Promises and Potential of BBRv3

Danesh Zeynali[1(✉)], Emilia N. Weyulu[1], Seifeddine Fathalli[1], Balakrishnan Chandrasekaran[2], and Anja Feldmann[1]

[1] Max-Planck-Institut für Informatik, Saarbrücken, Germany
{dzeynali,eweyulu,fathalli,anja}@mpi-inf.mpg.de
[2] Vrije Universiteit Amsterdam, Amsterdam, The Netherlands
b.chandrasekaran@vu.nl

Abstract. The Bottleneck-Bandwidth and Round-trip (BBR) congestion control algorithm was introduced by Google in 2016. Unlike prior congestion-control algorithms (CCAs), BBR does not rely on signals that are weakly correlated with congestion (e.g., packet loss and transient queue delay). Instead, it characterizes a path using two parameters, bottleneck bandwidth and round-trip propagation time, and is designed to converge with a high probability to Kleinrock's optimal operating point [34]. Essentially, in stable state, BBR maximizes throughput while minimizing delay and loss. Google has used BBR for a significant fraction of its network traffic both within its datacenters and on its WAN since 2017 [15]. BBR's interaction dynamics with Cubic, the widely used CCA in the Internet, has received intense scrutiny: Some studies observed BBR to be unfair to Cubic, or generally loss-based CCAs. Google, to its credit, has diligently revised BBR's design to address the criticisms. This paper focuses on characterizing the promises and potential of the third, and most recent, revision of BBR—introduced to the public in July 2023. We empirically evaluate BBRv3's performance across a range of network scenarios, e.g., considering different buffer sizes, round-trip times, packet losses, and flow-size distributions. We show that despite the improvements and optimizations introduced in BBRv3, it struggles to achieve an equitable sharing of bandwidth when competing with Cubic, the widely used CCA in the Internet, in a wide range of network conditions.

Keywords: Congestion control · BBR · fairness

1 Introduction

Bottleneck Bandwidth and Round-trip (BBR) is a relatively new congestion control algorithm (CCA) that takes a proactive approach in detecting congestion on a network path. It is designed to enable a sender to converge with a high probability to Kleinrock's optimal operating point [34], which maximizes throughput while minimizing both delay and loss. To this end, BBR estimates the bottleneck bandwidth and round-trip propagation time of a path, and paces the sender appropriately to avoid building any queue along the path. It has fast replaced Cubic for *all* TCP flows on Google's B4 WAN [9,13]; for context, Google's WAN

P. Richter et al. (Eds.): PAM 2024, LNCS 14538, pp. 249–272, 2024.
https://doi.org/10.1007/978-3-031-56252-5_12

traffic makes up 13.85% of all Internet traffic [41]. Since BBR runs purely on the sender side, a sample of Google's edge-to-end-user traffic (i.e., content served on google.com and youtube.com) has been delivered using BBR from as early as 2016 [9]. More recently, Google announced that they use BBR(v3) for *all* internal WAN traffic and public Internet traffic served from google.com [8].

It is quite challenging to design a CCA that works reasonably well in a wide range of network conditions. Traffic characteristics, round-trip times, and bottleneck buffers—to name a few parameters—may differ substantially from one network to another. Unsurprisingly, though Google designed BBR by analyzing traces of world-wide network traffic [13], several independent studies and reports have reported it to be highly unfair to other CCAs in the Internet [2,7,17,26,28,42,43,49,50]. In particular, its incompatibility with Cubic, one of the widely used loss-based CCAs in the Internet, has received much scrutiny [2,17,28]. Google, to its credit, has been diligently evolving BBR to address the concerns raised, and its efforts recently culminated with the release of BBRv3, in 2023 [12]. We focus on the performance and fairness claims (or "promises") of this most recent version of BBR in this work.

