Prudentia: Findings of an Internet Fairness Watchdog

Adithya Abraham Philip
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA

Rukshani Athapathu
UC San Diego
San Diego, California, USA

Ranysha Ware
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA

Fabian Francis Mkocheko
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA

Alexis Schlomer
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA

Mengrou Shou
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA

Zili Meng
HKUST
Hong Kong

Srinivasan Seshan
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA

Justine Sherry
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA

Abstract

With the rise of heterogeneous congestion control algorithms and increasingly complex application control loops (e.g. adaptive bitrate algorithms), the Internet community has expressed growing concern that network bandwidth allocations are unfairly skewed, and that some Internet services are 'winners' at the expense of 'losing' services when competing over shared bottlenecks. In this paper, we provide the first study of fairness between live, end-to-end services with distinct workloads. Rather than focusing on individual components of an application stack (e.g., studying the fairness of an individual congestion control algorithm), we want to provide a direct study over real-world deployed applications. Among our findings, we observe that services typically achieve less-than-fair outcomes: on average, the 'losing' service achieves only 72% of its max-min fair share of link bandwidth. We also find that some services are significantly more contentious than others: for example, one popular file distribution service causes competing applications to obtain as low as 16% of their max-min fair share of bandwidth when competing in a moderately-constrained setting.

CCS Concepts

• **Networks** → **Network measurement**; **Network performance analysis**.

Keywords

computer networks, internet fairness, internet measurement, internet services, congestion control algorithms

ACM Reference Format:
Adithya Abraham Philip, Rukshani Athapathu, Ranysha Ware, Fabian Francis Mkocheko, Alexis Schlomer, Mengrou Shou, Zili Meng, Srinivasan Seshan, and Justine Sherry. 2024. Prudentia: Findings of an Internet Fairness Watchdog. In *ACM SIGCOMM 2024 Conference (ACM SIGCOMM '24), August 4–8, 2024, Sydney, NSW, Australia.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3651890.3672229

ACM SIGCOMM '24, August 4–8, 2024, Sydney, NSW, Australia

ACM ISBN 979-8-4007-0614-1/24/08
https://doi.org/10.1145/3651890.3672229

1 Introduction

When two roommates log into competing video services of their choice, sharing the same bottleneck network link, what will their resulting experience be? Will one video play in high quality, while the other stutters and struggles with grainy video? Recent research suggests that we may have cause to be concerned that one roommate's service might thrive while starving the other roommate's stream of bandwidth. For example, some work has shown that novel congestion control algorithms (CCAs) lead to *unfair* outcomes when competing with legacy algorithms [52]; another study demonstrates that adaptive-bitrate (ABR) algorithms can unnecessarily harm competing cross-traffic due to bursty HTTP chunking [25].

We believe that ensuring heterogeneous services perform well side-by-side is essential to a stable and inclusive Internet. One of the Internet's core 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.

Hence, in this paper, we study a simple question: *Are there 'winners' and 'losers' when popular services compete for bandwidth on the Internet today?*

While many researchers are concerned with this question, existing research studies focus primarily on how a *single* aspect of a service's design (typically the CCA) impacts winners and losers. We argue that it is necessary to evaluate Internet services *as a whole*, as 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)[1]. For example, BBR has been broadly attacked in the research community and even the popular press [47, 52] because it leads to 'unfair' bandwidth allocations in deep-buffered networks with long-lived bulk flows. However, as we will show in §4, YouTube, which uses BBR for its congestion control, is in reality one of the *least* contentious

[1]We borrow the terms *contentiousness* and *sensitivity* from performance modeling literature [36, 37]

services that we tested. In short, choice of CCA fails to tell the whole story about congestion at the service level.

To evaluate deployed services in the wild, we present Prudentia, an independent 'watchdog' for Internet fairness. Prudentia evaluates contending Internet services by simultaneously accessing two live, deployed services through a controlled testbed, configured to emulate different link conditions. 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:

- The file-distribution service Mega [6] is substantially more contentious than any other application we tested. In our moderately-constrained setting, services running alongside Mega achieved on average only 63% of their Max-Min Fair [17] (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 [46] – 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 §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 §6, we discuss results from Prudentia that shed light on the various factors affecting fairness measurements. In §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 §6. Lastly, we discuss related work in §8, and conclude with future directions for Prudentia in §10.

2 Goals and Metrics

Before digging into the mechanics of our measurement methodology (§3) we first take a moment to ground the goals and philosophy behind our study.

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 [18] or ABR [44]. 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 paper 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 [15].

