Slowdown as a Metric for Congestion Control Fairness

Adrian Zapletal
Delft University of Technology

Fernando Kuipers
Delft University of Technology

ABSTRACT

The conventional definition of fairness in congestion control is flow rate fairness. However, Internet users typically care about flow completion times (FCTs) and flow rate fairness does not lead to equitable FCTs for different users. Therefore, we reconsider what it means for congestion control to be fair and posit a novel stance on fairness: it is fair when no flow unnecessarily prolongs another flow. Based on this stance, we propose an evaluation framework for congestion control fairness that uses slowdown (normalized FCT) as the metric.

We demonstrate the usefulness of our framework through surprising experiment results: in theory, prioritizing short flows should outperform fair queueing, but we show that this is not the case due to slow start dominating short flows. The framework can also analyze traditional flow rate fairness; we do so and verify well-known "fairness" issues, but additionally, we show that flow rate unfairness does not induce slowdown and is thus not a problem per se.

CCS CONCEPTS

• **Networks** → **Transport protocols**;

KEYWORDS

Congestion control, Fairness

ACM Reference Format:
Adrian Zapletal and Fernando Kuipers. 2023. Slowdown as a Metric for Congestion Control Fairness. In *The 22nd ACM Workshop on Hot Topics in Networks (HotNets '23), November 28–29, 2023, Cambridge, MA, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3626111.3628185

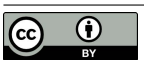
1 INTRODUCTION

With congestion control being crucial for the operability of the Internet and a panoply of new algorithms having been proposed recently [1, 5, 20, 24–26, 43, 46, 79–81, 84], it is paramount that we evaluate congestion control in a *realistic* manner that corresponds to the desires of *users*. An essential criterion for congestion control is fairness: every user shall be treated equitably. Conventionally, researchers evaluate fairness by starting multiple flows that share a bottleneck, making these flows send as much data as they can for some duration, and assessing for flow rate fairness, that is, whether the flows obtain equal throughput.

This conventional evaluation setup is not realistic and does not focus on the desires of users. It has three core problems:
① *Flow generation.* Flows that send indefinite amounts of data for some duration do not correspond to real flows in the Internet. Actual web flows have a certain size and end after transmitting their data.
② *Flow rate fairness is not optimal.* If a long flow and a short flow share a link, enforcing flow rate fairness prolongs the short flow's Flow Completion Time (FCT) unnecessarily. Theoretically, all FCTs could be shorter than or the same as under flow rate fairness [13, 59, 64] and, for most web applications, users mainly care about FCT [26]. Hence, optimizing for flow rate fairness does not optimize for the desires of users.
③ *Flow rate unfairness does not necessarily matter.* If one flow obtains more bandwidth than another flow of similar size, there is flow rate unfairness. However, this unfairness does not necessarily hurt user experience. The flow with a larger bandwidth share completes faster; after its completion, the other flow can utilize the total bandwidth and completes as fast as it would have completed under flow rate fairness.

The conventional approach to evaluate fairness is clearly deficient. Yet, the question remains: how should one evaluate congestion control fairness? To answer this question, we take a step back and ask what fairness really means. We argue that fairness is about the equitable treatment of users. Based on this, we posit a novel stance on congestion control fairness: **if no flow prolongs other flows unnecessarily, the situation is fair.** Equivalently, it is unfair if a flow induces an unnecessarily long FCT for another flow. By "unnecessarily" we mean that the other flow could complete faster without this affecting the first flow. We elucidate our stance on fairness with an example: Alice downloads a game while her roommate Bob browses the web. Alice's download clogs their home access link, which slows down Bob's traffic and deteriorates his experience. This situation is unfair to Bob because there is no need for this deterioration; Bob's traffic could be sped up without prolonging Alice's download.

Based on our stance on fairness and the aforementioned problems with the conventional evaluation approach, we



HotNets '23, November 28–29, 2023, Cambridge, MA, USA

ACM ISBN 979-8-4007-0415-4/23/11.
https://doi.org/10.1145/3626111.3628185

describe a *realistic* and *user-centric* evaluation framework for congestion control. This framework generates flows using flow sizes and flow inter-arrival times from real-world traces (solving problem ①) and uses *slowdown* (normalized FCT) as the metric (solving problems ② and ③). Slowdown has previously been considered for performance evaluation in data center research [3, 4, 6, 31, 39, 52, 63, 65]; given our new stance on fairness, we argue it is also the ideal metric for congestion control fairness. For example, if Bob's flows experience high slowdown because Alice's flow starves them, there is a fairness issue.

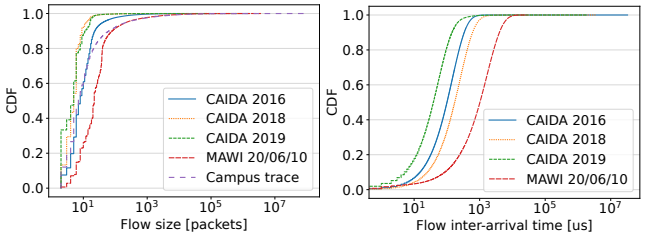
The goal of this paper is to encourage the networking community to embrace our stance on fairness and to use slowdown as a metric for evaluating fairness in future research on congestion control. One can use our proposed framework to evaluate congestion control for performance, fairness in our novel sense, and even traditional flow rate fairness. We implemented the framework and conducted first experiments with it to demonstrate how one can use it. Our experiments lead to novel insights:

- While flow rate fairness is not optimal in theory (problem ②), enforcing it works well in practice. Counterintuitively, *prioritizing short flows does not lead to lower slowdown than fair queueing* because the FCT of short flows is dominated by slow start.
- We verify problem ③ and show that *flow rate unfairness does not necessarily matter*. If a Congestion Control Algorithm (CCA) can quickly utilize free bandwidth (like BBR does, for instance), flows experience no unnecessary slowdown when there is flow rate unfairness.

