Accurately Parameterizing Internet Performance Testing for Realistic Evaluations

Adithya Abraham Philip

CMU-CS-25-138

August 2025

Computer Science Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Justine Sherry, Chair
Srinivasan Seshan
Theophilus A. Benson
Renata Teixeira, Netflix

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

This research was sponsored by: the National Science Foundation under award numbers 2212390 and 1850384; a VMWare Systems Research award; a Google Faculty Research Award; and a CMU Cylab Seed Grant. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

Keywords: computer networks, congestion control, internet measurement

Abstract

The performance of Internet services—be it file download completion times, video quality, or lag-free video conferencing—is heavily influenced by network parameters. These include the bottleneck bandwidth, network delays, and how fairly the bottleneck link is shared with other services. However, current techniques to evaluate service performance in emulated and simulated networks suffer from three major issues: (a) testing predominantly in settings representing the "edge" of the Internet, and not the core; (b) focus on evaluating Congestion Control Algorithms (CCAs), neglecting the impact of application-level controls like Adaptive-Bitrate (ABR) algorithms on network performance; (c) testing in settings that do not necessarily reflect the network conditions experienced by services with expansive CDNs. The goal of this thesis is to improve the state of the art in emulated testing for a more up-to-date evaluation of Internet service performance.

To highlight the need to perform Internet evaluations in settings representing congestion at the core of the Internet, we test CCAs with core Internet speeds and flow counts. We find that this dramatically alters fairness outcomes, and challenges long-standing assumptions about CCA behavior that were built on measurements performed at in settings representing the edge of the Internet, emphasizing the need to run Internet evaluations in more diverse settings.

We then challenge the implicit assumption that CCA evaluations alone are sufficient to predict the network behavior of services that use them. We perform this analysis through the lens of fairness, and build Prudentia, an Internet fairness watchdog, that measures how fairly two Internet services can share a bottleneck link. In addition to discovering extreme unfairness on the Internet today, we gain key insights into improving current testing methodology – (a) The most and least fair services both use variants of the same CCA, highlighting the need to test services in addition to CCAs; (b) network settings can drastically affect even service-level fairness outcomes, necessitating their careful selection.

Lastly, we infer the network conditions experienced by users of Netflix, a global video streaming provider, and contrast them with those used in typical Internet evaluations. We find that Netflix users experience shorter RTTs, greater maximum observed queuing delay, and greater ACK aggregation, all parameters that play an important role in determining CCA behavior. This highlights the need for more service operators to run similar analyses and share their respective perspectives of prevalent network conditions, so that the networking community can include these settings in the design and evaluation of Internet services.

Acknowledgments

I owe an immense debt of gratitude to my advisor, Justine Sherry, the reason I can sit here and write an acknowledgments section. I could not have asked for a better advisor; her patience and genuine care carried me through the roughest parts of my PhD. She has shaped—now and probably forever—how I approach research, writing, presentations, people, and life in general, and I wear that imprint with pride. Thank you, Justine!

Thank you, Srini, for your depth and breadth of knowledge and even-handedness, Vyas, for helping bring structure and clarity of thought to my more chaotic research tendencies, and Theo, for helping me shape my thesis into the best version it could be. Thank you, Reese, Renata, Grenville and TY, my collaborators at Netflix, who helped create one of the most enjoyable periods of my PhD. I was also lucky to work with excellent students throughout this journey: thank you, Ranysha, Rukshani, Cheko, Alexis, Zili, Aanand, Serena, and Kenny. You were a pleasure to work with, and I learned something different from each of you.

I would be remiss not to shout out the many folks who made my PhD and Pittsburgh so much more enjoyable; my roommates: Miguel, Ioannis, and Orestis; my brothers and sisters in arms: Hugo, Nirav, Anup, Maggie, Sagar, Aditi, Sanjith, Jovina, Ankush, and Deka (D-Dawg); the OGs: Don, Suhas, Bobby, Anish, Raut, Rama, Chigu and Pabba. And thank you, Chintu, for demonstrating that some bonds and factories stay strong through even the lossiest wireless networks.

I would also like to thank the professors who mentored my early research interests in my undergraduate years: thank you, Viraj Kumar, Ram Rustagi, Channa Bankapur and Nitin Pujari. And a big thank you to Vincent Breitmoser and Dominik Schürmann, my mentors at OpenKeychain, who rekindled my love for building things.

As this PhD draws to a close, it is fitting to remember how it started: thank you, Ranjita, for both being the reason I wanted to do a PhD (one could say I just wanted to be more like you), and for paving the way toward actually doing one.

And lastly, to my family, without whom completing this thesis would not have been possible. The journey was long and fraught, and you were my rock when the seas raged hard.

Contents

Chapter 1

Introduction

1.1 Motivation

Internet service operators are constantly improving their services to provide users with the best possible Quality-of-Experience (QoE). This includes improving network performance through changes to the Congestion Control Algorithm (CCA) used by the services, TCP-stack level parameter tuning and improvements such as RACK [34] and TLP [39], and application-layer improvements such as modification to the Adaptive Bitrate Algorithm (ABR) for video streaming services.

However, testing how these changes affect users is difficult. The Internet is constantly evolving, with changes in link bandwidths, usage patterns, increasing prevalence of newer link types such as cellular connections. Even the infrastructure used to serve users content is changing, with expansive user-proximal CDNs becoming increasingly common. Nevertheless, operators still need to be able to test their changes to understand if and by how much they improve QoE for users, both to know whether to deploy the changes and to understand how to iterate on these changes to further improve QoE. The goal of this thesis is to better enable Internet service operators and the research community to test emerging algorithms and understand their behavior and performance in the face of newly emerging Internet phenomena.

Network performance testing approaches lie on a spectrum, with A/B testing on one end, and emulated and simulated testbeds on the other, where interpretability and reproducibility must be traded off for realism. A/B tests, for example, provide the most realistic test results, as it allows performance to be measured on actual user devices on the networks used by those users. However, they provides little insight into *why* the change in performance occurred, or how close it is to the *best* possible performance the service could obtain. This is primarily due to the fact that A/B tests typically lack visibility into all the properties of the path, such as the available path capacity, jitter, or the queue size on the bottleneck link. Emulation and simulation in controlled testbeds, on the other

hand, provide much more visibility into the causes of the measured performance, and in root-causing issues that are discovered. However, most emulation and simulation based evaluations today are limited in scale, and do not necessarily reflect newer trends in how services are accessed on the Internet—including congestion at core Internet links, the testing of services and their complex network behavior in addition to CCAs, and the prevalence of expansive user-adjacent CDNs.

The goal of this thesis is to enhance emulated and simulated testbeds to enable diagnosable, debuggable insights in more relevant testing conditions. This work is not aimed at replacing A/B testing, but rather, complementing it—allowing service operators and researchers a deeper understanding of why services behave the way they do, in conditions that capture important emerging trends on the Internet.

Thesis Statement: ***Extending*** *emulation environments to capture* ***emerging trends in Internet evolution*** *enables* ***causal debugging and analysis*** *of congestion control algorithms and Internet services in more up-to-date settings, allowing us to build more robust CCAs and services for the Internet.*

Over the course of this thesis, we examine three key trends in Internet evolution that are currently unaccounted for in most Internet evaluation testbeds, that serve as our case-studies. We discuss the effects these trends have on popular assumptions about Internet performance, and how to account for them in future testbeds. Specifically, we examine:

- The comparatively recent discovery that there is persistent contention on peering links [37], which see magnitudes larger flow counts and path capacities than those used in typical testbeds.
- The dominance of services like video streaming and video conferencing, with their own complex application-level control loops, that change network behavior in ways that cannot be understood by evaluating the CCA alone.
- The rise of large services and CDNs using specialized user-proximal infrastructure that changes various parameters like the RTT ranges commonly used in testbeds, and the emergence of complex CCAs that are affected by network parameters not typically emulated in testbeds such as ACK aggregation.

1.2 Summary of Contributions

1.2.1 Evaluation at the Scale of the Core of the Internet

Does CCA behavior studied in residential link settings stay the same when there is congestion at the core of the Internet?

CCA testbeds have typically emulated link bandwidths with at most hundreds of Mbps, and with a few to tens of flows competing for bandwidth on a shared bottleneck link to

understand inter and intra-CCA interactions. This is due to the implicit assumption that congestion occurs primarily at the residential *edge* of the Internet, at the routers within people's homes or in the last few hops from ISPs to their users.

However, recent work has shown that congestion occurs even at peering links at the *core* of the Internet, where thousands of flows compete for bandwidth [37]. These peering links experience magnitudes greater bandwidths, often tens of Gbps, with thousands of flows simultaneously competing for that bandwidth. The question then is, do the beliefs we hold about CCA performance, based so far on evaluations in residential "edge" network conditions, continue to hold even at the scale of the core of the Internet?

Unfortunately, testing at core Internet scale isn't as simple as tweaking a few network properties in existing testbeds. At Gigabit bandwidths, state of the art simulated testbeds like ns3 [60] take days to run a single 10 minute experiment [74]. Similarly, state of the art emulated testbeds like mininet [22] and Mahimahi [113] fail to keep up with line rates as high as 10 Gbps. Luckily, all is not lost! We solve this problem by building our testbed on top of BESS [19], a highly performant software switch, that is capable of emulating Gbps bandwidths without performance degradation. Our testbed can emulate thousands of simultaneous flows at Gbps bandwidths. We use it to re-examine a number of fairness properties and CCA models believed to be true based on experiments in edge network conditions, and see if they still hold at scale.

These efforts were not in vain: we find that long-trusted CCA models must be applied with carefully considered caveats when there is congestion at core Internet links, and that CCA properties like throughput fairness can differ drastically at scale. These findings are especially important given that almost every single CCA and Internet service performance evaluation for the Wide Area Network (WAN) in the last decade has focused exclusively on the residential edge. As an example of the potentially devastating impact of this narrow focus—BBRv1, Google's new congestion control algorithm which has seen widespread adoption on the Internet [157, 106], seems perfectly fair to other BBRv1 flows at low flow counts and RTTs. However, at the scale of a core Internet link, fairness degrades to the extent that certain BBRv1 flows are almost completely starved, breaking the inherent promise of the Internet being a shared resource where multiple flows can seamlessly multiplex. This is in addition to bottleneck queues remaining almost completely filled for the duration of the competition between these flows, breaking the promise of low-latency that BBR broke onto the scene with.

In §3, we discuss these and more findings about how CCA behavior changes at scale, especially when it comes to inter and intra-CCA interactions, and make a case for including core-link congestion scenarios in standard CCA evaluations.

If we were to rely solely on A/B testing, we would never be able to tell if a BBRv1 flow was starving due to insufficient bandwidth on the path or due to a breakdown in competition dynamics at high flow counts, and as a result never discover this fatal flaw with the CCA. This is exemplified by the fact that BBRv1 was indeed deployed after being

thoroughly A/B tested. Without the queue occupancy visiblity provided by our emulated testbed, we would also not know what BBRv1 was failing to keep its promise of keeping queues low (which, as disucss in §3, is also the most likely explanation for its breakdown in fairness). This serves as a clear example of how emulated testbeds help us not just observe potentially damaging behavior, but help us understand *why* this behavior occurs in the first place.

1.2.2 Accounting for Application Stack Impact on Network Performance

Do complex Internet services result in emergent network behavior that is different from that of the underlying CCA?

Most CCA performance evaluations today focus almost exclusively on studying CCA behavior and properties with a bulk download workload, neglecting the wide variety of services the CCA may be actually deployed in. Once upon a time, this made sense—the majority of traffic on the Internet was simple bulk file transfers, and it was primarily the CCA that determined the manner and rate at which the data was transferred.

However, more than 60% of traffic on the modern Internet is sent by significantly more complex applications like video streaming and video conferencing. Such applications come with their own intricate application control loops to improve user Quality-of-Experience (QoE), such as Adaptive Bitrate Algorithms (ABRs) that reduce video quality in response to poor network conditions. Given how drastically such algorithms can change the quantity of data being transferred, it is quite possible that these play a significant role in determining network performance and behavior in addition to the CCA. However, modern CCA evaluations do not typically evaluate CCAs with these modern workloads, and instead with bulk file transfers.

To bridge this gap, one would need to be able test services in addition to CCAs. However, testing real deployed services in emulated testbeds is not trivial: it is typically infeasible to obtain proprietary server-side code for services to run in a fully contained environment, and creating emulated clients for popular services is a laborious and error-prone endeavor. Instead, we embrace a hybrid approach, with actual service clients running on commodity hardware, connecting to standard customer-facing instances of their respective servers like an actual user would. However, we route the traffic from and to these clients through BESS, which emulates our bottleneck link, and can therefore configure path properties like bottleneck path capacity and queue size, and Base RTT. This provides us with the best of both worlds—the fidelity of measuring actual services being consumed on commodity hardware in the same manner a real user would, while retaining the configurability and debugging capability of emulated testbeds. We use this testbed to build Prudentia, an Internet fairness watchdog that evaluates throughput fairness outcomes and QoE degradation when real world services, in addition to CCAs, compete over a shared

bottleneck link.

In §4, we discuss our results, and demonstrate the importance of testing deployed services in addition to CCAs—the most and least fair services we observe use variants of the same underlying CCA, showing that testing the CCA alone does not predict performance outcomes of the services that use those CCAs. Prudentia also finds that there is significant unfairness when a number of services compete on today's Internet, that network settings can have a large impact on service-level fairness outcomes, and that silent updates to CCAs and services on the Internet significantly changes their fairness properties. In a nutshell, this chapter shows how testing services in addition to CCAs enhances the realism of future testbeds, and provides insights into real-world performance that could not have been obtained by testing CCAs alone.

Fairness evaluations are where emulated testbeds shine, and where A/B testing has little to no chance of serving as a substitute. In an A/B test, it is next to impossible to know if the low throughput obtained by a service was due to competition with another service, or simply poor link conditions. If service operators were to rely on A/B tests alone, it can quickly lead to a race to the bottom, with various services deploying increasingly aggressive CCAs and application-layer algorithms as they find it provides an increase in performance. Howeve, in reality, they would actually be causing increased queuing and congestion when they compete with other services (and potentially even other instances of the same service!), leading to an overall reduction in QoE not just for one user, but other users of the same or different service on the shared bottleneck link.

1.2.3 Network Conditions Experienced by Users of a Large Internet Service

How do the network conditions experienced by users served by an expansive CDN differ from those used in typical emulated and simulated testbeds?

Emulated and simulated testbeds today evaluate a wide range of RTTs, path capacities, and maximum queuing delays, with each study often evaluating different ranges of these values. Newer CCAs like BBR are also affected by path properties that were previously considered to have little impact on CCA performance, like ACK aggregation, which is therefore not emulated in most testbeds, and whose extent and prevalence on the Internet is not well understood. This lack of a common benchmark for Internet evaluations stems from a combination of two factors: the Internet is wide and varied, with potentially as many unique network conditions as there are users, and we posses little to no understanding of the network conditions experienced by users when they actually consume a real Internet service.

These testbeds fail to keep up with a key trend on the modern Internet: large amounts of traffic today is served by a few large services and CDNs, that have built their own expansive infrastructure to serve their users more efficiently. This changes a number

of path property ranges commonly used in testbeds, such as Base RTT—which is likely significantly lower due to the edge-adjacent nature of these CDNs—potentially rendering obsolete decades of Internet performance evaluations and studies. However, the prevalence of these CDNs also presents a unique opportunity: data from a CDN could help us finally understand the network conditions most commonly experienced by real users. This would allow us to work towards a unified benchmark for Internet performance evaluations that focuses on the network conditions most likely to be experienced by real users of Internet services, allowing for more realistic and relevant testing and design of CCAs and services.

In §5, we leverage this opportunity by examining real user traffic from Netflix, a large video streaming on demand service with an expansive CDN, to infer the network conditions experienced by its users. We report properties such as the Base RTTs, peak observed queuing delay and ACK aggregation levels experienced by its users. We find that Netflix experiences significantly lower Base RTTs, higher peak queuing delay, and more ACK aggregation than used in most modern Internet evaluations. We hope that this finding encourages other large Internet service owners to perform similar analyses and release their own perspectives on what network conditions their users experience. In conjunction with measurements such as Internet speed tests, this sets on the path towards more realistic configuration of emulated and simulated testbeds, further bridging the gap in fidelity between emulated/simulated testbeds and A/B tests.

1.3 Thesis Outline

We begin by discussing important background and related work in §2, followed by a deep-dive into each of the major contributions of this thesis. We discuss the benefits and insights from evaluating CCAs at scale in §3. We then demonstrate the necessity to evaluate Internet services in addition to CCAs in §4, accompanied by a number of insights into the state of service-level throughput fairness on the Internet today. We then move on to examining the network conditions experienced by Netflix, a global video streaming service, in §5, with the hope of informing future Internet evaluations of the network conditions real users of a service experience. Finally, in §6, we discuss the overall implications of the work done in this thesis, the work that can be done to carry forward its torch, and a vision for the future of Internet performance evaluations as a whole.

Chapter 2

Background & Related work

> "It's like in the great stories, Mr. Frodo.
> The ones that really mattered."
>
> — Samwise Gamgee
> *The Lord of the Rings: The Two Towers*

To better contextualize the contributions in this work, it is important to understand how Internet performance evaluations are typically conducted (§2.1), the capabilities of the testbeds used to conduct these evaluations (§ 2.2), and what past measurement studies tell us about emerging Internet trends and parameterizing emulated and simulated testbeds (§ 2.3).

2.1 Internet Performance Evaluations

Internet performance evaluations typically test either the CCA or a service, and often with different performance metrics under consideration. CCAs are typically evaluated on their ability to maximize throughput or fully utilize available path capacity [170], keep queuing delays low (typically measured using median/average or tail latency) [170], minimize packet loss, and share path capacity fairly with other connections [78, 149, 146, 27]. Internet services on the other hand, tend to be evaluated with QoE metrics such as video quality and rebuffer rate [138, 90], and are almost never evaluated in the context of their inter-service interactions or fairness properties. However, both CCA and Internet service evaluations tend to be conducted in "residential" network conditions, as opposed to the scale at the core of the Internet, which is one the gaps we aim to bridge with our study of CCA performance with thousands of flows in § 3.

Some notable studies, such as that by Spang et al. on correct A/B testing practices for CCAs [137], evaluate the effect of CCA performance with real video streaming workloads to Netflix users. However, typical CCA studies evaluate the CCA in isolation from the

service it is to be used in, leaving unanswered questions about its performance when actually deployed in an application on the Internet. We demonstrate the drawbacks of this approach by testing the throughput fairness of inter-service interactions with Prudentia in §4.

A body of work aims to provably verify CCA performance outcomes in a variety of scenarios using analytical techniques [10, 2]. Similar to emulation and simulation-based testbeds, these approaches are limited by the realism of the network model they use. We expect that our work discussing the path properties experienced by Netflix users in §5 will aid this line of research too.

2.2 Emulation & Simulation Frameworks

Emulation: A number of emulation platforms such as Mininet‘[22] and Mahi-Mahi [113] exist to help quickly bootstrap emulated network testbeds. Another popular emulation platform is netem, which comes included with Linux. These testbeds typically support emulating a number of properties, from path capacities to packet loss rates, to adding artificial delays. However, both these approaches fail to scale to the Gigabit bandwidths and flow counts necessary in this thesis, which is why we used BESS [19], a software switch developed at Berkeley.

Simulation: While emulated testbeds provide more configurability, visibility and reproducibility than A/B tests, they still pale in comparison to the reproducibility and precise configurability provided by simulated networks. Network simulators like ns2, ns3 and htsim allow users to create complex network topologies, and parameterize them with an almost limitless range of popular network parameters.

Unfortunately, while network simulators provide the most reproducibility, it is typically difficult to run real-world applications on them. Even testing CCAs often requires a version of the CCA translated into simulation framework. Efforts have been made to run actual network stacks and binaries through capabilities such as Direct-Code-Execution (DCE) on ns3, but getting full-fledged GUI-based applications like a YouTube browser client running within these simulators remain an uphill task. Additionally, while theoretically possible to emulated multi-Gigabit links on such simulators, this can results in simple 10 minute simulations taking even *days* to complete depending on the processing power of the host machine. Making simulators performant for such tasks typically requires making workload-specific optimizations to simulator code.

2.3 Internet Measurement Studies

As the goal of this thesis is to better parameterize Internet testbeds to reflect emerging Internet trends, it is important to examine the sources of data that might help us track these trends.

A major source of guidelines for parameterizing emulated and similar testbeds comes

from Internet measurement studies such as speed tests. Notable sources of data for such studies include Ookla's Speed Test [116], the speed test data set offered by Measurement-Lab (M-Lab) [99], and the data gathered by Cloudlfare's speed test [36]. These tests have data from users all over the world, conducted across a variety of devices and ISPs.

Another rich source of data, though more geographically restricted, comes from the Measuring America Broadband dataset [44]. This is a United States-government sponsored initiative that uses specialized wired hardware in volunteer's homes to conduct periodic "active probes" to measure various path properties like path capacity, base RTT and jitter. A mobile application counterpart exists to measure connectivity from user's smartphones too.

While providing one of the best views we have into the network conditions experienced by users on the Internet at scale, speed tests are not without their issues. They can be biased towards more tech-savvy users who know how to run them, and towards times when the Internet is particularly bad (as users typically test their speed when something goes wrong). They are also from the vantage point of the speed-testing-entity's server infrastructure, which need not reflect the content serving infrastructure of the services that those users consume. That said, they are still one of the largest datasets the broader community has to parameterize testbeds for more realistic evaluations. We aim to further bolster this dataset by investigating and reporting the range of path properties experienced by users of an actual Internet service, Netflix, in §5.

Chapter 3

Evaluating Congestion Control Algorithms at Scale

> "Yesterday," he said, "we was not believing in giants, was we? Today we is not believing in snozzcumbers. Just because we happen not to have actually seen something with our own two little winkles, we think it is not existing."
>
> — Roald Dahl, *The BFG*

3.1 Chapter Overview

Much of our knowledge regarding the performance and fairness of congestion control algorithms (CCAs) is based on models and observations that assume congestion occurs primarily at the "edge" of the Internet, close to users. These studies therefore assess CCAs in small-scale edge environments involving a few tens of flows and bandwidths of up to a few hundred megabits per second [29, 57]. For instance, the model by Mathis et al. [97] and Padhye et al. [117] predict the throughput of a NewReno flow as a function of packet loss and round-trip time (RTT), and the BBR model by Ware et al. [162] predicts the throughput of BBR when competing with other CCAs. Application developers can use such results to decide which CCA best suits the network conditions they experience or to debug performance issues. The original BBR paper [28] used measurements performed at edge-scale to show that BBR is perfectly fair to other BBR flows. However, recent work [37] indicates that congestion can occur at the core of the Internet, where thousands of flows share links with tens of Gigabits of bandwidth. This raises the question: Are the beliefs we hold about CCA performance, based on measurements done solely in edge-scale scenarios,

still true at the scale of the core of the Internet?[1]

As discussed in §1, existing emulated and simulated testbeds are either simply not capable of emulating Gigabit bandwidths and thousands of flows, or take prohibitively wrong [74] to run even short simulations. We overcome with roadblock by building a testbed with BESS [19], a highly performant software switch, which serves as our Gbps-capable bottleneck link. We use this testbed to re-evaluate a number of beliefs based on edge-scale evaluations. Specifically, we ask:

- *Throughput Model:* The commonly accepted Mathis analytical model [97] for TCP throughout prediction says that the throughput depends only on the RTT and loss. Does this model accurately predict TCP NewReno's throughput at scale?
- *Intra-CCA fairness:* NewReno, Cubic, and BBR have shown to be fair at lower flow counts when all the flows have the same CCA and RTT [35, 57, 61, 103]. Does this continue to hold at scale?
- *Inter-CCA fairness:* Does the Inter-CCA unfairness observed in the home link setting, where Cubic takes up to 80% of link bandwidth when competing with NewReno [57], or BBR starves competing NewReno and Cubic flows [162, 176, 132, 61], continue to hold at scale?

And indeed, we find that some edge-derived expectations do not hold at scale:

- The Mathis model [97] for NewReno throughput relies on a parameter p which is commonly interpreted as the network loss rate [131, 129]. While using loss rate for p works well in edge settings, we find that using packet loss rate for p at scale results in more than 45% error in estimating throughput. Instead, operators should use direct measurements of the congestion window halving rate for throughput estimates at this scale.
- BBR surprisingly becomes unfair at scale even when competing with solely other BBR flows at the same RTT, with a Jain's Fairness Index (JFI) as low as 0.4. This is in contrast to the fairness observed by past research in the edge setting or at low flow counts, where the JFI is typically 0.99 [132, 176, 61].

On the other hand, our findings validate at scale prior claims about CCAs which were derived from analyses evaluating the edge:

- A single BBR flow takes up 40% of link capacity even when competing with thousands of NewReno or Cubic flows at scale. Prior work had only measured this phenomenon at up to 16 competing flows [162, 132, 160], and our measurements illustrate that this phenomenon persists even at scale. This confirms the prediction from the model by Ware et al. [162].
- The intra-CCA fairness of NewReno and Cubic and the inter-CCA unfairness of Cubic

[1]This research has been presented at the Internet Measurement Conference, 2021[122], and content including text and images in this chapter may originate verbatim from the published paper.

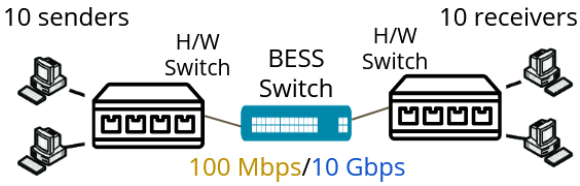


Figure 3.1: Testbed topology for emulated settings.

competing with NewReno, continue to hold at scale. The extreme inter-CCA unfairness when multiple BBR flows compete with multiple Cubic or NewReno flows also persists at scale.

Overall, our results emphasize the need to include core-scale evaluations in CCA performance tests. For example, applying the Mathis model over the Internet precisely will require end-host TCP instrumentation to obtain the congestion window values as one cannot rely on just measured packet loss. BBR's unexpectedly high unfairness when competing with just other BBR flows at scale further highlights the importance of explicitly including evaluations with thousands of flows and Gbps bandwidths as part of future CCA design and evaluation roadmaps. These additions will help emulated and simulated testbeds come closer towards achieving the fidelity that A/B tests provide, with the additional benefit of providing the visibility and debuggability that A/B tests cannot.

3.2 Related Work & Motivation

We begin this section with an introduction to CCAs, followed by past work on throughput models and fairness. Lastly, we discuss congestion at the core of the Internet and CCA results in data centers and high bandwidth settings.

CCA Background: Today there are many CCAs on the Internet, including NewReno [56], Cubic [129], Vegas [25], Copa [11], and BBRv1 [28] (hereafter referred to as 'BBR') as well as BBRv2 [16] (which remains a work in progress). Developers and network administrators evaluate CCAs for many important properties, including (1) *throughput*, or the rate at which a connection transfers data [97, 117, 57] and (2) *fairness*, or how equitably multiple connections share throughput when competing over a bottleneck link [35, 129, 103].

Throughput Models: To help us understand how well a CCA performs in a given network setting, analytical models predict the throughput of a connection as a function of key network properties (e.g., loss, delay). For example, the NewReno models by Mathis et al., [97] and Padhye et al., [117] predict the throughput of a NewReno flow given the network RTT and loss rate. Researchers have derived other models with similar goals for Cubic [57] and BBR [162]. In this chapter, we revisit the simpler model for NewReno throughput by Mathis et al. [97] and investigate the fairness implications of the BBR model by Ware et al. [162].

Fairness: Fairness determines how deployable a CCA is. Say, for example, that Cubic flows completely starve NewReno flows when competing for bandwidth over a shared link. This would result in Netflix streams (which use NewReno) seeing degraded performance every time they share a bottleneck link with large downloads using Cubic. Fairness is typically evaluated in two settings: (1) Intra-CCA fairness, where all competing flows have the same CCA; and (2) Inter-CCA fairness, where the competing flows have different CCAs.

Past research has found that in the wide-area context, NewReno, Cubic, and BBR all exhibit high intra-CCA fairness when all flows have the same RTT, with most flows getting the same throughput [35, 57, 28, 61]. There is also work on intra-CCA fairness when flows have different RTTs [93, 57, 103, 78]. In this chapter, as a simpler starting point, we specifically evaluate the same-RTT setting.

