its user-space nature, has significantly lowered the bar for developing and deploying new CCAs compared to those traditionally used with TCP [9]. Additionally, it was found that QUIC stacks differ in their CCA implementations, resulting in different CC behavior despite using the same algorithms on paper [10]. Overall, these developments have drastically complicated the already existing diversity in CC and further challenge the pillars of the Internet as many mechanisms, such as buffer sizing rules [11] or queuing mechanisms [12], [13], were designed for a more stable CC landscape dominated by loss-based CCAs.

One network management domain that is particularly affected by these observations is active queue management (AQM). With AQM, network operators actively manage router queues with the goal of assisting end host CC in finding a fair share of the bandwidth and, ultimately, improving throughput and latency. However, AQM often struggles with *unresponsive* flows [14], i.e., flows that do not respond to emitted congestion signals, such as packet drops or Explicit Congestion Notification (ECN) markings. Even mechanisms designed for constraining the advantages of flows not using CC typically only probabilistically identify misbehaving traffic which also hurts benign flows [14]–[16].

These issues are exacerbated with L4S [17] as operators put an increasing focus on improving latencies across the Internet with fine-grained AQM and DCTCP-style CC. For this, L4S first classifies flows based on their use of specific ECN code points before assigning them to two queues: one L4S queue with fine-grained AQM for L4S-capable traffic, the other for "classic" traffic with standard AQM. L4S flows can then benefit from the fine-grained marking while other flows are not harmed by them. However, flows mistakenly using the wrong ECN code point, e.g., caused by network interference [18], or doing it consciously with hopes of gaining an advantage, can easily void the benefits of L4S [19]–[21]. Hence, L4S could benefit from a robust mechanism for identifying misbehaving traffic to protect the L4S queue.

Our previous work *SpinTrap* [22] is a first step toward addressing this need. In short, *SpinTrap* assesses the actual congestion *responsiveness* of QUIC flows by first tracking their sending behavior and any emitted congestion signals. It then checks whether flows subject to signals respond with a decrease in their sending behavior: flows with a reduction are *responsive* to congestion, flows without are *unresponsive*.

In this paper, we draw inspiration from L4S and propose *CRQ* which embeds *SpinTrap* in a new dual-queue scheme that does not rely on ECN code points for flow separation. Instead, we use the responsiveness assessment pioneered by *SpinTrap* to isolate responsive flows in one dedicated queue while the remaining traffic is served by a standard queue. For this, we first design *TCPTrap*, which transfers the principles of the QUIC-specific *SpinTrap* to TCP, such that *CRQ* can handle QUIC and TCP alike. We then conceptualize *CRQ*'s dual-queue scheme which can use different underlying AQM mechanisms to optimize the performance of responsive flows while managing unresponsive traffic. In our evaluation, we first briefly study the efficacy of *TCPTrap* before focusing on *CRQ*, showing that basing a dual-queue scheme on a congestion responsiveness assessment can indeed provide performance benefits. Overall, this paper contributes the following:

- We propose *CRQ*, a dual-queue system that isolates traffic based on congestion responsivenesss.
- We design *TCPTrap*, the counterpart to *SpinTrap*, that can assess the responsiveness of TCP traffic.

- Evaluations with our eBPF prototype [23] show that *CRQ* can provide effective incentives for using responsive CC.

Structure. Sec. II discusses key concepts of and related works on congestion management. Sec. III then presents the design of *CRQ* and of *TCPTrap*. We extensively evaluate *CRQ* and *TCPTrap* in Sec. IV, discuss main takeaways in Sec. V, and conclude the paper in Sec. VI.

II. Internet Congestion Management

Hosts on the Internet have to share a limited bandwidth while individual flows usually aim to maximize their throughput. Hence, the decentralized bandwidth allocation is an essential component of the Internet. Originally, CC only focused on preventing a network overload for which CCAs typically maintain a congestion window (`cwnd`) to limit the amount of unacknowledged data that can be in-flight. CCAs then adjust the `cwnd` to the current network congestion status for which they rely on different congestion signals, such as packet loss.

ECN. In addition to avoiding congestion, Internet congestion management increasingly aims at improving end-to-end performance. ECN enables network devices to signal impending congestion without packet loss using two IP header bits. Hosts enable ECN by using either of two ECN-Capable Transport (ECT) code points, i.e., ECT(0) or ECT(1), and intermediaries signal congestion by changing ECT to Congestion Experienced (CE). In practice, CE markings are emitted by AQM to assist end hosts in finding a suitable sending rate. "Classic" CCAs [17] interpret CE markings equally to packet loss and react to such signals once per window of data [24].

L4S. The L4S framework [17] goes a step further and aims at providing very low latencies across the Internet. Its main idea is to give DCTCP-style ECN feedback [25] for "scalable" CCAs, such as Prague [26], which can react to CE markings more than once per round-trip time (RTT) and allow for a fine-grained sending rate control to avoid queue buildup. As not all traffic can and will use scalable CCAs, L4S currently targets a dual-queue (DualQ) operation [21]: one queue with fine-grained AQM dedicated for L4S traffic, the other serving L4S-incapable traffic with classic AQM. Flows are assigned to the queues based on the used ECT code points: ECT(1) identifies L4S traffic, ECT(0) standard traffic. Flows not using ECN are also assigned to the standard queue while packets with CE markings are served by the L4S queue [27].

Flow isolation challenges. While relying on ECT code points is easy and stateless, it comes with the significant challenge that ECT(0) and ECT(1) used to be equivalent such that (older) hosts might still be using either one. Furthermore, middleboxes interfering with ECN code points, e.g., changing ECT(0) to ECT(1), can still be found on the Internet [18]. Finally, malicious hosts could also illicitly use the wrong code point in the hope of gaining L4S advantages. Overall, there are several ways how L4S-incapable traffic could end up in the L4S queue. In such cases of unresponsive traffic overloads, the L4S DualQ falls back to the performance of a classic, single-queue AQM [19]–[21] and all benefits are wiped out.

Related work. There is general consensus that unresponsive traffic is a potential threat to AQM [28] and many designs constrain the advantages of unresponsive flows, e.g., through probabilistically identifying misbehaving traffic by equating high queue shares or flow arrival rates with unresponsiveness [29]–[33]. Yet, such solutions often also hurt benign flows [14]–[16]. Statically classifying flows, e.g., based on their UDP use [34], [35], is also insufficient, e.g., in light of QUIC [36], [37] or responsive UDP-based video conferencing [38]. Overall, we observe that today's AQM often struggles with unresponsiveness, leaving a gap for robust mechanisms that can reliably identify misbehaving traffic.

Identifying unresponsive traffic. In previous work, we have proposed *SpinTrap* [22], a first major building block for classifying flows based on their actual responsiveness. In particular, *SpinTrap* can identify QUIC flows that do not respond to congestion signals. For this, it tracks the sending behavior of individual flows as well as explicit (ECN) and implicit (packet loss) congestion signals that affect them. Once it detects such signals, it checks for corresponding reductions in the sending behavior two RTTs later: flows with a reduction are classified as responsive, flows without reduction as unresponsive.