Related Work. We are not the first to hint at the unrealistic nature of the flows generated in the conventional setup (problem ①) [77] or to challenge flow rate fairness (problem ②) [12, 13, 64, 70] although, to the best of our knowledge, we are the first to describe how flow rate unfairness does not necessarily matter (problem ③). Nonetheless, to this date, there is no practical alternative for the technical evaluation of congestion control fairness. Therefore, researchers simply continue to use the conventional evaluation setup. A recent proposal is to focus on whether a new CCA is deployable based on the harm it causes to existing CCAs in the network [77]. This approach is useful, but it focuses on inter-CCA deployability. We focus on both inter- and intra-CCA fairness, which is orthogonal to the approach in [77].

2 CONVENTIONAL EVALUATION APPROACH

The vast majority of congestion control research uses a similar setup to evaluate fairness [1, 2, 5, 7, 11, 16, 18–21, 24, 25, 28–30, 33–38, 45, 47–51, 53, 55, 57, 58, 61, 62, 67, 69, 71, 73–76, 79, 81–84]. In this setup, multiple senders send flows through a shared bottleneck link; these flows send data from an unlimited data backlog for a fixed duration using a tool such as iperf. While the flows transmit, one measures their throughput and evaluates whether all flows obtain equal throughput. This is called *flow rate fairness*. It is common to summarize the results using a fairness metric such as Jain's index [42]. This conventional evaluation setup has various problems related to how flows are generated and how fairness is evaluated. We highlight these problems hereafter.



(a) Flow sizes. **(b) Flow inter-arrival times.**

Figure 1: Flow characteristics in real-world datasets.

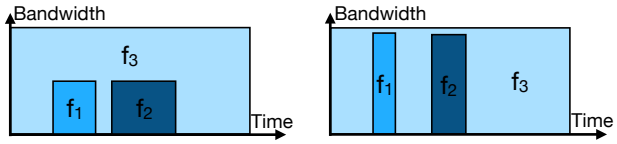
2.1 Flow Generation

Flows have varying sizes. Unlimitedly backlogged flows, as generated in the conventional setup, are not realistic. In the Internet, flows have a certain size; flows need to transmit some data, and after transmitting their data, they end. Additionally, flows arrive at varying times. One might argue that unlimitedly backlogged flows emulate long flows, but most flows in the Internet are short, and even for long flows, it is important to take their size into consideration, as we will discuss in Section 2.2. We verified that flows in the Internet vary greatly by analyzing flow sizes and flow inter-arrival times in various real-world traces by CAIDA [17], MAWI [60], and a campus trace from our institution. We used scripts based on [9] to conduct this analysis. Figure 1 illustrates our results, showing that all datasets follow similar heavy-tailed distributions and most flows are indeed short.

Flows may be rate-limited. The conventional setup assumes that every flow aims to achieve as high throughput as possible. However, a flow may be limited to some rate. Examples of such rate-limited flows are streams, file downloads that are throttled by the file host, and flows that have their bottleneck elsewhere in the network. The conventional evaluation setup does not account for rate-limited flows.

2.2 Flow Rate Fairness as a Metric

Flow rate fairness is not optimal. Establishing fairness in the Internet is a virtuous goal; every user shall have an equally good experience. Yet, the conventional fairness metric – flow rate fairness – does not represent user experience.



(a) Flow rate fairness. (b) Prioritize short flows.

Figure 2: Flow rate fairness leads to suboptimal FCTs. Each area depicts one flow, illustrating its bandwidth usage (y-axis) and time to complete (x-axis).

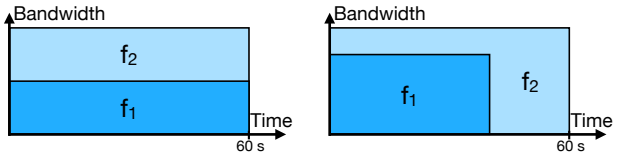
What users care about is receiving their data fast, and optimizing for flow rate fairness does not necessarily achieve this goal. Consider the example depicted in Figure 2 where two short flows (f_1 and f_2) share the network with one long flow (f_3)[1]. When all flows are granted equal throughput at all times (Figure 2a), the short flows complete slower than when short flows are prioritized (Figure 2b) while the long flow's completion time is the same in both scenarios[2]. Thus, flow rate fairness is not optimal for user experience. This issue with flow rate fairness has been recognized before [12, 13, 59, 64, 78], but it seems that the networking community does not heed this finding. We go one step further and argue that the scenario in Figure 2a is *unfair* because the short flows experience needless performance degradation.

Given the characteristics of web flows we derived in Figure 1, where most flows are short and few are long, the example depicted in Figure 2 is realistic. Since most flows are short and flow rate fairness puts short flows at a disadvantage, flow rate fairness as a metric is not fit for purpose.

Flow rate unfairness does not necessarily matter. Not only is optimizing for flow rate fairness not optimal, but it also does not always matter if one fails to achieve it. Consider two long flows f_1 and f_2 of the same size that start at the same time like depicted in Figure 3. In our example, the flows take 30 s to complete if they are on their own. If they share the bandwidth equally, both flows complete after 60 s (Figure 3a). If f_1 takes up a larger share of the bandwidth, it completes faster. For instance, if it takes up 75% of the bandwidth, it completes after 40 s. According to flow rate fairness, this is bad. However, f_2 can now utilize the bandwidth fully and still completes after 60 s (Figure 3b). Compared to ideal flow rate fairness, no flow gets prolonged. That is, in this scenario, flow rate unfairness does not cause any damage. Therefore, we argue that the scenario in Figure 3b is *fair* because no flow degrades another flow's performance. Additionally, this example shows how it is essential to take flow sizes into account even for long flows rather than just emulating long flows using unlimitedly backlogged flows.

[1]This example is inspired by a similar example by Briscoe [13].

[2]Moreover, it is proven that prioritizing short flows does not starve long flows if the flows follow a heavy-tailed distribution (as web flows do) [8, 59].



(a) Flow rate fairness. (b) Flow rate unfairness.