In the inter-CCA setting, prior work shows that Cubic flows compete unfairly with NewReno, with Cubic obtaining up to 80% of total bandwidth [57]. Past research also finds that BBR competes unfairly with both Cubic and NewReno, with a single BBR flow taking up 40% of link capacity irrespective of the number of competing Cubic and NewReno flows [162, 132]. Multiple BBR flows competing with an equal number of Cubic flows also result in the BBR flows obtaining 90% to 95% of link bandwidth with large buffers [130] and up to 99% with small buffers [61]. We re-evaluate all of these properties at scale.

Congestion in the core: Many prior CCA efforts implicitly assume that Internet con-

Table 3.1: Deriving the Mathis constant C using the packet loss rate results in different flow count-dependent constants in *CoreScale* vs *EdgeScale*, while using the CWND halving rate results in closer and more consistent values across settings and flow counts.

p	*EdgeScale*	*CoreScale* Flow Count		
		1000	3000	5000
Packet Loss	**1.78**	**3.95**	**3.64**	**3.24**
CWND Halving	1.47	1.36	1.36	1.34

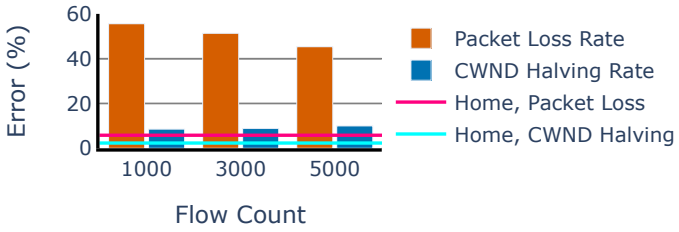


Figure 3.2: The median prediction error for the Mathis model in *CoreScale* is ≤ 10% using CWND halving rate, but 45% to 55% with packet loss rate. In *EdgeScale* both packet loss rate and CWND halving rate result in <10% error.

gestion occurs mostly at the network edge, evaluating only tens of flows at the scale of a hundred Mbps [28, 57, 97]. However, both older and more recent work [3, 37] show that there is persistent congestion on inter-provider links in the Internet core. This is significant in the light of analysis that the properties of CCAs can change as network parameters scale; e.g., the work of Appenzeller et al. [6] finds that when thousands, rather than tens, of NewReno flows compete over a "core" bottleneck link, they desynchronize, allowing the use of smaller router buffers compared to recommendations in the edge setting.

CCAs in data centers and high-bandwidth settings: While past research has investigated CCA properties in the data center setting [72, 82, 4, 33, 127], we are interested in the wide-area setting, which sees higher RTTs and has routers with larger buffers [6, 98]. There is also work on CCA fairness at Gbps bandwidths [61, 78, 103, 1], but they typically evaluate tens to a few hundred flows, not thousands. To the best of our knowledge, the Mathis model and fairness properties of CCAs when thousands of flows compete on Gbps links have not been rigorously studied in the wide-area setting.

3.3 Problem Scope and Methodology

In this section, we define the scope and methodology of our analysis, its relevance, and its limitations.

3.3.1 Problem Scope

Before we begin, we concretely define the two settings of interest for our study:

- *EdgeScale*: This represents the edge-link setting with a bottleneck bandwidth of 100

Mbps with 2 to 50 competing flows and a 3MB buffer.

- *CoreScale*: The “at scale" setting with a bottleneck bandwidth of 10 Gbps [55], 1000 to 5000 competing flows, and a 375MB buffer.

In both cases, a drop-tail queue is used at the bottleneck link, and the buffer size is approximately 1 BDP (bandwidth-delay product) based on the bandwidth of the bottleneck link and assuming a maximum RTT of 200ms. We choose this size based on the rule of thumb used to size router buffers [6]. It is the smallest buffer that would allow a single NewReno flow to saturate the link. While past work has shown that smaller buffers equal to a fraction of the BDP are sufficient to ensure upto 99% link utilization at scale [6, 17], recent work [98] has found that in practice ISPs still use extremely large buffers.

CCAs Analyzed: We focus our evaluation on three popular CCAs: NewReno, Cubic, and BBR. These CCAs are chosen based on both the depth of their research literature and their widespread usage on the Internet today [107, 111, 107]:

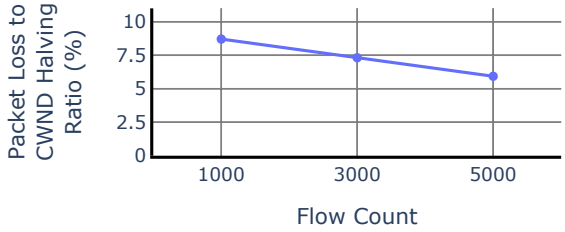
1. NewReno is a classic example of a loss-based CCA. It is widely used today, most notably by Netflix [107], which is believed to make up 13% of all traffic on the Internet [111].
2. Cubic is another loss-based CCA [57]. It is currently the default CCA on Linux and Windows Server and is the standard baseline almost every new CCA is compared with [93, 103, 162, 28].
3. BBR (specifically, BBRv1) is a comparatively new CCA proposed by Google [28]. However, it is used by YouTube [107], which accounts for 16% of all Internet traffic [111]. While a new version ‘BBRv2’ [16] exists, at the time of this study, it is was a work in progress. We, therefore, focus on the well-studied BBRv1 [61, 130, 162, 132].

3.3.2 Setup and Methodology

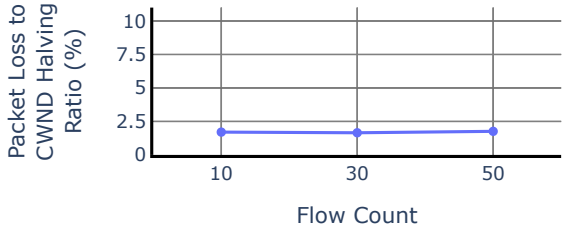
Studying TCP properties at scale is challenging; e.g., traditional packet-level simulators such as ns-3 [60] take several days for a simple Gbps-scale experiment [74], and past work on data-center networking that uses such simulators at scale typically run experiments modeling just a few seconds [84]. Approximations (e.g., flow or fluid model simulations [64]) may not accurately capture fine-grained dynamics. To achieve both fidelity (e.g., running actual TCP stacks) and scale, we use a simple testbed setup described below.

Our testbed uses a physical network with a dumbbell topology, with ten sender-receiver node pairs connected to a BESS software switch [19], as seen in Figure 4.1. We choose this topology as it is a common topology used to evaluate throughput models and fairness, and has been used to model a wide variety of scenarios [57, 103, 130, 97]. The bottleneck bandwidth for the experiments is varied between *EdgeScale* and *CoreScale* by changing the bandwidth and buffer size on the BESS software switch. We use a software switch as it allows greater control over the queue size and bottleneck bandwidth than the physical switches available to us, while still being closer to using physical network elements

than a simulator like ns-2 [153] or ns-3 [60]. The edge link bandwidths between the sender/receiver nodes and bottleneck link at the BESS switch is always 25 Gbps, which guarantees that congestion occurs at the BESS switch. The base RTT of flows is set using *netem* [147] to add the appropriate delay at the receiver, similar to past work [130, 103, 172]. We calculate the packet loss rate by logging packet drops at the bottleneck queue in the software switch, and use the Linux tool tcpprobe [151] to measure the congestion window halving rate to validate the Mathis throughput model. The testbed was hosted on CloudLab [40].

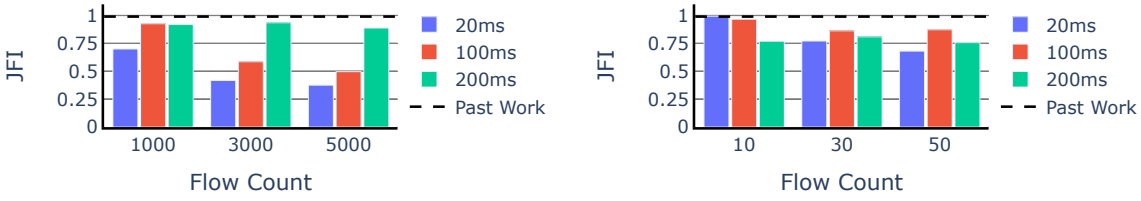


(a) *CoreScale*



(b) *EdgeScale*

Figure 3.3: The ratio between packet losses and congestion events (i.e., CWND halvings) changes between *CoreScale* and *EdgeScale*, and across across different flow counts within *CoreScale*.



(a) *CoreScale* **(b)** *EdgeScale*

Figure 3.4: BBR shows intra-CCA unfairness in *CoreScale*, with JFIs as low as 0.4. Milder unfairness can also be seen beyond 10 flows in *EdgeScale*, with JFIs as low as 0.7.

All TCP flows are distributed equally across each of the sender-receiver pairs and send infinite data, as common in past experiments [57, 103, 172, 61, 130]. The flows run for a maximum duration of 3 hours, significantly longer than past studies [162, 130, 61, 57], or until the metric being evaluated changes by less than 1% over 20 minutes. When an experiment starts, each flow waits a random period of time between 0 and 2 minutes before

it establishes a connection with the receiver, and the throughput obtained by all flows in the first 5 minutes of the experiment is ignored.

Limitations: As observed by many others, capturing the dynamics of Internet links with high fidelity — including random loss, arrival, and departures of new flows, application-level sending behaviors, etc — is perhaps impossible to achieve perfectly [170, 12, 167]. Furthermore, understanding the behaviors of CCAs can be challenging in "real" settings where many uncontrolled variables combine to influence CCA behavior. We instead opt to focus directly on *only two key variables*: the number of concurrent flows (which increases by two orders of magnitude between *EdgeScale* and *CoreScale*) and the link capacity (which increases similarly between *EdgeScale* and *CoreScale*). Therefore, when we say 'at scale', we refer to the setting where the bottleneck bandwidth is 10 Gbps and the flow count ranges from 1000 to 5000 flows. By controlling all other aspects of the experiment (all flows have the same, lengthy duration; there is no random loss; buffer sizes are approximately 1 BDP in both settings; all flows have the same RTT etc.) we can more easily inspect the impact of these two variables on CCA behavior.

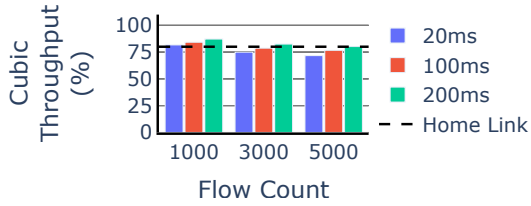


Figure 3.5: Cubic takes 70–80% of total throughput when competing with an equal number of NewReno flows in *CoreScale*.

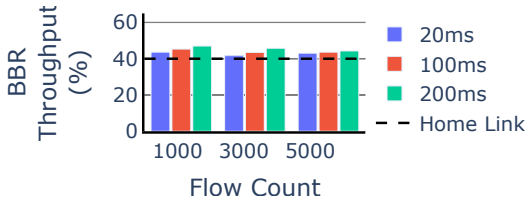


Figure 3.6: 1 BBR flow takes 40% of total throughput when competing with thousands of NewReno flows in *CoreScale*.

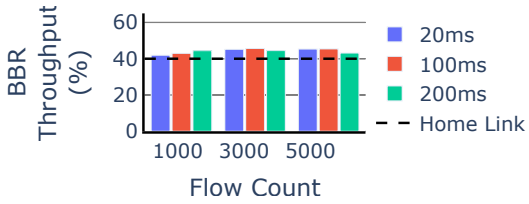


Figure 3.7: 1 BBR flow takes 40% of total throughput when competing with thousands of Cubic flows in *CoreScale*.

3.4 Revisiting the Mathis Throughput Model

Background: The Mathis model [97] predicts the throughput of a NewReno flow as a function of loss (p) and round-trip time (RTT). It depends on two constants: C, which may be different for different CCAs, and MSS (maximum segment size), which in our case is

fixed to 1448 bytes. The Mathis model equation can be expressed as:

$$Throughput = \frac{MSS * C}{RTT * \sqrt{p}} \tag{3.1}$$

The original paper by Mathis et al. [97] states that p refers to the congestion event rate. This can be interpreted in one of two ways: (a) the congestion window (CWND) halving rate or (b) the packet loss rate. While the original paper states that the CWND halving rate should be used for TCP with selective ACKs, subsequent research has often applied the packet loss rate instead [131, 129]. We, therefore, evaluate the Mathis equation with both the packet loss rate and the CWND halving rate.

The original paper derives a constant $C = 0.94$ for NewReno with delayed and selective ACKs [97]. The paper also demonstrates how to derive C empirically for varying NewReno configurations. For our modern NewReno [114, 95] stack we derive C empirically following the methodology described by Mathis: we calculate the C which minimizes the least squared prediction error of the Mathis equation at a given flow count and setting. For the following results, all flows run NewReno and have a 20ms RTT.

Observation 1: *Deriving C using packet loss rate results in flow-count dependent values and different values in CoreScale vs. EdgeScale. Using CWND halving rate produces consistent C values across both settings and flow counts. (Table 3.1)*

Table 3.1 shows the empirically derived "best-fit" constant C for NewReno in a few example settings. We see two main observations here. First, when using the packet loss rate the C value is quite different between *EdgeScale* and *CoreScale* and also changes between different flow counts in *CoreScale*. This violates the Mathis model which states that C depends only on the CCA being used, and should not change with the number of competing flows or bottleneck bandwidth. However, using the CWND halving rate produces a more consistent constant that changes only slightly between the *EdgeScale* and *CoreScale*, and does not change significantly between flow counts within *CoreScale*.

Observation 2: *Using the CWND halving rate results in accurate predictions ($\leq$ 10% median error) in CoreScale; using packet loss rate results in 45%-55% median error. In EdgeScale, however, both are accurate. (Fig 3.2)*

Fig 3.2 shows the median Mathis prediction error at different flow counts in *CoreScale*, while the two horizontal lines represent the median prediction error obtained in *EdgeScale*. These results show that the Mathis model does indeed hold at scale, as long as we use the

CWND halving rate for p. The 45%-55% median error when using the packet loss rate implies it cannot be used to accurately predict NewReno throughput at scale, even though the packet loss rate works well in *EdgeScale*. The error at scale is foreshadowed by the significantly different C values derived across settings and flow counts when using the packet loss rate.

Observation 3: *The ratio between packet losses and congestion events (i.e., CWND halvings) changes between CoreScale and EdgeScale, and across across different flow counts within CoreScale. (Fig 3.3)*

While investigating why the packet loss rate results in different constants in *EdgeScale* vs *CoreScale*, we discovered the ratio of packet loss rate to CWND halving rate is different in the two settings. As seen in Fig 3.3, in *EdgeScale*, the ratio of packet losses to CWND halvings is approximately 1.7 regardless of the number of concurrent flows. But in *CoreScale* the ratio varies between 6 and 9 and depends on the flow count. This explains why using packet loss rate results in different constants between *EdgeScale* and *CoreScale*, and different constants within *CoreScale* at different flow counts. While the idea that packet loss rate diverges from CWND halving rate is not new [46, 97, 117], we believe the drastic increase in divergence as we move from *EdgeScale* to *CoreScale* is a new finding.

Since the ratio is stable for *EdgeScale*, there is no reason to doubt past research that uses packet loss rate for p when evaluating links with tens of flows and only tens or hundreds of Mbps [131, 129]. *However, our results show one should not use the packet loss rate for estimating throughput over the Internet core.*

We hypothesize that the reason for different packet loss rate to CWND halving rate ratios is that losses are burstier at scale, causing multiple losses in the same burst or RTT which result in only one congestion window halving. We corroborate this hypothesis by measuring the burstiness of losses at the queue using the Goh-Barabasi burstiness score [51] which ranges from -1 to 1, where a higher score means the drops are burstier. We obtain median values close to 0.2 in *EdgeScale* and closer to 0.35 in *CoreScale*, implying that losses are indeed burstier at scale (Figure not shown).

Implications: Overall, we find that the Mathis model for throughput still holds in *CoreScale*, if C is calculated using CWND halving rate and not the more commonly used packet loss rate for the variable p. Unfortunately, this makes applying the Mathis model in practice more challenging, as obtaining the CWND halving rate requires end-host state reconstruction, where packet loss rate can be measured more easily via network-measurable loss. Furthermore, our findings also change our expectations regarding NewReno's performance with respect to loss: a flow on a congested core link can tolerate four times the

packet loss rate of a flow on a congested home link, and still obtain the same bandwidth because the CWND halving rate is the same.

3.5 Revisiting Fairness

In this section, we measure how fairly competing flows share bandwidth in our *CoreScale* setting.

3.5.1 Intra-CCA Fairness

Background: The classic metric used for measuring fairness is Jain's Fairness Index (JFI) [69], which ranges from 0 to 1 with a higher value indicating greater fairness. Past research in the edge setting has found Cubic, NewReno, and BBR to be intra-CCA fair – i.e., fair when competing only with other flows of the same CCA and RTT – with a JFI of 0.9 or more [35, 57, 28, 130, 176].

Observation 4: *NewReno & Cubic continue to show high intra-CCA fairness in CoreScale with a JFI > 0.99, as expected from past research. (Figure not shown)*

Both theoretical [35] and empirical studies [103, 57] have shown that when NewReno flows compete with other NewReno flows, or Cubic flows compete with other Cubic flows, throughput is shared almost equally when all flows have the same RTT. Our experiments confirm this in the *CoreScale* setting: NewReno and Cubic show high fairness with a JFI > 0.99.

Observation 5: *BBR surprisingly shows intra-CCA unfairness in CoreScale, with JFIs as low as 0.4, which is not expected from past research. Milder unfairness also occurs when more than 10 flows compete in EdgeScale, with JFI's as low as 0.7. (Fig 3.4)*

Fig 3.4 shows the JFI for BBR flows with the same RTT when they compete amongst themselves at different flow counts. It also shows the JFI based on results from past work (0.99) which finds BBR to be intra-CCA fair when all flows have the same RTT [28, 176, 132, 61].

We see that at scale BBR surprisingly becomes unfair at 20ms and 100ms RTTs, with the JFI going as low as 0.4. We investigate further and discover that BBR shows signs of unfairness even in *EdgeScale*, but at relatively higher flow counts (greater than 10) not examined by past research. This unfairness is exacerbated at scale.

Cardwell et al. [28] argue that BBR flows, share bandwidth fairly amongst each other at lower flow counts due to flow synchronization. While we have not verified it, we

hypothesize that the unfairness in *CoreScale* might be due to BBR flows desynchronizing at scale, similar to NewReno [6].

(a) NewReno **(b)** Cubic

Figure 3.8: BBR takes 99.9% of total throughput when competing with an equal number of NewReno or Cubic flows.

Implications: Prior work showed that BBR is unfair when competing with other CCAs (e.g. Cubic, NewReno) – however, it was assumed that if the entire Internet adopted BBR users could expect fair outcomes. Our *CoreScale* experiments show that this is not the case when thousands of flows compete in wide-area like settings; this emphasizes the need for CCA testing and evaluation at scale to understand whether a new algorithm is acceptable for deployment.

3.5.2 Inter-CCA Fairness

In this section, we evaluate how flows from different CCAs with the same RTT compete with each other.

Background: Past research in the edge link setting found that Cubic competes unfairly with NewReno, taking up to 80% of total throughput [57, 103] and that BBR is unfair to both Cubic and NewReno [162, 130, 61]. We revisit these properties at scale.

For the following results, we measure the aggregate throughput obtained by the flows of one CCA as a fraction of the total throughput obtained by all flows.

Observation 6: *A single BBR flow takes 40% of total throughput when competing with thousands of NewReno or Cubic flows in CoreScale, as predicted by past research in the edge setting. (Figs 3.6, 3.7)*

The BBR model by Ware et al. [162] shows that a single BBR flow could take 40% of total throughput irrespective of the number of competing NewReno or Cubic flows. We show that this result holds at scale and that a single BBR flow takes 40% of total throughput even when competing with thousands of NewReno or Cubic flows, as seen in Figs 5 and 6.

Observation 7: *BBR takes 99.9% of total throughput when competing with an equal number of NewReno (or Cubic) flows in CoreScale, confirming past research in the edge setting. (Fig 3.8)*

Past research in the edge setting has shown that BBR can take up to 99% of total throughput when competing with an equal number of Cubic flows [130, 176, 61]. Our results show that this inter-CCA unfairness persists even in *CoreScale*, with BBR obtaining up to 99.9% of total throughput when competing with an equal number of Cubic or NewReno flows, as seen in Fig 3.8.

Observation 8: *Cubic takes 70% to 80% of total throughput when competing with an equal number of NewReno flows in CoreScale, confirming the unfairness results of past research in the edge setting. (Fig 3.5)*

Past research [57, 103] expects Cubic flows to get around 80% of total throughput when competing with an equal number of NewReno flows in the edge setting. Our experiments show this holds true even in *CoreScale*, as seen in Figure 3.5.

Implications: Our results confirm that the inter-CCA unfairness displayed by Cubic to NewReno and BBR to both Cubic and NewReno persist at higher flow counts at scale. In these settings, the disparities between flows can be even more extreme than at the edge setting – with a single BBR flow attaining 4 Gbps and 5000 competing flows Reno or Cubic flows obtaining just 1.2 Mbps each. While past work shows that a few 'bad player' flows can impact fairness between a small number of users sharing an edge link (e.g. roommates in a shared house, co-workers in an office), the fact that prior unfairness findings extend to *CoreScale* suggests severely unfair outcomes where a single sender can impact thousands of physical neighbors with whom he or she shares a large inter-domain link.

3.6 Chapter Summary

Conventional wisdom about congestion control adopted by application and systems designers has been evaluated in settings implicitly assuming congestion at the edge. When congestion occurs in the core, as shown by many measurements, it is not clear if these accepted norms about throughput and fairness still hold. We revisit these and find that the widely accepted Mathis model can be applied using either loss or congestion window halving in edge settings, but these metrics diverge at scale. Similarly, we find that when BBR competes with other BBR flows, it goes from being completely fair in the edge setting to completely unfair at scale. This emphasizes the need for Internet performance evaluation testbeds to include network conditions that reflect congestion at the core of the Internet for more representative and complete testing, and further reduce the gap between emulated or simulated testbeds and A/B tests.

Chapter 4

Prudentia: Evaluating Services In Addition to CCAs

> "The whole is something else than the sum of its parts."
>
> — Kurt Koffka, Gestalt psychologist

4.1 Chapter Overview

Many emulated and simulated testbeds evaluating CCA performance almost exclusively study CCA behavior and properties with a bulk file transfer workload, not accounting for the fact that in practice, these CCAs are likely to also be used by other types of services like video streaming on demand and real time video conferencing. Once upon a time, this was unlikely to have been a major source of strife—the majority of traffic on the Internet was simple bulk file transfers, and it was primarily the CCA that determined the manner and rate at which the data was transferred. However, more than 60% of traffic on the modern Internet is sent by significantly more complex applications like video streaming [124], which come with their own intricate application control loops to improve user Quality-of-Experience (QoE), such as Adaptive Bitrate Algorithms (ABRs) that reduce video quality in response to poor network conditions. As a result, it is quite possible that the network performance we believe a CCA provides might not hold when it is actually deployed in these services.

To fix this gap in realism in modern emulated testbeds, we build Prudentia[1] , an Internet Fairness Watchdog which is capable of testing real services in addition to CCAs. We focus on fairness, as it is one the key pillars of Internet performance: one of the Internet's core

[1]The research paper describing Prudentia and its results was accepted and presented at SIGCOMM 2024 [121], and contents of this chapter may include text and images from that publication verbatim.

promises is to multiplex *shared* resources but this promise fails if a user has to pause their YouTube video every time their roommate needs to attend an online meeting. Furthermore, the economic implications of unequal performance outcomes are troubling: the Internet has been lauded over the past several decades as an open playing field for new entrants, with any startup having the same access as established players have to customer 'eyeballs'. If the established players deploy aggressive services that shunt their competitors aside under contention, new services may be unjustly perceived as low-quality and fail in the economic marketplace.

However, most research around fairness continues to focus exclusively on CCA performance, neglecting the impact application-level stacks can have on network performance. In the rest of this chapter, we will discuss the learnings obtained from Prudentia, which evaluates pairs of real services as they are deployed, accessing them through their browser-based clients just like a real user would, while routing traffic through a bottleneck link configured to emulate different link conditions. We use Prudentia to ask an overarching, simple question: *Are there 'winners' and 'losers' when popular services compete for bandwidth on the Internet today?*

A wide range of design choices can impact a service's *contentiousness* (*i.e.* how much 'pressure' it puts on competing services) and its *sensitivity* (*i.e.*, how much a service suffers under competition)[2]. For example, BBR has been broadly attacked in the research community and even the popular press [162, 154] because it leads to 'unfair' bandwidth allocations in deep-buffered networks with long-lived bulk flows. However, as we will show in §4.4, YouTube, which uses BBR for its congestion control, is in reality one of the *least* contentious services that we tested. In short, choice of CCA fails to tell the whole story about congestion at the service level.

We believe that it is important for a *public and independent* watchdog that identifies winners and losers to exist. Unfortunately, industry lacks incentives to do such monitoring on their own; for example, their engineers are rewarded for making services perform faster but not for making them kinder to competitors' traffic. Prudentia runs continuously with live experimental results available online at `https://www.internetfairness.net`. Over the two years Prudentia has been running, we observed changes in service stacks which have both improved and degraded fairness outcomes. Using Prudentia, we evaluated the behavior of several classes of applications under competition: video on demand, file distribution, web browsing, real time video streaming, and iPerf (which provides us a baseline to compare application-level testing against CCA-only testing).

Among our findings, presented in §4.4:

- The file-distribution service Mega [100] is substantially more contentious than any other application we tested. In our moderately-constrained setting, services running alongside

[2]We borrow the terms *contentiousness* and *sensitivity* from performance modeling literature [94, 92]

Mega achieved on average only 63% of their Max-Min Fair [18] (MmF) share of link bandwidth; with some services achieving less than *20% of their MmF share.*

- As mentioned above, YouTube – despite using much maligned BBR [146] – is among the least contentious. In the highly-constrained setting, *most applications competing against YouTube achieved more than their MmF share (117% on average).*
- Typically, losing services achieved on average 72% of their fair share (84% median) when subjected to contention from other services. Even when each service competed against another instance of itself (*e.g.*, one OneDrive download versus one OneDrive download), services achieved only an average of 88% of their MmF share.

Throughput is just one of the many metrics applications care about. In §4.5, we demonstrate Prudentia's ability to serve as a foundation for even more fairness metrics by observing the effect of contention on network metrics like loss, latency and jitter, and QoE metrics such as webpage load time.

In §4.6, we discuss results from Prudentia that shed light on the various factors affecting fairness measurements. In §4.7, we use these insights to make recommendations about how service providers might test their application for undesirable fairness outcomes so that problematic applications or algorithms can be patched. We highlight the need to test end-to-end applications, and not just CCAs; we also identify the need to test with multiple experimental trials to identify highly variable services (whose performance instability may be a problem on its own). To further encourage fairness testing, Prudentia allows externally submitted services to be evaluated as a part of its testbed. Unfortunately, one of our primary findings is many unfair outcomes are anomalous: *other than exhaustive all-to-all-testing, we are unable to find an approach to service testing that would identify these negative interactions.* We discuss challenges towards testing applications for fairness further in §4.6. Lastly, we discuss related work in §4.8, and conclude with future directions for Prudentia in §4.10.

4.2 Goals and Metrics

Before digging into the mechanics of our measurement methodology (§4.3) we first take a moment to ground the goals and philosophy behind our study.

4.2.1 Goals and Non-Goals

Our goals for this study are as follows:

(1) Provide a live, independent watchdog to highlight 'winner' and 'loser' performance outcomes between competing highly popular services, so that operators can take action to remediate these problems: To the best of our knowledge, this is the *first study* to consider end-to-end network performance outcomes under contention at the service level. Prior studies primarily focus on a single aspect of a service's design, such as CCA [23] or ABR [142]. Consequently, operators are often surprised at the negative outcomes – which are only visible at a service level – that we have reported. We hope that our data helps operators remediate these performance problems.

(2) Illuminate, where we can, common features and design decisions that might lead to unfair outcomes: At the network level, we can observe some traits that we suspect are leading to unequal performance outcomes. For example, some services rely on multiple parallel TCP connections, which is well-known to lead to unequal throughput allocations and is also visible from the network. Another observation we detect at the network level is that some services use 'bursty' transmission patterns that can cause intermittent packet loss. We hope that we can identify problematic design patterns that operators might seek to avoid.