Isolating traffic with *SpinTrap* for promoting CC. Conceptually, *SpinTrap* can be used for a myriad of applications. Drawing inspiration from the L4S DualQ, we focus on queue management. In particular, *SpinTrap* can first assess the responsiveness of a flow. The assessment can then guide traffic isolation and queuing: similar to L4S-capable traffic, responsive flows can be isolated and subjected to (fine-grained) AQM while unresponsive traffic can be served with standard AQM. Such a system would not only reward flows that support Internet stability, but could also provide concrete incentives for deploying CC as envisioned by Floyd and Fall [39].

***Congestion-Responsive Queuing* (*CRQ*).** In this paper, we study the proposed concepts in a focused system without direct interactions of the two queues as would be the case with the L4S *coupled* DualQ [21]. In particular, we propose a dual-queue system where flows are assigned based on their current congestion responsiveness at runtime — *Congestion-Responsive Queuing* (*CRQ*). For this, we pair *SpinTrap* with *TCPTrap*, a novel TCP responsiveness assessment tool, for providing the underlying responsiveness assessment that then informs the queue assignment. In the following, we present *CRQ*, its components, and our prototype in more detail.

III. *Congestion-Responsive Queuing* (*CRQ*)

The main motivation for *CRQ* is to reward flows with a responsive CCA as they are essential for avoiding a congestion collapse and generally healthier for the overall coexistence in the Internet. It is inspired by the DualQ operation of L4S but performs its traffic isolation based on the actual flow behavior and not based on two IP header bits. In particular, *CRQ* relies on two main components as illustrated in Fig. 1. First, it constantly assesses the responsiveness of QUIC and TCP flows. Second, it uses the assessment results to distribute traffic into dedicated queues: responsive flows are served by a

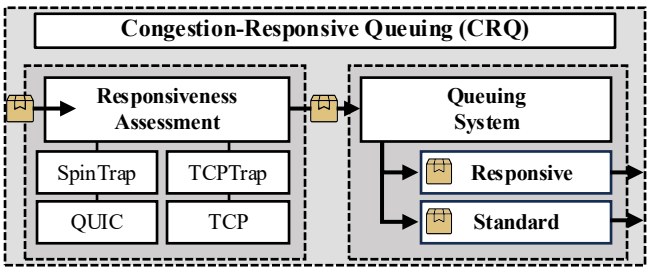

Fig. 1. *CRQ* relies on *SpinTrap* and *TCPTrap* for assessing the congestion responsiveness of flows. It then assigns responsive traffic to the responsive queue while the remaining traffic shares the standard queue.

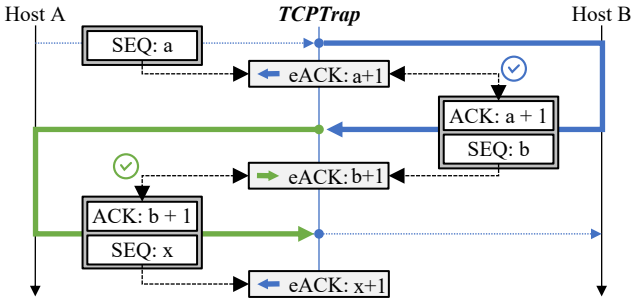

Fig. 2. *TCPTrap* matches TCP SEQs with their corresponding ACKs to determine individual cycles. Only half-cycles are possible for unidirectional traffic due to non-increasing SEQs in the reverse direction.

responsive queue while all other traffic remains in a *standard* queue. In the following, we discuss the conceptual considerations for *CRQ* in more detail starting with the responsiveness assessment in Sec. III-A, followed by the queuing system in Sec. III-B, before finally presenting our prototype in Sec. III-C.

A. Assessing Congestion Responsiveness

CRQ requires a robust responsiveness assessment. As already discussed in Sec. II, *SpinTrap* provides such an assessment for QUIC [22]. Yet, while QUIC is on the rise [36], [37], TCP still represents a large share of the Internet [40]. Thus, for widespread applicability, *CRQ* cannot solely rely on *SpinTrap* which is why we combine it with *TCPTrap*, our novel TCP responsiveness assessment tool. Next, we briefly revisit key concepts of *SpinTrap* and our lessons learned from previous work before we present the design of *TCPTrap*.

1) SpinTrap*:* The responsiveness assessment provided by *SpinTrap* conceptually relies on two key observations. First, classic CCAs react to congestion signals by reducing their `cwnd` once per window of data [24]. Second, any reaction only becomes visible on the wire one to two RTTs after the signals were emitted: if the signal is emitted at the *start* of a window, a reaction is perceivable in the *subsequent* window, i.e., one RTT later. If, however, the signal is emitted at the *end* of a window, the reaction might only become perceivable in the *next but one* window, i.e., two RTTs later (cf. [22]).

SpinTrap translates these considerations into a mechanism that constantly tracks the sending behavior of QUIC flows and corresponding congestion signals in the form of packet loss and CE markings. For this, it relies on the spin bit, an optional QUIC feature that explicitly shapes individual round trips ("cycles") on the transmission, originally with the purpose of enabling passive RTT measurements. *SpinTrap* leverages these explicit cycles to infer the sending window (ultimately governed by the `cwnd`) and compares the window at the time of a congestion signal with the window two cycles later.

Responsiveness criteria. In its basic form, *SpinTrap* classifies all flows as responsive that show any form of reduction, irrespective of the amount of reduction or the overall window size. However, in previous work [22], we found a more fine-grained configuration to work better. First, to protect flows that have already backed off due to heavy congestion, *SpinTrap* only assesses flows whose sending window is larger than four times the maximum segment size which corresponds to the current recommendation for the initial congestion window [41]. Additionally, to check for meaningful responses and exclude random sending rate fluctuations, we require a reduction to $90\,\%$ or less of the previous sending window. Finally, we use a majority voting for the responsiveness assessment to avoid repeated switching between the queues: a flow that has been classified as responsive ten times will stay responsive even if it is classified as unresponsive once.

While *SpinTrap* can reliably assess QUIC traffic, its use of the spin bit prohibits a direct application to TCP. Hence, to enable *CRQ*, we next present *TCPTrap* which translates our fundamental considerations from above to TCP.

2) TCPTrap*:* Plain TCP has no explicit measurement features that could help distinguish unique cycles. Yet, compared to QUIC, it has a more expressive wire image [42] as it does not encrypt its headers which enables measurements based on implicit protocol semantics [43]. Inspired by Dart [44], a mechanism designed to estimate TCP RTTs, we leverage the cyclic interaction of TCP sequence numbers (SEQs) and acknowledgement numbers (ACKs) for dividing the overall transmissions into dedicated cycles similar to *SpinTrap*.