BBR builds a model of the network path based on the bottleneck bandwidth and round-trip propagation time measured from each `ACK`. BBRv1 used this model to pace the sender at a rate equal to the estimated bottleneck bandwidth, while keeping the in-flight data quite close to the bandwidth-delay product (BDP). Google reported that BBRv1 achieves high throughput under random packet losses as high as 15% and maintains small queues *regardless* of buffer size [9]. In-depth evaluations revealed, however, BBRv1 to be extremely aggressive when competing with loss-based CCAs and unfair in shallow buffer scenarios [7,26,42,50]. Besides, studies also observed increased retransmission rate [7,26,42], high RTT unfairness and queue occupancy [7,26,49], and ACK aggregation issues in WiFi and cellular networks [11]. BBRv2 was released in 2019 to address these issues; chief among its changes was the inclusion of loss as a congestion signal. It reacted to explicit congestion notifications (ECNs) and optimized the congestion-window (`cwnd`) update logic. Despite the inter-CCA and RTT fairness improvements that Google reported [11,14], independent evaluations showed it suffering from low link utilization and being unfair to loss-based CCAs in deep buffer scenarios [30,37,45]. BBRv3, the most recent release introduced in July 2023, claimed to address these concerns. In particular, BBRv3 (a) claimed to offer quick bandwidth convergence to fair shares both with and without loss or ECN signals and (b) boasted of optimizations to minimize queuing delays and packet losses both during and shortly after slow start (i.e., `STARTUP`) [8].

Thoroughly evaluating a CCA to determine whether it falls short of its promises and, most importantly, along what dimensions and how, is an arduous task. The environment (e.g., a datacenter or private WAN) in which the CCA is originally designed, or for which it was specifically developed, may introduce implicit assumptions (for instance, about network conditions or traffic scenarios), which may not hold in the environment where the CCA is even-

tually deployed (e.g., public Internet). The literature has several examples of CCAs whose behavior in real-world network conditions significantly deviated from that observed in the controlled environments where they were designed and tested (e.g., [20,40,51]). In case of BBR, the conflicting observations between Google's internal evaluations and the various external evaluations—of BBRv1 and BBRv2—clearly attest to the challenges in accurately evaluating its performance. We bridge this conflict by designing a set of experiments—based on first principles—that unequivocally demonstrate whether BBRv3 holds its promises and where there is potential for improvement. Along the way, we outline a systematic and principled approach for evaluating a CCA in practice.

In this work, we empirically analyze the performance of BBRv3 in a wide range of network conditions. In particular, we investigate its claims of quick (throughput) convergence to fair shares in deep and shallow buffer scenarios, and of its improved compatibility (compared to BBRv1 and BBRv2) with loss-based CCAs. An accurate evaluation of these claims has substantial implications for BBR's deployment in the public Internet, since Google is transitioning more and more of its end-user facing traffic to use BBR. Consequently, we evaluate BBR in network scenarios where both long-lived (i.e., elephant) and short-lived (i.e., mouse) flows compete for bandwidth on a shared link. Our choice of using different sizes of flows is based on prior work that demonstrated that while short-lived flows contribute most packets to Internet traffic, long-lived flows contribute most bytes [4,56].

We summarize our contributions as follows.

⋆ We present a first independent empirical evaluation of BBRv3, Google's newest version of BBR, across a range of network conditions and justify the rationale behind our choice of evaluation settings. We focus our evaluation, in particular, on BBRv3's promises to (a) share bandwidth equitably with other loss-based CCAs, particularly in deep buffer scenarios, and (b) assure fairness when contending for throughput with flows that experience different RTTs, a common case in the public Internet where BBRv3 is increasingly being deployed.

⋆ Our evaluations show that BBRv3's throughput unfairness towards loss-based CCAs, i.e., Cubic, is nearly identical to BBRv1's behavior and is exacerbated in shallow buffers. We find that BBR's bias towards long-RTT flows persists in BBRv3, with the unfairness magnified when the difference in RTTs of competing flows is significant. Cubic, the widely used CCA in the Internet, in comparison, offers better throughput for flows with short RTTs than long RTTs.

⋆ We release our network testbed configuration, data sets, and scripts for experiment orchestration and analyses [54].

2 Background

BBR enables a sender to maximize delivery rate while minimizing delay and loss, i.e., converge to Kleinrock's optimal operating point [34]. To this end, it employs a sequential probing state machine that periodically alternates between probing for higher bandwidths (by increasing the delivery rate) and testing for

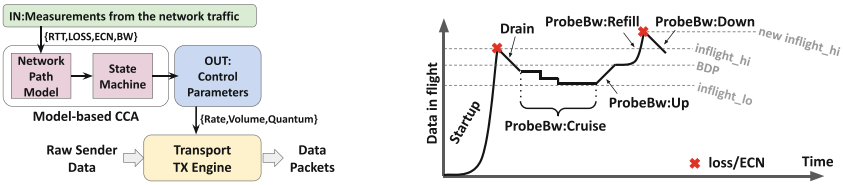
Table 1. An overview of the key changes between the three BBR versions.