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[2] Scaling further would 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.

[2]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.

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.

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 [33, 52, 54], 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 §5.

Measuring throughput fairness itself is highly debated as there are many competing definitions of fairness, *e.g.* equal-rate fairness [30], proportional fairness [31], RTT-fairness [46], and max-min fairness [19]. For better or worse, most Internet algorithms (including many TCP congestion controllers [5, 55] and fair queueing schemes [45]) 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 [30] 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 [51] 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.

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.[3] In the performance contention literature [36, 37], a given performance measure under contention is modeled as a function of the *contentiousness* of the contender, and of the *sensitivity* of the incumbent.

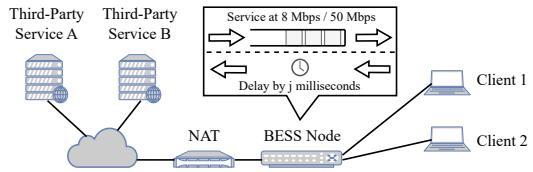


Figure 1: We use a dumbbell topology with two clients simultaneously receiving data from 2 services, with all traffic passing through the software switch BESS which acts as our controlled bottleneck link.

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 [36]. 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.[4]

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).

3 Methodology

Having described our high-level aims above, we now present our measurement methodology including our testbed design for network emulation (§3.1), how we ensure application fidelity in our automated environment (§3.3), the services and period we tested them in (§3.2), and finally how we measure statistical significance(§3.4).

3.1 Network Emulation

The Prudentia testbed is illustrated at a high level in Figure 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 [1] 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

[3] Betteridge's law of headlines states: 'Any headline that ends in a question mark can be answered by the word no.' [2]

[4] 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.

Table 1: Services supported in the Prudentia testbed.

Service	Category	CCA	Max Xput	# Flows[+]	Notes
YouTube	Video	BBRv1.1 [40]	13Mbps	1	*7 available bitrates, up to 4K. QUIC-based rather than TCP.*
Netflix[δ]	Video	NewReno [7]	8Mbps	4	*6 available bitrates, up to 4K.*
Vimeo	Video	BBR*	14Mbps	2	*7 available bitrates, up to 4K.*
DropBox	File Transfer	BBRv1.0 [41]	∞	1	
Google Drive	File Transfer	BBRv3 [5]	∞	1	
OneDrive	File Transfer	Cubic [13]	45 Mbps	1	*Extended version of Cubic* [13]. *Achieves a maximum average throughput of 45 Mbps.*
Mega	File Transfer	BBR*	∞	5	
Google Meet	RTC	GCC [21]	1.5Mbps	1	WebRTC-based.
Microsoft Teams	RTC	Unknown	2.6Mbps	1	WebRTC-based.
wikipedia.org	Web	BBRv1.0	∞	>5[β]	Mostly text with one or two images.
news.google.com	Web	BBRv3.0	∞	>20[β]	Text accompanied by thumbnail images.
youtube.com	Web	BBRv3.0	∞	>10[β]	Mostly images in the form of thumbnails, different CCA than YouTube video server.
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 [53].
+ 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 [28].
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.

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 [10] and (b) 8 Mbps represents the bottom 10% percentile of country-level median bandwidths [10]. 8 Mbps is also approximately the bandwidth that a 2K video would consume[5], allowing us to examine how contentious video services can be in a scenario where they can consume the entire link bandwidth.

While these are the primary bandwidths Prudentia uses to evaluate fairness, in §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 [24].[6] In §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 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.

[5]This number comes from the bitrates in Youtube's manifest files which we downloaded using [11].

[6]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.

3.2 Services & Period Under Test

Table 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. 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 §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 [14] 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 [3].

Prudentia has been evaluating fairness amongst these services since 2022. Unless otherwise noted, the numbers reported in this paper are from the latest set of experiments, run between June 2023 - September 2023, and the RTC service evaluation in January 2024.

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 [4] controlled by Selenium [8] 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 [12] 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.

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 §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 with every other service takes ~20 hours.