Tracking TCP cycles. The fundamental observation is that a new SEQ is answered by a new ACK. An intermediary with bidirectional visibility can, thus, track one half of the overall round trip by matching the SEQ in one direction with its ACK in the opposite direction. For bidirectional traffic, i.e., both end hosts transmitting payload, the ACK might be transmitted with a new SEQ of its own which causes another ACK on the original sender to complete the overall round trip.

TCPTrap leverages these semantics for tracking cycles as visualized in Fig. 2. A cycle starts when host A transmits a segment with a new SEQ a. *TCPTrap* stores the corresponding expected ACK (eACK) $a + 1$ and then monitors the reverse direction for ACKs from host B. An ACK matching the stored eACK completes the first half of the overall cycle (blue).

Ideally, the segment with the matching ACK also contains a new SEQ b for the reverse direction. In this case, *TCPTrap* again stores the corresponding eACK $b + 1$ and the second half of the cycle (green) is complete once an ACK from A matches the eACK. Together, this yields the overall cycle in

a way that is similar to the explicit spin bit cycle.

However, in the case of asymmetric traffic, it might be that one of the hosts, e.g., host B, mostly transmits ACKs. In this scenario, the SEQs from B mostly remain unchanged as will the corresponding eACK stored for segments from A. Hence, the next segment of A will directly trigger a new cycle. This way, *TCPTrap* implicitly supports uni- and bidirectional traffic but risks too short cycles in the case of unidirectional traffic.

Responsiveness criteria. *TCPTrap* directly borrows *SpinTrap*'s congestion signal tracking and assessment logic, reusing the responsiveness criteria described above.

Limitations and sources of inaccuracy. In contrast to *SpinTrap*, *TCPTrap* requires bidirectional traffic visibility to map SEQs and ACKs. While this theoretically limits its applicability, bidirectional visibility is often given, e.g., for university networks or residential access routers [44]. One source of inaccuracy is that *TCPTrap* determines cycles implicitly based on TCP semantics, such that it can be affected by any changes in the sending patterns. Additionally, as described above, *TCPTrap*'s cycle tracking could become imprecise for asymmetric connections as is often the case for web traffic. Hence, we study the resulting assessment accuracy of *TCPTrap* and the impact of traffic directionality in our evaluation.

B. Queue Mechanism

The main goal of *CRQ* is to provide means for improving the performance of flows with responsive CCAs, ideally giving strong incentives for the use of CC. We realize this vision through a dual-queue design that is inspired by the L4S DualQ: a *responsive* queue exclusively serves responsive traffic while a *standard* queue serves all other traffic. This way, responsive flows are isolated and protected from unresponsive traffic so that they can better manage their sending behavior, similar to the L4S queue with L4S. However, in contrast to L4S, *CRQ*'s flow isolation decides based on the responsiveness assessment of *SpinTrap* and *TCPTrap*, i.e., actual flow behavior, which reduces the likelihood of unresponsive traffic being assigned to the responsive queue. Overall, *CRQ* encourages flows to deploy CC and incentivizes using the spin bit, as QUIC flows without the mechanism are not assessed.

Configuration space. Assuming a robust responsiveness assessment, the main parameter space for *CRQ* lies in configuring the two queues and in defining how and when exactly flows are assigned to which queue. Conceptually, *CRQ* can support nearly arbitrary combinations, e.g., using different AQM mechanisms for the queues, assigning them different priorities and bandwidth shares, or mandating loss or ECN responsiveness. However, not all combinations make sense in practice. One example is combining a loss-based standard queue with an ECN-based responsive queue and only moving ECN-responsive traffic to the responsive queue. In this scenario, *CRQ* cannot assess the ECN responsiveness of the flows without other hops emitting CE signals. Thus, if no other hops signal CE, the responsive queue remains unused. Based on this observation, we argue that it is sensible to deploy queue combinations that can also work independent of other hops. Hence, in this paper, we focus on combinations where both queues use the same congestion signals. In the following, we discuss the choice of the congestion signals and further configuration options of *CRQ* in more detail.

1) Congestion signals: Packet loss and ECN are today's main congestion signals and we use them for *CRQ*, always combining two queues that support the same signal.

Packet loss. Packet loss is the standard congestion signal as it, e.g., naturally occurs once buffers overflow. If both queues use packet loss as their main congestion signal, we can check for loss responsiveness in the standard queue and move flows to the responsive queue as soon as they have shown a response.

ECN. ECN enables signaling congestion without packet loss. Yet, even AQM with ECN falls back to packet loss once certain thresholds are surpassed or the buffer overflows. Thus, when using two queues with ECN and moving ECN-responsive flows to the responsive queue, the standard queue will ultimately degrade to a loss queue as only non-ECN-responsive flows remain. Then, we could also assess the loss-responsiveness of the remaining flows. In this paper, we focus on the queues' main signal and leave exploring a combined use of ECN- and loss-responsiveness for future work.

2) Queuing mechanisms: There are various queuing mechanisms that support packet loss or ECN congestion signals.

Drop-tail. Many queues default to a drop-tail behavior which comes with performance penalties (cf. bufferbloat [45]). Thus, drop-tail only makes partial sense for the standard queue while we argue that it should not be used for the responsive queue.

AQM. There are many AQM mechanisms. Controlled Delay (CoDel) [13] is a prominent variant with a particular focus on delay that naturally supports ECN. Hence, we choose CoDel as the main AQM for the standard and responsive queues.

Queue combinations. In this work, we focus on three queue combinations: (1) a drop-tail standard queue with a loss-based CoDel responsive queue, (2) CoDel in both queues with packet drop, and (3) CoDel in both queues with ECN.

3) Queue assignment: Only flows classified as responsive are assigned to the responsive queue while all other traffic remains in the standard queue. However, responsive flows changing the queue will be subject to different conditions and might need time to acclimatize to the new queue. Additionally, reassessments of the flow behavior could cause flows to frequently switch the queues, in the worst case leading to oscillations with detrimental effects on performance and stability. To address such risks, *CRQ* adds two fundamental forms of inertia that dampen the frequency of queue switching. First, the responsiveness assessment deploys a majority voting which reduces the rate at which flows change their assessment if they show mostly consistent behavior (see III-A1). Second, we reset any observed congestion signals when flows change the queue to delay any possible reassessment by two round trips. The two round trip delay by the second measure represents the minimum sensible time frame while larger delays might be needed for flows showing an unstable behavior. We leave exploring the exact impact of this delay to future work.

4) General configuration: There are different ways for giving performance benefits to responsive flows. We could, e.g., strictly prioritize the responsive queue, similar to L4S, or we could assign it a larger share of the available bandwidth. Focusing on the efficacy of traffic isolation and aiming for a balanced evaluation of our concepts, we opt for a modest approach that gives equal bandwidth shares to both queues with similar configurations. Future work could study the benefits of different queue configurations in more detail.