(3) Develop a methodology for testing the Internet for undesirable outcomes under contention that operates at a service level: As we mentioned above, many operators were surprised when we discovered poor performance outcomes involving their own services – especially because they already test some aspects of fairness, such as CCA fairness. We believe that the methodology we explore in this work will be useful to service operators who should continuously test fairness outcomes for their *end-to-end* services as deployed. To further this objective, we have open-sourced all of the code used to run Prudentia [126].

It is also important to clarify *non-goals* for this project.

We do not aim to provide a comprehensive study of services, nor of all network conditions on the Internet: Our study covers 12 popular Internet services including file-sharing sites, video streaming, real-time video chat, and web browsing; we explore these services in the context of two network environments. It takes 2 weeks for our testbed to iterate over all pairs of services in both network environments[3] Scaling further would

[3] 12 services translates to almost 80 pairs to test, with 10 trials each, in 2 network settings, with more than 12 minutes between each experiment, adding up to more than 19,000 minutes or 13 days.

require additional resources beyond those available at our non-profit institution. Nonetheless, our website – https://www.internetfairness.net – does accept submissions of new web services for us to test and we can swap out services under study as feasible.

We do not aim to perform root cause analysis for every negative interaction we discover, nor do we aim to solve fairness problems on the Internet: Ultimately, only the operators of Internet services have the insight into their own end-to-end stacks to fully diagnose the cause of undesirable performance under contention. We can identify *some* problems, which are observable directly in the network, but we cannot, *e.g.*, identify that a proprietary ABR state machine chooses to 'back off' under contention too eagerly when we do not have access to the ABR implementation itself.

We do not aim to determine that any service is 'good' or 'bad' in a moral sense: Most operators we have spoken to about unfair outcomes have been genuinely surprised. Our end-to-end testing methodology is new, and we don't expect operators to have performed similar tests themselves. Hence, it is our operating assumption that any unfairness we observe is simply the result of intractable complexity in analyzing the performance impacts of a complete service stack, and not of any ill will on an operator's part.

4.2.2 What We Measure

There are many ways that service interactions can result in 'winners' and 'losers' or 'unfair outcomes.' Although the primary focus of the research community [80, 170, 162], has been on throughput, services can also have problematic performance outcomes due to interference inflating latency, causing persistent loss, introducing jitter, *etc.*. We provide the most in-depth analysis of throughput, but explore additional quality-of-experience metrics in §4.5.

Measuring throughput fairness itself is highly debated as there are many competing definitions of fairness, *e.g.* equal-rate fairness [70], proportional fairness [75], RTT-fairness [146], and max-min fairness [24]. For better or worse, most Internet algorithms (including many TCP congestion controllers [175, 65] and fair queueing schemes [144]) are designed with max-min fairness (MmF) as their target. Hence, with regard to throughput, we measure how closely outcomes achieve their MmF share – e.g., if a service's MmF share is 40 Mbps and it achieves 30 Mbps under contention, we would say it achieved 75% of its MmF share.

This means that every experiment we run results in *two* numbers – the MmF share attained by each competing service. *When we measure an MmF share, we refer to the service whose throughput is being measured as the 'incumbent' and the competing flow as the 'contender.'* We do *not* use Jain's Fairness Index [70] because it collapses these outcomes into one statistic: it can tell us that the outcome is imbalanced, but it cannot tell us which service is the 'winner.' We do *not* use harm [161] because it focuses on defining a 'deployability threshold' for services, and we do not aim to determine whether or not

services should be considered deployable here, merely to quantify their behavior under contention.

4.2.3 Does an unfair outcome mean that the contender is too aggressive?

Observing an unfair outcome does not mean that the contending service is too aggressive.[4] In the performance contention literature [94, 92], a given performance measure under contention is modeled as a function of the *contentiousness* of the contender, and of the *sensitivity* of the incumbent.

Contentiousness captures the idea that contenders place some 'pressure' on competing services, *e.g.* by using more than their fair share of bandwidth, or by choosing to send bursty sequences of traffic likely to induce loss. *Sensitivity* captures the idea that incumbents have some natural tendency to 'back off' given the presence of other services, *e.g.* choosing (or choosing not to) reduce sending rates in response to loss, jitter, or an increase in latency.

Whether or not a service is 'contentious' or 'sensitive' is a somewhat subjective concept. Most of the literature that attempts to model contentiousness and sensitivity does so by modeling them as *functions* rather than scalar values that can be ranked [92]. When we refer to a service as contentious, we mean that most services in our experiments that compete with it will attain less than 100% of their MmF share. When we refer to a service as sensitive, we mean that when that service competes with other services, it will generally attain less than 100% MmF share itself.[5]

When we observe an unfair outcome, it could be that the contender is a relatively contentious service. It could also mean that the incumbent is relatively sensitive. And, further muddying the situation, sometimes we observe 'idiosyncratic' outcomes in which the contender does not appear to be *generally* contentious and the incumbent does not appear to be *generally* sensitive, but we nonetheless observe poor performance for the incumbent (§4.4).

4.3 Methodology

Having described our high-level aims above, we now present our measurement methodology including our testbed design for network emulation (§4.3.1), how we ensure appli-

[4] Betteridge's law of headlines states: 'Any headline that ends in a question mark can be answered by the word no.' [20]

[5] Although we often see that more sensitive services are less contentious (and that more contentious services are less sensitive) it is also possible for a service to be both contentious and sensitive, or uncontentious and insensitive. A service that backs off in the face of a contender, but behaves in such a way (perhaps it is bursty) to cause the contender to also slow down could be both sensitive and contentious. A service that uses a very small amount of bandwidth and does not cede any bandwidth under competition is insensitive, but since by default it also consumes very little bandwidth it is likely to also be uncontentious.

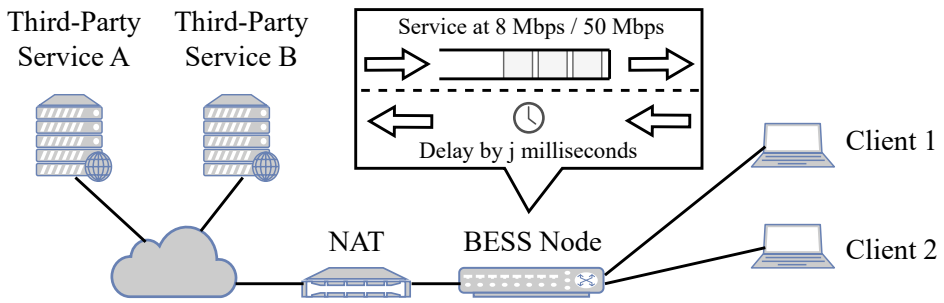


Figure 4.1: We use a dumbbell topology with two clients simultaneously receiving data from 2 services, with all traffic passing through the software switch BESS which acts as our controlled bottleneck link.

cation fidelity in our automated environment (§4.3.3), the services and period we tested them in (§4.3.2), and finally how we measure statistical significance(§4.3.4).

4.3.1 Network Emulation

The Prudentia testbed is illustrated at a high level in Figure 4.1. The simple idea behind this design is to have clients within the testbed access public Internet services over a controlled network connection which is likely to naturally be the bottleneck link. The upstream switch for our client is implemented using the BESS [19] software switch, which allows us to control the access link speed, queue size, and add delay to ingress and egress packets. BESS also allows us to measure queue occupancy and packet loss to enable deeper analysis of service behavior under competition. Other than this manipulation of the access link, all other traffic follows unmodified Internet paths from our institution to access live, deployed services.

To avoid variations due to network complexities, we use wired connections with no artificial loss or reordering; in the majority of our experiments all loss is due to queue overflows at the bottleneck link. Wireless settings introduce an additional and interesting setting to explore fairness outcomes, as the shared wireless channel becomes a new contended resource, however we consider it out of scope for this work.

Bandwidth Settings: We use BESS to emulate two network settings with 8 Mbps bottleneck bandwidth (which we refer to as a *highly-constrained)* and 50 Mbps (which we refer to as a *moderately-constrained)*. We choose these bandwidths because: (a) 50 Mbps is the median broadband speed experienced by more than half the countries in the world today [141] and (b) 8 Mbps represents the bottom 10% percentile of country-level median bandwidths [141]. 8 Mbps is also approximately the bandwidth that a 2K video would consume[6], allowing us to examine how contentious video services can be in a scenario where they can consume the entire link bandwidth.

[6]This number comes from the bitrates in Youtube's manifest files which we downloaded using [171].

Table 4.1: Services supported in the Prudentia testbed.

Service	Category	CCA	Max Xput	# Flows[+]
YouTube	Video	BBRv1.1 [107]	13Mbps	1
Netflix[δ]	Video	NewReno [125]	8Mbps	4
Vimeo	Video	BBR*	14Mbps	2
DropBox	File Transfer	BBRv1.0 [108]	∞	1
Google Drive	File Transfer	BBRv3 [65]	∞	1
OneDrive	File Transfer	Cubic [102]	45Mbps	1
Mega	File Transfer	BBR*	∞	5
Google Meet	RTC	GCC [31]	1.5Mbps	1
Microsoft Teams	RTC	Unknown	2.6Mbps	1
wikipedia.org	Web	BBRv1.0	∞	>5[β]
news.google.com	Web	BBRv3.0	∞	>20[β]
youtube.com	Web	BBRv3.0	∞	>10[β]
iPerf (BBR)	Baseline	BBRv1.0 (Linux 5.15)	∞	1
iPerf (Cubic)	Baseline	Cubic (Linux 5.15)	∞	1
iPerf (Reno)	Baseline	NewReno (Linux 5.15)	∞	1

* These CCAs were determined using a CCA classifier described in a related work [157].
+ The number of flows that are transferring service workload-related data (e.g. video chunks for video services) at the same time.
δ Netflix is run on Safari, as DRM prevents it from running at the highest quality on Google Chrome on MacOS [59].
β The number of flows used to load a webpage is variable and depends on the number of resources being loaded by the page and the number of distinct domains they are fetched from. We have listed the minimum number of flows we usually observe or these webpages.
Note: CCAs for YouTube, Netflix, Google Drive, Dropbox, and Wikipedia were confirmed with engineers at the respective companies.

While these are the primary bandwidths Prudentia uses to evaluate fairness, in §4.6 we run a one-off evaluation to examine how fairness evolves at other bandwidths. Prudentia's regular iterations over services (`http://www.internetfairness.net`) only include these two settings because including more settings would multiplicatively increase the time to cycle across all-pairs of services.

RTT Settings: We normalize round-trip times between services to 50ms; all services we tested had an RTT to/from the testbed of ≤ 50ms and we used the software switch to insert additional delay for all services to normalize to 50ms. We selected 50ms as the highest RTT we recorded for a service was 40ms and we can only increase, not reduce, the delay experienced by a service. For services that use multiple flows we normalize the RTT based on the first flow of that service.

Queue Sizing: Finally, we set the queue size of our Drop-tail FIFO bottleneck queue to approximately 4×BDP, based on input from large content providers who said that those

are the buffer sizes they see in practice, and past work which implies that queues are at least this big [47].[7] In §4.6 we briefly examine the effect an even larger buffer would have on fairness.

Background Noise: Although we fully control the client's access link, we do not control what happens over the Internet. Hence, it is impossible for us to prevent upstream bandwidth bottlenecks, throttling, or sources of loss. However, we do mitigate these effects using two techniques. First, to detect upstream throttling, we run all services 'solo' to detect their maximum transfer rate in the absence of contention; only one service is throttled either by its server or network upstream (OneDrive, which should have achieved higher throughput, see Table 4.1). Second, to mitigate the effects of upstream congestion caused by transient traffic, we run multiple experiments between every pair and repeat experiments every two weeks; we also discard any experiments with more than 0.05% packet loss external to our testbed. If we see experiments with high variability or a large number of 'outlier' results, our scheduler automatically re-queues the service pair for additional testing to achieve stronger statistical significance, up to a maximum of 30 trials.

4.3.2 Services & Period Under Test

Table 4.1 lists all of the services currently supported by the Prudentia testbed. These services can be broadly categorized as on-demand video services, file transfer services, real-time communication (RTC) services, web services, and baseline (iPerf) tests. We highlight throughput outcomes for on-demand video and file transfer services in §4.4. We focus on more application specific forms of performance interference (such as changes in frame rate or above the fold page load times) for RTC and Web traffic in our 'Beyond Throughput' discussion in §4.5. For each service, we list the CCA used by that service if known from references or direct contact with operators. For two services we were unable to obtain such ground truth information. Hence we used a CCA classification tool [148] which identified BBR as the CCA for Vimeo and Mega. We also confirm this by verifying the BBR bandwidth probe and RTT probe intervals in traces from our experiments with Vimeo and Mega. Video and RTC services have a maximum transmission rate depending upon their maximum bitrate encoding: 13 Mbps for YouTube, 14 Mbps for Vimeo, 8 Mbps for Netflix, 1.5 Mbps for Google Meet and 2.6 Mbps for Teams. One Drive is the only non-video service which otherwise had a throughput cap external to our testbed: downloads run on a 1 Gbps link were also able to achieve only an average throughput of 45 Mbps.

All file transfer services attempt to download the same 10 GB randomly generated file, and video streaming and RTC services play the reference Big Buck Bunny video [21].

Prudentia has been evaluating fairness amongst these services since 2022. Unless otherwise noted, the numbers reported in this work are from the latest set of experiments,

[7] A quirk of the BESS software is that it only allows queue sizes in powers of two, hence the queue is in reality set to the power of two nearest to 4×BDP.

run between June 2023 - September 2023, and the RTC service evaluation in January 2024.

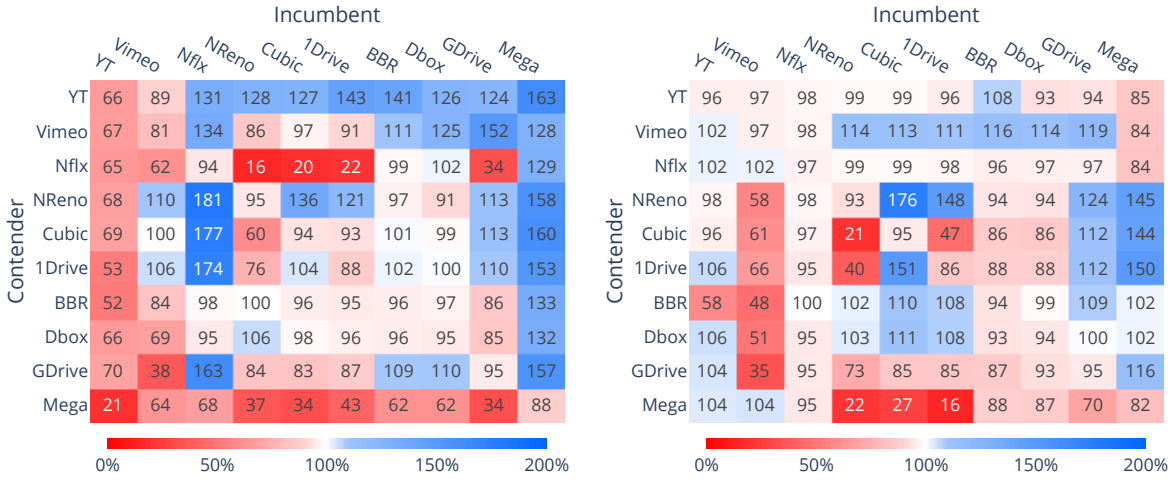
4.3.3 Application Fidelity

Automating end-to-end application behavior is challenging because seemingly simple concessions to automation, such as using command-line tools or running applications 'headless,' can result in different application behavior. We use Google Chrome [54] controlled by Selenium [133] rather than a command-line tool to make sure that service accesses result in the same sequence of TCP/QUIC connections as would be invoked by real client. Between experiments, we wipe all cookies and browser cache data to run all experiments in a consistent, repeatable state in which all application data must be fetched over the network.

Video playback was the most challenging class of services to automate. We were only able to generate realistic network traffic for video when using a full-fledged web-browser, on a server with a desktop-marketed GPU (we used Mac Mini Desktops), with a connected 4K monitor. The problem with other configurations, *e.g.*, headless configurations, is that video clients determine their bitrate selection not only on network connections, but also based on their perceived client rendering capacity. Because we wanted to measure only network effects, we needed a testbed which was not render-limited. For example, when we attempted to run video traffic without a real HDMI adapter – sending output instead to a virtual device `xbuf` – clients reduced their bitrate selection, percieving the device as unable to keep up with the highest (4K) video bitrate. Even with a real monitor, clients without GPUs or with GPUs that did not support native VP9 decoding [58] were unable to decode at a sufficient rate, once again triggering a lower bitrate request by the client. We provide these details because it is our understanding that video experiments in 'headless' modes using the above features are not uncommon but, from our experience, these automation tools are in reality a threat to validity of any experimental findings.

4.3.4 Statistical Significance

We run each experiment for a total duration of 10 minutes, and ignore the first and last two minutes of the experiments, as this gave us the most consistent results across trials. We run a minimum of 10 trials of each combination of contender and incumbent service, and then run more trials in sets of 10 up to a maximum of 30 until the 95% confidence interval of the median falls within +/- 0.5 Mbps in the highly-constrained setting and +/- 1.5 Mbps in the moderately-constrained setting. We find that almost all our experiments achieve these tight bounds, except for two services that display inherent instability in some fairness interactions. These are discussed in detail in §4.6. All our graphs show the inter-quartile range (difference between the 25th and 75th percentile measurements) as error bars. To limit the effect of temporally-localized performance issues, such as a service slowing down due to a data-center outage or external network performance degradation, we run the trials in a round-robin manner. A full run of one trial of every service competing



(a) Highly-constrained setting **(b)** Moderately-constrained setting

Figure 4.2: Median MmF share obtained by an incumbent service when competing with a given contender. Unless otherwise noted, all measured throughputs are within a 95% confidence interval of +/- 0.4 Mbps in the highly-constrained setting and +/- 1.5 Mbps in the moderately-constrained setting.

with every other service takes ~20 hours.

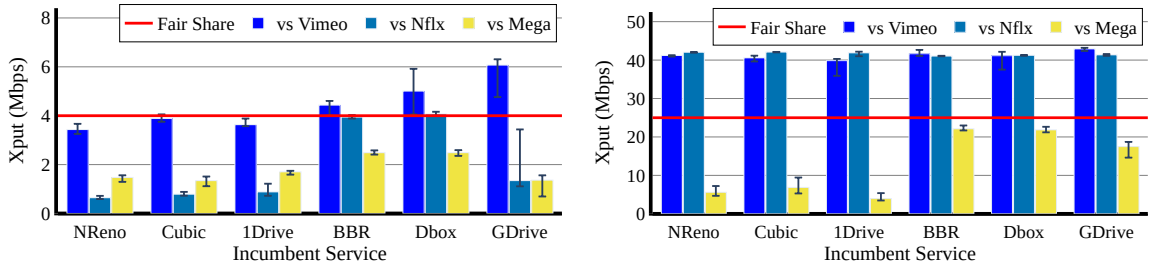
4.4 Throughput Under Contention

Having described our methodology (§4.3) we now explore the data from our testbed. In this section, we explore the traditional metrics of *throughput fairness* by looking at on-demand video and file transfer services. In §4.5, we explore other metrics which suffer under contention (such as latency, video resolution, and page load times) by inspecting our web and real time communication services.

Figure 4.2 plots a heatmap of the MmF share (that is, the fraction of the max-min fair allocation achieved by the incumbent service) for all-to-all experiments between video streaming and bulk download services in the 8 Mbps (highly-constrained) and 50 Mbps (moderately-constrained) settings. Numbers higher than 100 represent an outcome where the incumbent achieves *more* than its MmF allocation; numbers lower than 100 represent an outcome where the incumbent achieves less than its MmF allocation.

In the majority of experiments, the MmF allocation is simply 50% of the link capacity. However, video services in the 50 Mbps setting are application-limited by their maximum achievable bitrate and, in this setting, their MmF allocation is between 8 Mbps and 14 Mbps, and their contenders' allocation correspondingly higher, depending on the service as shown in Table 4.1. Each datapoint represents a median of at least ten trials, with additional experiments performed to ensure a 95% confidence interval of +/- 0.5 Mbps (in the highly-constrained setting) or +/- 1.5 Mbps (in the moderately-constrained setting).

To read this graph, it is easier to look at rows and columns than individual datapoints.



(a) Highly-constrained setting **(b)** Moderately-constrained setting

Figure 4.3: Mega, Netflix and Vimeo use up to 5, 4 and 2 concurrent flows respectively. In the highly-constrained setting, this causes Netflix and Mega to be unfair to other services. In the moderately-constrained setting, Netflix being application-limited prevents it from causing unfairness. Vimeo (using 2 BBR flows) does not cause unfairness in either setting, potentially due to the influence of its ABR algorithm.

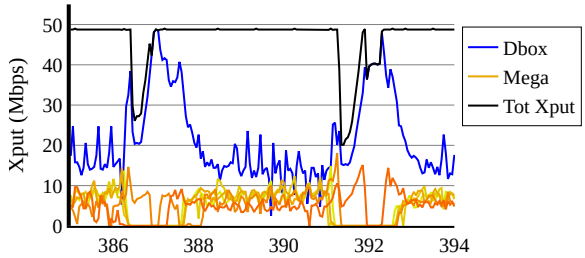


Figure 4.4: When Dropbox competes with Mega, it is able to ramp up quickly and utilize the extra bandwidth between Mega's bursts, allowing it to obtain 3–4× the throughput against Mega compared to traditional CCAs like NewReno or Cubic. The x-axis is the time in seconds since the start of the experiment.

Each row reflects the *contentiousness* of its respective service: how well incumbent applications performed when competing with the service labeled in that row as a contender. In the highly-constrained setting, for example, we can see that the YouTube row highlights all services (except YouTube itself) in blue with values ≥100, reflecting that most services perform well when competing against YouTube. On the other hand, each column reflects the *sensitivity* of the service. Looking at the YouTube column, we can see that it is entirely red: most of the time, YouTube performs poorly when it is competing against other services. We therefore consider YouTube as both a *generally sensitive* and *generally uncontentious* service.

Observation 9: *Unfair outcomes are common in bandwidth-contended environments. (Fig 4.2)*

Across our heatmaps in both the moderately-constrained setting and the highly-constrained setting, it is the *uncommon* case for both incumbent and contender to receive exactly 100% of their MmF share. In the highly-constrained setting setting, the median 'losing' service

achieved 69% of their MmF share. 73% of losing services achieved 90% or less than their MmF share, and 22% of losing services achieved 50% or less than their MmF share. In the moderately-constrained setting, the skew is less but still often unfair: the median 'losing' service achieved 86% of its MmF share. Although these numbers are not meant to be representative statistics for the Internet as a whole, they suggest that unfair outcomes are common on the Internet.

Observation 10: *The most and least contentious services we measured use variants of the same underlying CCA; CCA alone cannot account for the differing fairness outcomes.*

Since both Mega and YouTube use variants of BBR as their underlying CCA[8], one might expect them to display similar fairness behavior. However, we see exactly the opposite: Mega is one of the most contentious services we evaluate, while YouTube is one of the least contentious (Fig 4.2). This is best observed in the highly-constrained setting, where both YouTube and Mega are capable of fully utilizing the link. Services that compete against Mega obtain less than 50% of their fair share on average, while YouTube allows most competing services to get more than 120% of their fair share. Mega's contentiousness is most likely due to its use of multiple flows, while YouTube's sensitivity is most likely due to its ABR's desire for stability and its discrete bitrate ladder, both of which are application-level characteristics. These results justify our core argument that fairness testing for the Internet must encapsulate the entire application stack, both to capture the behaviour of potential CCA variants in deployment (e.g. Google Drive uses an updated version of BBR), and because analyzing CCAs alone would fail to predict the outcomes we observe for BBR-based or NewReno-based services.

Observation 11: *Concurrent TCP flows – known to have negative fairness consequences – are one cause of Mega's unfairness. Concurrent TCP flows are also used by other services but with less impact. (Fig 4.3)*

Mega uses a custom javascript framework to open up to 5 concurrent BBR flows to download files. For Mega, this can result in extreme disparities between the 'winner' (Mega) and 'loser' (any other incumbent). In the most extreme case, a competing One Drive download is able to obtain only 16% of its fair share in the moderately-constrained setting (Fig 4.2b).

Netflix and Vimeo also use up to 4 and 2 concurrent flows respectively. Note that the fact that different video services use different flow counts (YouTube (1), Vimeo (2) and Netflix (4)) indicates that while the browser typically controls the number of simultaneous flows a webpage can use, services can implement additional client-side controls to further

[8]We have confirmed with contacts at Google that YouTube continues to use an older version of BBRv1 (rather than BBRv3), which is what we believe to be true for Mega as well.

limit this. Fig 4.3 shows how these services using multiple flows impact services that only use one. In the moderately constrained setting, neither of them are contentious since they are both application-limited. In the highly-constrained setting, Netflix is more contentious due to its use of multiple flows but Vimeo is not. We hypothesize that Vimeo's ABR algorithm chooses a more conservative bitrate than Netflix in the highly-constrained setting, reducing its contentiousness.

Observation 12: *Application-level scheduling and request patterns can shape fairness outcomes. (Fig 4.4)*

Since Mega uses five BBR flows, one might expect its fairness properties to match that of five iPerf BBR flows. However, in separate experiments in the moderately-constrained setting, we find that the two behave very differently. Dropbox achieves only 33% of its MmF share against five BBR flows but achieves almost 90% of its fair share against Mega – suggesting that Mega is less contentious than BBR alone. However, NewReno and Cubic fare much better against five BBR flows (80-90% of fair share) as compared to Mega (22-27% of fair share) – suggesting that Mega is *more* contentious than BBR alone.

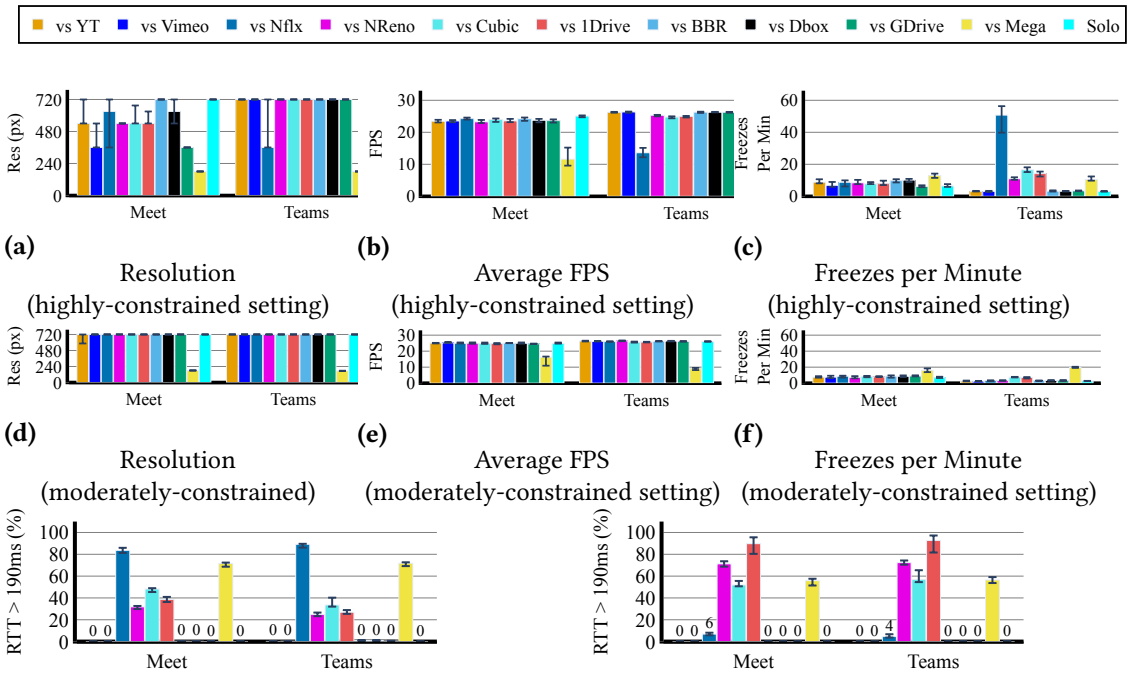
We believe this odd behavior is most likely due to Mega's "batching" behaviour; Mega downloads files in batches of five chunks, with each of its five flows downloading a separate chunk. If one flow finishes downloading a chunk early, Mega does not start downloading a new chunk right away; it waits for all of the flows in a batch to finish before starting another batch. This results in "bursty" traffic patterns, as shown in Fig 4.4. Dropbox (which uses BBR) is able to ramp up sufficiently in-between bursts (Fig 4.4) to achieve an almost fair outcome. In contrast, NewReno and Cubic are unable to ramp up significantly before Mega's next burst starts. It is also possible that Mega is running a slightly different version of BBR– one of our later observations is that being a new, still frequently patched CCA [155], even kernel updates can change BBR's fairness outcomes.