Figure 3: Flow rate *un*fairness is not a problem per se because it does not necessarily prolong FCTs.

3 OUR EVALUATION FRAMEWORK

We have derived problems with the conventional evaluation setup and have argued for a novel stance on fairness: *it is fair when no flow prolongs other flows unnecessarily*. Based on these prolegomena, we propose an evaluation framework for congestion control fairness. We want to encourage the community to take the final leap away from using flow rate fairness as the primary metric for congestion control fairness by providing a practical and simple alternative and showcasing its usefulness. Our evaluation framework requires a setup where multiple senders share a bottleneck, similarly to the conventional setup. It differs from the conventional setup in how flows are generated and in the metric: flows are representative of actual web flows, and the metric is slowdown (i.e., the normalized FCT).

3.1 Flow Generation

Flows are generated either with a predefined size and start time (e.g., to enforce a long flow that clogs the bottleneck) or by leveraging the flow size and inter-arrival time distributions shown in Figure 1 to generate randomized flows that resemble real-world workloads. To generate a randomized flow, we pick one random value from a flow size distribution and one from a flow inter-arrival time distribution. This setup allows us to conduct realistic but controlled experiments in which flows correspond to actual web flows. Additionally, our framework supports generating rate-limited flows by limiting a sender's outgoing bandwidth.

3.2 Slowdown as a Metric

For most applications, users mainly care about their flows finishing fast (e.g., web browsing, e-commerce, social apps, messaging, file sharing). So, a fairness metric that reflects user experience should be based on FCT. However, FCT by itself is not useful because it depends on the flow size, available bandwidth, and Round-Trip Time (RTT), making it hard to compare FCTs of differently sized flows. Additionally, FCTs are not intuitive: it is not intuitively clear to people what constitutes a "good" FCT in a given scenario.

These problems with FCT as a metric arise because FCT is an absolute value. Normalizing an FCT by the theoretically

optimal FCT (given flow size, bandwidth, and RTT) yields the *slowdown*. We define it as $slowdown = \frac{measured\_fct}{optimal\_fct}$ where $optimal\_fct = \frac{flow\_size}{\min\{bandwidth, rate\_limit\}} + rtt$. The *bandwidth* is the bottleneck capacity, and the *rate_limit* is the flow's rate limit if it has one (∞ if not). Slowdown is a relative value, making it easy to compare the slowdown of differently sized flows. Additionally, slowdown is intuitive: prolonging a flow that could take 500 ms by 2 s annoys users (slowdown of 5), while prolonging a flow that could take 1 min by 2 s is fine (slowdown of 1.033). As an added bonus, slowdown is useful for both evaluation of performance (does a single flow complete fast?) and evaluation of fairness (do coexisting flows degrade each other's performance?). In fact, slowdown has been used as a *performance* metric before. We are the first to propose using it as a *fairness* metric.

Slowdown intuitively represents the needs of users. However, slowdown is a per-flow value, whereas fairness experiments involve multiple flows. Therefore, we compute both the mean slowdown (to see whether flows perform equitably on average) and the maximum slowdown (to see whether any flow gets starved) across all flows.

When there are multiple active flows and the bottleneck is congested, some slowdown is inevitable, and it is unclear what the best values for mean and maximum slowdown are. Then, it is useful to compare the slowdown results of experiments to the theoretically optimal slowdown. In our framework, we simulate an idealized network that uses Shortest Remaining Processing Time (SRPT) scheduling at the bottleneck router as an optimum to compare results of experiments to. SRPT optimizes the mean FCT [72] and is 2-competitive for mean slowdown [66][3].

3.3 Implementation

We implemented our evaluation framework using Python. In our implementation, one server acts as a controller that establishes ssh connections to servers that act as senders, router, and receiver, and instructs them to configure parameters and start processes for sending and receiving flows. We use Linux sockets in a C script for sending and receiving flows. We make our implementation publicly available[4].

4 EXPERIMENTS

Our framework is useful for experiments on the performance of flows that are alone in the network, on fairness in our novel sense (flows should not prolong other flows), and on fairness in the traditional sense (flow rate fairness). We demonstrate such experiments using our implementation on six physical servers that are directly connected via Gigabit Ethernet. Four servers act as senders, one as router, and one as receiver. The bottleneck link between the router and the receiver is limited to 100 Mbit/s. We deployed Linux kernel 5.13.12 with BBRv2 alpha enabled[5] on all servers. Table 1 summarizes the values of the parameters we used.

CCAs	Reno, Cubic, BBR, BBRv2
RTTs	10 ms, 20 ms, 50 ms, 100 ms
Buffer sizes	50 packets, 250 packets, 1000 packets (1 packet = 1500 Bytes)
QDiscs	FIFO, FQ/FQ_CoDel, HHF

Table 1: Parameter values used in our experiments.

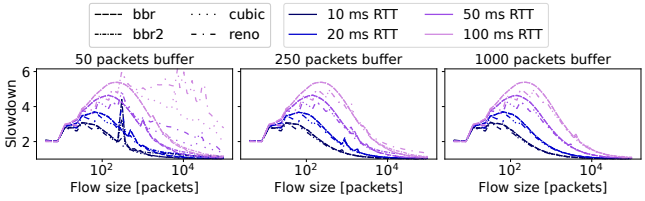


Figure 4: Using our framework for *performance analysis*: the slowdown per flow size of flows on their own.

Performance of solo flows. We started flows of varying sizes on their own and measured their slowdown with different CCAs, RTTs, and buffer sizes to evaluate their solo performance. Figure 4 summarizes the slowdown per flow size. Generally, the shortest flows experience a slowdown of 2 because they complete in two RTTs: one RTT for the TCP handshake and one for the data transmission. For short flows larger than the Initial Window (IW), slow start's exponential growth is inefficient and leads to notable slowdown. For long flows, the effect of this inefficiency diminishes. Following our flow size analysis in Figure 1, about half of all flows are short enough to complete within two RTTs. Almost all remaining flows, i.e., *about half of all flows, are sized such that slow start induces significant slowdown for them.* Only *about 1% of flows are large enough that steady state overshadows slow start* and they experience only minuscule slowdown.