Life cycle phases	Property	BBRv1	BBRv2	BBRv3
Startup	`cwnd_gain`	$2/ln\,2$ (~ 2.89)	$2/ln\,2$ (~ 2.89)	2.00
		$2/ln\,2$ (~ 2.89)	$2/ln\,2$ (~ 2.89)	2.77
	Max. `cwnd`	3x`BDP`	-	-
	`inflight_hi`	-	max.`cwnd`	max(est. `BDP`, last `cwnd`)
	Exit	send. rate <25% for 3 consec. RTTs	loss/`ECN` rate >= thresh. (8)	loss/`ECN` rate >= thresh. (6)
Drain		0.35		
	Exit	`cwnd` <= 1x`BDP`		
ProbeBW	Phases	8 fixed gain cycles	*{Cruise, Refill, Up, Down}*	*{Cruise, Refill, Up, Down}*
	*	Cycle=RTprop	`cwnd` limits={`inflight_hi`,`inflight_lo`}	-
	`cwnd_gain`	-	`cwnd_gain(Up)`=2.0	`cwnd_gain(Up)`=2.25
	`pacing_gain`	[1.25,0.75,1,1,1,1,1,1]	`pacing_gain(Down)`=0.75	`pacing_gain(Down)`=0.90
	Exit	`cwnd`>= (`pacing_gain` x `BDP`) *or* loss	loss/`ECN` rate >= thresh	loss/`ECN` rate >= thresh.
ProbeRTT	Frequency	10 s	5 s	5 s
	`cwnd`	4	`BDP`/2	
	Duration	200 ms + `RTprop`	-	-

lower RTTs (by draining the bottleneck queue) of a path. It then uses these bandwidth and RTT measurements to build a path model that determines various congestion control parameters of a TCP implementation such as `cwnd` and pacing rate [9]. Since its introduction in 2016, BBR has underwent three major iterations, and Table 1 summarizes the key differences between these versions.

BBRv1. The first version of BBR used four phases: *Startup*, *Drain*, *ProbeBW* (or *Probe Bandwidth*), and *ProbeRTT* [9]. The Startup phase grows the sending rate exponentially, similar to NewReno, Vegas, and Cubic, but it uses a window increase factor (i.e., `cwnd_gain`) of $2/ln\,2$. Once the bottleneck bandwidth estimate "plateaus" (i.e., three attempts to double the delivery rate results in increasing it by less than 25%), it exits the Startup phase. To drain the queue that the Startup may have built, BBRv1 then enters the Drain phase, by reducing its to 0.75. A BBR flow spends the majority of its time in the ProbeBW phase, where it probes for bandwidth via *gain cycling*. Essentially, BBRv1 cycled through a sequence of eight values for -5/4, 3/4, 1, 1, 1, 1-where a gain higher than one indicates a probe for higher bandwidth and a gain lesser than one represents the attempt to drain any queue built earlier. With the average of the eight values being one, the approach allowed ProbeBW to maintain an average pacing rate that is equal to the estimated bandwidth. BBRv1 continuously sampled the bottleneck bandwidth during this phase. Every 10 s, BBRv1 stopped the ProbeBW phase and entered the ProbeRTT phase. It reduced its `cwnd` to four packets for as long as 200 ms and one round-trip. The low rate drains the bottleneck queue and allows the sender to sample the minimum propagation RTT of the path; it is only in this phase that BBR updates its minimum RTT estimate. Several studies revealed, however, that BBRv1 was extremely aggressive when competing with loss-based CCAs, and highly unfair in shallow buffer scenarios [7,26,28,42,50].