C. eBPF Prototype

We implement *CRQ* in an eBPF prototype that is partially based on our previous implementation of *SpinTrap* [22]. In short, our prototype first leverages Linux kernel tracepoints for tracking packet loss and ECN congestion signals. We then combine *SpinTrap* and *TCPTrap* in a joint responsiveness assessment tool that can simultaneously assess loss- and ECN-responsiveness based on the observed congestion signals. We attach the tool as a `tc filter` and it assigns flows different `tc classids` based on the assessment result. Finally, *CRQ*'s queuing is implemented with a class-full Hierarchical Token Bucket (HTB) with two child classes: flows are enqueued based on their assigned `tc classid`. For reproducibility, we publish the source code of our implementation [23]. In the following, we provide additional details.

Tracking congestion signals. We track congestion signals using Linux tracepoints. At packet enqueue, we monitor packet drops caused by overflowing buffers. At dequeue, we track packet drops or CE markings caused by AQM.

Estimating sending windows. We track the sending windows of QUIC and TCP flows with *SpinTrap* and *TCPTrap*, respectively. In essence, they report new estimates as soon as a new cycle is completed. In the case of *SpinTrap*, this is whenever the spin bit flips, i.e., QUIC traffic that disables the spin bit by using a constant value of 0 is never evaluated. In contrast, *TCPTrap* always yields a new estimate for TCP traffic based on the methodology described in Sec. III-A2.

Responsiveness assessment. Whenever *SpinTrap* or *TCPTrap* signal the end of a cycle, we check if a flow has seen congestion signals before assessing its responsiveness. Combining the tracked congestion signals with the estimated sending windows, we look for a corresponding window reduction as described in Sec. III-A1; traffic that has never seen congestion signals or has never triggered a cycle remains unclassified.

In the following, we first evaluate the assessment accuracy of *TCPTrap*, followed by the evaluation of *CRQ* overall.

IV. Evaluation

CRQ is designed to bring benefits to responsive flows by isolating them from unresponsive traffic in a dedicated queue. The achievable benefits largely depend on the specific system configuration and could be achieved by, e.g., strictly prioritizing responsive traffic over the standard queue or assigning a larger bandwidth share to the responsive queue. Aiming for a general assessment of *CRQ*, we focus on its traffic isolation capabilities in this paper. In particular, we study the benefits and behavior of *CRQ* when giving both queues equal bandwidth shares in a controlled testbed and leave fully exploring the configuration space for future work. In this section, we first describe our methodology in Sec. IV-A. We then analyze *CRQ*'s responsiveness assessment as the basis for its traffic isolation in Sec. IV-B, focusing on *TCPTrap* as we have already evaluated *SpinTrap* [22]. Finally, we study the overall performance of *CRQ* in Sec. IV-C.

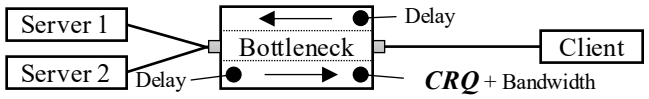

Fig. 3. Our testbed consists of four machines. The *Bottleneck* emulates different network scenarios and deploys our *CRQ* prototype. The *Server* machines transmit data to the *Client* machine using QUIC, TCP, and UDP.

A. Methodology

We conduct our evaluation in the testbed illustrated in Fig. 3 consisting of four Ubuntu machines that are interconnected with Gigabit Ethernet. We deploy QUIC and TCP servers on the *Server* machines (left) which then transmit data to the *Client* machine (right). The *Bottleneck* machine shapes network characteristics and deploys our *CRQ* prototype.

Network configuration. We add delay on the ingress of both *Bottleneck* interfaces using Intermediate Functional Block (IFB) interfaces and `tc netem`, adding half of the overall delay to each interface. We emulate bandwidths on the egress of the interface from *Bottleneck* to *Client* for which we leverage *CRQ*'s HTB. In particular, we attach two child `tc qdiscs` representing the responsive and standard queue of *CRQ* which we can then configure as needed. Throughout all settings, we configure a bottleneck buffer size equal to the bandwidth delay product (BDP) of the corresponding queue.

Studied queues. We study the performance of *CRQ* for the three queue combinations stated in Sec. III-B2. For a focused evaluation of *TCPTrap* and as a baseline, we also use scenarios where we feed all traffic into the standard queue which then uses drop-tail, CoDel with ECN, or CoDel with packet drop.

Traffic generation. We generate TCP traffic with a custom application on top of the Linux TCP stack which allows us to create uni- and bidirectional traffic patterns. As the spin bit enables meaningful unidirectional measurements, we only consider unidirectional traffic patterns for QUIC which we generate with *picoquic* [46], a well-maintained QUIC stack that has shown interesting insights in our previous work [22]. Additionally, we use *iperf3* to generate unresponsive UDP traffic. For simplicity, we refer to *picoquic* as QUIC and to the Linux TCP stack as TCP for the remainder of the evaluation.

Studied CCAs. Similar to our evaluation of *SpinTrap* [22], we focus on two CCAs. First, we use *Cubic* as a classic loss-based CCA that relies on the `cwnd`. It is the TCP default on major operating systems, such as Linux, Windows, and MacOS. Second, we use *BBR* as a modern model-based CCA that does not use the `cwnd` to track congestion, but estimates the available capacity via measurements of the BDP. For

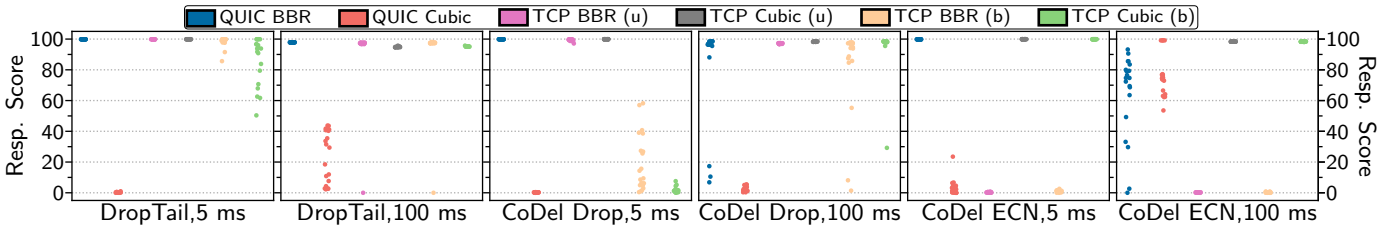

Fig. 4. Responsiveness scores for single flows using different stacks and CCAs at a drop-tail queue (left), a CoDel queue with packet drop (middle), and a CoDel queue with ECN (right) subject to different RTTs. A score of 100 corresponds to always being classified as responsive.

TCP, we use the current Linux kernel version BBRv1 while picoquic has an adapted BBRv1 variant described to add *some* reaction to packet loss and ECN. Thus, we expect picoquic and TCP BBR to behave differently but refrain from attempting to equalize these differences to avoid introducing a bias.

Data collection. During our tests, we capture logging information provided by the TCP and the QUIC traffic generators. Additionally, we collect detailed output of *CRQ*, including the timestamp of each assessment, the observed congestion signals, and the classification result which we then analyze.

Experiments. We perform twenty independent measurement runs for each configured scenario.