As one might expect, shorter RTTs lower the slowdown induced by slow start. For Reno and, to a lesser extent, Cubic, even long flows experience notable slowdown when the buffer is small and the RTT is long because their loss-based mechanisms struggle to keep up the necessary throughput. We have also tested different IW configurations; as expected, a larger IW value generally reduces slowdown when flows are alone (results omitted here for brevity).

Novel type of fairness experiments. A well-known issue where flows prolong other flows unnecessarily, which we

[3]While SRPT is not theoretically optimal for slowdown, we still use it as optimum because it is proven that no online algorithm exists that optimizes slowdown in every scenario [10, 66], and SRPT is close to optimal for mean [66] and maximum slowdown [10]. SRPT only leads to suboptimal slowdown when a short flows starts shortly before a long flow ends [66].

[4]https://gitlab.tudelft.nl/lois/cc-fct-eval-framework

[5]https://github.com/google/bbr/tree/v2alpha at commit `a23c4bb`.

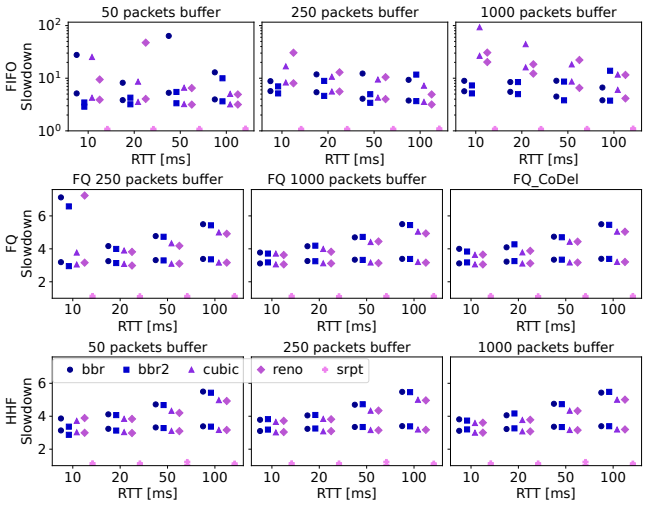


Figure 5: Using our framework for *novel fairness experiments*: the mean and maximum slowdown when a bulk flow and short flows share a bottleneck. We compare FIFO (on a log-scale y-axis), FQ, and HHF.

deem unfair, is bufferbloat [32]: bulk flows fill buffers and increase the latency for coexisting flows. This increased latency causes unnecessary slowdown. To test this issue, we set up experiments where one bulk flow coexists with short flows that we generate by picking random flow sizes and inter-arrival times from a MAWI trace [60]. We used the same seed for the random number generation across repetitions of the experiment to ensure that the results are comparable. Different seeds lead to similar results (omitted here for brevity). We tested different CCAs, RTTs, buffer sizes, and different queueing disciplines at the bottleneck: First In First Out (FIFO), Fair Queueing (FQ) and FQ_CoDel [41], and Heavy-Hitter Filter (HHF). HHF tries to detect long flows (using classic heavy hitter detection [27]), puts long and short flows into two separate queues, and prioritizes the short flow queue.

Figure 5 illustrates the mean and the maximum slowdown for each combination of parameters. It also shows the mean and maximum slowdown of an idealized network with SRPT as the optimum (mean ≈ 1.01 and maximum ≈ 1.1). When using FIFO, the bulk flow fills up the buffer to some extent with all tested CCAs, thereby slowing down the short flows. Buffer-filling CCAs (Reno and Cubic) and large buffers aggravate this issue and can lead to tremendous performance degradation for short flows (up to a slowdown of 93 in our experiment). We argue that this performance degradation is highly unfair. BBR and BBRv2 do not fully fill the buffer, but they do create a queue that causes notable slowdown for short flows (maximum slowdown between 6 and 12).

Separating flows resolves bufferbloat. This is evident in the results where the router runs FQ or FQ_CoDel. When using FQ, we set the maximum per-flow queue size to 1/5 of the total buffer size. For small buffers, this leads to tiny queues, which cause notable packet loss and, thus, slowdown. With FQ_CoDel, the queues are handled by CoDel, and the buffer is simply large enough to never overflow (10000 packets). Apart from situations with high packet loss due to small FQ buffers, no flow experiences dramatic slowdown with FQ or FQ_CoDel. Nonetheless, the results are still not optimal.

Based on the example in Figure 2, where prioritizing short flows leads to lower FCTs than FQ, we expected HHF to outperform FQ because it prioritizes short flows. However, our results show that *the slowdowns with HHF hardly differ from the slowdowns with FQ*. This is because with bufferbloat being resolved through flow separation, the main factor contributing to the slowdown of short flows is slow start. In fact, *the slowdown coexisting flows experience when separated into different queues is almost the same as the slowdown flows experience when they are alone*. In both our solo flow experiments (Figure 4) and our experiments where coexisting flows are separated (Figure 5), the maximum slowdown is about 3 for 10 ms RTT flows and about 5.5 for 100 ms RTT flows.

There is another problem with HHF: it requires some parameters to be configured, and a configuration that works well in one setting might not work in others. The heavy hitter detection that HHF employs classifies a flow as a heavy hitter if the flow has sent b bytes within a time period t; the parameters b and t need to be configured. We configured HHF such that it works in the setting we show here, but in other settings, the same configuration either fails to detect heavy hitters or falsely detects short flows as heavy hitters. A scheme like FQ_CoDel, on the other hand, always works[6].