4.5 Beyond Throughput Fairness

While throughput fairness is the standard metric evaluated by most studies of network contention [80, 170, 162], there are other important performance metrics that can be impacted by cross-traffic contention. In this section, we investigate the impact on QoE metrics in real-time communication (RTC) services (§4.5.1), page load times in webpage browsing (§4.5.2), as well as metrics such as link utilization and loss for our throughput-intensive services (§4.5.3).

4.5.1 RTC Services

RTC services typically track a number of metrics that impact user perception. Here, we examine the impact that contention has on video resolution, frames per second, freezes



(g) High Delay Packets (%) (highly-constrained setting)

(h) High Delay Packets (%) (moderately-constrained setting)

Figure 4.5: The degradation, or lack thereof, in various metrics when Google Meet and Teams compete against other services. In the moderately-constrained setting, in most cases, both services perform well in metrics other than latency. Howver, in the highly-constrained setting, many competing services cause varying degrees of QoE degradation.

per minute, and high-delay packets – metrics which are often incorporated into higher order 'QoE' measures. We define these metrics in Table 4.2; we defer evaluation of higher order QoE metrics (*e.g.* VMAF [85], SSIM [159]) to future work.

Observation 13: *Differing trade-offs made by applications can lead to different perceived sensitivity at the user level (Fig.4.5)*

In Fig 4.5 we show the resolution, FPS, FPM, and fraction of high delay packets for both Google Meet and Microsoft Teams, under both the highly-constrained setting and the moderately-constrained setting. In the highly-constrained setting, Google Meet degrades in resolution more so than Teams and (not shown) correspondingly in bandwidth attained. However, Google Meet suffers less degradation in FPS compared to Teams. Also, while Google Meet tends to show a higher baseline of freezes per minute, it nevertheless suffers fewer freezes per minute than Teams when exposed to certain competitors such as Netflix.

Hence, from a video quality perspective, Meet can be seen as more sensitive than

Table 4.2: Quality metrics for real-time communication

Resolution	The resolution the video played at for the majority of the stream, represented by the height in pixels (e.g. 720p, 480p).
Average Frames Per Second (FPS)	Average number of frames rendered per second. A higher average FPS indicates smoother video [136].
Average Freezes Per Minute (FPM)	The number of times a frame "freezes" on the user's screen. Measured using the WebRTC definition of a freeze [136], which checks if the frame inter-arrival time exceeds $max(3 * \delta, \delta + 150ms)$, where δ is the average frame inter-arrival time.
Fraction High Delay Packets	Fraction of packets that experience greater than the ITU requirement of a 190ms RTT for RTC [66].

Teams – but from meeting a 'real time' bar for communication, Teams can end up being more sensitive to certain services due to the lower FPS and increased occurrence of freezes.

Observation 14: *Services using loss-based CCAs can cause as much as 92% of the packets to exceed ideal RTT requirements (Fig 4.5g,4.5h).*

We find that when competing against loss-based CCAs (and Mega), 40% to 90% of packets can experience high delay beyond the requirements defined in ITU publications [66]. This replicates a well-known and old finding – namely, that loss-based CCAs are problematic for real-time networking – but we find it worth calling out in an era in which many major providers seem to finally be shifting towards CCAs with lower queue occupancy demands [107, 52]. We observe that all but one of the BBR based services cause almost no latency anomalies for our RTC traffic. Nonetheless, our results with Mega reveal that the deployment of low queue occupancy CCAs (or at least, the deployment of BBR) is not a panacea for cross-traffic latency inflation: application layer decisions from Mega lead it to cause just as much latency inflation as services using buffer-filling algorithms.

Observation 15: *Layered and complex control loops in on-demand video and real time video streaming services make predicting or understanding contention challenging.*

Not shown in our figures, we also investigated the impact of RTC traffic on our throughput-intensive services. Surprising us, in the highly-constrained setting, Teams causes Vimeo to obtain a throughput of 2.5 Mbps, which is almost half of what it gets when competing against iPerf flows running NewReno or Cubic. We did not expect to see this result because

Teams is inherently bandwidth-limited to less than half of the bottleneck bandwidth link. Unfortunately, further investigation would likely require a better understanding of Team's rate selection and pacing, as well as Vimeo's ABR algorithm. Perhaps the root cause has something to do with pacing, rate selection, or buffer filling. With more components to analyze – and more of these components under proprietary domain – identifying the root cause of outcomes under contention is now more complex than ever.

4.5.2 Web Browsing

We now turn to another complex metric, page load time (PLT) for web sites. We measure PLT as the time it takes for 95% of a page's default visible region ("above-the-fold") to load for a user, based on Google's SpeedIndex technique [139]. The pages are loaded on a 4K display. In each trial between a webpage and a contender service, we first start the contender service, and after 30 seconds load the page in a new Google Chrome instance. We then repeat this page load 10 times, with a gap of 45 seconds between each webpage load. Each time the page is loaded it is through a new Google Chrome instance with its cache and cookies wiped. This is so we can better understand the impact competing traffic has on a fresh page load in a reproducible manner; we would expect cached pages to perform differently. Each trial is then repeated at least five times, providing a total of at least 50 data points per service-webpage pair.

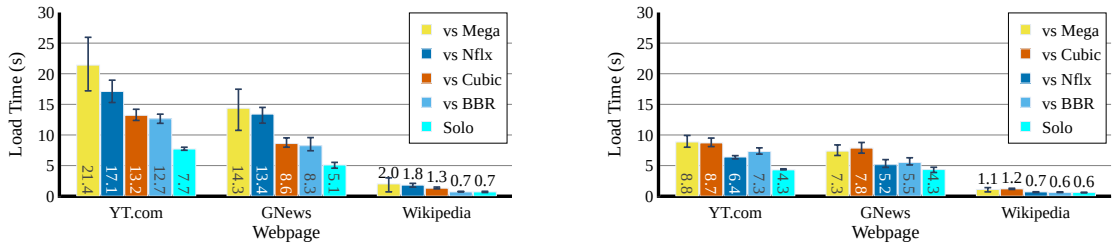
Observation 16: *Competing traffic can double page load times in the 50 Mbps setting, and triple it in the 8 Mbps setting, adding additional wait times of up to 4 and 14 seconds respectively in the worst case (Fig 4.6).*

We find that competing traffic can increase page load times, especially in the highly-constrained setting. In the presence of Mega and Netflix, users visiting youtube.com may have to wait for 21 seconds instead of just 8 seconds (median), a difference of 162%. The increase in loading time is also clearly correlated with how many images are on the webpage. Wikipedia, which is mostly text, is only minimally affected by competing traffic. In contrast, YouTube, which consists of mostly images, sees the greatest increase in load times.

In the moderately-constrained setting, aside from Netflix which is application-limited and cannot utilize the full link, BBR has the least impact on page load times. This is likely because BBR maintains small queues, allowing the bursty nature of webpage traffic to fill the queue and quickly obtain the bandwidth it needs.

4.5.3 Link Utilization & Loss

We now briefly consider two other performance metrics, returning to our more throughput intensive applications from the on-demand video and file transfer datasets.



(a) Highly-constrained setting **(b)** Moderately-constrained setting

Figure 4.6: Page load times are increased by competing traffic in both bandwidth settings, almost doubling it in the worst case. The greatest increase is seen with multi-flow services like Mega and Netflix, and the least by delay-based CCAs like BBR. In the highly-constrained setting, Mega and Netflix, both contentious, bursty services, cause high variance in page load times.

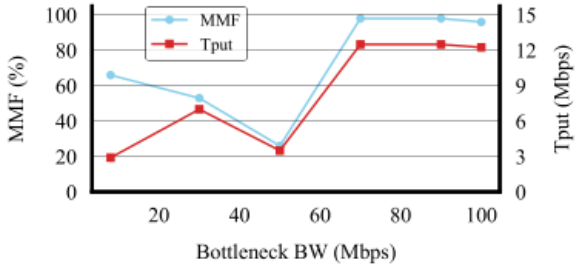


Figure 4.7: The unfairness YouTube suffers against Dropbox initially increases with the bottleneck bandwidth, then suddenly becomes fair beyond 70 Mbps.

Observation 17: *Application-level behaviors can cause both unfairness and under-utilization.*

In most scenarios we see 95% or higher link utilization (a complete heatmap is in the appendix of the SIGCOMM'24 Prudentia paper [121]). However, in some scenarios, we observe *both* unfairness and under-utilization: not only are these services unable to obtain their fair share, but this lost bandwidth is not utilized by contenders, and is effectively wasted. In the moderately-constrained setting, we see this with Mega causing NewReno, Cubic and One Drive to get less than 27% (Fig 4.2b) of their fair share while simultaneously resulting in less than 85% total link utilization in all cases. We believe this is due to the previously mentioned interaction of loss-based CCAs with Mega's bursty traffic (see Observation 4.4). The sudden burst of traffic causes NewReno and similar loss-based CCAs to experience loss and back off, but unlike Dropbox, they are unable to recover in time to utilize the unused bandwidth available between bursts. This is in spite of our buffer size being 4×BDP, which is traditionally considered a "deep" buffer.

We also see under-utilization in the highly-constrained setting when video services compete against each other. We suspect this is due to ABR algorithms prioritizing stabil-

ity over maximal throughput and hence choosing to play a video consistently at lower quality than potentially having to switch back and forth between higher and lower quality video [142].

Observation 18: *Multi-flow services induce the most loss, while BBR-based services induce the least, resulting in no loss for single-flow BBR-based services competing with other single-flow BBR services.*

We obtain the loss rate for a service by measuring the fraction of packets of that service that arrived at the bottleneck queue but were dropped (a complete heatmap is in the appendix of the SIGCOMM'24 Prudentia paper [121]). When single-flow BBR services such as Dropbox or Google Drive compete with other single-flow BBR services, they do not end up filling the queue and as a result experience no loss in both settings. On the other hand, in the highly-constrained setting, BBR does not prevent Mega from causing the most loss of any service (8%), reflecting our observation above that multiple BBR flows can also inflate latency. Aside from Netflix (which induces a loss rate of 4%), most other service interactions result in loss rates close to or below 1%. In the moderately-constrained setting, loss rates are even lower – close to 0% in almost all interactions.

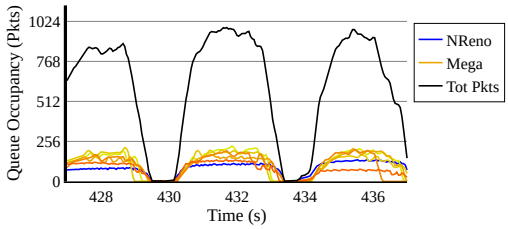
4.6 Lessons for Testing

The primary lesson from Prudentia for operators is the importance of testing *applications* for their side-effects on competing applications. In this section, we also highlight a few other aspects of testbed design and methodology which we draw from our experiences.

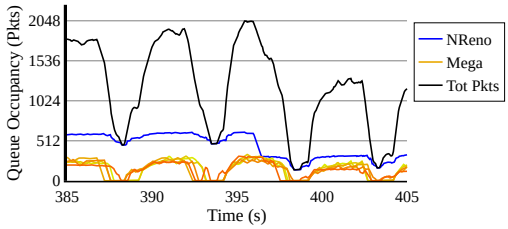
Observation 19: *Buffer sizing significantly influences fairness and utilization outcomes, underscoring the need to profile the properties of contended links in the wild.*

We repeated our experiments with a doubled buffer size. This led to significant changes in some of our results.

With a larger buffer, we find that Mega competing with both loss-based CCAs (NewReno and Cubic) in the moderately-constrained setting no longer results in link under-utilization; both cases achieve more than 95% link utilization: the queue was now large enough to absorb bursts from Mega without forcing NewReno and Cubic to experience loss and back off each time, as seen for NewReno in Fig 4.8. The large queue also allows these CCAs to have enough packets in the queue to keep throughput high until they recover from a loss. NewReno and Cubic consequently obtain more than 92% and 97% of their fair share respectively when competing with Mega, up from the 22% and 27% obtained when using the original 4×BDP buffer.



(a) (moderately-constrained setting, 4×BDP (1024 packet) buffer) In spite of using what is traditionally considered a “deep" buffer, we see link under-utilization when NewReno competes with Mega. This is due to a combination of Mega’s bursty traffic pattern suddenly draining the queue, and NewReno not having enough packets in the queue at the time to compensate for this drain.



(b) (moderately-constrained setting, 8×BDP (2048 packet) buffer) Doubling the buffer size results in NewReno-based iPerf obtaining a larger share of the queue when competing with Mega, preventing under-utilization.

Figure 4.8: Switching from a 4×BDP to a 8×BDP buffer results in NewReno-based iPerf obtaining a larger share of the queue when competing with Mega, preventing under-utilization.

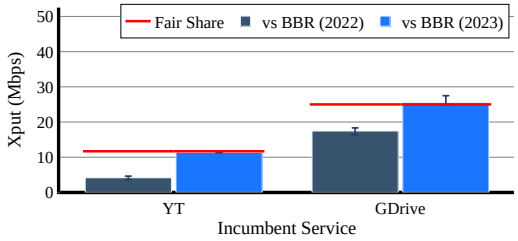
Conversely, NewReno’s MmF share against Cubic drops from 60% to 28% in the highly-constrained setting when larger queues are used. This is unfortunate but understandable as Cubic is well-optimized for larger buffers [57]. Larger buffers also increase the queuing delay experienced by all services when competing against a loss based CCA, which can negatively affect latency-sensitive services such as RTC.

These findings underscore the need for continued measurement studies (*e.g.* [37, 79]) so that operators can test their services given appropriate real-world parameters.

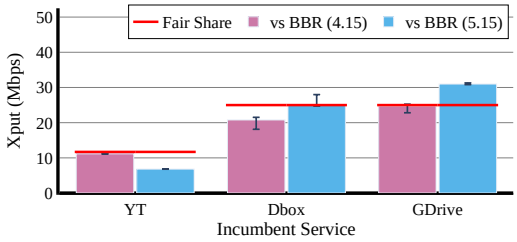
Observation 20: *Contentiousness can have a non-monotonic relationship with increasing bandwidth availability.* [9] *(Fig 4.7)*

We performed all-pairs experiments at a range of bandwidths between 8 Mbps and 100Mbps. Overall, we did observe a general trend of fairness improving with higher bandwidths. However, this was *not always the case* and in some scenarios we even observed fairness degrade with increased bandwidth. For example, we find that as we increase the bottleneck bandwidth from 8 Mbps to 50 Mbps, the MmF share acquired by YouTube from Dropbox actually decreases (Fig 4.7). Even more surprisingly, when we go from 30 Mbps to 50 Mbps, the raw throughput obtained by YouTube itself decreases. This means that YouTube plays

[9] This observation is based on experiments from the 2022 period referred to in §4.3.2.



(a) The throughput obtained by both YouTube and Google Drive when competing against BBR-based iPerf (Linux 4.15) increased between our measurements in 2022 and 2023. This coincided with BBRv3 being deployed to Google Drive, and QUIC-stack tuning for YouTube.



(b) Changes to BBR introduced in kernel updates between Linux 4.15 and Linux 5.15 made it less contentious against Dropbox and Google Drive, but more contentious against YouTube.

Figure 4.9: Changes to services, and even kernel updates, can change fairness properties, necessitating the use of a live watchdog that constantly monitors services.

at a lower quality when competing against Dropbox at 50 Mbps compared to 30 Mbps. These results suggest that testing for equitable services will persist as necessary even as broadband capacities increase with time.

Observation 21: *Incremental changes in CCA design can lead to noticeable changes in contentiousness. (Fig 4.9)*

Through Prudentia's live experiments, we were able to detect changes in Google Drive and YouTube's deployments between 2022 and 2023. We found that compared to 2022, Google Drive and YouTube performed 46% and 172% better in 2023 against iPerf-based BBR (see Fig 4.9a). Google engineers confirmed that this coincided with the deployment of BBRv3 to Google Drive [65] and parameter tuning in YouTube's QUIC stack.

We find similar changes in contentiousness when comparing BBR implementations in different versions of the Linux kernel – the version of BBR available in Linux 5.15 causes different fairness outcomes than that found in Linux 4.15 (see Fig 4.9b) – despite both of these versions supposedly representing 'BBRv1'. This serves as a word of caution for service owners – when using an actively developed CCA like BBR, it is possible that an innocent kernel upgrade might actually change the fairness properties of services running on it.

Table 4.3: Unfairness and fairness are not necessarily transitive. Service α may cause β to get an unfair (or fair) share, and β may cause γ to get an unfair (or fair) share, but this does not guarantee that α causes unfairness (or fairness) to γ. The lack of transitivity exemplifies the difficulty in classifying most services as generally contentious or sensitive.

α	β	γ	BW (Mbps)	MmF Obtained (%)		
				β (vs α)	γ (vs β)	γ (vs α)
Mega	NReno	Vimeo	50	22%	58%	104%
Cubic	Dbox	NReno	8	99%	106%	60%
BBR	1Drive	YT	50	108%	106%	58%

These findings underscore the need for live and continuous testing to keep up with the constant evolution of services and their underlying CCAs.

Observation 22: *Many of the most harmful outcomes are anomalous: they are not the result of one service being generally sensitive or contentious, but instead, the result of idiosyncratic interactions between the two services under test. (Table 4.3)*

While some services can be classified as generally “contentious" (e.g Mega) or generally “sensitive" (e.g. YouTube) with a tendency to grab or yield resources against *all* competing applications, we find that most services do not clearly fall into either of these categories. For example, we can see that Cubic lets most incumbent services obtain close to their fair share of bandwidth when competing with them. However, when competing with NewReno, the latter receives only 21% and 60% of its fair share of throughput in the moderately-constrained setting and highly-constrained setting, respectively (Fig 4.2). This highlights the need to evaluate each contender against a wide variety of incumbents. Simply extrapolating a service’s fairness from its interactions against a few incumbents can lead to erroneous conclusions.

This is further reinforced by our finding that unfair outcomes do not follow a transitive structure. A service α that is unfair to service β need not be unfair to another service γ, even if β is unfair to γ. Table 4.3 shows a few examples of this lack of transitivity, extracted from the set of results in Fig 4.2.

The above findings give us valuable guidance about testing new Internet services. For example, we should reject claims that a service is ‘safe’ to deploy alongside video streaming just because an experiment shows that the service is safe along a particular instance of video streaming.

Observation 23: *Service instability can lead to sometimes-harmful sometimes-not outcomes between the same services. (Fig 4.10)*

We observed that certain services exhibit a wider variance in the throughput outcomes obtained when competing with other services, and do not meet the +/- 0.5 Mbps and +/-

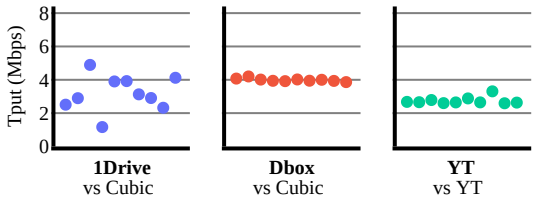


Figure 4.10: Each data point represents the throughput obtained by the service in **bold** in a single trial. Certain services such as One Drive show unstable outcomes when interacting with other services, while others are relatively stable.

1.5 Mbps 95% confidence interval range thresholds we place on the highly-constrained setting and moderately-constrained setting respectively. We provide an example of what this instability looks like in Fig 4.10. We observe this most consistently with Vimeo in the highly-constrained setting and One Drive in both the highly-constrained and moderately-constrained settings. We observe similar variance in outcomes with various RTC metrics in §4.5.1. Operators should be concerned about services which are 'sometimes' overly contentious, and run multiple trials to capture these issues.

4.7 Recommendations

Given our findings above, we now turn to making recommendations for future testing of deployed Internet services by both service owners and the research community at large.

Application developers need to test for fairness, not just CCA developers: While congestion control developers typically *do* test their services for fairness, application developers do not under the assumption that CCA developers have 'taken care' of the issue. Our findings show that application-layer decisions – such as ABR algorithms, the use of multiple connections, or unexpected browser interactions – can lead to different fairness outcomes than what one would expect given the underlying CCA. To this end, we allow the submission of custom URLs for testing on the Prudentia website. More details can be found at `https://internetfairness.net/testing`.

Pairwise testing – in a wide range of settings – is necessary: A surprising result from our findings is that there are no "bellweather" Internet services that can predict the general fairness properties of a service. In fact, many fairness outcomes were anomalous and unpredictable. This combined with our finding that buffer size and bottleneck bandwidth can affect fairness outcomes highlights the needs for thorough pair-wise fairness testing of a large set of popular Internet services in a wide variety of network settings.

Services should be tested continuously: Many of the fairness properties we saw change with small shifts in design. For example, we observed that QUIC parameter tuning for YouTube, incremental updates to BBR in the Linux kernel, and the deployment of BBRv3 to Google Drive changed their fairness properties. Other small shifts, such as changes in application behavior, may also influence service outcomes. Hence, service

testing is not a 'one and done' endeavor.

Involve service owners in root-causing unfairness: The proprietary nature of CCAs and ABR algorithms today limits third-party visibility into the precise causes of unfairness, and consequently, the fixes for it. In conversations with various service owners, we found that unfairness was usually an unintended and undesirable outcome, and one they are keen to rectify. It is therefore in the mutual interest of both the research community and service owners to work together to gain a better understanding of the underlying causes that result in specific instances of unfairness. Prudentia aids this effort by identifying and surfacing these instances for further investigation by both parties. To this end, the Prudentia website makes potentially useful data like bottleneck queue logs and client PCAPs for every experiment publicly accessible.

Should browsers play an active role in fairness?: Given that the most extreme cases of unfairness we observe are due to the use of multiple connections by the browser, we wonder if there are changes to be made to browsers themselves to enable fairer outcomes.

4.8 Other Related Work

There are two broad types of related work 1) fairness evaluations of CCAs [11, 29, 38, 156] and 2) frameworks for testing CCAs and deployed services [170, 90, 80].

Several studies have conducted experiments to evaluate the co-existence of CCAs. There is the evaluations done in proposals for new CCAs to legacy CCAs where the deployability has been justified through the lens of TCP friendliness using infinitely backlogged flows [11, 29, 38]. Turkovic et al. [156] did a detailed study of CCA interactions by first grouping them into loss-based, delay-based, and hybrid groups and then studying the interactions among them with bulk traffic. These studies have largely ignored and overlooked the other traffic patterns like video streaming when evaluating CCAs. As we've shown in this work, the workload used to evaluate CCAs impacts the fairness outcomes.

Several studies have built frameworks for studying the performance of CCAs and services in a variety of network settings. Pantheon [170] is a framework built to test CCAs under a variety of network settings, however this framework seeks to compare the performance of CCAs in isolation; it does not test the interactions between CCAs. MacMillan et al. [90] aimed at studying three modern video conferencing applications (VCAs): Zoom, Google Meet, and Microsoft Teams to understand how they perform under different network conditions. Apart from that, they have also studied how VCAs perform in the presence of other applications like iperf3, YouTube, and Netflix. Kunze et al. [80] conducted a study of how different content providers like Akamai interact with other content providers as out-of-the-box CCAs on Linux servers. We distinguish ourselves from this prior work by providing a study of broader scope over different application types, including video services. Our testbed interacts with external services using a scripted Google Chrome instance, and therefore should be easily extendable to other services which

can be accessed through the browser.

In addition, some prior work has shown that CCA implementations differ from specifications, including silent updates to algorithms like BBR in deployment [107, 105]. This motivates our conjecture that services need to be evaluated periodically and constantly.

4.9 Future Work

Going forward, we would like to scale Prudentia to test more services, networks settings, and vantage points.

Services: To keep up with an evolving Internet, Prudentia is designed to allow the easy addition of new browser-based services to its testbed. Drawing inspiration from Pantheon[170], we allow public PRs to our Github repository that automate the consumption of new services. This is in addition to the existing capability Prudentia provides for service owners to submit URLs for testing on its website.

Beyond pairwise testing: Past work has shown that a single BBRv1 flow can take up to half the link capacity even when competing against up to a thousand NewReno and Cubic flows [122, 162]. This behavior can be seen even in Prudentia's results – when BBR-based services compete against Netflix, which uses multiple NewReno flows, single-flow BBR services get close to half the link capacity in spite of being at a flow-count disadvantage. This raises the question of whether services that compete fairly against one other service would continue to be fair when competing against multiple services.

Network settings: Past work has shown that fairness outcomes can depend on network settings such as queue size, RTT, and background packet loss. It would be interesting to examine how the fairness outcomes observed by Prudentia change when these parameters are varied. For example, background packet loss would likely reduce the throughput obtained by services using loss-based CCAs such as Netflix and One Drive. Similarly, past work has shown that BBR's fairness when competing against loss-based CCAs can vary based on the queue size [29], and that NewReno suffers from poor performance in networks with high RTTs [97]. Testing these varied network settings would require modifying Prudentia to run multiple tests in parallel to ensure they all finish within a feasible time-frame.

Vantage points: To limit confounding effects from Prudentia's presence at a single vantage point, and to help us better understand fairness outcomes, we normalize the RTT of all competing services to 50ms. However, it is possible that in the real world services with widespread CDN deploymets will consistently experience lower RTTs than other services. Therefore it would be interesting to deploy Prudentia at various locations over the world without RTT normalization, and examine how that changes fairness outcomes. We hope by making Prudentia's source code publicly available, we can aid efforts in deploying Prudentia globally.

4.10 Chapter Summary

In this work we presented Prudentia, a watchdog for Internet fairness. Using Prudentia, we showed that to accurately predict service-level performance outcomes, it is important to test not just CCAs but also the services that use them.

Some of Prudentia's findings are altogether novel: for example, we are the first we know of to characterize javascript file transfer applications like Mega, and our tests of the interactions between RTC and On-Demand Video are counter-intuitive in that they result in low latency for both players. However, other findings of Prudentia are not novel – and perhaps should not exist in 2024. The networking community has known for decades that using multiple flows can cause negative outcomes (see Observation **11**) and that buffer-filling algorithms are bad for real-time communication (see Observation **14**). Here, Prudentia serves as a reminder to operators and the community that these design choices are nevertheless deployed on the Internet in large-scale, popular services.

Perhaps the most surprising aspect of operating Prudentia has been how many results are anomalous or hard to diagnose. Many of our expectations – *e.g.*, that more bandwidth would always reduce contention, or that CCAs are the ultimate driver of fairness outcomes – turned out to be entirely wrong. Given the proprietary nature of most services, it is important for the research community to work hand-in-hand with servie owners to better understand and rectify the causes of unfairness.

Prudentia also serves to highlight the importance of emulated testbeds even when A/B testing is available. Most of the findings in this chapter would be impossible to obtain using just A/B tests. One would never know whether poor throughput in an A/B test was due to competition with a contentious service, or simple low path capacity. It is scenarios like this where the controlled and observable nature of emulated testbeds, augmented with the capability to test real services, serves to enhance the testing process, as opposed to hold it back.

Prudentia runs continuously and is available online at
`http://internetfairness.net`.

Chapter 5

Network Conditions Experienced by Users of a Large Internet Service

> "If you know the enemy and know yourself, you need not fear the result of a hundred battles."
>
> —Sun Tzu, *The Art of War*

5.1 Chapter Overview

Developers of novel congestion control algorithms (CCAs) and application-level mechanisms such as Adaptive Bitrate Algorithms (ABRs) include countless assumptions about their deployment environment both in their process of design and testing. Testbeds [121, 60, 113, 170] carefully emulate what are believed to be "real world" conditions, allowing developers to explore the behavior of their CCAs under parameterized throughput and latency conditions, but also under complex network settings such as the presence of traffic filters or ACK-aggregating hardware [170]. Ideally, testing environments should emulate every detail of the messiness that exists between service and client in the real world.

Unfortunately, the network edge—from service infrastructure, typically hosted out of a CDN or off-net, to eyeball infrastructure, passing through cellular or residential broadband infrastructure—is a continually evolving and convolutedly managed datapath. Hence, this ideal is almost certainly impossible: different Internet Service Providers (ISPs), of which there are tens of thousands, have diverse policies (e.g., pacers, buffers, filters, and proxies) and these policies are dynamic and proprietary. Together, the scale, dynamism, and secretiveness about real networks make perfect replication an untenable proposition.