B. Responsiveness Assessment

The traffic isolation capabilities of *CRQ* crucially depend on the underlying responsiveness assessment provided by *SpinTrap* and *TCPTrap*. We have already evaluated *SpinTrap* in previous work [22], which is why we focus on *TCPTrap* in this paper. Yet, for reference and completeness, we also include assessment results of *SpinTrap*.

Experimental setup. We start a single flow that transmits 200 MB of data from *Server 1* to the *Client* machine. We perform dedicated measurement runs for QUIC and TCP, each with both CCAs. To further probe the impact of traffic directionality on *TCPTrap*, we distinguish two cases for TCP. In addition to the *unidirectional* (u) case above, we also emulate a *bidirectional* (b) case where the *Client* mirrors half of the received data back to the *Server*. As in our evaluation for *SpinTrap* [22], we configure a bottleneck bandwidth of 50 Mbps and evaluate two different RTTs (5 ms, 100 ms) as RTTs in between showed no considerable differences in our evaluation for *SpinTrap*. For each run, we finally compute the *responsiveness score* as the fraction of cycles in which a flow has been classified as responsive, such that responsive flows should ideally get a high score. We use the majority voting variant throughout the evaluation (cf. Sec. III-A).

Results. Fig. 4 shows the achieved responsiveness scores for a drop-tail queue (left) and CoDel with packet drop (middle) and ECN (right). Each point represents one measurement run.

Starting with drop-tail, we observe that most flows are classified as responsive in a large number of runs with QUIC Cubic being the major exception. As discussed in our previous work [22], the main reason is that the specific Cubic implementation has a small sending window after the initial large burst loss, yet does not react to small loss events that happen afterward such that it arguably behaves unresponsively. In contrast, QUIC BBR is almost always classified as responsive: in our setup, it only periodically creates loss at the start of its ProbeBW [47] cycle after which it reduces the sending rate. For TCP, there is a high classification accuracy for Cubic which shows that the *SpinTrap* concept is generally transferable to TCP. Similar to QUIC, BBR is also classified as responsive which we attribute to the same reasons as before. Surprisingly, results fluctuate more for the bidirectional case for lower RTTs, indicating that the unidirectional cycle detection might be a better fit for our responsiveness criteria.

Moving to CoDel with packet drop (middle), it can be seen that *SpinTrap* assesses QUIC Cubic as unresponsive and QUIC BBR as responsive due to similar reasons as above. Considering TCP, we observe that the unidirectional cases are always classified as responsive while both bidirectional cases are unresponsive for low RTTs. Inspecting the behavior in depth, we do indeed find sensible rate reductions for Cubic but these do not result in a responsive classification due to a combination of two reasons. First, the frequent congestion signals of CoDel cause Cubic to operate with a low cwnd which generally complicates showing an adequate response to 90 % or less (cf. Sec. III-A1). Second, the window reduction already occurs in the first cycle after the congestion signal such that Cubic already increases the sending behavior when *TCPTrap* makes its assessment. These results indicate that more lenient responsiveness criteria, e.g., regarding the rate reduction or when the response occurs, could be used to optimize the responsiveness assessment results.

SpinTrap and *TCPTrap* achieve the best classification results for CoDel with ECN. In particular, *TCPTrap* correctly classifies that Cubic is responsive to ECN and that the used BBR version is not. QUIC BBR is seemingly responsive as the ECN markings coincide with the start of the ProbeBW cycle. In contrast, QUIC Cubic at low RTTs operates at low sending windows and exhibits very nuanced responses which are often assessed as insufficient by our 90 % threshold.

The majority voting expectedly causes sharp distributions, mostly always or never classifying a flow as responsive. Thus, it seems to fulfill the intended task of providing inertia.

Takeaway. SpinTrap*'s concept is generally transferable to TCP as* TCPTrap *mostly provides sensible results. Contra-*

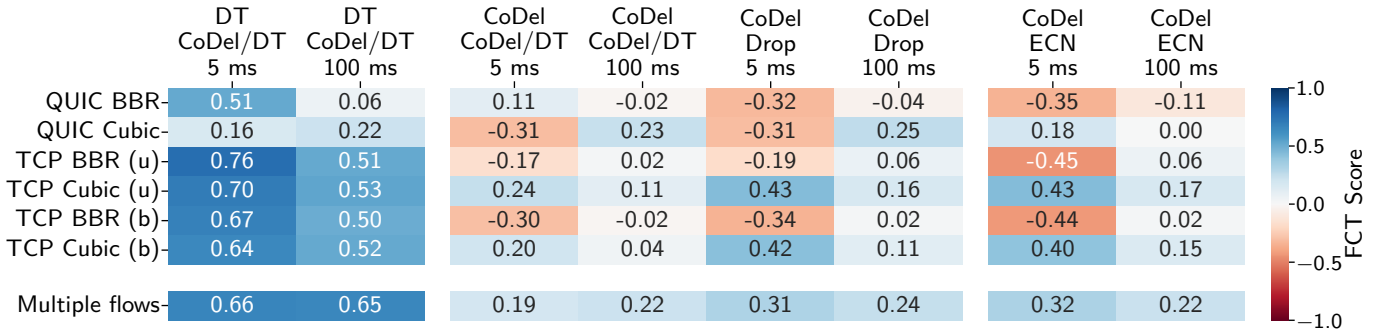

Fig. 5. Mean FCT score for one (Sec. IV-C1) and multiple (Sec. IV-C2) responsive flows across different queue combinations when competing with unresponsive UDP traffic in settings with and without *CRQ*. Positive FCT scores correspond to a performance improvement, negative scores to performance detriments.

dicting our intuition, bidirectionality of the traffic does not aid TCPTrap*'s assessment as classification results are generally worse than for unidirectional traffic. Overall, the assessment performs best with ECN because it requires responses when other congestion signals, such as packet loss or RTT estimates used by BBR, are usually not yet emitted or noticeable.*

C. Prioritizing Congestion Responsive Flows

CRQ is designed to provide traffic isolation based on the congestion responsiveness of flows. In the previous section, we have shown that *SpinTrap* and *TCPTrap* can reasonably assess flow responsiveness. Hence, in this section, we turn our attention to the effective performance that the dual-queue design of *CRQ* can achieve based on the responsiveness assessment. For this, we compare the performance of flows when competing against unresponsive UDP traffic: once in a setup without *CRQ* and once with *CRQ*.

Experimental setup. In *baseline* scenario *B*, we configure a single queue with the full bandwidth of $1\,\mathrm{Gbps}$ and one queue scheme, i.e., drop-tail, CoDel with packet drop, or CoDel with ECN. In *CRQ* scenario *C*, we deploy our *CRQ* prototype and assign half of the overall bandwidth, i.e., $500\,\mathrm{Mbps}$, to each of its two queues which we then configure with three queue combinations (Sec. III-B2). In both scenarios, we let a single (Sec. IV-C1) or twenty responsive flows (Sec. IV-C2) compete against unresponsive *iperf3* UDP traffic with an overall rate of $50\,\%$ of the corresponding queue bandwidth.