Traditional fairness experiments. While the novel type of fairness experiments we showed above is our framework's primary use case, one can also use the framework to evaluate traditional flow rate fairness and verify conclusions that previous work has drawn using the conventional setup. However, from the same experiments, one can also draw novel conclusions. To demonstrate this, we verified well-known results regarding RTT unfairness (i.e., flows with different RTTs obtain different shares of the bandwidth), but we show that this "unfairness" has no negative impact per se because it does not necessarily induce additional slowdown. We conducted these experiments by starting two flows of the same size simultaneously. If they achieve flow rate fairness, both obtain a slowdown of 2. We fixed the first flow's RTT to 20 ms and varied the RTT of the second flow.

[6]Ironically, we started this work expecting that we would show how a scheme like HHF outperforms FQ and should thus be deployed in all routers, but we ended up showing that this is not the case!

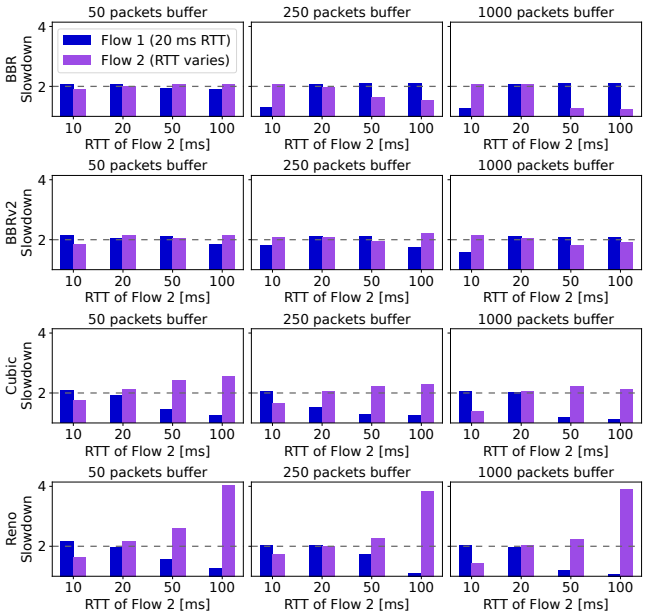


Figure 6: Using our framework for *traditional fairness experiments*: the slowdown per flow for two long flows of the same size, where the RTT of the first flow is fixed to 20 ms and the RTT of the second flow varies. The dotted line indicates the slowdown both flows obtain if they achieve flow rate fairness. We verify RTT unfairness, but also show that it does not always matter.

Figure 6 illustrates the slowdown per flow in these experiments and verifies various results that researchers have found using the conventional setup: BBR suffers from an RTT unfairness that favors flows with long RTTs [37, 71]. BBRv2 still suffers from some RTT unfairness, but the issue is less pronounced than with its predecessor [33, 73]. With Reno and Cubic, short RTT flows obtain more bandwidth [15]. When the RTTs of flows are equal, Reno's Additive Increase Multiplicative Decrease (AIMD) scheme leads to approximate flow rate fairness [22]. Lastly, the results verify again that Reno, as well as, to a lesser extent, Cubic, struggle to fully utilize the bandwidth when the buffer is small and the RTT is long [20]. While Figure 6 confirms well-known issues regarding RTT unfairness, it also shows a new result: *flow rate unfairness does not necessarily matter as long as the CCA can quickly utilize newly available bandwidth.* Flow rate unfairness does not imply that a flow gets slowed down unnecessarily. In our experiments, BBR and BBRv2 flows utilize bandwidth fast in all scenarios and never experience a slowdown notably larger than 2, which is close to optimal. Reno, in particular, struggles with utilizing bandwidth when a flow has a long RTT because AIMD raises the sending rate too slowly.

5 DISCUSSION

Why flows as the core entity? Some researchers have argued that flows are not the correct entity to look at because (1) they are not the economic actors in the Internet [12, 14], and (2) users can (and do) start multiple flows per application [12]. While both points are true, (1) economic entities in the Internet do not map well to users because an economic entity can comprise an arbitrary number of users, and the foundation for fairness should be users. Flows also do not map directly to users, but they are more representative of actual applications and, thus, of user experience than economic entities. Hence, flows are more useful as a foundation for fairness. (2) The problem with starting multiple flows is that it "games" flow rate fairness as a metric. However, it does not "game" slowdown: in common cases where multiple flows are started (e.g., loading a website using multiple flows), users effectively care about the completion time of the whole transfer (e.g., when the website has loaded fully). Hence, for such applications, one can group flows using an abstraction such as coflow [23], for which our framework can, in principle, also be applied.

For which applications is this framework useful? The focus of our evaluation framework are classic web applications. Another popular application is streaming. While our framework can model simple streams (e.g., audio streams) using rate-limited flows, this approach is often not sufficient for a realistic evaluation, especially for video content. Video streams typically use adaptive bitrate schemes and change the video quality during a session. However, throughput or flow rate fairness are also unsuitable metrics for video. What matters for video is the perceived Quality of Experience (QoE). There are application-specific metrics for video QoE and QoE fairness [40, 44, 54, 56, 68]. Arguably, these metrics are better suited for evaluating video streams. Another type of applications are interactive applications that keep connections alive and transmit data when needed. For such applications, one can evaluate the slowdown of each data transmission within an active connection.

How can we achieve lower slowdown? Current CCAs perform suboptimally even when flows are alone (Figure 4). A faster flow startup mechanism would improve performance (but must not cause slowdown for coexisting flows). With faster flow startup, a scheduler that prioritizes short flows could achieve the desired effect and improve fairness compared to FQ (unlike in Figure 5). Finally, CCAs must ensure they utilize newly available bandwidth quickly (Figure 6).

Acknowledgements. We thank Chenxing Ji, Anup Bhattacharjee, George Iosifidis, Georgios Smaragdakis, and the anonymous reviewers for their valuable comments. This work was supported by the Netherlands Organisation for Scientific Research (NWO) under the CATRIN project.

REFERENCES

[1] Soheil Abbasloo, Chen-Yu Yen, and H. Jonathan Chao. 2020. Classic Meets Modern: A Pragmatic Learning-Based Congestion Control for the Internet. In *ACM SIGCOMM*.