BBRv2. Google introduced BBRv2 in 2019 to alleviate the problems with BBRv1 [13,14]. BBRv2 split the ProbeBW phase into four new sub-phases: *Down*, *Cruise*, *Refill*, and *Up*. Unlike BBRv1, it reacted to loss or ECN sig-



(a) *BBRv3's System Architecture* (b) *Congestion control phases of BBRv3*

Fig. 1. An overview of (a) BBRv3's high-level implementation architecture & (b) the life cycle of the CCA showing how it transitions phases of congestion control.

nals to facilitate an equitable sharing of bandwidths with widely used CCAs such as Cubic and NewReno [14]. In BBRv2, the Startup phase ends not only when the bandwidth estimate "plateaus," but also if loss or ECN mark rate exceeds a threshold (i.e., 2%). It capped the maximum in-flight data volume to the maximum congestion window observed, prior to entering the Drain phase. During ProbeBW, BBRv2 picks the maximum bandwidth measured in the last two ProbeBW attempts instead of the last 10 RTTs as in the case of BBRv1. BBRv2 maintained three different values to bypass the issue of 2xBDP in-flight cap, which was identified in prior work as the primary source of unfairness in BBRv1 [50]: 1.25 for Probe:Down, 0.75 for ProbeBW:Up, and 1 for both ProbBW:Cruise and ProbeBW:Refill. BBRv2 also replaced the 8-phase cycle for ProbeBW with an adaptive probing time, i.e., a value picked at random between 2–3 s, for improving compatibility with Cubic and NewReno.

BBRv3. BBRv3 was introduced in July 2023 at the 117$^{\text{th}}$ IETF meeting [8]. Figure 1 illustrates both the high-level implementation architecture and the life cycle of the CCA, showing how it transitions between the different congestion control phases. BBRv3 is a minor revision of BBRv2, and it addresses two key performance issues of BBRv2. First, an implementation change caused BBRv2 to prematurely exit the ProbeBW phase that in turn prevented it from sharing bandwidth equitably with BBRv1 or loss-based CCAs such as Cubic [8]. The issue resulted in under-utilization of the link even after the link was no longer congested. BBRv3, hence, continues probing for bandwidth until (a) the loss rate or ECN mark rate is below a set threshold (i.e., 2%) or (b) the bandwidth saturates even if the delivery rate is not restricted by the `inflight_hi` threshold. Second, the choice of parameter values (e.g., `cwnd_gain` and) used in the ProbeBW phase made BBRv2 highly unfair: Especially in deep buffer scenarios if the sender received no loss or ECN signals, it caused the sender to be unfair to competing flows [8]. To mitigate these fairness concerns, Google adjusted several parameters, e.g., `cwnd_gain`, , and `inflight_hi`, used in the different congestion control phases, which per their report resulted in improving the CCA's performance [8].

3 Methodology

We empirically evaluate the performance of the three versions of BBR and compare them to that of Cubic using a custom network testbed. The testbed consists of six Dell R6515 blade servers, each with 16 cores and 128 GiB of RAM. We installed one or two Broadcom NetXtreme BCM5720 25 GbE NICs, as required, on each server and used 25 Gbps DACs for interconnecting the servers and four APS Networks BF6064X-T switches (as shown in Fig. 2). We configured the servers to run Linux kernel v5.4 (Debian 10). We used two VMs on one of the servers to evaluate different versions of BBR implementations using the same machine: One VM ran BBRv2 with Linux kernel v5.13.12 from [22], while the other ran BBRv3 with Linux kernel v6.4.0 from [23]. We did not enable ECN support in our testbed, and so neither BBRv2 nor BBRv3 benefited from ECN in our evaluations. Since BBRv3 promises quick (bandwidth) convergence to fair share even without loss or ECN signals [8], we assume such a scenario in our setup and leave the evaluations in the presence of ECN to future work.

To evaluate under a wide range of network conditions, we varied the capacity of the bottleneck link (in red in Fig. 2) and one-way delays as needed for the different experiments using the Linux traffic control (`tc`) utility. Specifically, we used `netem` [47] on an intermediate node, designated as the network latency emulator (NLE), between the end hosts to avoid issues with TCP small queues [31,38]. We also used `tc-tbf` [1] on the NLE to configure the bottleneck bandwidth and buffer size. Unless otherwise stated, we added half of the configured delay before the bottleneck and half after the bottleneck. The base RTT between sender-receiver pairs in the testbed, without any added delay, was 1.5 ms on average.