As a result, one approach to dealing with this complexity would be avoiding emulated or simulated network testing altogether, and instead relying exclusively on A/B testing [165]. As noted previously, A/B testing is a common industry-standard [138, 137] for verifying

the performance of CCAs and application-level changes in deployment, and to a lesser extent in academia through platforms such as Puffer [169]. However, A/B tests provide a longer turn-around time as they often require careful scrutiny before deployment, and risk negative performance outcomes for real users. While A/B tests can provide the final verdict about whether a new or updated CCA or ABR improves performance for users, they are of limited utility in understanding *why* a change might have caused a performance regression, as the path properties that caused this behavior are still unknown. As a result, testing in realistic conditions in emulated or simulated network testbeds is still critical to the development process of a CCA or application-level network algorithm.

As a result, the state of the art is for researchers, both industrial and academic, to perform opaque-box, end-to-end measurements of Internet connections and to infer individual "path properties" to be emulated in the network. Services like M-Lab's NDT, CloudFlare Radar, Ookla SpeedTest, Netflix's Fast.com, and RIPE Atlas offer invaluable measurements of baseline RTTs, loss rates, and queuing delays; these observed measurements are often then used by developers and testers of CCA services to configure their testbed or emulation environment.

However, there are few tools to observe the hyperscaler and CDN edge, in spite of almost two-thirds of Internet traffic today being served over these paths [76, 7]. Speed tests such as those from Ookla and M-Lab do not necessarily reflect the CDN infrastructure that is used to serve Internet traffic. While some speed tests are hosted on CDNs used to serve real Internet traffic, like Cloudflare Radar and Fast.com, they still need to be actively run by users. This leads to potential biases towards more tech-savvy users who know to run speed tests, who might also run them more often when their Internet connection is experiencing difficulties. They might also not run it on the device that is actually consuming the Internet service—a buffering video on a user's television might cause them to run the test on their smartphone, instead of the television itself, as evidenced by device statistics reported in past work [118].

In this work [1] we aim to augment the community's understanding of path properties at the far edge (from a hyperscaler content distribution network to individual clients) by offering novel findings from over 2.4 million user sessions to Netflix, a large-scale on-demand video streaming service. Netflix serves a diverse user base worldwide, with sessions in more than a hundred countries, served from almost 20,000 servers spread across 6,000 distinct locations, and is consumed over a variety of devices, including televisions, laptops and smartphones, providing a rich and varied dataset. By estimating path properties *passively* from normal user traffic, we avoid the biases introduced by tests users need to actively conduct, and are able to report the path properties experienced by users at the time they are actually consuming an Internet service. This diversity offers a

[1] This work is currently under submission, and text and images from the submitted work may be used verbatim in this chapter.

broad and representative view of path properties. While this approach does not entirely eliminate demographic bias—Netflix's users may not fully represent the entire Internet population, and its CDN infrastructure and client device distribution might differ from other large services—it provides an important perspective that enriches our understanding of path properties beyond speed tests. We hope our contribution is a step towards other large Internet service operators sharing their own insights, leading to a more complete understanding of the path properties faced by Internet users today.

Over the course of this chapter, we measure various path properties from Netflix user traffic, and compare the results with those from other measurement studies and CCA evaluations. Compared to typical CCA evaluations and testbeds, Netflix experiences lower Base RTTs, higher peak observed queuing delays relative to the Base RTT, greater ACK aggregation, and TCP proxies. Compared to other measurement studies, Netflix experiences lower queuing delays, and Base RTT that is higher than those from M-Lab tests, but similar to those from Ookla and Measuring Broadband America [44]. Netflix also encounters TCP proxies in 20% of its cellular sessions. ACK aggregation, a path property that is increasingly important due to its interference with packet-train based throughput measurement techniques employed by newer CCAs like BBR [10, 30], is surprisingly omitted in most measurement studies and CCA evaluations. We discuss these results in more detail in §5.4 through §5.6.

In the rest of this chapter, we explain the related work in §5.1, describe our data sources and their limitations in §5.3, and elaborate on our findings on various path properties in §5.4 through §5.6. We conclude by summarizing our findings and describing directions for future work in §5.9.

5.2 Inferring Path Properties: Related Work

The networking community employs two main methods to estimate path properties: active and passive measurements. Active measurements send test traffic to probe the network, while passive measurements analyze existing traffic. Each method has strengths and limitations.

Active speed tests are particularly useful for individual users seeking to understand the capacity of their own network link. They provide immediate insights into available bandwidth and help diagnose connectivity issues. Among the most widely used tools are Ookla's Speedtest [116], M-Lab's Network Diagnostic Tool (NDT) [49], fast.com [110], and Cloudflare's Radar [36], which run speed tests and report downlink and uplink speeds as well as RTTs. Netalyzr tested for connectivity issues and speed, offering a broader diagnostic perspective [79]. Numerous studies have examined the probing methods and accuracy of speed tests, as well as compared different speed test methodologies [15, 91, 118]. The Measuring Broadband America project, sponsored by the FCC, measures directly from home routers and conducts a more diverse set of measurements, including web and video downloads, and RTT under load [44].

While some these datasets measure a magnitude lower number of client locations than the hundreds of thousands used in this chapter, they have driven numerous analyses, providing valuable insights into network performance trends and informing our understanding of broadband coverage and performance [145, 115, 118, 79]. They have revealed, for example, that users on wireless links often do not achieve the subscription speeds they pay their ISPs for [118], that there is occasional congestion in the core of the Internet [37], and have contributed to measuring latency variation on the internet [62].

We present a comparison of the path property values experienced by Netflix sessions to those from these large measurement datasets in the sections dedicated to the respective path properties (§5.4, §5.5, §5.6).

Given the limitations of active measurements, this work adopts a complementary passive approach. Previous studies have developed tools to facilitate the passive observation of TCP traffic, enabling the inference of various path properties. Some of these tools instrument the TCP stack on the server [96, 123, 143], while others reconstruct network behavior from packet traces traversing the network [119, 101, 120]. Utilizing these passive techniques, researchers can perform root cause analysis of factors limiting TCP throughput [135, 174], detect TCP proxies in cellular networks [50], and estimate path capacity using packet inter-arrival times [42].

While passive measurements offer the advantage of capturing path properties as users actively consume an Internet service, they are not without limitations. Our study is influenced by our vantage point, specifically, Netflix infrastructure and users. First, Netflix's edge CDN is often in close proximity to users, which can result in lower base RTTs and fewer potential bottlenecks. Second, we only observe video streaming traffic, which

affects how we interpret certain path property estimates. For example, the peak observed queuing delay might be lower than the maximum possible queuing delay on the path, as video streaming does not necessarily saturate the link like a speed test would [138]. We address these caveats when discussing each path property. Additionally, our measurement demographic is limited to Netflix users, whose distribution may differ from the general Internet user base. To enhance the relevance of our findings, we segment the data by device type, network type (wired, wireless, cellular, where available), and country where necessary. This segmentation allows services with specific user profiles, such as those primarily serving smartphone users, to apply the most pertinent portions of our results.

5.3 Dataset

Our passive measurements originate from video on-demand streaming sessions at Netflix, a global video streaming provider with hundreds of millions of users. A session corresponds to an uninterrupted viewing of a specific video title by the user. Most viewing hours occur on “Fixed Entertainment Devices" (FEDs) such as TVs and gaming consoles, followed by laptops/desktops and smartphones. Content is delivered using TCP with a variant of NewReno, a loss-based CCA, through an extensive CDN that aims to deliver traffic as close to users as possible.

Where necessary, we contextualize the path properties experienced by our sessions using the following per-session metadata:

1. Device Type: We categorize devices into three types: Fixed Entertainment Devices (FEDs), such as smart TVs, set-top boxes, and gaming consoles; PCs, which include laptops and desktops accessing Netflix through a browser-based client; and Mobile devices, encompassing smartphones and tablets.

2. Client Network Type: The Netflix client logs the network type the device is connected to at the start of each session, with some exceptions. For FEDs, we can identify whether the connection is wired or wireless, such as Wi-Fi. Mobile devices are typically connected via wireless networks, and we can distinguish between Wi-Fi and cellular connections. However, for PCs, we lack network type information due to the limitations of the browser-based client.

3. ISP Type: For some sessions, we can identify the Internet Service Provider (ISP) type, distinguishing between cable, fiber (US only), and cellular ISPs. This allows us to examine differences in path properties across ISP types.

4. Country: Netflix geolocates customers by (1) extracting the public IP address they stream from and (2) using multiple commercially available databases to geolocate their IP address. We use regional metadata to highlight differences in certain path properties across various countries.

For all path properties we rely on per-packet data enhanced with the TCP stack's internal state and measurements [143] that Netflix CDN servers log for 0.01% randomly sampled streaming sessions. Since a single session may stream from multiple CDN nodes, for simplicity, we consider only the traffic to the CDN node that transferred the most data for each session. To ensure sufficient data for our estimation techniques, we limit our analysis to sessions lasting at least two minutes. This leaves us with 2.4 million unique sessions, from more than a hundred different countries and hundreds of thousands of distinct devices, that we use in our analysis, sampled over the entirety of September 2024.

Ethical considerations. We follow strict ethical guidelines to ensure user privacy and data protection. Netflix collects data solely to enhance user experience. Our analysis is limited to network property metrics and excludes sensitive personal information; for instance, our dataset does not include details about the content being watched. Our research does not alter or interfere with users' streaming experiences; we passively observe network performance as it naturally occurs. Sessions are annotated with metadata upon collection, retaining only the high-level information necessary for analysis. This approach respects user privacy while providing valuable insights into network conditions.

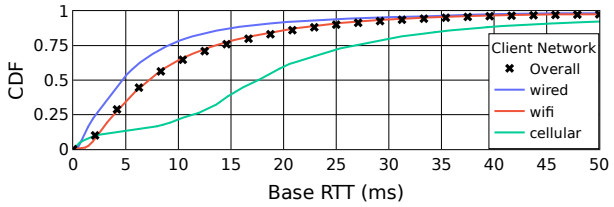


Figure 5.1: The CDF of Base RTTs seen by sessions from various device/home network combinations. Wired sessions see lower RTTs than their Wi-fi counterparts, almost 30% lower at the 50th percentile, while cellular sees the highest RTTs, more than three times those of wired sessions.

5.4 Base RTT

Base RTT is the minimum possible round-trip time over a given path when there is no queuing or congestion, and is the sum of the forward and reverse propagation and transmission delays on the path. Emulating a realistic Base RTT is critical for congestion control testing, as it impacts fairness between CCAs [26], time to convergence [35], CCA performance due to shorter or longer feedback loops [97], and buffer sizing [6]. It is therefore one of the most commonly used parameters when designing emulated or simulated testbeds for CCA evaluation.

Estimation Technique: We estimate the Base RTT of a session as the minimum of all the RTT samples measured by Netflix TCP connections in that session. The TCP stack generates these samples for every ACK received by taking the difference between the time the ACK was received and the corresponding packet that is being acknowledged was sent. When a cumulative ACK acknowledges more than one packet, the stack uses the latest send time of the data being acknowledged. This is a common technique used in other measurement datasets [86, 88, 89].

5.4.1 Base RTT by Client Network Type

The CDF of RTTs experienced by various network types is shown in Fig 5.1. As expected, wired devices experience lower RTTs (median: 5ms) than those on Wi-Fi links (median: 7ms), with cellular links having the highest base RTTs (median: 18ms). The 90th percentile of Base RTTs is 30ms for wired and Wi-Fi links and just 50ms for cellular links. A small fraction of cellular sessions experience sub-ms RTTs, which we believe are largely due to TCP proxies, and discussed later in §5.4.4.

To put our results in perspective, we compare the Base RTTs obtained by Netflix sessions with those from popular measurement studies. For example, Ookla's Speedtest Global Index reports median RTTs of 9ms for fixed broadband connections and 25ms for cellular connections [140]. The Measuring Broadband America Report [44] finds that the median Base RTTs of Cable ISPs is typically between 12ms and 22ms while that of Fiber ISPs is between 7ms and 13ms. The median RTTs reported from both these sources are similar to but higher than the median Base RTT of 7ms experienced by non-cellular Netflix

sessions. This discrepancy is likely due to Netflix's CDN deployment being closer to users than the vantage points used by either of these measurement tools. We analyzed M-Lab (NDT) data for the same time period as the Netflix sessions and found the median Base RTT to be 23ms, higher than Netflix, Speedtest and Measuring Broadband America. We were unable to distinguish between cellular and non-cellular sessions in the data, but this is still high given that the median cellular RTT from both Ookla's Speedtest and Netflix sessions is 22ms and 18ms respectively. The 90th percentile of Base RTT in NDT tests corresponds to 82ms, which is greater than the 90th percentile of Netflix's non-cellular sessions (<30ms) and even cellular sessions (approx. 50ms). This is likely because M-Lab's test servers do not have the wide CDN coverage that Netflix does, with its test servers being located primarily at Tier-1 IXPs, and some virtual servers in cloud networks such as Google Cloud [87]. On the other hand, Ookla and Measuring Broadband American both make use of test servers that are sometimes embedded within the user's ISP, similar to Netflix. Overall, these results show that while external measurement sources such as speed tests present a useful perspective into the RTT a user is likely to experience, the nature of the vantage points employed can cause the observed RTT to differ from the RTT a user experiences when they actually consume a service such as Netflix.

While Netflix sessions, Ookla Speedtest and the Measuring Broadband America report show that sub 10-ms Base RTTs account for almost 50% or more of sessions/tests, existing work evaluating CCAs in WAN settings [122, 121, 29, 162] use RTTs as high as 100ms to 200ms. The lowest Base RTT used in past work is typically 10ms, which is still greater than the Base RTT reported by more than half of Netflix sessions or Ookla's Broadband Speedtest results. While the difference between 5ms and 10ms can seem low in absolute terms, in relative terms it is a doubling in latency, which would halve the rate at which RTT-based CCAs like NewReno grow their congestion window, and consequently, their sending rate.

We now discuss specific examples of RTT ranges used by past evaluations: work evaluating how well various CCAs perform in the presence of AQMs [45] used RTTs between 10ms and 100ms. Pantheon [170] is a state-of-the-art CCA testbed which was used by newer CCAs like Copa [11], Vivace [38], Aurora [71] and TCP-TACK [83] to evaluate their performance. More than 80% of the emulated settings offered by Pantheon use Base RTTs greater than 50ms. BBRv1 [29] was tested primarily in settings with a base RTT of 40ms, with subsequent work that attempted to model BBR [162, 108] using 40ms or between 20ms and 60ms for most of their evaluations. Work that examined congestion control behavior at scale [122] used RTT settings ranging from 20ms to 200ms. The trend of not testing at sub-10ms RTTs continues in recent work—an analysis of the fairness properties of BBRv3 [173] uses a Base RTT of 100ms in almost all their evaluations, an Internet fairness watchdog that evaluates inter-service fairness outcomes [121] uses an RTT of 50ms, and a recently proposed improvement to TCP Slow Start to better aid short-lived flows tests only RTTs between 25ms and 200ms [8].

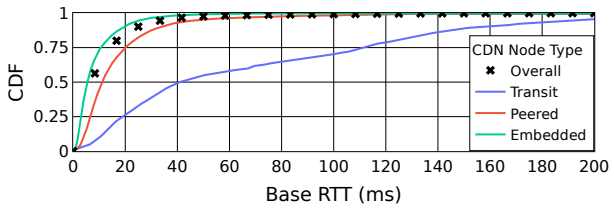


Figure 5.2: Sessions served by CDN Nodes deployed within an ISP ("Embedded") or at an Internet Exchange location ("Peered") experience significantly lower RTTs than those embedded in a Transit ISP. Embedded and Peered CDN nodes account for more than 99% of Netflix's sessions. This shows that services that use edge-adjacent CDN nodes similar to Netflix can likely test in settings under 50ms, while those using Transit ISPs might need to test RTTs as high as 200ms.

We find that Netflix should be testing its service in much lower RTTs than used in most CCA and Internet service evaluations: testing RTTs less than 50ms would cover more than 99% of wired and Wi-Fi sessions, and 90% of cellular sessions. Specifically, Netflix requires increased testing in sub-10ms Base RTT ranges, as that is what is experienced more than 50% of Netflix sessions. This is not without consequence—NewReno, the CCA used by Netflix, is typically dismissed as a CCA to be used in WANs due to its congestion window (CWND) growth rate, and consequently its throughput, being inversely proportional to the RTT of the link [97]. This can cause it to under-utilize high bandwidth, high latency links, which is one of the reasons it was replaced by Cubic [57] as the default CCA on Linux. However, the lower RTTs that Netflix experiences in practice make NewReno more than capable of serving its traffic, a conclusion one would not arrive at if it were tested in the 100-200ms ranges that many modern CCA evaluations use. As a concrete example, consider a path with a capacity of 100 Mbps. A NewReno flow in congestion avoidance would take around 80 seconds to grow its CWND from 1 MSS to the 1 BDP (Bandwidth-Delay Product) required to fully utilize the link if the Base RTT was 100ms, but just 0.8 seconds if the Base RTT were 10ms.[2] This highlights how testing a CCA in conditions it is more likely to encounter can lead to drastically different and more realistic conclusions about the performance of the CCA in practice.

It is possible that other services with similarly extensive CDNs and largely static content, including video streaming services which account for more than 60% of Internet traffic [124], should also be testing in such low RTTs. On the other hand, services that serve more real-time content might still need to test wider RTT ranges—for example, a video conferencing service that often connects users across continents will likely experience RTTs greater than 200ms.

[2]The 100x increase from 0.8s to 80s when the Base RTT increases by 10x is due to a combnation of: 1. The CWND to fully utilize the link (1 BDP) grows by 10x, 2. NewReno increases it CWND by 1 MSS per RTT, and as a result the rate at which NewReno increases its CWND decreases by 10x.

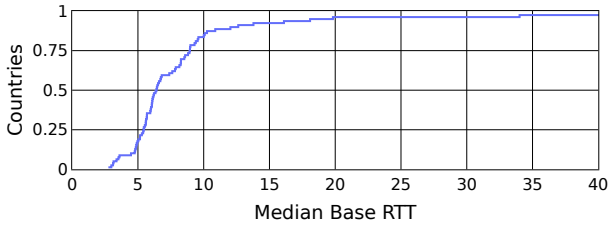


Figure 5.3: The CDF of countries with median session Base RTT less than various values. Almost 80% of the 108 countries shown here have their median Base RTT less than 10ms, showing that the low RTTs observed by Netflix are not limited to a specific few countries where Netflix's CDN is especially expansive.

5.4.2 Base RTT by CDN Node Type

We further quantify the impact a more edge-adjacent CDN can have on the RTTs experienced by Netflix sessions by comparing RTTs for node types with varying degrees of edge-adjacency. Not all Netflix CDN nodes are built equal—some are "embedded" within user-facing ISP networks, others hosted at peering locations such as Internet Exchange (IX) points, and others rely on Transit ISPs [67] for Internet access. We refer to these different types as **embedded**, **peered** and **transit** nodes respectively, with embedded nodes typically being closest to users and transit nodes being the furthest. Many large service operators have similar hierarchies in their CDN infrastructure [53, 63, 112]. We hope that by separating the RTTs experienced by Netflix sessions by various CDN node types, we can increase the generalizability of these results to service operators with varying levels of edge-adjacency in their CDNs.

We show the RTTs experienced by each of these types in Fig 5.2. As one might expect, embedded nodes show lower RTTs in general compared to peered or transit nodes. For a service with embedded nodes similar to Netflix, testing between 0-20ms would cover more than 90% of wired and Wi-Fi connected sessions. If the service uses predominantly peered CDNs, on the other hand, it would be necessary to test a larger range of 0-50ms for the same amount of coverage.

CDN nodes connected by a transit ISP, on the other hand, experience a wide range of RTTs, with almost a quarter of connections exceeding 100ms RTT, and going as high as 200ms. As a result, services that rely on transit ISP-connected CDNs will likely have to test in larger RTT ranges. This puts into perspective the latency benefits of using more edge-adjacent peered or embedded CDN nodes. Netflix's CDN nodes are primarily embedded or peered, with transit nodes accounting for a minuscule fraction of sessions (< 0.5%).

5.4.3 Base RTT by Country

We investigate the possibility that the low Base RTTs Netflix experiences is due to a majority of sessions arising from a small set of countries where it might have exceptionally

edge-adjacent CDN infrastructure. In Fig 5.3, we show that for almost 80% of the 108 countries that have at least 500 Netflix sessions in our dataset,[3] the median RTT is less than 10ms. Clearly, while specific countries might experience higher RTTs, more than half the users in 80% of the countries that Netflix serves experience sub-10ms Base RTTs. This shows that the nature of Netflix's CDN infrastructure is not limited to a few countries and is more generalizable.

5.4.4 Low RTT Cellular Sessions: TCP Proxies

As seen in Fig 5.1, a number of cellular sessions have very low (<5ms) RTTs. We confirmed that these low RTT values are not the result of one-off measurement anomalies within a session as they appear even when using the 1st-percentile of RTTs as opposed to the minimum. This was surprising, since cellular connections are typically expected to have higher RTTs, as evidenced by the higher percentiles of the cellular RTT CDF. While it is possible that there exist extremely good cellular connections that offer such low latencies, another possibility is that they are the product of Connection-Terminating TCP proxies deployed in cellular networks [50]. Such TCP proxies are known to improve TCP performance for users [43, 68, 128], and actively deployed by cellular carriers [50]. In such a case, the server measures the RTT from the server to the TCP proxy, which we assume goes over fully wired links, instead of the RTT to the client device, and therefore does not observe the higher RTTs that are commonly expected to accompany cellular networks [140, 81]. This theory is supported by the jump in RTTs in the CDF: the 10th percentile corresponds to 3ms RTT, but the 15th percentile immediately jumps to 15ms, suggesting a drastic difference in network conditions between the sub-5ms and higher RTT sessions.

We further strengthen this hypothesis with a TCP receive window (RWND) based TCP proxy detection technique inspired by past work [50], which identifies the majority of low RTT cellular sessions as using TCP proxies. The technique is based on the insight that, in the absence of a TCP proxy, devices typically advertise a consistent, device-specific initial RWND when they open a connection [150, 50]. However, a proxy will this change this RWND to that of the proxy's before the packet reaches the server. Therefore, a discrepancy between the initial RWND seen by the server for a session on a given device and the "typical" RWND expected for that device can indicate the presence of a TCP proxy. We provide a detailed description and evaluation of the technique in Appendix 5.8, but it can be briefly summarized as follows: sessions which both show an RWND discrepancy and whose origin (client-side) Autonomous System (AS) has at least 10% of sessions with an RWND discrepancy are tagged as proxied.

As seen in Fig 5.4, the proxy detection technique is effective in separating low-RTT

[3]We placed a session count threshold to avoid drawing conclusion from countries with insufficient data points.

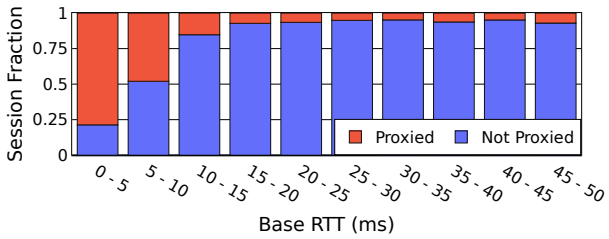


Figure 5.4: The fraction of cellular sessions with various Base RTTs that are categorized as using TCP proxies or not. More than 75% of sessions with less than 5ms Base RTT are classified as using TCP proxies, lending credence to both the TCP proxy detection technique, and the hypothesis the low RTT cellular session we observe are due to TCP proxies.

sessions from higher RTT sessions, strengthening the hypothesis that the low RTT sessions are due to TCP proxies. Suspected proxy sessions also experience significantly lower peak queuing delays in most cases, which we discuss in the queuing delay section (§5.5, Fig 5.9). A TCP proxy would explain both the low RTTs and low peak queuing delay (which we measure using RTT variation), as the proxy would insulate the server's TCP stack from the effects of the cellular link layer, and consequently, the higher RTTs and RTT variation associated with it [81].

We find that roughly 20% of Netflix's cellular sessions were tagged as using TCP proxies. As one might expect, this behavior is often AS-specific, with just 10% of ASs accounting for almost 90% of proxied sessions. However, it is not necessary that an AS that has a proxy have all its sessions proxied—at least 10% of the ASs that cellular sessions originated from had just 20-60% of their sessions marked as proxied, as seen in Fig 5.5.

The presence of TCP proxies on cellular networks presents an interesting dilemma for service operators. While it is ostensibly deployed with the intention of improving cellular performance [50, 43, 68, 128], it further limits a service operator's ability to implement optimizations of their own for their users. It can be especially challenging for server-side applications that rely on accurate feedback loops from the network, as the Base RTT and the RTT variation experienced by the user due to the cellular link is hidden from the server. Given the potential impact TCP proxies can have on such applications, and their prevalence in cellular networks noted both in this and past work [50, 164], we recommend that these proxies and any AQMs they deploy be further characterized, so that they can be included in future CCA evaluations and testbeds.

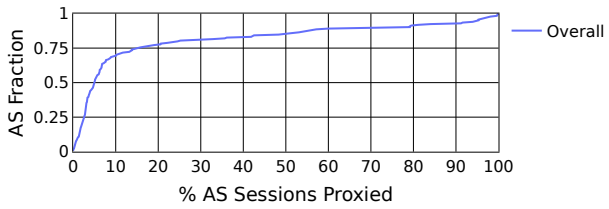


Figure 5.5: A CDF showing the fraction of cellular-serving ASs that had at least a given % of sessions within it proxied. Clearly, it is not necessary for an AS to serve exclusively proxied traffic, with 10% of cellular ASs having only 20-60% of their sessions being served by proxies.

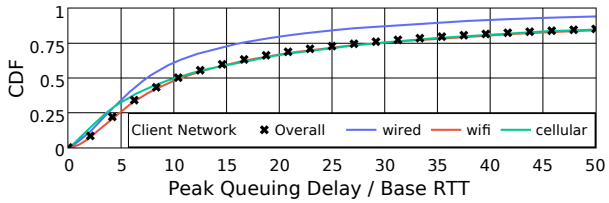


Figure 5.6: The CDF of peak queuing delay relative to Base RTT, measured at session-level granularity, showing that the peak delay can be several times the Base RTT of the link. Almost half of Netflix's Wi-Fi and cellular sessions experience peak delay that is more than 8 times the base RTT, while many CCA evaluations configure their testbeds to have peak delay that is only 1-4 times the Base RTT.

5.5 Peak Observed Queuing Delay

Queuing delay occurs when packets arrive at a router faster than they can be sent over the bottleneck link, causing the router to queue them. This delay introduces RTT variation, which serves as a congestion signal for many delay-aware CCAs like BBR and COPA [11]. In testing environments, the buffer size at the bottleneck link typically determines the maximum possible queuing delay, significantly influencing fairness outcomes. This has been shown to be true when loss-based CCAs like NewReno and Cubic compete [57], and even more so when loss-based and delay-based CCAs compete [29]. For instance, BBRv3 tends to be particularly unfair to Cubic in shallow buffers [173]. Conversely, large buffers can lead to bufferbloat [48], adversely affecting latency and QoE for real-time services like video conferencing and cloud gaming. To accurately test the effects and interactions that a CCA or service can experience, it is crucial that the queuing delay emulated in the testbed mirrors the queuing delay it is likely to encounter in deployment.

In this section, we report the peak observed queuing delay on paths traversed by Netflix. This observed delay effectively serves as a lower bound on the maximum possible queuing delay, as the on-off nature of video traffic [138] can prevent queues from being fully filled, as supported by comparisons to past studies in § 5.5.1. Surprisingly, even this lower bound exceeds the queuing delay values used in many CCA and Internet service evaluations. Most CCA evaluations configure the maximum possible queuing delay in relation to the Base RTT of the path, and the most commonly used value across is evaluations is 1-2 times the Base RTT. However, 50% of Netflix sessions experience peak queuing delay that is at least 10 times, and 25% experience peak queuing delay that is at least 25 times the Base RTT. We discuss these findings in detail in § 5.5.1.

Estimation Technique: We measure the peak observed queuing delay as the difference between the 99th percentile of RTT samples measured by the server's TCP stack and the Base RTT (the minimum of these RTT samples). The 99th percentile is used to avoid outliers, as maximum RTTs can be inflated due to Delayed Acknowledgments.