[2] Mohammad Alizadeh, Albert Greenberg, David A. Maltz, Jitendra Padhye, Parveen Patel, Balaji Prabhakar, Sudipta Sengupta, and Murari Sridharan. 2010. Data Center TCP (DCTCP). *ACM CCR* 40, 4 (2010).

[3] Mohammad Alizadeh, Shuang Yang, Milad Sharif, Sachin Katti, Nick McKeown, Balaji Prabhakar, and Scott Shenker. 2013. pFabric: Minimal Near-Optimal Datacenter Transport. In *ACM SIGCOMM*.

[4] Serhat Arslan, Yuliang Li, Gautam Kumar, and Nandita Dukkipati. 2023. Bolt: Sub-RTT Congestion Control for Ultra-Low Latency. In *USENIX NSDI*.

[5] Venkat Arun and Hari Balakrishnan. 2018. Copa: Practical Delay-Based Congestion Control for the Internet. In *USENIX NSDI*.

[6] Wei Bai, Li Chen, and Kai Chen. 2015. Information-Agnostic Flow Scheduling for Commodity Data Centers. In *USENIX NSDI*.

[7] Andrea Baiocchi, Angelo P. Castellani, and Francesco Vacirca. 2007. YeAH-TCP: Yet Another Highspeed TCP. In *PFLDnet*.

[8] Nikhil Bansal and Mor Harchol-Balter. 2001. Analysis of SRPT Scheduling: Investigating Unfairness. In *ACM SIGMETRICS*.

[9] Simon Bauer, Benedikt Jaeger, Fabian Helfert, Philippe Barias, and Georg Carle. 2021. On the Evolution of Internet Flow Characteristics. In *ACM/IRTF ANRW*.

[10] Michael A. Bender, Soumen Chakrabarti, and S. Muthukrishnan. 1998. Flow and Stretch Metrics for Scheduling Continuous Job Streams. In *ACM-SIAM Symposium on Discrete Algorithms*.

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

[12] Bob Briscoe. 2007. Flow Rate Fairness: Dismantling a Religion. *ACM CCR* 37, 2 (2007).

[13] Bob Briscoe. 2019. Per-Flow Scheduling and the End-to-End Argument. Discussion Paper TR-BB-2019-001, bobbriscoe.net. (2019).

[14] Lloyd Brown, Ganesh Ananthanarayanan, Ethan Katz-Bassett, Arvind Krishnamurthy, Sylvia Ratnasamy, Michael Schapira, and Scott Shenker. 2020. On the Future of Congestion Control for the Public Internet. In *ACM HotNets*.

[15] P. Brown. 2000. Resource sharing of TCP connections with different round trip times. In *IEEE INFOCOM*.

[16] Hadrien Bullot, Roger Leslie Cottrell, and Richard Hughes-Jones. 2003. Evaluation of Advanced TCP Stacks on Fast Long-Distance Production Networks. *Springer Journal of Grid Computing* 1 (2003).

[17] CAIDA. 2016–2019. The CAIDA UCSD Anonymized Internet Traces – 2016 to 2019. https://www.caida.org/catalog/datasets/passive_dataset. (2016–2019).

[18] Carlo Caini and Rosario Firrincieli. 2004. TCP Hybla: A TCP Enhancement for Heterogeneous Networks. *Wiley Int. J. Satell. Commun. Netw.* 22, 5 (2004).

[19] Yi Cao, Arpit Jain, Kriti Sharma, Aruna Balasubramanian, and Anshul Gandhi. 2019. When to Use and When Not to Use BBR: An Empirical Analysis and Evaluation Study. In *ACM IMC*.

[20] Neal Cardwell, Yuchung Cheng, C. Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. 2016. BBR: Congestion-Based Congestion Control: Measuring Bottleneck Bandwidth and Round-Trip Propagation Time. *ACM Queue* 14, 5 (2016).

[21] Daniela M. Casas-Velasco, Fabrizio Granelli, and Nelson L. S. da Fonseca. 2022. Impact of background traffic on the BBR and CUBIC variants of the TCP protocol. *IEEE Networking Letters* (2022).

[22] Dah-Ming Chiu and Raj Jain. 1989. Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks. *Computer Networks and ISDN Systems* 17 (1989).

[23] Mosharaf Chowdhury and Ion Stoica. 2012. Coflow: A Networking Abstraction for Cluster Applications. In *ACM HotNets*.

[24] Mo Dong, Qingxi Li, Doron Zarchy, Brighten Godfrey, and Michael Schapira. 2015. PCC: Re-architecting Congestion Control for Consistent High Performance. In *USENIX NSDI*.

[25] Mo Dong, Tong Meng, Doron Zarchy, Engin Arslan, Yossi Gilad, Brighten Godfrey, and Michael Schapira. 2018. PCC Vivace: Online-Learning Congestion Control. In *USENIX NSDI*.

[26] Nandita Dukkipati and Nick McKeown. 2006. Why Flow-Completion Time is the Right Metric for Congestion Control. *ACM CCR* 36, 1 (2006).

[27] Cristian Estan and George Varghese. 2002. New Directions in Traffic Measurement and Accounting. In *ACM SIGCOMM*.

[28] Ferenc Fejes, Gergő Gombos, Sándor Laki, and Szilveszter Nádas. 2019. Who Will Save the Internet from the Congestion Control Revolution?. In *ACM Workshop on Buffer Sizing*.

[29] Ferenc Fejes, Gergő Gombos, Sándor Laki, and Szilveszter Nádas. 2020. On the Incompatibility of Scalable Congestion Controls over the Internet. In *IFIP Networking*.

[30] Cheng Peng Fu and S.C. Liew. 2003. TCP Veno: TCP Enhancement for Transmission Over Wireless Access Networks. *IEEE Journal on Selected Areas in Communications* 21, 2 (2003).

[31] Peter X. Gao, Akshay Narayan, Gautam Kumar, Rachit Agarwal, Sylvia Ratnasamy, and Scott Shenker. 2015. pHost: Distributed near-Optimal Datacenter Transport over Commodity Network Fabric. In *ACM CoNEXT*.