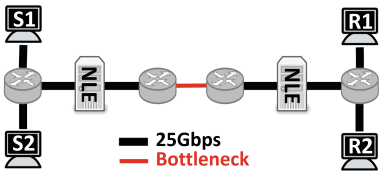
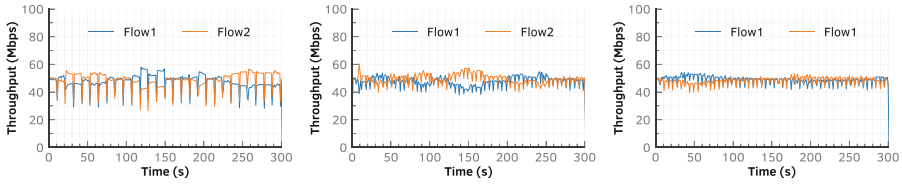


Fig. 2. Network testbed for evaluating performance of CCAs in a single bottleneck setup. (Color figure online)

To emulate traffic representative of real-world network conditions, we generated workloads by using flow sizes from a MAWI trace [32] downloaded on January 5, 2023. We extracted the flows using Zeek [29] and scaled up the inter-arrival times by a factor of 5 to account for the difference in link capacities between the MAWI infrastructure and our testbed. The flow sizes in this data set followed the Pareto distribution, while the inter-arrival times followed the Lognormal distribution. We then used Harpoon [44] to generate traffic using samples from these empirical distributions. Unless otherwise stated, we repeated each experiment three times and reported the statistics across the three runs.

4 Evaluation

We now evaluate the three versions of BBR and Cubic in a wide range of network conditions. Our goal is to ascertain the ability of each CCA to share the bandwidth in a fair or equitable manner with other flows-regardless of whether the



(a) *BBRv1* (b) *BBRv2* (c) *BBRv3*

Fig. 3. Throughput achieved by two similar, synchronous (i.e., starting at the same time) flows when competing for bandwidth on a bottleneck with a deep buffer of size 16×BDP.

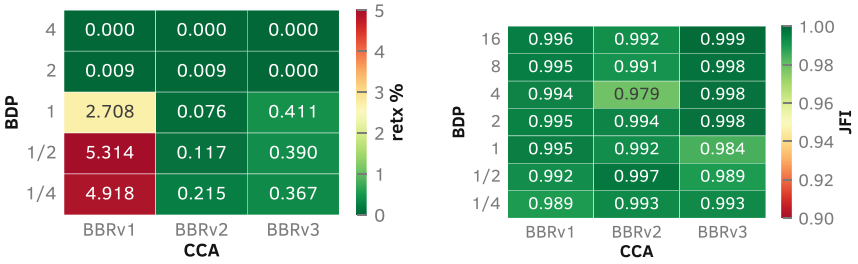
contending flows use the same CCA or a different one. To this end, we evaluate fairness in two broad settings: *similar* and *dissimilar* flows. Similar flows use the same CCA and experience the same RTT, but may vary from one another in size or arrival times (at a bottleneck link). Dissimilar flows, in contrast, differ from one another based on one or more factors, e.g., choice of CCAs and RTTs experienced by the flows.

4.1 Similar and Synchronous Flows

We begin our evaluation with a simple scenario where two similar flows, i.e., using the same CCA and experiencing the same RTT (of 100 ms), contend for bandwidth on a bottleneck (refer Fig. 2). We start both flows *at the same time* and let them run for a duration of 300 s and plot the throughput experienced by the flows as a function of time in Fig. 3. Overall, all BBR variants achieve an equitable bandwidth share when competing with a similar and synchronous flow. BBRv3 significantly reduces the throughput oscillations and allows the flows to converge much quicker than either of the earlier two versions. We repeated this experiment by varying the bottleneck queue size from one-fourth to 16 times the BDP, doubling the queue size once for each experiment. Although not shown here, our inferences hold across all these settings.