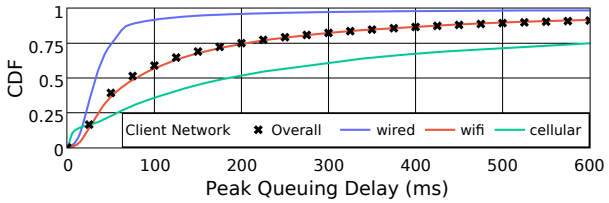


Figure 5.7: The CDF of peak observed queuing delay for Netflix sessions, which serves as a lower bound on the maximum queuing delay on these paths. Wired connections experience significantly lower queuing delay than Wi-Fi connections, while cellular connections experience the most.

5.5.1 Queuing Delay by Client Network Type

Most CCA evaluations configure the maximum amount of queuing delay possible in the testbed as a function of the Base RTT of the path. This is typically set implicitly by defining the maximum queue size of the bottleneck link on the path as a multiple of the Bandwidth-Delay-Product (BDP). For example, if a queue is sized to be 2 times the BDP, the maximum queuing delay $QD_{max} = Q\_SIZE/BW = 2 \times BW \times RTT/BW = 2 \times RTT$. While the range of maximum queuing delays in testbeds varies greatly across evaluations, almost every evaluation supports maximum queuing delays that are 1-2 times the Base RTT. However, in Fig 5.6, the median session-level peak queuing delay is 10 times the Base RTT of the path, and the 75th percentile is 25 times the Base RTT.

Since the ratio of queuing delay to Base RTT is high partially due to the low Base RTTs experienced by Netflix sessions, we also present the peak queuing delay in absolute terms in Fig 5.7. Surprisingly, a small number of cellular sessions show lower peak queuing delay than wired or Wi-Fi sessions. Similar to what we see with low Base RTTs in §5.4.4, we suspect that these are due to TCP proxies. We elaborate on this in §5.5.3.

We show the absolute peak observed queuing delay experienced by Netflix sessions in Fig 5.7. A comparison to other measurement studies reinforces the hypothesis that the peak observed queuing delay from Netflix sessions is likely a lower bound on the maximum possible queuing delay on these paths. Netflix sessions report a median and 75th-percentile queuing delay of 80ms and 200ms respectively, while NDT speed tests [62] report median and 75th-percentile queuing delays of 200ms and 500ms respectively—NDT tests experience more than twice as much peak queuing delay as Netflix sessions. Measuring Broadband America [44], however, reports lower queuing delays, between 50-400ms, but still several times the Base RTT. The lower absolute queuing delays could be due to the nature of the specific ISP links it measures, such as certain ISPs deploying AQMs like L4S [152], and the fact that it uses completely wired client networks, avoiding the increased delays associated with Wi-Fi links (Fig 5.7). A weighted average (by number of tests) of the country-level queuing delay from Cloudflare Radar corresponded to 135ms, lower than other measurement datasets. This could be due to Cloudflare speed tests not saturating the path as aggressively as Ookla's tests, for example.

Internet performance evaluations use a wide range of parameters when it comes to setting the maximum possible queuing delay. While evidence from both Netflix sessions and past studies [62, 44] suggest we should be using larger queuing delay values, numerous CCA and Internet evaluations use significantly smaller queuing delays. A number of popular QUIC evaluations use queuing delays between 0.5 to 2 times the Base RTT [32, 73, 109], with some studies going as high as 5 times the Base RTT [104]. A study measuring congestion control performance at core Internet links uses queuing delay that is equal to the Base RTT of the link for all its evaluations, and some of the early empirical evaluations of BBRv1 tested it with maximum queuing delays of 0.8 and 8 BDP [61]. Examining Internet service measurement studies paints a similar picture—an evaluation of Cloud Gaming services [168] uses maximum queuing delays between 0.5 and 7 times the Base RTT. Pantheon [170] offers two kinds of emulators: 'artificial" emulators, intended to test specific network conditions, and calibrated emulators, designed to mimic real world links. The artificial emulators use buffer sizes between 0.1 to 1 BDP, and even the calibrated emulators use queue sizes between 0.1 to 6 BDP.[4]

That said, many Internet evaluations avoid this pitfall, either by using high absolute queuing delay values instead of relative to the Base RTT, or testing in a wide range of queue sizes. The original BBRv1 evaluation [29] used queues with as much as 200-500ms of queuing delay, and an Internet fairness watchdog [121] used a queuing delay of 200ms, greater than the median queuing delay reported by both Netflix sessions and NDT [62]. On the other hand, work modeling BBRv1 tends to evaluate a wide range of maximum possible queuing delays ranging from 0.25 to 128 times the Base RTT [162, 108], and a study about BBRv3 performance used queuing delays ranging from 1 to 32 times the Base RTT. However, in many evaluations that sweep a large range of buffer sizes, there is wide variance in the performance of the evaluated CCA or Internet service based on the maximum possible queuing delay [162, 27]. For example, when a single BBRv1 flow competes with a single Cubic flow, a testbed with a maximum queuing delay that is 0.25 times the Base RTT causes a single BBR flow to take up more than 90% of the bandwidth, while if it is 64 times the Base RTT, it is Cubic that takes up more than 90% of the bandwidth [162], the exact opposite fairness result. When there is such wide variance in the outcome based on the maximum possible queuing delay, we recommend attaching more importance to the results in buffer sizes that more accurately reflect the queuing delay measured in settings where the CCA or service in question is intended to be deployed.

In some evaluations that set the queuing delay as a function of the Base RTT, the use of high Base RTT values can result in the absolute queuing delay being high enough to fall within ranges reported by measurement studies. However, the pitfall we must eliminate

[4]While we do not know for sure why Pantheon's calibrated emulator parameters differ from the typical queuing delays reported by measurement studies, it's possible that it is due to the specific characteristics of the small number of links that Pantheon chose to replicate.

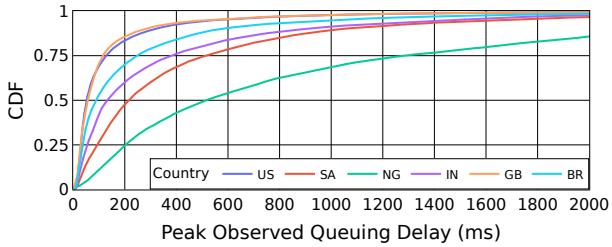


Figure 5.8: The CDF of peak observed queuing delay by sessions in various countries, identified by their country ISO codes. Clearly, there can be wide variance in the peak queuing delay by country, with more than a quarter of sessions in Nigeria experiencing greater than 1 second of peak delay.

is the pervasive assumption in many evaluations that queue sizes, a key contributor to queuing delay, is inherently tied the Base RTT of the path. This likely originates from the old rule of thumb which states that a single NewReno flow requires at least 1 BDP (Bandwidth $\times$ RTT) sized queues at a given link to be able to fully utilize it [158, 6]. However, as CDNs grow closer to users' homes, reducing RTTs, it is not necessary that ISP hardware configurations or user's home router queue size changes at the same rate. Additionally, past work has shown that due to a combination of ISP service-level agreements (SLAs) penalizing packet loss, and switch/router manufacturer advertising focusing on being able to provide larger queue sizes, queues on the Internet have only become larger [98]. Therefore it is not just important to configure testbeds for larger queuing delays—we also need to disconnect it from the Base RTT of the path, and instead rely on sizing buffers using the absolute queuing delay values reported from measurements such as speedtests.

5.5.2 Queuing Delay By Country

We find that peak observed queuing delay can indeed vary with country. We show this in detail for specific countries selected to provide a representative but diverse view of sessions in various continents (Fig 5.8). Nigeria, along with some other South African countries (not shown), showed particularly high queuing delay. There are a number of possible explanations for this. It is possible that the network paths in these countries are typically congested, and as a result queues are more full on average, causing Netflix traffic to also experience increased queuing delay. Paths to users in these countries could be longer, causing traffic to pass through multiple queues, each of which could also be experiencing high queuing delays. It could be due to lower capacity paths being present in these countries, due to which Netflix traffic tends to be able to saturate the queue to greater extents in spite of its on-off traffic pattern and comparatively lighter (to a large bulk download) video streaming workload. This is supported by internal studies at Netflix which showed that many countries with higher median peak queuing delays also experienced lower median client-reported throughput. Lastly, it could also be due to network equipment with larger queue sizes being present in these countries.

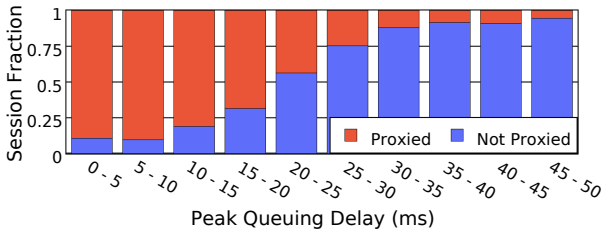


Figure 5.9: The fraction of cellular sessions with various peak queuing delays that are categorized as using TCP proxies. Almost 90% of sessions experiencing less than 10ms of peak delay are tagged as using TCP proxies, further strengthening the hypothesis that some cellular Netflix sessions traverse TCP proxies which are responsible for surprisingly low delays.

5.5.3 Low Queuing Delay: TCP Proxies

Similar to what we saw with Base RTTs in § 5.4.4, around 10% of cellular Netflix sessions display very low peak queuing delay that is less than 10ms, lower than wired or Wi-Fi connected sessions at the same percentiles. This is surprising, since it contradicts the general belief that cellular links have high RTT variation [81]. We believe that these occur due to the same root cause as the low Base RTT cellular sessions: the presence of TCP proxies. Classifying the sessions using the same proxy detection technique described in §5.4.4, we find that almost 90% of low peak queuing delay cellular sessions (Fig 5.9) were tagged as traversing a TCP proxy. The fact that this TCP proxy detection technique was able to isolate a significant majority of both low Base RTT (Fig 5.4) and low peak queuing delay sessions lends further credence to the premise these low Base RTTs and peak queuing delays are likely due to TCP proxies.

5.5.4 Time of Day Effects

Many past studies have found that latency on the Internet varies with time of day [37, 5, 118], likely due to greater usage during certain hours of the day resulting in more queuing. We therefore examine the differences in peak observed queuing delay during “busy" vs “non-busy" hours. We find that only specific client network types in a few countries show a statistically significant increase in peak observed queuing delay of at least 10ms, and that just one country+client network combination shows a statistically significant increase of 50ms, during busy hours as compared to non-busy hours.

Methodology: We divide each day into 3-hour segments, and for a given country identify the 3-hour segment with the most sessions as the busy period for that country, and the segment with the least as the non-busy period. For most countries, we find that the non-busy periods tend to be between 1 AM and 6 AM, while the busy periods tend to be between 6 PM and 12 AM in the local timezone. We then apply the Mann–Whitney U test [166] with a significance level of $p < 0.05$ to determine if there is a statistically significant increase in peak queuing delay between sessions that started during busy

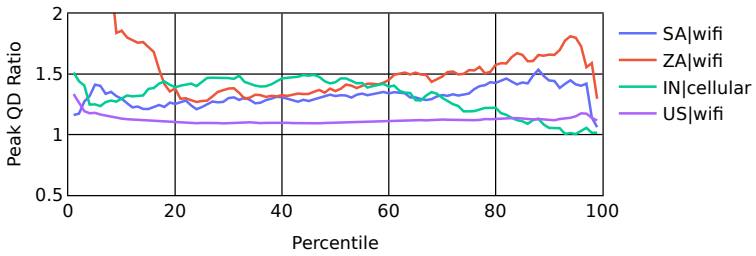


Figure 5.10: The ratio of the peak queuing delay in busy vs non-busy sessions at corresponding percentiles for various country+client network combinations. The SA (Saudi Arabia), ZA (South Africa) and IN (India) sessions show a statistically significant increase in queuing delay of at least 20ms (50ms for IN) during busy hours, while the US link does not.

periods vs non-busy periods for a given client network type and country. [5]. We test for increases in peak queuing delay in busy periods compared to non-busy periods of at least 10ms, 20ms etc. up to 100ms. To ensure robust conclusions, we limit our analysis to the 31 country+client network pairs that have at least 500 sessions during non-busy hours. The 31 pairs are spread across 25 countries, and consist of 25 Wi-Fi, 4 wired and 2 cellular links.

5.5.5 Results

We find that out of the 31 pairs that met our 500 non-busy session requirement, only Wi-Fi links in Saudia Arabia and South Africa, and cellular links in India, showed a statistically significant ($p < 0.05$) increase in queuing delay of at least 10ms. Of these, only the cellular links in India showed statistically significant increases up to 50ms, while the Wi-Fi links in Saudi Arabia and South Africa only showed statistically significant increases of 20ms. We take a closer look at the ratio of busy to non-busy peak queuing delays at various percentiles in Fig 5.10, and contrast it against Wi-Fi links in the US which don't show statistically significant evidence of contention. As expected, we can see the queuing delay ratio is much higher for those links that showed a statistically significant increase in queuing delay. The IN|cellular link shows an interesting trend—at lower percentiles the increase in queuing delay is substantial, but at higher percentiles the difference is less pronounced. This could potentially be due to the higher percentiles consisting of poor cellular connections that show high queuing delay regardless of time of day.

There are a number of potential explanations for why these country+client network pairs show increased peak queuing delay. It is possible that during busy periods more low throughput or oversubscribed links are active, such as public Wi-Fi access points at coffee shops, increasing the likelihood that Netflix traffic exceeds the available path capacity and causes more queuing. We verified this hypothesis by comparing the client-reported

[5]We perform our analysis at a client network-country level instead of just country level to normalize for the variation in peak queuing delay across client network type

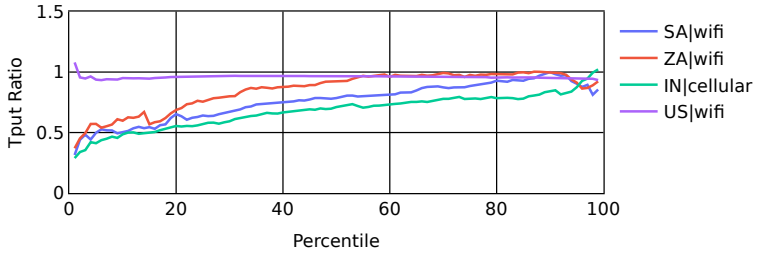


Figure 5.11: The ratio of average throughput reported by sessions from busy periods to those from non-busy periods at corresponding percentiles for the country+client network pairs that showed statistically significant increased in queuing delay (Fig 5.10). The US serves as a comparison point showing very little variation between busy and non-busy periods.

average throughput [6] reported by Netflix sessions during busy and non-busy periods for the 31 country+client network pairs. We found that pairs do indeed report lower average throughput, as shown in Fig 5.11. It is also possible that the paths used during busy periods have larger queues and therefore a higher ceiling on the peak observed queuing delay, but it is unfortunately difficult to verify this hypothesis without data from the bottlenecks themselves.

The finding of no statistically-significant increase in peak queuing delay of at least 10ms on most country+client network pairs analyzed should also be interpreted in the context of Netflix's CDN and the lightweight nature of video streaming traffic. Netflix CDN nodes are located relatively close to users, minimizing the number of potential bottlenecks. A service with a CDN further away from users might be more likely to face contention as it traverses more links on its path to the user. Past work that studied time-of-day effects from throughput measurements from Ookla's SpeedTest, which we believe uses similarly edge-adjacent test servers due to its low RTT measurements (§5.4), also reported only slightly decreased throughput with time of day [118]. In addition, video streaming path capacity requirements of around 15 Mbps for the highest 4K quality video can often be low compared to the available path capacity, preventing it from saturating the link for long durations. Further, Netflix's Adaptive-Bitrate algorithm (ABR) can prevent link saturation, as it automatically reduces video quality and consequently capacity requirements in response to persistent insufficient throughput. For these reasons, a bulk download service capable of fully utilizing a given link for long stretches of time, for example, would be more likely to experience contention than Netflix.

[6] This is the average of the time taken to transfer individual video chunks to the client

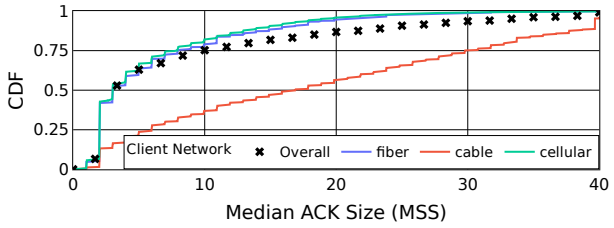


Figure 5.12: The CDF of the median ACK size witin a session on various ISP link types in the US. Cable links show significantly more ACK aggregation than fiber or cellular links.

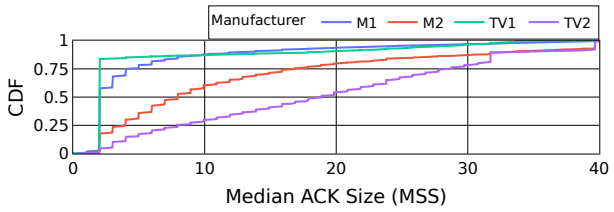


Figure 5.13: The CDF of median ACK sizes by device manufacturer, for two Mobile (M1 and M2) and two FED (TV1 and TV2) manufacturers. To normalize/limit the ACK aggregation due to the network path, these results are from sessions that were connected by Wi-Fi to non-cable ISPs. The extent of ACK aggregation clearly varies by device manufacturer, with devices from M2 and TV2 experiencing significantly more ACK aggregation than those from M1 and TV1. This could be caused by a variety of device specific configurations, such as Large Receive Offload (LRO) on the client device, long polling intervals to the device NIC/selective interrupt coalescing resulting in larger bursts of packets being received, or a change to the typical 2 MSS delayed ACK threshold on the device's TCP stack.

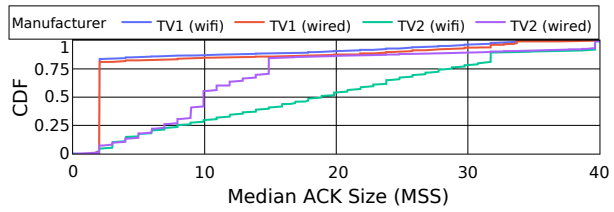


Figure 5.14: The CDF of the median ACK size on wired vs Wi-Fi client networks for devices from two distinct FED manufacturers on non-Cable connections. Using a Wi-Fi network does not change the low ACK aggregation levels experienced by devices from TV1, but increases ACK aggregation drastically for TV2.

5.6 ACK Aggregation Levels

ACK aggregation is the phenomenon due to which a sender might receive a single TCP Acknowledgement from the receiver that acknowledges the sequence space of multiple transmitted TCP segments. This can occur due to optimizations such as delayed ACKs on receivers [13], and techniques that "decimate" or coalesce multiple ACKs into a single ACK [30] in equipment like cable modems [134], wireless networks [77], and satellite links

[9]. ACK compression, when multiple distinct ACKs arrive at the sender in a short period of time due to queuing on the ACK path, is a related but distinct phenomenon that is out of scope for this analysis.

ACK aggregation is known to interfere with CCAs like BBR which perform ACK-rate based path capacity estimation [30], where large ACKs can lead to over(or under)-estimating the delivery rate, and unless such samples are discarded, they can lead to send rates that exceed (or fall short of) the path capacity. It contributes to jitter in RTT estimates, and past work on theoretically verifying CCA properties has shown that jitter from high levels of ACK aggregation would make it fundamentally impossible to prevent starvation in end-to-end congestion control [10]. This is a serious enough issue for the latest version of BBR to attempt to measure and account for the ACK aggregation seen on its path [30]. Further, for window-based CCAs, large ACKs can lead to sending large bursts of data, which is more likely to fill queues and cause latency spikes for real-time services such as video conferencing or cloud gaming sharing the bottleneck link.

However, despite its significance, ACK aggregation is almost always absent in state-of-the-art CCA testbeds and evaluations [29, 57, 122, 121], and Internet measurement studies [116, 44, 118, 62]. Pantheon [170] serves as an exception, and tests a few "pathological" paths with high levels of ACK aggregation—where a single ACK is received once every 100ms or 200ms [163].

By quantifying the extent and frequency of ACK aggregation at scale, and examining how ISP types, device types and client networks affect it, we aim to provide valuable guidelines for future CCA evaluations and testbeds that seek to include ACK aggregation as part of their emulated or simulated testbeds.

Estimation technique: We quantify ACK aggregation as the median ACK size, where the ACK size is defined as the number of bytes that is newly acknowledged by a given ACK, divided by the Maximum Segment Size (MSS). As the MSS represents a common amount of data payload in packets containing TCP segments, dividing the median ACK size by the MSS approximates how many packets are acknowledged by each ACK. Specifically, we use cumulative ACK sizes, as ACK decimation relies on the fact that TCP ACKs are cumulative [14]. We exclude ACKs that contain a Selective Acknowledgment (SACK) because ACK aggregation mechanisms may avoid decimating such ACKs, distorting our measurements. In addition, we exclude ACKs that are received immediately after an ACK with a SACK, which may be large due to the end of a recovery episode but unrelated to on-path ack aggregation. By isolating ACKs that reflect data transfer without evidence of loss, we gain a more accurate understanding of typical aggregation behavior. Note that our estimates include the total aggregation behavior, including on-path ACK aggregation, as well as well as due to potential device effects, such as client TCP stacks sending larger ACKs than the standard 2 MSS [41]. ACKs being lost, rather than explictly decimated, on the path from the receiver is also perceived as ACK aggregation by senders.

5.6.1 Results

We find extensive ACK aggregation on the paths traversed by Netflix, with the median ACK size for some sessions as high as 40MSS on Cable links (Fig 5.12).[7] Fiber and Cellular links show lower ACK aggregation levels that are similar (Fig 5.12).

Much to our surprise, we discovered that the device itself plays a key role in determining the extent of ACK aggregation it should expect. Specifically, we found that devices from the same manufacturer, in the case of both FEDs and Mobile devices, showed similar ACK aggregation behaviors. As shown in Fig 5.13, devices from certain manufacturers experience more ACK aggregation than others. Changing the client network from wired to Wi-Fi can increase the ACK aggregation on devices from certain manufacturers, but not on others, as shown in Fig 5.14. While we lack the instrumentation to verify the cause of this behavior, we hypothesize that it likely has to do with TCP or OS optimizations such as Large Receive Offload (LRO), long polling intervals to the NIC/interrupt coalescing, all of which serve to reduce overheads associated with networking and improve performance and power consumption.

These findings suggest the need to consider the expected range of ISP types, client devices, and client networks when configuring a CCA testbed, so that various expected ACK aggregation levels can be accounted for. In the case of physical testbeds, it also urges caution in extrapolating the behavior from a single test client device to other devices with different ACK aggregation properties—testbeds that use actual client devices should test a variety of devices with different ACK aggregation properties.

Lastly, while still biased by Netflix's user demographic, ACK aggregation is likely less influenced by the nature of Netflix's edge-adjacent CDN (compared to Base RTT and peak queuing delay), and therefore even more generalizable to most CCA and Internet testbeds. This is based on our results which show that ACK aggregation depends primarily on the last mile—the user's ISP network type, client network, and client device—which should be shared across most paths to the user.

[7]The ceiling of 40MSS is due to server-side configuration and not an inherent property of the network.

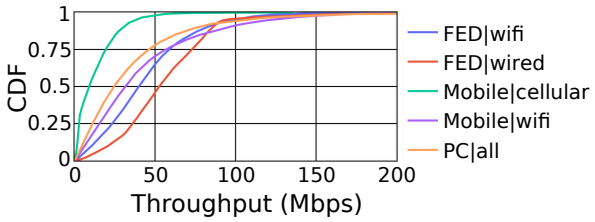


Figure 5.15: The CDF of client-reported throughput experienced by sessions from various device/home network combinations. Wired FEDs achieve the highest throughput, while cellular connections achieve the least.

5.7 Throughput

In this section, we analyze the client-reported throughput, which is the rate at which the client received data. While speed tests are designed to saturate the link and accurately measure the network's path capacity, our results reflect the data rate that Netflix users actually experience, which can be often limited by Netflix's sending rate (due to its CCA NewReno, and Netflix's video streaming workload) rather than the network itself. For example, Netflix requires just 15 Mbps of throughput to stream the highest quality 4K video, and 5 Mbps for HD video. Consequently, users are often able to enjoy the highest quality video that their device supports, even when the throughput obtained does not fully utilize the network's path capacity as reported by most speed tests.

The client-reported average throughput for a session is calculated by dividing the total size of all chunks received from Netflix CDN servers by the active download time for those chunks. This calculation excludes the "off" periods when the client video buffer is full and no data is fetched, a typical pattern in streaming video traffic [138]. It provides a lower-bound estimate of the path capacity of the link, as we have found that Netflix may not consistently saturate the link during all chunk downloads.

5.7.1 Results by Device & Network Type

Figure 5.15 describes the throughput observed in sessions from various device types and home network combinations. Consistent with past studies measuring the effect of the client network on throughput [118], wired FEDs achieve higher throughput than wireless FEDs or mobile devices, while cellular links show the lowest throughput. This lower throughput on cellular links can be partially attributed to Netflix's data-saving measures, which limit the maximum available video bitrate to 750 Kbps. For comparison, 4K video streaming typically requires at least 15 Mbps, and HD requires 5 Mbps.

Ookla's Speed Index [140] reports the median fixed broadband download speed as 101 Mbps, and the median cellular download speed as 92 Mbps. A weighted average, by number of tests, of the per-country download speeds reported by Cloudflare Radar's speed test is 168 Mbps. Clearly, Netflix's client-reported throughput is lower than those values. One reason for this is that speed tests are designed to saturate paths to reliably measure

path capacity, using strategies such as multiple flows [140]. On the other hand, Netflix uses a single NewReno flow to download video data on all sessions originating from its native TV, Android or iOS apps (which account for the bulk of Netflix sessions). Internal evaluations have also shown that Netflix's throughput can also be limited by the bitrate selected by its ABR algorithm, which is typically less than 20 Mbps. Past studies have shown that M-Lab speed test results, which also use a single TCP flow, suffer from similar limitations, showing a 20-50% reduction in measured download speed compared to Ookla's speed tests for the same bandwidth subscription tier [118].

5.8 TCP Proxy Detection Details

This section contains specific implementation details about the proxy detection technique for interested readers.

Manufacturer	**RWND: 20th, 80th-ptile**
Apple	131 KB, 131 KB
Google	67 KB, 88 KB
Honor	67 KB, 74 KB
Huawei	82 KB, 90 KB
Oppo	67 KB, 88 KB
Oneplus	67 KB, 77 KB
Tecno	67 KB, 74 KB
Xiaomi	67 KB, 89 KB
Motorola	67 KB, 127 KB
Realme	67 KB, 88 KB
Samsung	68 KB, 89 KB
Vivo	67 KB, 88 KB

Table 5.1: The initial RWNDs reported by devices using Netflix via cellular connections, aggregated by manufacturer. We leverage the consistency of their reported RWNDs between the 20th and 80th percentile, and the fact that TCP proxies likely advertise a different initial RWND, to build an RWND-based TCP Proxy detection technique.

To detect potential TCP proxies on sessions, we leverage insights from past work on TCP proxy detection [50] which noted that TCP proxy-reported initial RWNDs can vary from device-reported initial RWNDs. We find that devices from a given manufacturer often show a consistent initial RWND between the 20th and 80th percentile, as shown in Table 5.1. We categorize sessions that deviate from this initial RWND by more than 10KB as using a *suspected proxy*. We tag Autonomous Systems (ASs) with more than 10% of their sessions using a *suspected proxy* as an *AS with a suspected proxy*. We then tag sessions which are both using a *suspected proxy* and originating from an *AS with a suspected proxy* as sessions using TCP proxies.

We also find that while non-proxied sessions advertise RWNDs between 50KB to 150KB, almost half of proxied-sessions advertise initial RWNDs greater than 1000KBs. This is likely because the middleboxes that run TCP proxies are configured to dedicate more memory to TCP connections, and consequently advertise a larger RWND than commodity mobile devices. This combined evidence makes us believe that our RWND-based proxy detection technique correctly identifies TCP proxies in most of the cellular sessions in our dataset, and consequently, that the low RTT cellular sessions are likely caused by TCP proxies.