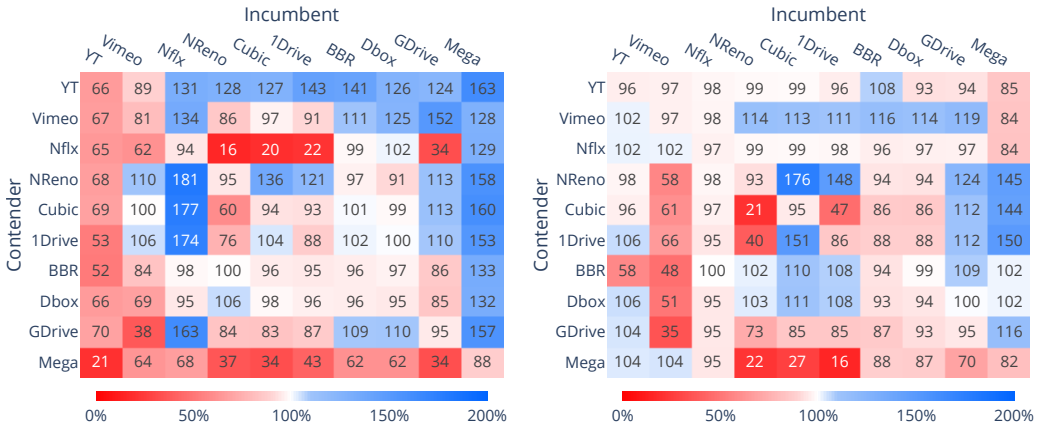
4 Throughput Under Contention

Having described our methodology (§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 §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 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.

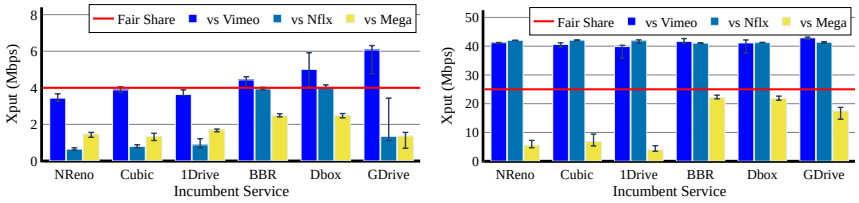
In the majority of experiments, the MmF allocation is simply 50% of the link capacity. However, video services in the 50 Mbps setting are application-limited by their maximum achievable bitrate and, in this setting, their MmF allocation is between 8 Mbps and 14 Mbps, and their contenders' allocation correspondingly higher, depending on the service as shown in Table 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. 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 $\geq$100, reflecting that most services perform well when competing



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

Figure 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.



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

Figure 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.

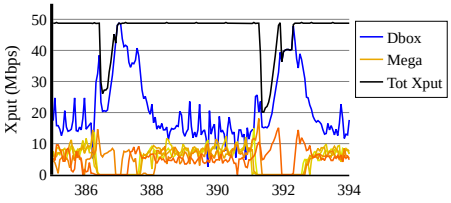


Figure 4: When Dropbox competes with Mega, it is able to ramp up quickly and utilize the extra bandwidth between Mega's bursts, allowing it to obtain 3-4x the throughput against Mega compared to traditional CCAs like NewReno or Cubic.

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 1: *Unfair outcomes are common in bandwidth-contended environments. (Fig 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 2: *The most and least contentious services we measured use variants of the same underlying CCA; CCA alone cannot account for the di ering fairness outcomes.*

Since both Mega and YouTube use variants of BBR as their underlying CCA[7], 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 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

[7]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.

the outcomes we observe for BBR-based or NewReno-based services.

Observation 3: *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 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 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 limit this. Fig 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 4: *Application-level scheduling and request patterns can shape fairness outcomes. (Fig 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. Dropbox (which uses BBR) is able to ramp up sufficiently in-between bursts (Fig 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 [48], even kernel updates can change BBR's fairness outcomes.

5 Beyond Throughput Fairness

While throughput fairness is the standard metric evaluated by most studies of network contention [33, 52, 54], 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 (§5.1), page load times in webpage browsing (§5.2), as well as metrics such as link utilization and loss for our throughput-intensive services (§5.3).

Table 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 [43].
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 [43], which checks if the frame inter-arrival time exceeds $<OG(3*X•X+150<B)$, where X 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 [29].

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 per minute, and high-delay packets – metrics which are often incorporated into higher order 'QoE' measures. We define these metrics in Table 2; we defer evaluation of higher order QoE metrics (*e.g.* VMAF [34], SSIM [50]) to future work.