[32] Jim Gettys and Kathleen Nichols. 2011. Bufferbloat: Dark Buffers in the Internet. *ACM Queue* 9, 11 (2011).

[33] Jose Gomez, Elie Kfoury, Jorge Crichigno, Elias Bou-Harb, and Gautam Srivastava. 2020. A Performance Evaluation of TCP BBRv2 Alpha. In *IEEE International Conference on Telecommunications and Signal Processing (TSP)*.

[34] Luigi A. Grieco and Saverio Mascolo. 2004. Performance Evaluation and Comparison of Westwood+, New Reno, and Vegas TCP Congestion Control. *ACM CCR* 34, 2 (2004).

[35] Sangtae Ha, Yusung Kim, Long Le, Injong Rhee, and Lisong Xu. 2006. A Step toward Realistic Performance Evaluation of High-Speed TCP Variants. In *PFLDnet*.

[36] Sangtae Ha, Injong Rhee, and Lisong Xu. 2008. CUBIC: A New TCP-Friendly High-Speed TCP Variant. *ACM OSR* 42, 5 (2008).

[37] Mario Hock, Roland Bless, and Martina Zitterbart. 2017. Experimental Evaluation of BBR Congestion Control. In *IEEE ICNP*.

[38] Mario Hock, Felix Neumeister, Martina Zitterbart, and Roland Bless. 2017. TCP LoLa: Congestion Control for Low Latencies and High Throughput. In *IEEE LCN*.

[39] Chi-Yao Hong, Matthew Caesar, and P. Brighten Godfrey. 2012. Finishing Flows Quickly with Preemptive Scheduling. In *ACM SIGCOMM*.

[40] Tobias Hoßfeld, Lea Skorin-Kapov, Poul E. Heegaard, and Martín Varela. 2018. A new QoE fairness index for QoE management. *Springer Quality and User Experience* 3, 4 (2018).

[41] Toke Høiland-Jørgensen, Paul McKenney, Dave Täht, Jim Gettys, and Eric Dumazet. 2018. The Flow Queue CoDel Packet Scheduler and Active Queue Management Algorithm. RFC 8290. (2018).

[42] Raj Jain, Dah Ming Chiu, and Hawe WR. 1998. A Quantitative Measure Of Fairness And Discrimination For Resource Allocation In Shared Computer Systems. *Eastern Research Laboratory, Digital Equipment Corporation, Hudson, MA* cs.NI/9809099 (1998).

[43] Nathan Jay, Noga Rotman, Brighten Godfrey, Michael Schapira, and Aviv Tamar. 2019. A Deep Reinforcement Learning Perspective on Internet Congestion Control. In *ICML*.

[44] Junchen Jiang, Vyas Sekar, and Hui Zhang. 2012. Improving Fairness, Efficiency, and Stability in HTTP-based Adaptive Video Streaming with FESTIVE. In *ACM CoNEXT*.

[45] Kazumi Kaneko, Tomoki Fujikawa, Zhou Su, and Jiro Katto. 2007. TCP-Fusion: A Hybrid Congestion Control Algorithm for High-speed Networks. In *PFLDnet*.

[46] Dina Katabi, Mark Handley, and Charlie Rohrs. 2002. Congestion Control for High Bandwidth-Delay Product Networks. *ACM CCR* 32, 4 (2002).

[47] Elie F. Kfoury, Jose Gomez, Jorge Crichigno, and Elias Bou-Harb. 2020. An Emulation-based Evaluation of TCP BBRv2 Alpha for Wired Broadband. *Elsevier Computer Communications* 161 (2020).

[48] R. King, R. Baraniuk, and R. Riedi. 2005. TCP-Africa: an adaptive and fair rapid increase rule for scalable TCP. In *IEEE INFOCOM*.

[49] Ike Kunze, Jan Rüth, and Oliver Hohlfeld. 2020. Congestion Control in the Wild – Investigating Content Provider Fairness. *IEEE Transactions on Network and Service Management* 17, 2 (2020).

[50] Douglas J. Leith and Robert N. Shorten. 2004. H-TCP: TCP for high-speed and long-distance networks. In *PFLDnet*.

[51] Douglas J. Leith, Robert N. Shorten, and G. McCullagh. 2007. Experimental evaluation of Cubic-TCP. In *PFLDnet*.

[52] Yuliang Li, Rui Miao, Hongqiang Harry Liu, Yan Zhuang, Fei Feng, Lingbo Tang, Zheng Cao, Ming Zhang, Frank Kelly, Mohammad Alizadeh, and Minlan Yu. 2019. HPCC: High Precision Congestion Control. In *ACM SIGCOMM*.

[53] Yee-Ting Li, Douglas J. Leith, and Robert N. Shorten. 2007. Experimental Evaluation of TCP Protocols for High-Speed Networks. *IEEE/ACM ToN* 15, 5 (2007).

[54] Zhi Li, Anne Aaron, Ioannis Katsavounidis, Anush Moorthy, and Megha Manohara. 2016. Toward A Practical Perceptual Video Quality Metric. Netflix Technology Blog. (2016).

[55] Shao Liu, Tamer Başar, and R. Srikant. 2008. TCP-Illinois: A loss- and delay-based congestion control algorithm for high-speed networks. *Elsevier Performance Evaluation* 65, 6 (2008).

[56] Ahmed Mansy, Marwan Fayed, and Mostafa Ammar. 2015. Network-layer Fairness for Adaptive Video Streams. In *IFIP Networking*.

[57] Gustavo Marfia, Claudio Palazzi, Giovanni Pau, Mario Gerla, M. Y. Sanadidi, and Marco Roccetti. 2007. TCP Libra: Exploring RTT-Fairness for TCP. In *IFIP Networking*.

[58] Saverio Mascolo, Claudio Casetti, Mario Gerla, M. Y. Sanadidi, and Ren Wang. 2001. TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links. In *ACM MobiCom*.