Performance metric. We evaluate the flow performance by comparing the flow completion time (FCT), i.e., the time from the initial request until the responsive flow has completed, in the *CRQ* scenario to the corresponding FCT in the *baseline* scenario. To fairly compare performance benefits and detriments, we compute an FCT score similar to [48] as

$$FCTscore = \begin{cases} 1 - \frac{FCT(C)}{FCT(B)} & \text{if FCT(B)} > \text{FCT(C)} \\ -1 + \frac{FCT(B)}{FCT(C)} & \text{if FCT(C)} \geq \text{FCT(B)} \end{cases}$$

where FCT(B) is the FCT in the baseline scenario and FCT(C) is the corresponding FCT in the *CRQ* scenario. Intuitively, our FCT score depicts the ratio of the two FCTs by mapping the behavior in the two settings to the range $[-1, 1]$ with zero indicating absolute parity. Positive values correspond to a performance benefit of *CRQ*, negative values to a performance detriment. Note that the FCT score has a non-linear behavior with larger absolute values signaling an increasingly higher impact. For example, 0.5 indicates that *CRQ* improves, i.e., reduces, the FCT to $50\,\%$ of the baseline while -0.75 indicates that *CRQ* causes a FCT that is larger by a factor of 4. For each setup, we first compute the FCT score for each run individually before taking the mean over all runs in that setup.

1) Single-Flow Behavior: In the single flow scenario, we let a single responsive flow between *Client* and *Server 1* transmit $1\,\mathrm{GB}$ of data. It competes against a single *iperf3* UDP flow between *Client* and *Server 2* which we start with a delay of $1\,\mathrm{s}$. Similar to Sec. IV-B, we study QUIC and TCP with both CCAs and consider TCP's sending directionality.

Results. The first six rows of Fig. 5 show the mean single flow FCT scores for four sets of queue combinations. The first two columns compare a drop-tail (DT) queue in the baseline with CoDel with drop (responsive queue) and DT (standard queue) in the *CRQ* scenario. The next two columns compare the same *CRQ* scenario configuration to CoDel with packet drop in the baseline. Columns five to six and seven to eight deploy CoDel in all queues with packet drop or ECN, respectively.

As can be seen, *CRQ* strongly improves the FCT in a large number of cases. The main beneficiary is TCP Cubic with a better performance in all scenarios which corresponds to the responsiveness assessment results from Sec. IV-B. In contrast, we notice performance detriments for TCP BBR as competing with iperf seems to unveil that BBR does not truly respond to packet loss or ECN. Similar observations can be made for QUIC BBR while QUIC Cubic does not benefit as much as TCP Cubic due to its more nuanced responses to congestion signals which often fall short of the required $90\,\%$ threshold.

Comparing the different queue combinations, we observe the largest benefits for the first queue scenario which we attribute to the strong differences in queuing between the DT queue in the baseline and CoDel for responsive flows. TCP Cubic shows similar results for both pure CoDel scenarios while TCP BBR expectedly struggles more when being as-

sessed based on ECN responsiveness. The two QUIC CCAs show a diverse behavior across all queue combinations which reflects the diverse behavior from Sec. IV-B.

Overall, our findings show the general feasibility of isolating flows based on their actual congestion responsiveness as *CRQ* can indeed provide benefits to responsive flows.

2) Multi-Flow Behavior: In our multi-flow scenario, we generate 20 random combinations of 20 responsive flows out of the available pool of stacks, CCAs, and traffic patterns while reusing the queue combinations. Half of the flows run between *Client* and *Server 1* and the other half between *Client* and *Server 2*, each transmitting 100 MB. We split the unresponsive *iperf3* UDP traffic to two individual flows, one per *Server* machine, which we again start with a delay of 1 s.

Results. The bottom row in Fig. 5 shows the mean FCT score across all flows and flow combinations; we first compute the score for each flow in each combination individually and then aggregate the mean scores across all flows to compare the performance in the *CRQ* scenario with the baseline.

We observe that *CRQ* improves the performance in all scenarios when jointly considering all flows. Similar to the single flow setting, the impact is larger for the pure CoDel scenarios with smaller RTTs although the difference is less nuanced than before. Digging deeper into our results, we expectedly find that the involved TCP Cubic flows see the largest improvements in all scenarios (not shown). Furthermore, most of the previously seen detriments for the other stacks and CCAs vanish. TCP BBR sees the only small performance decrease in a single scenario. We attribute these findings to the complex interaction of the many different flows which causes more congestion signals and also a sustained impact on the RTT which BBR uses for adapting its sending rate.

Takeaway. *The responsiveness-based dual-queue operation of* CRQ *can indeed provide performance benefits to flows with responsive CCAs. In particular, these benefits do not only manifest when using a single flow, but are also visible for a larger number of flows. Overall, our results indicate that basing queuing on flow responsiveness has general utility.*

V. Discussion

Our evaluation shows the general usefulness of *CRQ* and the feasibility of a responsiveness assessment for traffic isolation. In the following, we discuss a selection of further considerations, especially regarding *CRQ*'s operational use and the impact of our work on queue management in general.

Applicability to L4S. *CRQ* is inspired by the L4S DualQ, but in contrast to L4S, we deliberately use non-coupled queues to provide strict traffic isolation. Yet, we could also transfer some of our concepts to L4S, e.g., by extending the static classification based on ECT(1) use with our responsiveness assessment in order to more effectively protect against unresponsive traffic negatively affecting the L4S queue. Thus, L4S deployments could trade off additional state needed for tracking flows with a more effective traffic protection.

Deployment considerations. CE markings can be lost on the way to the receiver as there are still network devices deployed on Internet paths that change or clear code points [18]. Hence, similar to our considerations for *SpinTrap* [22], we argue that *CRQ* should ideally be deployed close to the destinations to minimize the impact of such interference. Additionally, it is still commonly assumed that most congestion happens at the edges of the network. With many services being provided by large content distribution networks with well provisioned links close to customers, the customer access side is the likely bottleneck in many connections such that providing flow isolation at these locations, e.g., by moving responsive video conferencing streams to the responsive queue, could help in improving the performance of real-time applications.

***CRQ* configuration space.** In this paper, we study a focused configuration of *CRQ* to provide a fair assessment of using congestion responsiveness for queuing. Conceptually, however, *CRQ* has a rich configuration parameter space. For example, we could explore combining queues emitting different congestion signals, or we could also extend *CRQ* to a three-queue design: one standard queue, one loss-responsive queue, and one ECN-responsive queue. Additionally, different bandwidth assignment strategies, e.g., giving the responsive queue a larger share of the overall bandwidth, could be used to tune the benefits responsive flows can receive when being processed by *CRQ*. Finally, we could also dynamically adjust the queue sizes based on queue fill rates to adapt to traffic changes. Future work can thus leverage these opportunities for creating more distinct incentives for responsive traffic.