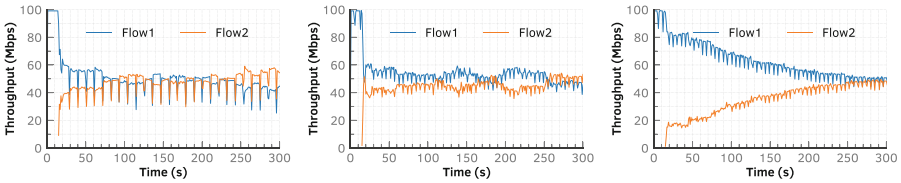
Figure 4a indicates whether the flows experience high retransmissions when competing with another similar, synchronous flow.[1] The heatmap reports the percentage of retransmissions (i.e., ratio of the aggregate number of retransmitted packets to the total number of packets delivered by both flows) experienced by the two flows; higher values indicate higher retransmissions, and, hence, lower values (in green) imply better throughput or utilization. Per this figure, although BBRv1 experiences high retransmissions when competing with another BBRv1 flow started at the same time in low buffer settings, across all other settings, the CCAs converge quickly without incurring much loss. This observation of high retransmissions when using BBRv1 in shallow buffers is in line with observations made by prior work [7,26,42]: They are due to BBRv1' aggressive probing

[1] We omit the results for 8 × BDP and 16 × BDP as we see virtually no retransmissions in such settings.



(a) *Retransmissions* (b) *Jain's fairness index (JFI)*

Fig. 4. (a) Retransmissions experienced by and (b) Jain's fairness index (JFI) of two similar, synchronous flows competing for bandwidth on a bottleneck link as a function of bottleneck queue size.



(a) *BBRv1* (b) *BBRv2* (c) *BBRv3*

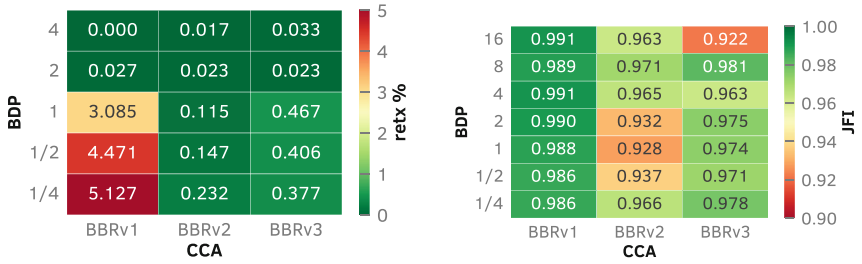
Fig. 5. Throughput achieved by two similar but staggered flows (i.e., second flow starts 15 s after the first) when competing for bandwidth on a bottleneck with a deep buffer of size 16×BDP.

behavior and its indifference to loss. The Jain's fairness index (JFI) for the flows in Fig. 4b, however, shows that all three versions equitably share the bottleneck with a similar and synchronous flow, regardless of bottleneck buffer size; BBRv3 shows only a marginal improvement, if any, over the prior two versions.

Takeaways. BBR achieves an equitable sharing of bandwidth with similar synchronous flows, and BBRv3 address the well-documented issues of aggressiveness and unfairness in BBRv1.

4.2 Similar But Staggered Flows

We now repeat the earlier evaluation, but with one change: We *stagger* the flows so that the second flow starts 15 s after the first. In the Internet flows may arrive at a bottleneck at random times, when other contending flows on the link have already started or reached their stable state. The staggering of flows simply emulates this typical traffic condition.



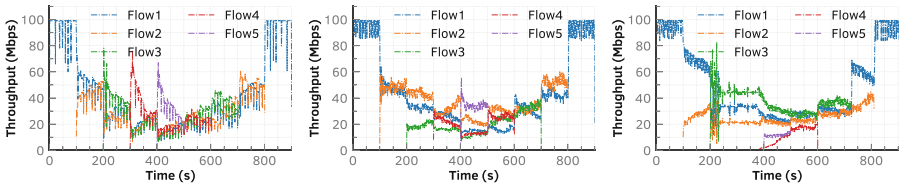
(a) *Retransmissions* (b) *Jain's fairness index (JFI)*

Fig. 6. (a) Retransmissions experienced by and (b) JFI of two similar but staggered flows (i.e., second one starts 15 s after the first flow) competing for bandwidth on a bottleneck link as a function of bottleneck buffer size.

Per Fig. 5, BBR implementations do not converge quickly to an equitable share when flows do not start at the same time—quite unlike the case of similar and synchronous flows (Sect. 4.1). For instance, when a BBRv3 flow joins 15 s later at a bottleneck link with a 16×BDP buffer, it takes *more than* 4 *minutes* for it to achieve an equitable bandwidth sharing. Such large buffers are not uncommon in the internet: Router manufacturers may (by default) provide large buffers [19], and administrators may configure larger buffers on transcontinental links to cope with the high RTTs [16] or improve video QoE [46].