[59] L. Massoulié and J Roberts. 2000. Bandwidth sharing and admission control for elastic traffic. *Springer Telecommunication Systems* 15 (2000).

[60] MAWI Working Group. 2020. Packet traces from WIDE backbone, samplepoint G, 2020/06/10. https://mawi.wide.ad.jp/mawi/samplepoint-G/2020/202006101400.html. (2020).

[61] Dimitrios Miras, Martin Bateman, and Saleem Bhatti. 2008. Fairness of High-Speed TCP Stacks. In *IEEE International Conference on Advanced Information Networking and Applications (AINA)*.

[62] Radhika Mittal, Vinh The Lam, Nandita Dukkipati, Emily Blem, Hassan Wassel, Monia Ghobadi, Amin Vahdat, Yaogong Wang, David Wetherall, and David Zats. 2015. TIMELY: RTT-Based Congestion Control for the Datacenter. In *ACM SIGCOMM*.

[63] Radhika Mittal, Alexander Shpiner, Aurojit Panda, Eitan Zahavi, Arvind Krishnamurthy, Sylvia Ratnasamy, and Scott Shenker. 2018. Revisiting Network Support for RDMA. In *ACM SIGCOMM*.

[64] Gautam Raj Moktan, Sebastian Siikavirta, Mikko Särelä, and Jukka Manner. 2012. Favoring the Short. In *IEEE INFOCOM Workshops*.

[65] Aisha Mushtaq, Radhika Mittal, James McCauley, Mohammad Alizadeh, Sylvia Ratnasamy, and Scott Shenker. 2019. Datacenter Congestion Control: Identifying what is essential and making it practical. *ACM CCR* 49, 3 (2019).

[66] S. Muthukrishnan, R. Rajaraman, A. Shaheen, and J.E. Gehrke. 1999. Online Scheduling to Minimize Average Stretch. In *IEEE Symposium on Foundations of Computer Science*.

[67] Aarti Nandagiri, Mohit P. Tahiliani, Vishal Misra, and K. K. Ramakrishnan. 2020. BBRv1 vs BBRv2: Examining Performance Differences through Experimental Evaluation. In *IEEE LANMAN*.

[68] Vikram Nathan, Vibhaalakshmi Sivaraman, Ravichandra Addanki, Mehrdad Khani, Prateesh Goyal, and Mohammad Alizadeh. 2019. End-to-End Transport for Video QoE Fairness. In *ACM SIGCOMM*.

[69] Adithya Abraham Philip, Ranysha Ware, Rukshani Athapathu, Justine Sherry, and Vyas Sekar. 2021. Revisiting TCP Congestion Control Throughput Models & Fairness Properties At Scale. In *ACM IMC*.

[70] Sudarsanan Rajasekaran, Manya Ghobadi, Gautam Kumar, and Aditya Akella. 2022. Congestion Control in Machine Learning Clusters. In *ACM HotNets*.

[71] Dominik Scholz, Benedikt Jaeger, Lukas Schwaighofer, Daniel Raumer, Fabien Geyer, and Georg Carle. 2018. Towards a Deeper Understanding of TCP BBR Congestion Control. In *IFIP Networking*.

[72] Linus E. Schrage and Louis W. Miller. 1966. The Queue M/G/1 with the Shortest Remaining Processing Time Discipline. *INFORMS Operations Research* 14, 4 (1966).

[73] Yeong-Jun Song, Geon-Hwan Kim, Imtiaz Mahmud, Won-Kyeong Seo, and You-Ze Cho. 2021. Understanding of BBRv2: Evaluation and Comparison With BBRv1 Congestion Control Algorithm. *IEEE Access* 9 (2021).

[74] K.N. Srijith, Lillykutty Jacob, and A.L. Ananda. 2005. TCP Vegas-A: Improving the performance of TCP Vegas. *Elsevier Computer Communications* 28 (2005).

[75] K. Tan, J. Song, Q. Zhang, and M. Sridharan. 2006. A Compound TCP Approach for High-Speed and Long Distance Networks. In *IEEE INFOCOM*.

[76] Belma Turkovic, Fernando Kuipers, and Steve Uhlig. 2019. Interactions between Congestion Control Algorithms. In *IFIP TMA*.

[77] Ranysha Ware, Matthew K. Mukerjee, Srinivasan Seshan, and Justine Sherry. 2019. Beyond Jain's Fairness Index: Setting the Bar For The Deployment of Congestion Control Algorithms. In *ACM HotNets*.

[78] Christo Wilson, Hitesh Ballani, Thomas Karagiannis, and Ant Rowtron. 2011. Better Never than Late: Meeting Deadlines in Datacenter Networks. In *ACM SIGCOMM*.

[79] Keith Winstein and Hari Balakrishnan. 2013. TCP Ex Machina: Computer-Generated Congestion Control. In *ACM SIGCOMM*.

[80] Keith Winstein, Anirudh Sivaraman, and Hari Balakrishnan. 2013. Stochastic Forecasts Achieve High Throughput and Low Delay over Cellular Networks. In *USENIX NSDI*.

[81] Yaxiong Xie, Fan Yi, and Kyle Jamieson. 2020. PBE-CC: Congestion Control via Endpoint-Centric, Physical-Layer Bandwidth Measurements. In *ACM SIGCOMM*.

[82] Lisong Xu, K. Harfoush, and Injong Rhee. 2004. Binary Increase Congestion Control (BIC) for Fast Long-Distance Networks. In *IEEE INFOCOM*.

[83] Lin Xue, Suman Kumar, Cheng Cui, and Seung-Jong Park. 2014. A study of fairness among heterogeneous TCP variants over 10 Gbps high-speed optical networks. *Elsevier Optical Switching and Networking* 13 (2014).

[84] Yasir Zaki, Thomas Pötsch, Jay Chen, Lakshminarayanan Subramanian, and Carmelita Görg. 2015. Adaptive Congestion Control for Unpredictable Cellular Networks. In *ACM SIGCOMM*.