TCP responsiveness assessment. *SpinTrap* is fueled by explicit information on individual cycles based on unidirectional spin bit measurements. In contrast, *TCPTrap* relies on implicit TCP semantics and requires bidirectional visibility. Thus, in addition to shared difficulties with the spin bit, such as interference of application delay, *TCPTrap*'s cycle detection is also subject to fluctuations when misinterpreting the implicit signals and it is inapplicable when only one traffic direction can be observed. Hence, we argue that *TCPTrap* could benefit from adding the spin bit as a TCP extension to enable broader applicability and less noisy assessments.

VI. Conclusion

Unresponsive traffic is a potential threat to AQM as it can wipe out the benefits provided by latest innovations such as L4S which crucially require a fine-grained interaction between AQM and congestion control. One of the main reasons is that existing approaches either use probabilistic or static mechanisms to, e.g., filter high bandwidth flows or specific protocols, or, worse, have no protection whatsoever.

In this paper, we propose *CRQ*, a dual-queue system inspired by L4S that can isolate unresponsive traffic based on the actual flow behavior. Relying on an assessment if flows respond to congestion signals, *CRQ* assigns responsive traffic to one queue and the remaining traffic to another, thus, protecting responsive flows from unresponsive traffic taking a too large bandwidth share. Our evaluation shows that *CRQ* can give effective benefits to responsive traffic, thus, positioning it as an incentive giver for using responsive congestion control.

References

[1] V. Jacobson, "Congestion Avoidance and Control," *ACM SIGCOMM Computer Communication Review*, vol. 18, no. 4, pp. 314–329, 1988. [Online]. Available: https://doi.org/10.1145/52325.52356

[2] N. Cardwell, Y. Cheng, S. H. Yeganeh, I. Swett, and V. Jacobson, "BBR Congestion Control," IRTF, Internet-Draft, 2022, work in progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-cardwell-iccrg-bbr-congestion-control

[3] N. Cardwell, I. Swett, and J. Beshay, "BBR Congestion Control," IETF, Internet-Draft, 2024, work in progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-cardwell-ccwg-bbr

[4] M. Duke and G. Fairhurst, "Specifying New Congestion Control Algorithms," IETF, Internet-Draft, 2024, work in progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-ccwg-rfc5033bis

[5] A. Mishra, X. Sun, A. Jain, S. Pande, R. Joshi, and B. Leong, "The Great Internet TCP Congestion Control Census," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 3, no. 3, pp. 45:1–45:24, 2019. [Online]. Available: https://doi.org/10.1145/3366693

[6] A. Mishra, L. Rastogi, R. Joshi, and B. Leong, "Keeping an Eye on Congestion Control in the Wild with Nebby," in *Proceedings of the 2024 ACM SIGCOMM Conference*, 2024. [Online]. Available: https://doi.org/10.1145/3651890.3672223

[7] J. Rüth, I. Kunze, and O. Hohlfeld, "An Empirical View on Content Provider Fairness," in *Proceedings of the 2019 IEEE/IFIP Network Traffic Measurement and Analysis Conference (TMA)*, 2019. [Online]. Available: https://doi.org/10.23919/TMA.2019.8784684

[8] A. A. Philip, R. Athapathu, R. Ware, F. F. Mkocheko, A. Schlomer, M. Shou, Z. Meng, S. Seshan, and J. Sherry, "Prudentia: Findings of an Internet Fairness Watchdog," in *Proceedings of the 2024 ACM SIGCOMM Conference*, 2024. [Online]. Available: https://doi.org/10.1145/3651890.3672229

[9] A. Langley, A. Riddoch, and A. Wilk et al., "The QUIC Transport Protocol: Design and Internet-Scale Deployment," in *Proceedings of the 2017 ACM SIGCOMM Conference*, 2017. [Online]. Available: https://doi.org/10.1145/3098822.3098842

[10] A. Mishra and B. Leong, "Containing the Cambrian Explosion in QUIC Congestion Control," in *Proceedings of the 2023 ACM Internet Measurement Conference (IMC)*, 2023. [Online]. Available: https://doi.org/10.1145/3618257.3624811

[11] G. Appenzeller, I. Keslassy, and N. McKeown, "Sizing Router Buffers," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 4, pp. 281–292, 2004. [Online]. Available: https://doi.org/10.1145/1030194.1015499

[12] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, 1993. [Online]. Available: https://doi.org/10.1109/90.251892

[13] K. Nichols and V. Jacobson, "Controlling Queue Delay," *ACM Queue*, vol. 10, no. 5, pp. 20–34, 2012. [Online]. Available: https://doi.org/10.1145/2208917.2209336

[14] G. Abbas, Z. Halim, and Z. H. Abbas, "Fairness-Driven Queue Management: A Survey and Taxonomy," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 324–367, 2016. [Online]. Available: https://doi.org/10.1109/COMST.2015.2463121

[15] R. Pan, B. Prabhakar, and K. Psounis, "CHOKe - a stateless active queue management scheme for approximating fair bandwidth allocation," in *Proceedings of the 2000 IEEE Conference on Computer Communications (INFOCOM)*, 2000. [Online]. Available: https://doi.org/10.1109/INFCOM.2000.832269

[16] G. Abbas, U. Raza, Z. Halim, and K. Kifayat, "ARCH: A dual-mode fairness-driven AQM for promoting cooperative behaviour in best effort Internet," *IET Networks*, vol. 8, no. 6, pp. 372–380, 2019. [Online]. Available: https://doi.org/10.1049/iet-net.2018.5089

[17] B. Briscoe, K. De Schepper, M. Bagnulo, and G. White, "Low Latency, Low Loss, and Scalable Throughput (L4S) Internet Service: Architecture," IETF, RFC 9330, 2023. [Online]. Available: https://doi.org/10.17487/rfc9330

[18] 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," *arXiv.2208.14523*, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2208.14523

[19] K. De Schepper, O. Albisser, O. Tilmans, and B. Briscoe, "Dual Queue Coupled AQM: Deployable Very Low Queuing Delay for All," *arXiv.2209.01078*, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2209.01078

[20] M. Letourneau, G. Doyen, R. Cogranne, and B. Mathieu, "A Comprehensive Characterization of Threats Targeting Low-Latency Services: The Case of L4S," *Journal of Network and Systems Management*, vol. 31, no. 1, p. 19, 2022. [Online]. Available: https://doi.org/10.1007/s10922-022-09706-z

[21] K. D. Schepper, B. Briscoe, and G. White, "Dual-Queue Coupled Active Queue Management (AQM) for Low Latency, Low Loss, and Scalable Throughput (L4S)," IETF, RFC 9332, 2023. [Online]. Available: https://doi.org/10.17487/rfc9332

[22] I. Kunze, C. Sander, L. Tissen, B. Bode, and K. Wehrle, "SpinTrap: Catching Speeding QUIC Flows," in *Proceedings of the 2024 IEEE/IFIP International Symposium on Network Operations and Management (NOMS)*, 2024. [Online]. Available: https://doi.org/10.1109/NOMS59830.2024.10575719