Observation 5: *Differing trade-offs made by applications can lead to different perceived sensitivity at the user level (Fig.5)*

In Fig 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 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 6: *Services using loss-based CCAs can cause as much as 92% of the packets to exceed ideal RTT requirements (Fig 5g,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 [29]. 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 [26, 40]. 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

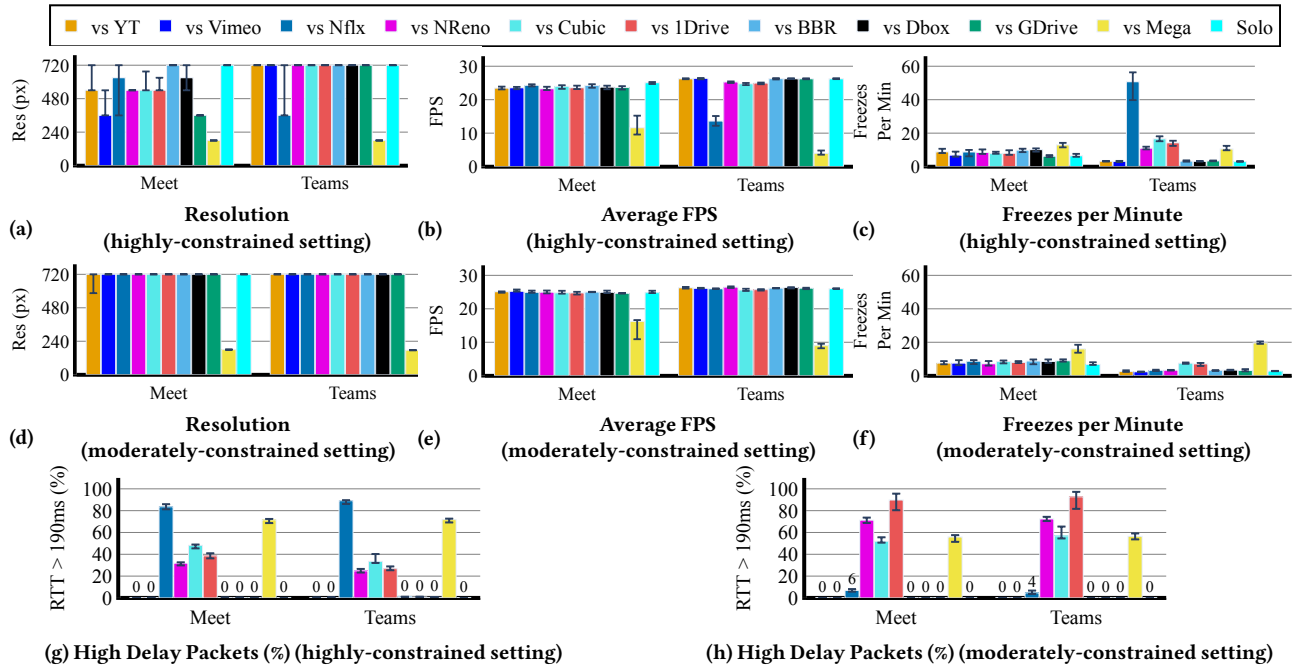


Figure 5: The degradation, or lack thereof, in various metrics when Google Meet and Teams compete against other services. In the moderately-constrained setting, in most cases, both services perform well in metrics other than latency. Howver, in the highly-constrained setting, many competing services cause varying degrees of QoE degradation.

as much latency inflation as services using buffer-filling algorithms.

Observation 7: *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.

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 [9]. 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 8: *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 6).*

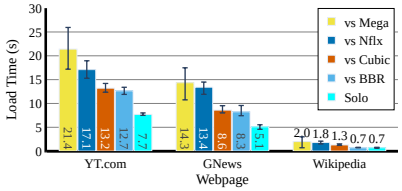
We find that competing traffic can increase page load times, especially in the highly-constrained setting. In the presence of Mega and Netflix, users visiting youtube.com may have to wait for 21 seconds instead of just 8 seconds (median), a difference of 162%. The increase in loading time is also clearly 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.

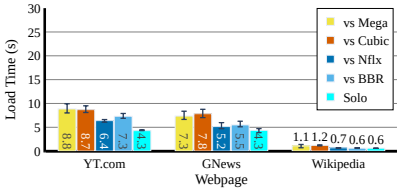
5.3 Link Utilization & Loss

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

Observation 9: *Application-level behaviors can cause both unfairness and under-utilization.*



(a) Highly-constrained setting



(b) Moderately-constrained setting

Figure 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.

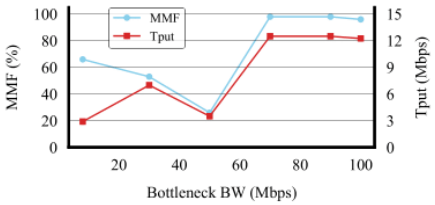


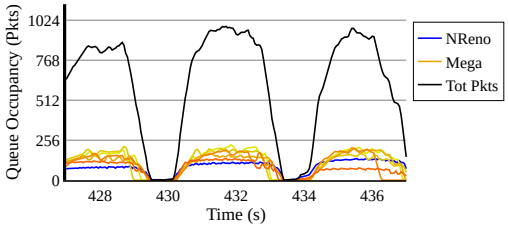
Figure 7: The unfairness YouTube suffers against Dropbox initially increases with increase in the bottleneck bandwidth, then suddenly becomes fair beyond 70Mbps.