5.9 Chapter Summary

In this work, we presented a number of passive path property estimates from a global video streaming service, Netflix, with an expansive CDN presence. We found that for accurate testing of Netflix's performance in the real world, the Base RTTs, queuing delays, the presence of TCP proxies, and ACK aggregation levels used by Netflix sessions, would have to differ significantly from those used by a large fraction of CCA and Internet evaluations. We hope that this perspective from a hyperscaler helps enrich the next generation of CCA and Internet service evaluations to emulate even more representative paths for testing, further bridging the gap between emulated and simulated testbeds and A/B tests. We expect this work to lay the foundations for more large service owners to contribute their own views of the Internet so we can build a much better understanding of network conditions needed to realistically test an Internet that is diverse in its users, their networks, and the services they consume.

Chapter 6

Conclusion

6.1 Charted Territory

> "All right, Doc. What's going on? Where are we? … When are we?"
>
> — *Marty McFly, Back to the Future Part II* (1989)

In §3 and §4, we demonstrated the necessity of expanding typical experimental setups for CCA evaluations to include both congestion scenarios at scale, and the evaluation of CCAs to include the services they are to be deployed in. We saw how BBRv1, which was touted as a fair, low delay, high throughput solution for CCA on the Internet if uniformly deployed, results in catastrophic fairness outcomes when there is congestion at the core of the Internet, which past work has shown occurs with surprising frequency on certain inter-domain links. Through Prudentia, we saw that YouTube, which uses the much maligned BBR, was actually one of the most fair services in practice due to its bitrate ladder and application-limited nature. It also showed us how the prevalence of multi-flow services turned decades of per-flow fairness research on its head, suggesting the need for potential browser or OS-level control. But more than anything, it showed us the unknown-unknowns we would have never discovered otherwise, underscoring the need for service-level evaluations when testing CCAs.

In §5, we addressed the question of how closely typical evaluation environments align with the emerging trend of CDNs using user-adjacent infrastructure, by a case study of the network conditions experienced by a real Internet service, Netflix. We found that Netflix experiences lower RTTs, higher queuing delays, and more ACK aggregation than evaluated in many experimental testbeds. This suggested the need for more service owners to take a closer look at the network conditions their users experience, and share them with the

community at large, so we can all be sure that we are testing in network conditions that are actually relevant.

In summary, this thesis shows that keeping Internet performance evaluation in emulated environments up-to-date requires testing full-stack services, not just CCAs, across both residential and core Internet link settings. Moreover, the specific network conditions we choose to perform these evaluations in must mirror those real services face, lest the victories we proclaim be illusory, and the failures we dread exist only in theory.

6.2 Future Expeditions

> "There's always money in the banana stand."
>
> — *George Bluth Sr.*, Arrested Development

This thesis lays the groundwork for a number of directions of continued investigation to further improve the sphere of Internet performance evaluations. This section is intended come by the way of a budding, starry-eyed PhD student or researcher eager to take up the mantle of realistic, dependable CCA and Internet service evaluations.

Expanding the parameter space: While we explored different ranges of common path properties like RTT, buffer sizes, and path capacity, we also discovered the neccessity for future evaluations to include less commonly considered path properties like ACK aggregation. There are a number of other potential path properties that need to be studied, such as path capacity variation, jitter, packet re-ordering and random packet loss (the necessity of some of which Pantheon [170] has established, but their use in other testbeds has not been widespread). These extent and prevalence of these properties as experienced by users must be measured, and given their limited presence in network evaluations, new experiments conducted to understand their impact on CCAs and services. Additionally, while Netflix is a major provider of video streaming on-demand content, other providers can experience different network conditions due to differing infrastructure and user demographics. The service type plays an important role too—a video conferencing call across the globe will experiences hundreds of milliseconds of RTT irrespective of the infrastructure of the the service operator. Therefore the network conditions experienced by the users of these other services must be studied and accounted for too.

Constraining the parameter space: A large portion of these thesis has encouraged the expansion of the testbed parameter space—testing in larger ranges of RTTs, path capacities, buffer sizes and with services in addition to CCAs. However, while it might be tempting to test all possible values of all potentially relevant path properties, keeping such evaluations feasible inevitably require constraining the parameter space to the most relevant values. Path profiles—the idea that certain values of path properties are likely to co-occur with specific values of other path properties, such as low bandwidth links being

correlated high jitter and RTT—would be an ideal line of investigation towards further constraining this space.

Searching the parameter space: Once we have expanded and constrained the parameter space as necessary, there is still the question of what granularity of values to test within this space. For example, should we test with RTT values in intervals of 10ms or 1ms? The answer likely depends on the service or CCA—we would not want to a test a CCA with a number of network conditions that all yield the same performance outcome, but rather the ones most likely to show a change in behavior. For example, NewReno's throughput has a multiplicative relationship with RTT [97], with a change of 1m->2ms representing the same decrease in its window growth rate as 20ms->40ms. An interesting empirical approach to this could involve a gradient-descent-esque exploration, where some initial tests with different values in the parameter space determine the next parameter values most likely to cause the greatest difference in the CCA or service's performance. These parameter values could then be tested in the next iteration. This will likely also depend on what performance metric is being evaluated: while BBR's throughput and delay might not be affected significantly by an increase in RTT, it exhibits intra-CCA unfairness at higher RTTs.

6.3 Distant Horizons

> "You've got the makings of greatness in you, but you've gotta take the helm and chart your own course!"
>
> — *John Silver*, *Treasure Planet* (2002)

An important parallel to empirical evaluations has been recent developments in provably verifying congestion control performance [10]. While validating *existing* applications or CCA implementation comes with the overhead and associated errors of translating real-world implementations into models compatible with the verification framework, the same framework can be used to *synthesize* provably performant implementations of CCAs [2] and applications. The aspect in question then is the realism of the network settings used in the verification model: synthesizing a CCA that performs well in all network conditions likely comes at the cost of a better CCA that performs well only in common network conditions, and synthesizing a CCA that does not cover the network conditions experienced by users in practice can potentially result in poor performance in practice. As a result, measurement studies from actual user traffic to services they consume will be critical going forward; this will not just accurately parameterize empirical testbeds, but also inform CCA designers about the network conditions they should be designing for—both human and machine.

6.4 Parting Words

"And that's why you always leave a note." [1]

— J. Walter Weatherman
Arrested Development (2004)

There was a time when there were simple, elegant ways to reason about CCA behavior. An entire class of CCAs that used the Additive/Multiplicative Increase/Decrease paradigm could have their performance and fairness properties predicted analytically [35]. A simple model for NewReno [97, 117] held up for decades, and with the focus on certain caveats, even at the scale of the core of the Internet. On the other hand, CCAs like BBR have had multiple analytical models derived [162, 108], all of which fail to predict its unfairness to other BBR flows, even when all the flows are at the same RTT. This is no fault of the modelers—these new CCAs are so complex that by the time anyone truly understands them the next version of the same CCA is already released. As a result, the future of CCA testing and design both likely lie in the hands of machines—empirical evaluations to verify service performance in deployment, and synthesis by automated reasoning techniques to generate CCAs and application-control loops with verifiable performance guarantees [2]. These approaches are most likely to keep up with the constantly changing landscape of the Internet, ensuring the performance of our services does not let us down when we most need them.

[1] https://aphilip.cc/always-leave-a-note

Bibliography

[1] I. Abdeljaouad, H. Rachidi, S. Fernandes, and A. Karmouch. "Performance analysis of modern TCP variants: A comparison of Cubic, Compound and New Reno". In: *2010 25th Biennial Symposium on Communications*. 2010, pp. 80–83. DOI: 10.1109/BSC.2010.5472999.

[2] Anup Agarwal, Venkat Arun, Devdeep Ray, Ruben Martins, and Srinivasan Seshan. "Towards provably performant congestion control". In: *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*. Santa Clara, CA: USENIX Association, Apr. 2024, pp. 951–978. ISBN: 978-1-939133-39-7. URL: https://www.usenix.org/conference/nsdi24/presentation/agarwal-anup.

[3] Aditya Akella, Srinivasan Seshan, and Anees Shaikh. "An Empirical Evaluation of Wide-Area Internet Bottlenecks". In: *Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement*. IMC '03. New York, NY, USA: Association for Computing Machinery, 2003, 101–114. ISBN: 1581137737. DOI: 10.1145/948205.948219. URL: https://doi.org/10.1145/948205.948219.

[4] Mohammad Alizadeh, Adel Javanmard, and Balaji Prabhakar. "Analysis of DCTCP: Stability, Convergence, and Fairness". In: 39.1 (June 2011), 73–84. ISSN: 0163-5999. DOI: 10.1145/2007116.2007125. URL: https://doi.org/10.1145/2007116.2007125.

[5] Daniel S.F. Alves and Katia Obraczka. "An Empirical Characterization of Internet Round-Trip Times". In: Q2SWinet '17. New York, NY, USA: Association for Computing Machinery, 2017, 23–30. ISBN: 9781450351652. DOI: 10.1145/3132114.3132123. URL: https://doi.org/10.1145/3132114.3132123.

[6] Guido Appenzeller, Isaac Keslassy, and Nick McKeown. "Sizing Router Buffers". In: *Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*. SIGCOMM '04. New York, NY, USA: Association for Computing Machinery, 2004, 281–292. ISBN: 1581138628. DOI: 10.1145/1015467.1015499. URL: https://doi.org/10.1145/1015467.1015499.

[7] HTTP Archive. *CDN | 2024 | The Web Almanac by HTTP Archive — almanac.httparchive.org*. https://almanac.httparchive.org/en/2024/cdn. [Accessed 15-05-2025]. 2024.

[8] Mahdi Arghavani, Haibo Zhang, David Eyers, and Abbas Arghavani. "SUSS: Improving TCP Performance by Speeding Up Slow-Start". In: ACM SIGCOMM '24. New York, NY, USA: Association for Computing Machinery, 2024, 151–165. ISBN: 9798400706141. DOI: 10.1145/3651890.3672234. URL: https://doi.org/10.1145/3651890.3672234.

[9] Vivek Arora, Narin Suphasindhu, John Baras, and Douglas Dillon. "Asymmetric internet access over satellite-terrestrial networks". In: *16th International Communications Satellite Systems Conference*. 1996, p. 1044.

[10] Venkat Arun, Mohammad Alizadeh, and Hari Balakrishnan. "Starvation in end-to-end congestion control". en. In: *Proceedings of the ACM SIGCOMM 2022 Conference*. Amsterdam Netherlands: ACM, Aug. 2022, pp. 177–192. ISBN: 9781450394208. DOI: 10.1145/3544216.3544223. URL: https://dl.acm.org/doi/10.1145/3544216.3544223 (visited on 09/21/2022).

[11] Venkat Arun and Hari Balakrishnan. "Copa: Practical Delay-Based Congestion Control for the Internet". In: *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*. Renton, WA: USENIX Association, Apr. 2018, pp. 329–342. ISBN: 978-1-939133-01-4. URL: https://www.usenix.org/conference/nsdi18/presentation/arun.

[12] Sachin Ashok, Sai Surya Duvvuri, Nagarajan Natarajan, Venkata N. Padmanabhan, Sundararajan Sellamanickam, and Johannes Gehrke. "IBox: Internet in a Box". In: *Proceedings of the 19th ACM Workshop on Hot Topics in Networks*. HotNets '20. New York, NY, USA: Association for Computing Machinery, 2020, 23–29. ISBN: 9781450381451. DOI: 10.1145/3422604.3425935. URL: https://doi.org/10.1145/3422604.3425935.

[13] Wikipedia Authors. *TCP Delayed Acknowledgement*. Dec. 2024. URL: https://en.wikipedia.org/wiki/TCP_delayed_acknowledgment.

[14] Hari Balakrishnan, Venkata N Padmanabhan, and Randy H Katz. "The effects of asymmetry on TCP performance". In: *Proceedings of the 3rd annual ACM/IEEE international conference on Mobile computing and networking*. 1997, pp. 77–89.

[15] Steven Bauer, David D. Clark, and William Lehr. "Understanding Broadband Speed Measurements". In: *TPRC 2010*. Available at SSRN: https://ssrn.com/abstract=1988332. Aug. 2010.

[16] *BBR v2: A Model-based Congestion Control*. https://datatracker.ietf.org/meeting/104/materials/slides-104-iccrg-an-update-on-bbr-00. Accessed: 2021-03-04. 2021.

[17] Neda Beheshti, Yashar Ganjali, Monia Ghobadi, Nick McKeown, and Geoff Salmon. "Experimental Study of Router Buffer Sizing". In: *Proceedings of the 8th ACM SIGCOMM Conference on Internet Measurement*. IMC '08. New York, NY, USA: Association for Computing Machinery, 2008, 197–210. ISBN: 9781605583341. DOI: 10.1145/1452520.1452545. URL: https://doi.org/10.1145/1452520.1452545.

[18] Dimitri P. Bertsekas and Robert G. Gallager. *Data Networks, Second Edition*. Prentice Hall, 1992.

[19] *BESS: A Software Switch*. https://github.com/NetSys/bess. Accessed: 2021-03-04. 2021.

[20] *Betteridge's law of headlines - Wikipedia*. https://en.wikipedia.org/wiki/Betteridge%27s_law_of_headlines.

[21] *Big Buck Bunny*. https://peach.blender.org/about/.

[22] Bob Lantz, Brandon Heller, Nikhil Handigol, Vimal Jeyakumar and other contributors. *Mininet: An Instant Virtual Network on Your Laptop (or Other PC)*. https://mininet.org/. Accessed: 22 July 2025. 2025.

[23] Thomas Bonald. "Comparison of TCP Reno and TCP Vegas: efficiency and fairness". In: *Performance Evaluation* 36 (1999), pp. 307–332.

[24] Thomas Bonald, Laurent Massoulié, Alexandre Proutiere, and Jorma Virtamo. "A queueing analysis of max-min fairness, proportional fairness and balanced fairness". In: *Queueing systems* 53 (2006), pp. 65–84.

[25] Lawrence S. Brakmo, Sean W. O'Malley, and Larry L. Peterson. "TCP Vegas: New Techniques for Congestion Detection and Avoidance". In: *SIGCOMM Comput. Commun. Rev.* 24.4 (Oct. 1994), 24–35. ISSN: 0146-4833. DOI: 10.1145/190809.190317. URL: https://doi.org/10.1145/190809.190317.

[26] Patrick Brown. "Resource sharing of TCP connections with different round trip times". In: *Proceedings ieee infocom 2000. conference on computer communications. nineteenth annual joint conference of the ieee computer and communications societies (cat. no. 00ch37064)*. Vol. 3. IEEE. 2000, pp. 1734–1741.

[27] Yi Cao, Arpit Jain, Kriti Sharma, Aruna Balasubramanian, and Anshul Gandhi. "When to use and when not to use BBR: An empirical analysis and evaluation study". en. In: *Proceedings of the Internet Measurement Conference*. Amsterdam Netherlands: ACM, Oct. 2019, pp. 130–136. ISBN: 9781450369480. DOI: 10.1145/3355369.3355579. URL: https://dl.acm.org/doi/10.1145/3355369.3355579 (visited on 09/20/2022).

[28] Neal Cardwell, Yuchung Cheng, C. Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. "BBR: Congestion-Based Congestion Control". In: *ACM Queue* 14, September-October (2016), pp. 20 –53. URL: http://queue.acm.org/detail.cfm?id=3022184.

[29] Neal Cardwell, Yuchung Cheng, C. Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. "BBR: congestion-based congestion control". In: *Commun. ACM* 60.2 (Jan. 2017), 58–66. ISSN: 0001-0782. DOI: 10.1145/3009824. URL: https://doi.org/10.1145/3009824.

[30] Neal Cardwell, Yuchung Cheng, Soheil Hassas Yeganeh, and Van Jacobson. *BBR Congestion Control*. Internet-Draft. Work in Progress. Internet Engineering Task Force, Oct. 2024. URL: https://datatracker.ietf.org/doc/html/draft-ietf-ccwg-bbr (visited on 01/29/2025).

[31] Gaetano Carlucci, Luca De Cicco, Stefan Holmer, and Saverio Mascolo. "Congestion control for web real-time communication". In: *IEEE/ACM Transactions on Networking* (2017).

[32] Gaetano Carlucci, Luca De Cicco, and Saverio Mascolo. "HTTP over UDP: an experimental investigation of QUIC". In: *Proceedings of the 30th Annual ACM Symposium on Applied Computing*. SAC '15. New York, NY, USA: Association for Computing Machinery, 2015, 609–614. ISBN: 9781450331968. DOI: 10.1145/2695664.2695706. URL: https://doi.org/10.1145/2695664.2695706.

[33] Yanpei Chen, Rean Griffith, Junda Liu, Randy H. Katz, and Anthony D. Joseph. "Understanding TCP Incast Throughput Collapse in Datacenter Networks". In: *Proceedings of the 1st ACM Workshop on Research on Enterprise Networking*. WREN '09. New York, NY, USA: Association for Computing Machinery, 2009, 73–82. ISBN: 9781605584430. DOI: 10.1145/1592681.1592693. URL: https://doi.org/10.1145/1592681.1592693.

[34] Yuchung Cheng, Neal Cardwell, Nandita Dukkipati, and Priyaranjan Jha. *The RACK-TLP Loss Detection Algorithm for TCP*. RFC 8985. Feb. 2021. DOI: 10.17487/RFC8985. URL: https://www.rfc-editor.org/info/rfc8985.

[35] Dah-Ming Chiu and Raj Jain. "Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks". In: *Comput. Netw. ISDN Syst.* 17.1 (June 1989), 1–14. ISSN: 0169-7552. DOI: 10.1016/0169-7552(89)90019-6. URL: https://doi.org/10.1016/0169-7552(89)90019-6.

[36] Cloudflare Blog. *Introducing the Cloudflare Radar Internet Quality Page*. https://blog.cloudflare.com/introducing-radar-internet-quality-page/. [Online; accessed 14-May-2025]. May 2025.

[37] Amogh Dhamdhere, David D. Clark, Alexander Gamero-Garrido, Matthew Luckie, Ricky K. P. Mok, Gautam Akiwate, Kabir Gogia, Vaibhav Bajpai, Alex C. Snoeren, and Kc Claffy. "Inferring persistent interdomain congestion". In: *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*. SIGCOMM '18. New York, NY, USA: Association for Computing Machinery, 2018, 1–15. ISBN: 9781450355674. DOI: 10.1145/3230543.3230549. URL: https://doi.org/10.1145/3230543.3230549.

[38] Mo Dong, Tong Meng, Doron Zarchy, Engin Arslan, Yossi Gilad, Brighten Godfrey, and Michael Schapira. "PCC Vivace: Online-Learning Congestion Control". In: *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*. Renton, WA: USENIX Association, Apr. 2018, pp. 343–356. ISBN: 978-1-939133-01-4. URL: https://www.usenix.org/conference/nsdi18/presentation/dong.

[39] Nandita Dukkipati, Neal Cardwell, Yuchung Cheng, and Matt Mathis. *Tail Loss Probe (TLP): An Algorithm for Fast Recovery of Tail Losses*. Internet-Draft draft-dukkipati-tcpm-tcp-loss-probe-01. Work in Progress. Internet Engineering Task Force, Feb. 2013. 20 pp. URL: https://datatracker.ietf.org/doc/draft-dukkipati-tcpm-tcp-loss-probe/01/.

[40] Dmitry Duplyakin, Robert Ricci, Aleksander Maricq, Gary Wong, Jonathon Duerig, Eric Eide, Leigh Stoller, Mike Hibler, David Johnson, Kirk Webb, Aditya Akella, Kuangching Wang, Glenn Ricart, Larry Landweber, Chip Elliott, Michael Zink, Emmanuel Cecchet, Snigdhaswin Kar, and Prabodh Mishra. "The Design and Operation of CloudLab". In: *Proceedings of the USENIX Annual Technical Conference (ATC)*. July 2019, pp. 1–14. URL: https://www.flux.utah.edu/paper/duplyakin-atc19.

[41] W Eddy. *Rfc 9293: Transmission control protocol (tcp)*. 2022.

[42] T. En-Najjary and G. Urvoy-Keller. "PPrate: A Passive Capacity Estimation Tool". In: *2006 4th IEEE/IFIP Workshop on End-to-End Monitoring Techniques and Services*. 2006, pp. 82–89. DOI: 10.1109/E2EMON.2006.1651283.

[43] Viktor Farkas, Balázs Héder, and Szabolcs Nováczki. "A Split Connection TCP Proxy in LTE Networks". In: *EUNICE Open European Summer School*. 2012. URL: https://api.semanticscholar.org/CorpusID:1948515.

[44] Federal Communications Commission. *Measuring Broadband America*. https://www.fcc.gov/general/measuring-broadband-america. Accessed: Jan 29, 2025. 2025.

[45] Ferenc Fejes, Gergő Gombos, Sándor Laki, and Szilveszter Nádas. "Who will Save the Internet from the Congestion Control Revolution?" In: *Proceedings of the 2019 Workshop on Buffer Sizing*. BS '19. Association for Computing Machinery, 2020. ISBN: 9781450377454. DOI: 10.1145/3375235.3375242. URL: https://doi.org/10.1145/3375235.3375242.

[46] Sally Floyd, Mark Handley, Jitendra Padhye, and Jörg Widmer. "Equation-Based Congestion Control for Unicast Applications". In: *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*. SIGCOMM '00. Stockholm, Sweden: Association for Computing Machinery, 2000, 43–56. ISBN: 1581132239. DOI: 10.1145/347059.347397. URL: https://doi.org/10.1145/347059.347397.

[47] Yashar Ganjali and Nick McKeown. "Update on Buffer Sizing in Internet Routers". In: *SIGCOMM Comput. Commun. Rev.* 36.5 (2006), 67–70. ISSN: 0146-4833. DOI: 10.1145/1163593.1163605. URL: https://doi.org/10.1145/1163593.1163605.

[48] Jim Gettys and Kathleen Nichols. "Bufferbloat: dark buffers in the internet". In: *Communications of the ACM* 55.1 (2012), pp. 57–65.

[49] Phillipa Gill, Christophe Diot, Lai Yi Ohlsen, Matt Mathis, and Stephen Soltesz. *M-Lab: User initiated Internet data for the research community*. 2022.

[50] Utkarsh Goel, Moritz Steiner, Mike P. Wittie, Martin Flack, and Stephen Ludin. "Detecting Cellular Middleboxes Using Passive Measurement Techniques". In: *Passive and Active Measurement - 17th International Conference, PAM 2016, Heraklion, Greece, March 31 - April 1, 2016. Proceedings*. Ed. by Thomas Karagiannis and Xenofontas A. Dimitropoulos. Vol. 9631. Lecture Notes in Computer Science. Springer, 2016, pp. 95–107. DOI: 10.1007/978-3-319-30505-9_8. URL: https://doi.org/10.1007/978-3-319-30505-9_8.

[51] K.-I. Goh and A.-L. Barabási. "Burstiness and memory in complex systems". In: *EPL (Europhysics Letters)* 81.4 (2008), p. 48002. DOI: 10.1209/0295-5075/81/48002. URL: https://doi.org/10.1209/0295-5075/81/48002.

[52] Sishuai Gong, Usama Naseer, and Theophilus A Benson. "Inspector Gadget: A Framework for Inferring TCP Congestion Control Algorithms and Protocol Configurations." In: *Network Traffic Measurement and Analysis Conference*. 2020.

[53] Google. *Google Edge Network — peering.google.com*. https://peering.google.com/#/infrastructure. [Accessed 31-03-2025]. 2025.

[54] *Google Chrome*. https://www.google.com/chrome/.

[55] *Google LLC Peering Data*. https://www.peeringdb.com/net/433. Accessed: 2021-03-04. 2021.

[56] Andrei Gurtov, Tom Henderson, Sally Floyd, and Yoshifumi Nishida. *The NewReno Modification to TCP's Fast Recovery Algorithm*. RFC 6582. Apr. 2012. DOI: 10.17487/RFC6582. URL: https://rfc-editor.org/rfc/rfc6582.txt.

[57] Sangtae Ha, Injong Rhee, and Lisong Xu. "CUBIC: a new TCP-friendly high-speed TCP variant". en. In: *ACM SIGOPS Operating Systems Review* 42.5 (July 2008), pp. 64–74. ISSN: 0163-5980. DOI: 10.1145/1400097.1400105. URL: https://dl.acm.org/doi/10.1145/1400097.1400105 (visited on 09/20/2022).

[58] *Harware accelaration for VP9 not working*. Apr. 2017. URL: https://community.intel.com/.../td-p/291546.

[59] Help Center. *How to use Netflix on your Mac computer*. https://help.netflix.com/en/node/55764. Accessed: Feb. 02, 2024.

[60] Thomas R Henderson, Mathieu Lacage, George F Riley, Craig Dowell, and Joseph Kopena. "Network simulations with the ns-3 simulator". In: *SIGCOMM demonstration* 14.14 (2008), p. 527.

[61] Mario Hock, Roland Bless, and Martina Zitterbart. "Experimental evaluation of BBR congestion control". In: Oct. 2017, pp. 1–10. DOI: 10.1109/ICNP.2017.8117540.

[62] Toke Høiland-Jørgensen, Bengt Ahlgren, Per Hurtig, and Anna Brunstrom. "Measuring Latency Variation in the Internet". In: *Proceedings of the 12th International on Conference on Emerging Networking EXperiments and Technologies*. CoNEXT '16. New York, NY, USA: Association for Computing Machinery, 2016, 473–480. ISBN: 9781450342926. DOI: 10.1145/2999572.2999603. URL: https://doi.org/10.1145/2999572.2999603.

[63] *How CloudFront delivers content - Amazon CloudFront*. https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/HowCloudFrontWorks.html. [Accessed 31-03-2025].

[64] *htsim*. https://nrg.cs.ucl.ac.uk/mptcp/implementation.html. Accessed: 2020-03-12. 2020.

[65] *IETF - BBRv3*. https://datatracker.ietf.org/meeting/117/materials/slides-117-ccwg-bbrv3-algorithm-bug-fixes-and-public-internet-deployment-00.

[66] International Telecommunication Union. *ITU Workshop on Voice and Video Services Interoperability over Fixed-Mobile Hybrid Environments*. https://www.itu.int/en/ITU-T/Workshops-and-Seminars/conformity-interoperability/20150112/Documents/Summary-of-the-Workshop/Summary-of-the-event_V3.docx. Accessed: Feb. 02, 2024. Jan. 2015.

[67] *Internet transit - Wikipedia*. https://en.wikipedia.org/wiki/Internet_transit. [Accessed 31-03-2025].

[68] Milosh Ivanovich, Philip W. Bickerdike, and Jonathan C. Li. "On TCP performance enhancing proxies in a wireless environment". In: *IEEE Communications Magazine* 46.9 (2008), pp. 76–83. DOI: 10.1109/MCOM.2008.4623710.

[69] Raj Jain, Dah Ming Chiu, and Hawe WR. "A Quantitative Measure Of Fairness And Discrimination For Resource Allocation In Shared Computer Systems". In: *CoRR* cs.NI/9809099 (Jan. 1998).

[70] Rajendra K Jain, Dah-Ming W Chiu, William R Hawe, et al. "A quantitative measure of fairness and discrimination". In: *Eastern Research Laboratory, Digital Equipment Corporation, Hudson, MA* 21 (1984).

[71] Nathan Jay, Noga Rotman, Brighten Godfrey, Michael Schapira, and Aviv Tamar. "A Deep Reinforcement Learning Perspective on Internet Congestion Control". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, June 2019, pp. 3050–3059. URL: https://proceedings.mlr.press/v97/jay19a.html.

[72] Glenn Judd. "Attaining the Promise and Avoiding the Pitfalls of TCP in the Datacenter". In: *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*. Oakland, CA: USENIX Association, May 2015, pp. 145–157. ISBN: 978-1-931971-218. URL: https://www.usenix.org/conference/nsdi15/technical-sessions/presentation/judd.

[73] Arash Molavi Kakhki, Samuel Jero, David Choffnes, Cristina Nita-Rotaru, and Alan Mislove. "Taking a long look at QUIC: an approach for rigorous evaluation of rapidly evolving transport protocols". In: 62.7 (June 2019), 86–94. ISSN: 0001-0782. DOI: 10.1145/3330336. URL: https://doi.org/10.1145/3330336.

[74] Simon Kassing, Debopam Bhattacherjee, André Baptista Águas, Jens Eirik Saethre, and Ankit Singla. "Exploring the "Internet from Space" with Hypatia". In: *Proceedings of the ACM Internet Measurement Conference*. IMC '20. Virtual Event, USA: Association for Computing Machinery, 2020, 214–229. ISBN: 9781450381383. DOI: 10.1145/3419394.3423635. URL: https://doi.org/10.1145/3419394.3423635.