[23] I. Kunze, C. Sander, M. Kosek, L. Tissen, J. Pennekamp, and K. Wehrle, "Source code for 'Congestion-Responsive Queuing for Internet Flows'," 2025. [Online]. Available: https://github.com/COMSYS/congestion-responsive-queuing

[24] K. K. Ramakrishnan, S. Floyd, and D. L. Black, "The Addition of Explicit Congestion Notification (ECN) to IP," IETF, RFC 3168, 2001. [Online]. Available: https://doi.org/10.17487/rfc3168

[25] S. Bensley, D. Thaler, P. Balasubramanian, L. Eggert, and G. Judd, "Data Center TCP (DCTCP): TCP Congestion Control for Data Centers," IETF, RFC 8257, 2017. [Online]. Available: https://doi.org/10.17487/rfc8257

[26] K. De Schepper, O. Tilmans, B. Briscoe, and V. Goel, "Prague Congestion Control," IRTF, Internet-Draft, 2024, work in progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-briscoe-iccrg-prague-congestion-control/

[27] K. De Schepper and B. Briscoe, "The Explicit Congestion Notification (ECN) Protocol for Low Latency, Low Loss, and Scalable Throughput (L4S)," IETF, RFC 9331, 2023. [Online]. Available: https://doi.org/10.17487/rfc9331

[28] F. Baker and G. Fairhurst, "IETF Recommendations Regarding Active Queue Management," IETF, RFC 7567, 2015. [Online]. Available: https://doi.org/10.17487/rfc7567

[29] G. Chatranon, M. A. Labrador, and S. Banerjee, "BLACK: Detection and Preferential Dropping of High Bandwidth Unresponsive Flows," in *Proceedings of the 2003 IEEE International Conference on Communications (ICC)*, 2003. [Online]. Available: https://doi.org/10.1109/ICC.2003.1204258

[30] I. Yeom, "A rate-based drop policy for punishing unresponsive flows," *Computer Communications*, vol. 29, no. 10, pp. 1868–1878, 2006. [Online]. Available: https://doi.org/10.1016/j.comcom.2005.05.012

[31] J. Zheng, L. Zhao, and T. Zhang, "Improving Unresponsive Flow Control by Active Queue Management Algorithm," in *Proceedings of the 2007 IEEE Wireless Communications and Networking Conference (WCNC)*, 2007. [Online]. Available: https://doi.org/10.1109/WCNC.2007.795

[32] S. Yi, X. Deng, G. Kesidis, and C. R. Das, "A dynamic quarantine scheme for controlling unresponsive TCP sessions," *Springer Telecommunication Systems*, vol. 37, no. 4, pp. 169–189, 2008. [Online]. Available: https://doi.org/10.1007/s11235-008-9104-2

[33] G. Aldabbagh, M. Rio, and I. Darwazeh, "Fair Early Drop: An Active Queue Management Scheme for the Control of Unresponsive Flows," in *Proceedings of the 2010 IEEE International Conference on Computer and Information Technology (CIT)*, 2010. [Online]. Available: https://doi.org/10.1109/CIT.2010.449

[34] S. Yilmaz and I. Matta, "On Class-based Isolation of UDP, Short-livedand Long-lived TCP Flows," in *Proceedings of the 2001 International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, 2001. [Online]. Available: https://doi.org/10.1109/MASCOT.2001.948894

[35] T. Yamaguchi and Y. Takahashi, "A queue management algorithm for fair bandwidth allocation," *Computer Communications*, vol. 30, no. 9, pp. 2048–2059, 2007. [Online]. Available: https://doi.org/10.1016/j.comcom.2007.04.002

[36] J. Rüth, I. Poese, C. Dietzel, and O. Hohlfeld, "A First Look at QUIC in the Wild," in *Proceedings of the 2018 International Conference on Passive and Active Network Measurement (PAM)*, 2018. [Online]. Available: https://doi.org/10.1007/978-3-319-76481-8_19

[37] J. Zirngibl, P. Buschmann, P. Sattler, B. Jaeger, J. Aulbach, and G. Carle, "It's Over 9000: Analyzing Early QUIC Deployments with the Standardization on the Horizon," in *Proceedings of the 2021 ACM Internet Measurement Conference (IMC)*, 2021. [Online]. Available: https://doi.org/10.1145/3487552.3487826

[38] C. Sander, I. Kunze, K. Wehrle, and J. Rüth, "Video Conferencing and Flow-Rate Fairness: A First Look at Zoom and the Impact of Flow-Queuing AQM," in *Proceedings of the 2021 International Conference on Passive and Active Network Measurement (PAM)*, 2021. [Online]. Available: https://doi.org/10.1007/978-3-030-72582-2_1

[39] S. Floyd and K. Fall, "Promoting the Use of End-to-End Congestion Control in the Internet," *IEEE/ACM Transactions on Networking*, vol. 7, no. 4, pp. 458–472, 1999. [Online]. Available: https://doi.org/10.1109/90.793002

[40] S. Bauer, B. Jaeger, F. Helfert, P. Barias, and G. Carle, "On the Evolution of Internet Flow Characteristics," in *Proceedings of the 2021 ACM/IRTF Applied Networking Research Workshop (ANRW)*, 2021. [Online]. Available: https://doi.org/10.1145/3472305.3472321

[41] M. Allman, S. Floyd, and C. Partridge, "Increasing TCP's Initial Window," IETF, RFC 3390, 2002. [Online]. Available: https://doi.org/10.17487/rfc3390

[42] B. Trammell and M. Kühlewind, "The Wire Image of a Network Protocol," IETF, RFC 8546, 2019. [Online]. Available: https://doi.org/10.17487/rfc8546

[43] M. Allman, R. Beverly, and B. Trammell, "Principles for Measurability in Protocol Design," *ACM SIGCOMM Computer Communication Review*, vol. 47, no. 2, pp. 2–12, 2017. [Online]. Available: https://doi.org/10.1145/3089262.3089264

[44] S. Sengupta, H. Kim, and J. Rexford, "Continuous In-Network Round-Trip Time Monitoring," in *Proceedings of the 2022 ACM SIGCOMM Conference*, 2022. [Online]. Available: https://doi.org/10.1145/3544216.3544222

[45] J. Gettys and K. Nichols, "Bufferbloat: Dark Buffers in the Internet," *ACM Queue*, vol. 9, no. 11, pp. 40–54, 2011. [Online]. Available: https://doi.org/10.1145/2063166.2071893

[46] "Picoquic," 2024. [Online]. Available: https://github.com/private-octopus/picoquic

[47] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR: Congestion-Based Congestion Control: Measuring bottleneck bandwidth and round-trip propagation time," *ACM Queue*, vol. 14, no. 5, pp. 20–53, 2016. [Online]. Available: https://doi.org/10.1145/3012426.3022184

[48] I. Kunze, J. Ruth, and O. Hohlfeld, "Congestion Control in the Wild—Investigating Content Provider Fairness," *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp. 1224–1238, 2020. [Online]. Available: https://doi.org/10.1109/TNSM.2019.2962607