In most scenarios we see 95% or higher link utilization (a complete heatmap is in Appendix B.1). 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 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). 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.

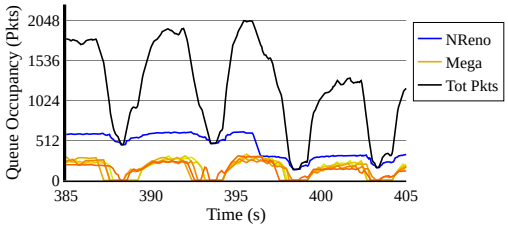
We also see under-utilization in the highly-constrained setting when video services compete against each other. We suspect this is due to ABR algorithms prioritizing stability over maximal throughput and hence choosing to play a video consistently at lower quality than potentially having to switch back and forth between higher and lower quality video [44].

Observation 10: *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 in Appendix B.2). 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.



(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 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.

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 11: *Bu er sizing signi cantly influences fairness and utilization outcomes, underscoring the need to pro le 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 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.

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 [27]. 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.* [22, 32]) so that operators can test their services given appropriate real-world parameters.

Observation 12: *Contentiousness can have a non-monotonic relationship with increasing bandwidth availability.* [8] *(Fig 7)*

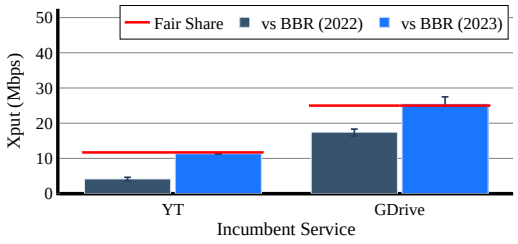
We performed all-pairs experiments at a range of bandwidths between 8 Mbps and 100Mbps. Overall, we did observe a general trend of fairness improving with higher bandwidths. However, this was *not always the case* and in some scenarios we even observed fairness degrade with increased bandwidth. For example, 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 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 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 13: *Incremental changes in CCA design can lead to noticeable changes in contentiousness. (Fig 9)*

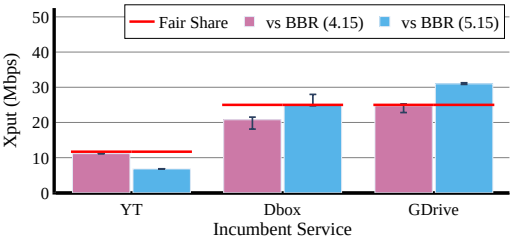
Through Prudentia's live experiments, we were able to detect changes in Google Drive and YouTube's deployments between 2022 and 2023. We found that compared to 2022, Google Drive and YouTube performed 46% and 172% better in 2023 against iPerf-based BBR (see Fig 9a). Google engineers confirmed that this coincided with the deployment of BBRv3 to Google Drive [5] 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 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.

[8] This observation is based on experiments from the 2022 period referred to in §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 9: Changes to services, and even kernel updates, can change fairness properties, necessitating the use of a live watchdog that constantly monitors services.

Table 3: Unfairness and fairness are not necessarily transitive. Service U may cause V to get an unfair (or fair) share, and V may cause W to get an unfair (or fair) share, but this does not guarantee that U causes unfairness (or fairness) to W. The lack of transitivity exemplifies the difficulty in classifying most services as generally contentious or sensitive.

U	V	W	BW (Mbps)	MmF Obtained (%)		
				V (vs U)	W (vs V)	W (vs U)
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 14: *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 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 2). This highlights the need to evaluate each contender against a wide variety of incumbents. Simply extrapolating a service's fairness

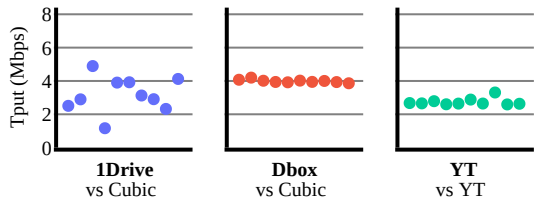


Figure 10: Each data point represents the throughput obtained by the service in bold in a single trial. Certain services such as One Drive show unstable outcomes when interacting with other services, while others are relatively stable.