[75] Frank P Kelly, Aman K Maulloo, and David Kim Hong Tan. "Rate control for communication networks: shadow prices, proportional fairness and stability". In: *Journal of the Operational Research society* 49 (1998), pp. 237–252.

[76] Gary Kim. *CDNs Will Carry 72% of Total Internet Traffic by 2022*. https://ipcarrier.blogspot.com/2018/12/cdns-will-carry-72-of-total-internet.html. [Accessed 15-05-2025]. 2025.

[77] Hyogon Kim, Heejo Lee, Sangmin Shin, and Inhye Kang. "On the cross-layer impact of TCP ACK thinning on IEEE 802.11 wireless MAC dynamics". In: *IEEE Vehicular Technology Conference*. IEEE. 2006, pp. 1–6.

[78] Tomoki Kozu, Yuria Akiyama, and Saneyasu Yamaguchi. "Improving RTT Fairness on CUBIC TCP". In: *2013 First International Symposium on Computing and Networking*. 2013, pp. 162–167. DOI: 10.1109/CANDAR.2013.30.

[79] Christian Kreibich, Nicholas Weaver, Boris Nechaev, and Vern Paxson. "Netalyzr: Illuminating the edge network". In: *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*. 2010, pp. 246–259.

[80] Ike Kunze, Jan Rüth, and Oliver Hohlfeld. "Congestion Control in the Wild—Investigating Content Provider Fairness". In: *IEEE Transactions on Network and Service Management* 17.2 (2020), pp. 1224–1238. DOI: 10.1109/TNSM.2019.2962607.

[81] Natalie Larson, Džiugas Baltrunas, Amund Kvalbein, Amogh Dhamdhere, kc claffy, and Ahmed Elmokashfi. "Investigating Excessive Delays in Mobile Broadband Networks". In: *Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges*. AllThingsCellular '15. New York, NY, USA: Association for Computing Machinery, 2015, 51–56. ISBN: 9781450335386. DOI: 10.1145/2785971.2785980. URL: https://doi.org/10.1145/2785971.2785980.

[82] Soojeon Lee, Myungjin Lee, Dongman Lee, Hyungsoo Jung, and Byoung-Sun Lee. "TCPRand: Randomizing TCP payload size for TCP fairness in data center networks". In: *2015 IEEE Conference on Computer Communications (INFOCOM)*. 2015, pp. 1697–1705. DOI: 10.1109/INFOCOM.2015.7218550.

[83] Tong Li, Kai Zheng, Ke Xu, Rahul Arvind Jadhav, Tao Xiong, Keith Winstein, and Kun Tan. "TACK: Improving Wireless Transport Performance by Taming Acknowledgments". In: SIGCOMM '20. New York, NY, USA: Association for Computing Machinery, 2020, 15–30. ISBN: 9781450379557. DOI: 10.1145/3387514.3405850. URL: https://doi.org/10.1145/3387514.3405850.

[84] Yuliang Li, Rui Miao, Hongqiang Harry Liu, Yan Zhuang, Fei Feng, Lingbo Tang, Zheng Cao, Ming Zhang, Frank Kelly, Mohammad Alizadeh, and Minlan Yu. "HPCC: High Precision Congestion Control". In: *Proceedings of the ACM Special Interest Group on Data Communication*. SIGCOMM '19. Beijing, China: Association for Computing Machinery, 2019, 44–58. ISBN: 9781450359566. DOI: 10.1145/3341302.3342085. URL: https://doi.org/10.1145/3341302.3342085.

[85] Zhi Li, Anne Aaron, Ioannis Katsavounidis, Anush Moorthy, and Megha Manohara. *Toward A Practical Perceptual Video Quality Metric | Netflix TechBlog*. https://netflixtechblog.com/toward-a-practical-perceptual-video-quality-metric-653f208b9652. 2016.

[86] M-Lab. *BigQuery Schema - M-Lab*. https://shredtechular.github.io/m-lab.github.io/data/bq/schema/. [Accessed 12-05-2025]. 2025.

[87] M-Lab. *FAQ - M-Lab – Platform*. https://www.measurementlab.net/frequently-asked-questions/. [Accessed 12-05-2025]. 2025.

[88] M-Lab. *ndt5 Data - NDT (Network Diagnostic Tool)*. https://www.measurementlab.net/tests/ndt/ndt5/. [Accessed 12-05-2025]. 2025.

[89] M-Lab. *ndt7 Protocol - NDT (Network Diagnostic Tool)*. https://www.measurementlab.net/tests/ndt/ndt7/. [Accessed 12-05-2025]. 2025.

[90] Kyle MacMillan, Tarun Mangla, James Saxon, and Nick Feamster. "Measuring the Performance and Network Utilization of Popular Video Conferencing Applications". In: *Proceedings of the 21st ACM Internet Measurement Conference*. IMC '21. New York, NY, USA: Association for Computing Machinery, 2021, 229–244. ISBN: 9781450391290. DOI: 10.1145/3487552.3487842. URL: https://doi.org/10.1145/3487552.3487842.

[91] Kyle MacMillan, Tarun Mangla, James Saxon, Nicole P Marwell, and Nick Feamster. "A comparative analysis of ookla speedtest and measurement labs network diagnostic test (ndt7)". In: *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 7.1 (2023), pp. 1–26.

[92] Antonis Manousis, Rahul Anand Sharma, Vyas Sekar, and Justine Sherry. "Contention-Aware Performance Prediction For Virtualized Network Functions". en. In: *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*. Virtual Event USA: ACM, July 2020, pp. 270–282. ISBN: 9781450379557. DOI: 10.1145/3387514.3405868. URL: https://dl.acm.org/doi/10.1145/3387514.3405868 (visited on 09/20/2022).

[93] Gustavo Marfia, Claudio E. Palazzi, Giovanni Pau, Mario Gerla, M. Y. Sanadidi, and Marco Roccetti. "TCP *Libra* : Exploring RTT-Fairness for TCP". In: *NETWORKING 2007. Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet, 6th International IFIP-TC6 Networking Conference, Atlanta, GA, USA, May 14-18, 2007, Proceedings*. Ed. by Ian F. Akyildiz, Raghupathy Sivakumar, Eylem Ekici, Jaudelice Cavalcante de Oliveira, and Janise McNair. Springer, 2007. URL: https://doi.org/10.1007/978-3-540-72606-7_86.

[94] Jason Mars, Lingjia Tang, Robert Hundt, Kevin Skadron, and Mary Lou Soffa. "Bubble-Up: increasing utilization in modern warehouse scale computers via sensible co-locations". In: *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*. MICRO-44. New York, NY, USA: Association for Computing Machinery, Dec. 2011, pp. 248–259. ISBN: 9781450310536. DOI: 10.1145/

2155620.2155650. URL: https://doi.org/10.1145/2155620.2155650 (visited on 09/20/2022).

[95] Matt Mathis, Nandita Dukkipati, and Yuchung Cheng. *Proportional Rate Reduction for TCP*. RFC 6937. May 2013. DOI: 10.17487/RFC6937. URL: https://rfc-editor.org/rfc/rfc6937.txt.

[96] Matt Mathis, John Heffner, and Raghu Reddy. "Web100: Extended TCP instrumentation for research, education and diagnosis". In: *ACM SIGCOMM Computer Communication Review* 33.3 (2003), pp. 69–79.

[97] Matthew Mathis, Jeffrey Semke, Jamshid Mahdavi, and Teunis Ott. "The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm". In: *SIGCOMM Comput. Commun. Rev.* 27.3 (July 1997), 67–82. ISSN: 0146-4833. DOI: 10.1145/263932.264023. URL: https://doi.org/10.1145/263932.264023.

[98] Nick McKeown, Guido Appenzeller, and Isaac Keslassy. "Sizing Router Buffers (Redux)". In: *SIGCOMM Comput. Commun. Rev.* 49.5 (Nov. 2019), 69–74. ISSN: 0146-4833. DOI: 10.1145/3371934.3371957. URL: https://doi.org/10.1145/3371934.3371957.

[99] *Measurement Lab*. https://www.measurementlab.net/. 2023.

[100] *Mega: Secure Cloud Storage and Communication Privacy by Design*. https://mega.io/.

[101] Marco Mellia, R Lo Cigno, and Fabio Neri. "Measuring IP and TCP behavior on edge nodes with Tstat". In: *Computer Networks* 47.1 (2005), pp. 1–21.

[102] *Microsoft: Algorithmic improvements boost TCP performance on the Internet*. https://techcommunity.microsoft.com/.../ba-p/2347061. 2021.

[103] Dimitrios Miras, Martin Bateman, and Saleem Bhatti. "Fairness of High-Speed TCP Stacks". In: Jan. 2008, pp. 84–92. DOI: 10.1109/AINA.2008.143.

[104] Ayush Mishra and Ben Leong. "Containing the Cambrian Explosion in QUIC Congestion Control". In: *Proceedings of the 2023 ACM on Internet Measurement Conference*. IMC '23. New York, NY, USA: Association for Computing Machinery, 2023, 526–539. ISBN: 9798400703829. DOI: 10.1145/3618257.3624811. URL: https://doi.org/10.1145/3618257.3624811.

[105] Ayush Mishra, Sherman Lim, and Ben Leong. "Understanding Speciation in QUIC Congestion Control". In: *Proceedings of the 22nd ACM Internet Measurement Conference*. IMC '22. New York, NY, USA: Association for Computing Machinery, 2022, 560–566. ISBN: 9781450392594. DOI: 10.1145/3517745.3561459. URL: https://doi.org/10.1145/3517745.3561459.

[106] Ayush Mishra, Lakshay Rastogi, Raj Joshi, and Ben Leong. "Keeping an Eye on Congestion Control in the Wild with Nebby". In: *Proceedings of the ACM SIGCOMM 2024 Conference, ACM SIGCOMM 2024, Sydney, NSW, Australia, August 4-8, 2024*. ACM, 2024, pp. 136–150. DOI: 10.1145/3651890.3672223. URL: https://doi.org/10.1145/3651890.3672223.

[107] Ayush Mishra, Xiangpeng Sun, Atishya Jain, Sameer Pande, Raj Joshi, and Ben Leong. "The Great Internet TCP Congestion Control Census". In: *Proc. ACM Meas. Anal. Comput. Syst.* 3.3 (Dec. 2019). DOI: 10.1145/3366693. URL: https://doi.org/10.1145/3366693.

[108] Ayush Mishra, Wee Han Tiu, and Ben Leong. "Are we heading towards a BBR-dominant internet?" In: *Proceedings of the 22nd ACM Internet Measurement Conference*. IMC '22. New York, NY, USA: Association for Computing Machinery, Oct. 2022, pp. 538–550. ISBN: 9781450392594. DOI: 10.1145/3517745.3561429. URL: https://doi.org/10.1145/3517745.3561429 (visited on 02/15/2023).

[109] Naveenraj Muthuraj, Nooshin Eghbal, and Paul Lu. "Replication: "Taking a long look at QUIC"". In: *Proceedings of the 2024 ACM on Internet Measurement Conference*. IMC '24. New York, NY, USA: Association for Computing Machinery, 2024, 375–388. ISBN: 9798400705922. DOI: 10.1145/3646547.3688453. URL: https://doi.org/10.1145/3646547.3688453.

[110] Netflix. *Fast.com*. https://fast.com. Accessed: Jan 29, 2025. 2025.

[111] *Netflix falls to second place in global internet traffic share as other streaming services grow*. https://www.sandvine.com/inthenews/netflix-falls-to-second-place-in-global-internet-traffic-share. Accessed: 2021-03-04. 2021.

[112] *Netflix Open Connect Overview*. https://openconnect.netflix.com/Open-Connect-Overview.pdf. [Accessed 31-03-2025].

[113] Ravi Netravali, Anirudh Sivaraman, Somak Das, Ameesh Goyal, Keith Winstein, James Mickens, and Hari Balakrishnan. "Mahimahi: accurate {Record-and-Replay} for {HTTP}". In: *2015 USENIX Annual Technical Conference (USENIX ATC 15)*. 2015, pp. 417–429.

[114] *NewReno RFC History*. https://www.rfc-editor.org/search/rfc_search_detail.php?title=NewReno. Accessed: 2021-03-04. 2021.

[115] OECD. *Broadband Networks of the Future*. OECD Digital Economy Papers 327. Paris: OECD Publishing, 2022. DOI: 10.1787/755e2d0c-en. URL: https://doi.org/10.1787/755e2d0c-en.

[116] Ookla. *Speedtest by Ookla*. http://www.speedtest.net. Accessed: Jan 29, 2025. 2025.

[117] J. Padhye, V. Firoiu, Towsley DF, and J. Kurose. "Modeling TCP through-put: A simple model and its empirical validation". In: *Computer Communication Review* 28 (Jan. 2000).

[118] Udit Paul, Jiamo Liu, Mengyang Gu, Arpit Gupta, and Elizabeth Belding. "The Importance of Contextualization of Crowdsourced Active Speed Test Measurements". In: *Proceedings of the 22nd ACM Internet Measurement Conference*. IMC '22. New York, NY, USA: Association for Computing Machinery, 2022, 274–289. ISBN: 9781450392594. DOI: 10.1145/3517745.3561441. URL: https://doi.org/10.1145/3517745.3561441.

[119] Vern Paxson. "Automated packet trace analysis of TCP implementations". In: *Proceedings of the ACM SIGCOMM'97 conference on Applications, technologies, architectures, and protocols for computer communication*. 1997, pp. 167–179.

[120] Vern Paxson. "Bro: a system for detecting network intruders in real-time". In: *Computer networks* 31.23-24 (1999), pp. 2435–2463.

[121] Adithya Abraham Philip, Rukshani Athapathu, Ranysha Ware, Fabian Francis Mkocheko, Alexis Schlomer, Mengrou Shou, Zili Meng, Srinivasan Seshan, and Justine Sherry. "Prudentia: Findings of an Internet Fairness Watchdog". In: *Proceedings of the ACM SIGCOMM 2024 Conference, ACM SIGCOMM 2024, Sydney, NSW, Australia, August 4-8, 2024*. ACM, 2024, pp. 506–520. DOI: 10.1145/3651890.3672229. URL: https://doi.org/10.1145/3651890.3672229.

[122] Adithya Abraham Philip, Ranysha Ware, Rukshani Athapathu, Justine Sherry, and Vyas Sekar. "Revisiting TCP Congestion Control Throughput Models & Fairness Properties at Scale". In: *Proceedings of the 21st ACM Internet Measurement Conference*. IMC '21. Association for Computing Machinery, 2021, 96–103. ISBN: 9781450391290. DOI: 10.1145/3487552.3487834. URL: https://doi.org/10.1145/3487552.3487834.

[123] Pittsburgh Supercomputing Center. *White Paper: Web10G Taking TCP Instrumentation to the Next Level*. https://www.psc.edu/research/networking/web10g/white-paper/. Accessed: [Insert date of access here]. 2023.

[124] PR Newswire. *Sandvine's 2023 Global Internet Phenomena Report Shows 24% Jump in Video Traffic, with Netflix Volume Overtaking YouTube*. https://www.prnewswire.com/news-releases/sandvines-2023-global-internet-phenomena-report-shows-24-jump-in-video-traffic-with-netflix-volume-overtaking-youtube-301723445.html. [Accessed 2025-01-04]. 2025.

[125] *Private communication with Renata Texeira.*

[126] *Prudentia Github Repository*. https://github.com/adithyaphilip/prudentia. Accessed: 2024-06-12. June 2024.

[127] Costin Raiciu, Sebastien Barre, Christopher Pluntke, Adam Greenhalgh, Damon Wischik, and Mark Handley. "Improving Datacenter Performance and Robustness with Multipath TCP". In: 41.4 (Aug. 2011), 266–277. ISSN: 0146-4833. DOI: 10.1145/2043164.2018467. URL: https://doi.org/10.1145/2043164.2018467.

[128] Fengyuan Ren and Chuang Lin. "Modeling and Improving TCP Performance over Cellular Link with Variable Bandwidth". In: *IEEE Transactions on Mobile Computing* 10.8 (2011), pp. 1057–1070. DOI: 10.1109/TMC.2010.234.

[129] Injong Rhee, Lisong Xu, Sangtae Ha, Alexander Zimmermann, Lars Eggert, and Richard Scheffenegger. *CUBIC for Fast Long-Distance Networks*. RFC 8312. Feb. 2018. DOI: 10.17487/RFC8312. URL: https://rfc-editor.org/rfc/rfc8312.txt.

[130] Kanon Sasaki, Masato Hanai, Kouto Miyazawa, Aki Kobayashi, Naoki Oda, and Saneyasu Yamaguchi. "TCP Fairness Among Modern TCP Congestion Control Algorithms Including TCP BBR". In: Oct. 2018, pp. 1–4. DOI: 10.1109/CloudNet.2018.8549505.

[131] Stefan Savage, Tom Anderson, Amit Aggarwal, David Becker, Neal Cardwell, Andy Collins, Eric Hoffman, John Snell, Amin Vahdat, Geoff Voelker, and John Zahorjan. "Detour: a case for informed internet routing and transport". In: *IEEE Micro* 19 (1999), pp. 50–59.

[132] Dominik Scholz, Benedikt Jaeger, Lukas Schwaighofer, Daniel Raumer, Fabien Geyer, and G. Carle. "Towards a Deeper Understanding of TCP BBR Congestion Control". In: *2018 IFIP Networking Conference (IFIP Networking) and Workshops* (2018), pp. 1–9.

[133] *Selenium*. https://www.selenium.dev/.

[134] Neel Shah, Demetres Kouvatsos, Jim Martin, and Scott Moser. "A tutorial on DOCSIS: protocol and performance models". In: *Proceedings of the international working conference on performance modeling and evaluation of heterogeneous networks, Ikley, UK*. 2005.

[135] Matti Siekkinen, Guillaume Urvoy-Keller, Ernst W Biersack, and Denis Collange. "A root cause analysis toolkit for TCP". In: *Computer Networks* 52.9 (2008), pp. 1846–1858.

[136] Varun Singh and Harald Alvestrand. *Identifiers for WebRTC's statistics API*. Jan. 2024. URL: https://www.w3.org/TR/webrtc-stats/.

[137] Bruce Spang, Veronica Hannan, Shravya Kunamalla, Te-Yuan Huang, Nick McKeown, and Ramesh Johari. "Unbiased experiments in congested networks". In: *Proceedings of the 21st ACM Internet Measurement Conference*. IMC '21. New York, NY, USA: Association for Computing Machinery, 2021, 80–95. ISBN: 9781450391290. DOI: 10.1145/3487552.3487851. URL: https://doi.org/10.1145/3487552.3487851.

[138] Bruce Spang, Shravya Kunamalla, Renata Teixeira, Te-Yuan Huang, Grenville Armitage, Ramesh Johari, and Nick McKeown. "Sammy: smoothing video traffic to be a friendly internet neighbor". In: *Proceedings of the ACM SIGCOMM 2023 Conference*. ACM SIGCOMM '23. New York, NY, USA: Association for Computing Machinery, 2023, 754–768. ISBN: 9798400702365. DOI: 10.1145/3603269.3604839. URL: https://doi.org/10.1145/3603269.3604839.

[139] *Speed Index*. https://developer.chrome.com/en/docs/lighthouse/performance/speed-index/.

[140] *Speedtest Global Index – Internet Speed around the world – Speedtest Global Index*. https://www.speedtest.net/global-index. [Accessed 01-04-2025].

[141] *Speedtest Global Speed Index*. https://web.archive.org/web/20230731005025/https://www.speedtest.net/global-index.

[142] Kevin Spiteri, Rahul Urgaonkar, and Ramesh K. Sitaraman. "BOLA: Near-Optimal Bitrate Adaptation for Online Videos". In: *IEEE/ACM Transactions on Networking* 28.4 (2020), pp. 1698–1711. DOI: 10.1109/TNET.2020.2996964.

[143] Randall Stewart and Michael Tüxen. "Adventures in TCP/IP: TCP Black Box Logging". In: *FreeBSD Journal* May/June 2024 (2024). Configuration Management Showdown. URL: https://freebsdfoundation.org/our-work/journal/browser-based-edition/configuration-management-2/adventures-in-tcp-ip-tcp-black-box-logging/.

[144] Ion Stoica, Scott Shenker, and Hui Zhang. "Core-stateless fair queueing: Achieving approximately fair bandwidth allocations in high speed networks". In: *Proceedings of the ACM SIGCOMM'98 conference on Applications, technologies, architectures, and protocols for computer communication*. 1998, pp. 118–130.

[145] Srikanth Sundaresan, Walter De Donato, Nick Feamster, Renata Teixeira, Sam Crawford, and Antonio Pescapè. "Broadband internet performance: a view from the gateway". In: *ACM SIGCOMM computer communication review* 41.4 (2011), pp. 134–145.

[146] Yuechen Tao, Jingjie Jiang, Shiyao Ma, Luping Wang, Wei Wang, and Bo Li. "Unraveling the RTT-fairness Problem for BBR: A Queueing Model". In: *2018 IEEE Global Communications Conference (GLOBECOM)*. ISSN: 2576-6813. Dec. 2018, pp. 1–6. DOI: 10.1109/GLOCOM.2018.8647260.

[147] *tc-netem(8) - Linux Manual Page*. https://man7.org/linux/man-pages/man8/tc-netem.8.html. Accessed: 2021-03-04. 2021.

[148] *TCP Congestion Control Algorithms Comparison*. https://www.speedguide.net/articles/tcp-congestion-control-algorithms-comparison-7423. 2021.

[149] *TCP Rtt-Fairness In NS-2*. http://intronetworks.cs.luc.edu/current/html/dynamics.html. Accessed: 2020-03-25.

[150] *TCP/IP Fingerprinting Methods Supported by Nmap*. https://nmap.org/book/osdetect-methods.html#osdetect-w. [Accessed 07-04-2025].

[151] *tcpprobe*. Accessed 2021-09-10. 2021. URL: https://wiki.linuxfoundation.org/networking/tcpprobe.

[152] *Test and Measurement: Comcast touts 'ultra-low lag connectivity' using L4S*. https://www.rcrwireless.com/20250129/uncategorized/comcast-l4s. [Accessed 14-04-2025].

[153] *The Network Simulator - ns-2*. https://www.isi.edu/nsnam/ns/. Accessed: 2021-03-04. 2021.

[154] James Titcomb. "Google algorithm 'hogs' internet traffic, researchers show". In: *The Telegraph* (Oct. 27, 2019). URL: https://www.telegraph.co.uk/technology/2019/10/27/google-algorithm-hogs-internet-traffic-researchers-show (visited on 09/18/2022).

[155] Linus Torvalds and other contributors. *TCP BBR congestion control algorithm in the Linux kernel*. https://github.com/torvalds/linux/commits/master/net/ipv4/tcp_bbr.c. Accessed: 2024-02-03. 2024.

[156] Belma Turkovic, Fernando A Kuipers, and Steve Uhlig. "Fifty Shades of Congestion Control: A Performance and Interactions Evaluation". In: *arXiv* (2019).

[157] *Under submission. Details omitted for double-blind reviewing.*

[158] Curtis Villamizar and Cheng Song. "High performance TCP in ANSNET". In: *SIGCOMM Comput. Commun. Rev.* 24.5 (Oct. 1994), 45–60. ISSN: 0146-4833. DOI: 10.1145/205511.205520. URL: https://doi.org/10.1145/205511.205520.

[159] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. "Image quality assessment: from error visibility to structural similarity". In: *IEEE Transactions on Image Processing* (2004).

[160] R. Ware, M. K. Mukerjee, J. Sherry, and S. Seshan. "The Battle for Bandwidth: Fairness and Heterogeneous Congestion Control. In Poster at NSDI 2018". In: 2018.

[161] Ranysha Ware, Matthew K. Mukerjee, Srinivasan Seshan, and Justine Sherry. "Beyond Jain's Fairness Index: Setting the Bar For The Deployment of Congestion Control Algorithms". In: *Proceedings of the 18th ACM Workshop on Hot Topics in Networks*. HotNets '19. New York, NY, USA: Association for Computing Machinery, Nov. 2019, pp. 17–24. ISBN: 9781450370202. DOI: 10.1145/3365609.3365855. URL: https://doi.org/10.1145/3365609.3365855 (visited on 09/19/2022).

[162] Ranysha Ware, Matthew K. Mukerjee, Srinivasan Seshan, and Justine Sherry. "Modeling BBR's Interactions with Loss-Based Congestion Control". In: *Proceedings of the Internet Measurement Conference*. IMC '19. New York, NY, USA: Association for Computing Machinery, 2019, 137–143. ISBN: 9781450369480. DOI: 10.1145/3355369.3355604. URL: https://doi.org/10.1145/3355369.3355604.

[163] Pantheon Website. *Pantheon FAQ*. URL: https://pantheon.stanford.edu/faq/.

[164] Wei Wei, Chun Zhang, Hui Zang, Jim Kurose, and Don Towsley. "Inference and evaluation of split-connection approaches in cellular data networks". In: *Passive and Active Measurement Conference*. Citeseer. 2006.

[165] Wikipedia. *A/B testing - Wikipedia*. https://en.wikipedia.org/wiki/A/B_testing. [Accessed 15-05-2025]. 2025.

[166] Wikipedia. *Mann–Whitney U test — Wikipedia*. http://en.wikipedia.org/w/index.php?title=Mann%E2%80%93Whitney%20U%20test&oldid=1284635664. [Online; accessed 15-May-2025]. 2025.

[167] Keith Winstein, Anirudh Sivaraman, and Hari Balakrishnan. "Stochastic Forecasts Achieve High Throughput and Low Delay over Cellular Networks". In: *10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*. Lombard, IL: USENIX Association, Apr. 2013, pp. 459–471. ISBN: 978-1-931971-00-3. URL: https://www.usenix.org/conference/nsdi13/technical-sessions/presentation/winstein.

[168] Xiaokun Xu and Mark Claypool. "Measurement of cloud-based game streaming system response to competing TCP cubic or TCP BBR flows". In: *Proceedings of the 22nd ACM Internet Measurement Conference*. IMC '22. New York, NY, USA: Association for Computing Machinery, 2022, 305–316. ISBN: 9781450392594. DOI: 10.1145/3517745.3561464. URL: https://doi.org/10.1145/3517745.3561464.

[169] Francis Y Yan, Hudson Ayers, Chenzhi Zhu, Sadjad Fouladi, James Hong, Keyi Zhang, Philip Levis, and Keith Winstein. "Learning in situ: a randomized experiment in video streaming". In: *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. 2020, pp. 495–511.

[170] Francis Y. Yan, Jestin Ma, Greg D. Hill, Deepti Raghavan, Riad S. Wahby, Philip Levis, and Keith Winstein. "Pantheon: the training ground for Internet congestion-control research". In: *2018 USENIX Annual Technical Conference (USENIX ATC 18)*. Boston, MA: USENIX Association, July 2018, pp. 731–743. ISBN: 978-1-939133-01-4. URL: https://www.usenix.org/conference/atc18/presentation/yan-francis.

[171] *youtube-dl*. https://github.com/ytdl-org/youtube-dl.

[172] Zhaojuan Yue, Xiaodan Zhang, Yongmao Ren, Jun Li, and Qianli Zhong. "The performance evaluation and comparison of TCP-based high-speed transport protocols". In: *2012 IEEE International Conference on Computer Science and Automation Engineering*. 2012, pp. 509–512. DOI: 10.1109/ICSESS.2012.6269516.

[173] Danesh Zeynali, Emilia N Weyulu, Seifeddine Fathalli, Balakrishnan Chandrasekaran, and Anja Feldmann. "Promises and Potential of BBRv3". In: *International Conference on Passive and Active Network Measurement*. Springer. 2024, pp. 249–272.

[174] Yin Zhang, Lee Breslau, Vern Paxson, and Scott Shenker. "On the characteristics and origins of internet flow rates". In: *Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*. 2002, pp. 309–322.

[175] Yongguang Zhang and Thomas R Henderson. "An implementation and experimental study of the explicit control protocol (XCP)". In: *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.* Vol. 2. IEEE. 2005, pp. 1037–1048.

[176] Yuxiang Zhang, Lin Cui, and Posco Tso. "Modest BBR: Enabling Better Fairness for BBR Congestion Control". In: June 2018, pp. 00646–00651. DOI: 10.1109/ISCC.2018.8538521.