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 3 shows a few examples of this lack of transitivity, extracted from the set of results in Fig 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 15: *Service instability can lead to sometimes-harmful sometimes-not outcomes between the same services. (Fig 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 +/- 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 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 §5.1. Operators should be concerned about services which are 'sometimes' overly contentious, and run multiple trials to capture these issues.

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 in Appendix A.

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.

8 Other Related Work

There are two broad types of related work 1) fairness evaluations of CCAs [16, 20, 23, 49] and 2) frameworks for testing CCAs and deployed services [33, 35, 54].

Several studies have conducted experiments to evaluate the coexistence 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 [16, 20, 23]. Turkovic et al. [49] 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 [54] 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. [35] 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. [33] 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 [39, 40]. This motivates our conjecture that services need to be evaluated periodically and constantly.

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[54], 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 [42, 52]. 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 [20], and that NewReno suffers from poor performance in networks with high RTTs [38]. 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.

10 Conclusion

In this work we presented Prudentia, a watchdog for Internet fairness. Using Prudentia, we observe that unequal bandwidth outcomes are not a rarity, but in fact a common case in contended bandwidth settings. We also explored other aspects of performance degradation, such as spikes in latency and loss.

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 **3**) and that buffer-filling algorithms are bad for real-time communication (see Observation **6**). 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.

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

Ethics Statement: Prudentia does not use any data generated by real users, and downloads from only targeted high-capacity service providers. We believe this work does not raise any ethical issues.

Acknowledgments

We would like to thank the anonymous reviewers and our shepherd Zafar Ayyub Qazi for their valuable feedback and assistance in revising the paper. We owe Hugo Sadok, Anup Agarwal, Christopher Canel, Nirav Atre, Miguel Alves Ferreira, Ioannis Anagnostides and Margarida Ferreira a debt of gratitude for their great suggestions and help with the paper. We are also extremely grateful to Neal Cardwell, Ian Swett, TY Huang, Renata Teixeira, Randall Stewart, Grenville Armitage and Nikita Shirokov for their help understanding various phenomena we observed over the course of Prudentia's measurements. This research was sponsored by the following awards: NSF grants (award numbers 2212390 and 2007733), a CMU Cylab Seed Grant, a Google Faculty Research Award, and a Facebook Fellowship. 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.

References

[1] [n. d.]. BESS: A Software Switch. https://github.com/NetSys/bess. ([n. d.]).
[2] [n. d.]. Betteridge's law of headlines - Wikipedia. https://en.wikipedia.org/wiki/Betteridge%27s_law_of_headlines. ([n. d.]).
[3] [n. d.]. Big Buck Bunny. https://peach.blender.org/about/. ([n. d.]).
[4] [n. d.]. Google Chrome. https://www.google.com/chrome/. ([n. d.]).
[5] [n. d.]. IETF - BBRv3. https://datatracker.ietf.org/meeting/117/materials/slides-117-ccwg-bbrv3-algorithm-bug-fixes-and-public-internet-deployment-00. ([n. d.]).
[6] [n. d.]. Mega: Secure Cloud Storage and Communication Privacy by Design. https://mega.io/. ([n. d.]).
[7] [n. d.]. Private communication with Renata Texeira. ([n. d.]).
[8] [n. d.]. Selenium. https://www.selenium.dev/. ([n. d.]).
[9] [n. d.]. Speed Index. https://developer.chrome.com/en/docs/lighthouse/performance/speed-index/. ([n. d.]).
[10] [n. d.]. Speedtest Global Speed Index. https://web.archive.org/web/20230731005025/https://www.speedtest.net/global-index. ([n. d.]).
[11] [n. d.]. youtube-dl. https://github.com/ytdl-org/youtube-dl. ([n. d.]).

[12] 2017. Harware accelaration for VP9 not working. (Apr 2017). https://community.intel.com/.../td-p/291546

[13] 2021. Microsoft: Algorithmic improvements boost TCP performance on the Internet. https://techcommunity.microsoft.com/.../ba-p/2347061. (May 11, 2021).

[14] 2021. TCP Congestion Control Algorithms Comparison. https://www.speedguide.net/articles/tcp-congestion-control-algorithms-comparison-7423. (2021).

[15] 2024. Prudentia Github Repository. https://github.com/adithyaphilip/prudentia. (June 2024). Accessed: 2024-06-12.

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

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

[18] Thomas Bonald. 1999. Comparison of TCP Reno and TCP Vegas: e ciency and fairness. *Performance Evaluation* 36 (1999), 307–332.

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

[20] Neal Cardwell, Yuchung Cheng, C. Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. 2017. BBR: congestion-based congestion control. *Commun. ACM* 60, 2 (Jan. 2017), 58–66. https://doi.org/10.1145/3009824

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

[22] Amogh Dhamdhere, David D Clark, Alexander Gamero-Garrido, Matthew Luckie, Ricky KP Mok, Gautam Akiwate, Kabir Gogia, Vaibhav Bajpai, Alex C Snoeren, and Kc Cla y. 2018. Inferring persistent interdomain congestion. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*. 1–15.

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

[24] Yashar Ganjali and Nick McKeown. 2006. Update on Bu er Sizing in Internet Routers. *SIGCOMM Comput. Commun. Rev.* 36, 5 (oct 2006), 67–70. https://doi.org/10.1145/1163593.1163605

[25] Monia Ghobadi, Yuchung Cheng, Ankur Jain, and Matt Mathis. 2012. Trickle: Rate Limiting YouTube Video Streaming. In *USENIX ATC'12*. USENIX Association, USA, 17.

[26] Sishuai Gong, Usama Naseer, and Theophilus A Benson. 2020. Inspector Gadget: A Framework for Inferring TCP Congestion Control Algorithms and Protocol Con gurations.. In *Network Tra c Measurement and Analysis Conference*.

[27] Sangtae Ha, Injong Rhee, and Lisong Xu. 2008. CUBIC: a new TCP-friendly high-speed TCP variant. *ACM SIGOPS Operating Systems Review* 42, 5 (July 2008), 64–74. https://doi.org/10.1145/1400097.1400105

[28] Help Center. [n. d.]. How to use Net ix on your Mac computer. https://help.net ix.com/en/node/55764. ([n. d.]). Accessed: Feb. 02, 2024.

[29] International Telecommunication Union. 2015. 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. (January 2015). Accessed: Feb. 02, 2024.

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

[31] Frank P Kelly, Aman K Maulloo, and David Kim Hong Tan. 1998. Rate control for communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research society* 49, 3 (1998), 237–252.

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

[33] Ike Kunze, Jan Rüth, and Oliver Hohlfeld. 2020. Congestion Control in the Wild–Investigating Content Provider Fairness. *IEEE Transactions on Network and Service Management* 17, 2 (2020), 1224–1238. https://doi.org/10.1109/TNSM.2019.2962607

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

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

[36] Antonis Manousis, Rahul Anand Sharma, Vyas Sekar, and Justine Sherry. 2020. Contention-Aware Performance Prediction For Virtualized Network Functions. 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*. ACM, Virtual Event USA, 270–282. https://doi.org/10.1145/3387514.3405868

[37] Jason Mars, Lingjia Tang, Robert Hundt, Kevin Skadron, and Mary Lou So a. 2011. 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)*. Association for Computing Machinery, New York, NY, USA, 248–259. https://doi.org/10.1145/2155620.2155650

[38] Matthew Mathis, Je rey Semke, Jamshid Mahdavi, and Teunis Ott. 2001. The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm. *ACM Computer Communication Review* 27 (02 2001). https://doi.org/10.1145/263932.264023

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

[40] Ayush Mishra, Xiangpeng Sun, Atishya Jain, Sameer Pande, Raj Joshi, and Ben Leong. 2019. The Great Internet TCP Congestion Control Census. *3, 3, Article 45 (dec 2019), 24 pages*. https://doi.org/10.1145/3366693

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

[42] Adithya Abraham Philip, Ranysha Ware, Rukshani Athapathu, Justine Sherry, and Vyas Sekar. 2021. 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, New York, NY, USA, 96–103. https://doi.org/10.1145/3487552.3487834

[43] Varun Singh and Harald Alvestrand. 2024. Identi ers for WebRTC's statistics API. (Jan 2024). https://www.w3.org/TR/webrtc-stats/

[44] Kevin Spiteri, Rahul Urgaonkar, and Ramesh K. Sitaraman. 2020. BOLA: Near-Optimal Bitrate Adaptation for Online Videos. *IEEE/ACM Transactions on Networking* 28, 4 (2020), 1698–1711. https://doi.org/10.1109/TNET.2020.2996964

[45] Ion Stoica, Scott Shenker, and Hui Zhang. 1998. 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*. 118–130.

[46] Yuechen Tao, Jingjie Jiang, Shiyao Ma, Luping Wang, Wei Wang, and Bo Li. 2018. Unraveling the RTT-fairness Problem for BBR: A Queueing Model. In *2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE. https://doi.org/10.1109/GLOCOM.2018.8647260

[47] James Titcomb. [n. d.]. Google algorithm 'hogs' internet tra c, researchers show. *The Telegraph* ([n. d.]). https://www.telegraph.co.uk/technology/2019/10/27/google-algorithm-hogs-internet-traf c-researchers-show

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

[49] Belma Turkovic, Fernando A. Kuipers, and Steve Uhlig. 2019. Fifty Shades of Congestion Control: A Performance and Interactions Evaluation. (2019). https://doi.org/10.48550/ARXIV.1903.03852

[50] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing* 13, 4 (2004).

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

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

[53] Ranysha Ware, Adithya Abraham Philip, Nicholas Hungria, Yash Kothari, Justine Sherry, and Srinivasan Seshan. 2024. CCAnalyzer: An E cient and Nearly-Passive Congestion Control Classi er. In *Proceedings of the 38th ACM Special Interest Group on Data Communication (SIGCOMM) (SIGCOMM '24)*. Association for Computing Machinery, New York, NY, USA.

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

[55] Yongguang Zhang and Thomas R Henderson. 2005. 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, 1037–1048.

Appendices are supporting material that has not been peer-reviewed.

A Third Party Service Evaluation

The Prudentia website can be accessed at https://internetfairness.net. Instructions for submitting a service to be evaluated by Prudentia can be found at https://internetfairness.net/testing. Access codes are required to run third party tests, and have been provided below:

- `KD4p1Z8GslSVPHUrTOVTMNHtvUnMSmvZ`
- `A7mH2gHPmtlhbpb8ajfe48oCzA7hp6VB`
- `5PWWIvTUxZSYVhIuEiBEmOOOog8zgrGa`
- `XrVzJ3evvkVpoAf3k54mYuY0tCgjTD2k`
- `bTXmWjSdAmQf4ULItqH2JCR5oX8jZvhL`

B Additional Results

B.1 Link Utilization Heatmap

Fig 11 summarizes the link utilization obtained when two services compete. It is symmetric across the diagonal as it measures the total fraction of link capacity used, and is obtained by adding the throughputs obtained by both competing services in a given experiment and dividing by the total bandwidth. The heatmap shows the median link utilization obtained across multiple trials between the same pair of services.

B.2 Loss Rate Heatmap

Fig 12 summarizes the packet loss rates obtained when two services compete. The heatmap shows the median packet loss rates obtained across multiple trials between the same pair of services.

B.3 Queueing Delay Heatmap

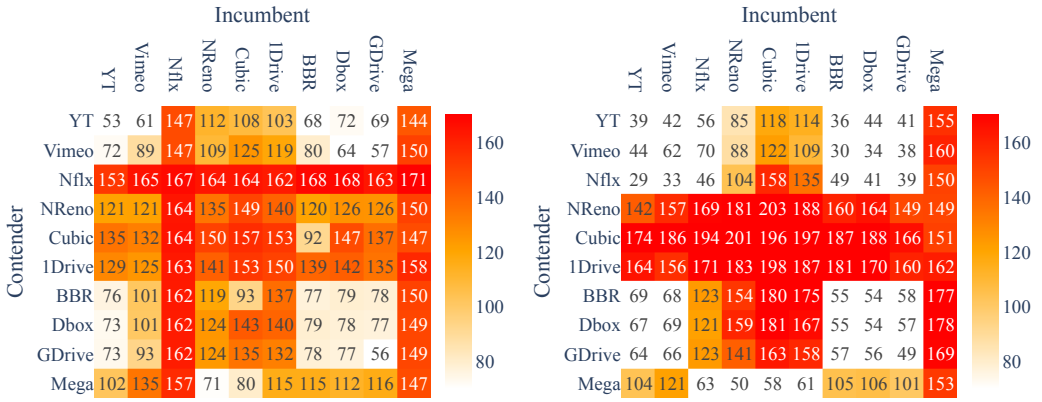
Fig 13 shows the average queueing delay experienced by packets of an incumbent service when competing against a contender (median over multiple trials shown).

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

Figure 11: Median link utilization obtained when two services compete. Except for Mega and certain pairs of video services, all services achieve at least 95% link utilization.

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

Figure 12: Packet loss rates obtained by an incumbent service when competing with a given contender (median over multiple trials shown).



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

Figure 13: Average queuing delay (ms) experienced by packets of an incumbent service when competing with a given contender (median over multiple trials shown).