We observe that when flow starts are staggered by an interval of 15 s, the flows experience higher retransmissions (Fig. 6a) than when they are started at the same time.[2] The heatmap in the plot reports, as in Sect. 4.1, the percentage of retransmissions experienced by the two flows; higher values (in yellow and red) indicate higher retransmissions. BBRv1 in particular suffers high retransmissions in shallow buffer settings, again owing to its aggressive probing behavior and indifference to losses. A key change in BBRv2 (compared to BBRv1) was the addition of using loss as a congestion signal; this change enables it to reduce the loss rates in shallow buffer settings and avoid unnecessary retransmissions [10]. BBRv2 experiences, as a result, the lowest retransmissions of all three implementations. BBRv3 inherits BBRv2's changes and, as a consequence, experiences low retransmissions, which matches Google's claims about BBRv3 [8]. We observe that BBRv3 experiences more retransmissions than BBRv2 in shallow buffer settings, but the former achieves better fairness than the latter across a range of buffer sizes (Fig. 6b), with the exception of the largest buffer size (of 16-times BDP). BBRv1, which does not use packet loss to infer congestion, leads to higher fairness than BBRv2 and BBRv3, both of which take loss into consideration. These observations could partly be explained by BBRv3's less aggressive behavior compared to BBRv1 (compare BBRv3's `cwnd_gain` and `pacing_gain` to those of BBRv1 in Table 1).

[2] As before, we omit the results for 8 times BDP and 16 times BDP as we see virtually no retransmissions in such settings.



(a) *BBRv1* (b) *BBRv2* (c) *BBRv3*

Fig. 7. When we stagger the arrivals and departures of five flows at a bottleneck link of 100 Mbps and buffer size 32×BDP by 100 s, (a) BBRv1 flows achieve fair throughput shares faster than (b) BBRv2 and (c) BBRv3.

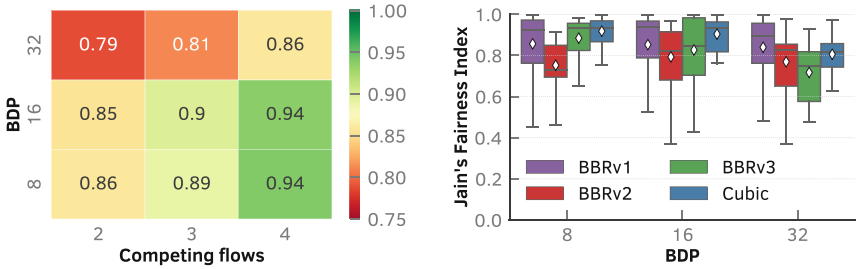
(a) *BBRv1* (b) *BBRv2* (c) *BBRv3*

Fig. 8. When we stagger the arrivals and departures of five flows at a bottleneck link of 100 Mbps by 100 s, (a) BBRv1 flows achieve better shares of throughput than (b) BBRv2 and (c) BBRv3.

Takeaways. While BBR's refinements seem to allow it reduce losses, with recent versions taking losses into account to control their probing, even the most recent version of BBR struggles to achieve high fairness when flow arrivals are staggered by a few seconds. This observation has crucial implications for BBRv3's adoption in the Internet, since in the public Internet arrivals of competing flows might often be staggered with respect to one another.

4.3 Co-existence in Deep Buffer Scenarios

To evaluate how quickly different CCAs converge to fair bandwidth share, we run a staggered-flow experiment (similar to that of HPCC [35]), where we introduce five flows, one after another. Again, the flows are similar and we add a new flow to the testbed every 100 s until we reach five concurrent flows, and then remove one flow every 100 s until we have only one left. This experiment enables us to compare and contrast how quickly different CCAs react to dynamic changes in traffic conditions and converge to the fair share of the bottleneck (100 Mbps). In this evaluation, we specifically focus on deep buffer scenarios to verify Google's claim that BBRv3 can equitably share bandwidth in such conditions [8].



(a) *Cubic* (b) *At t=400-500 s*

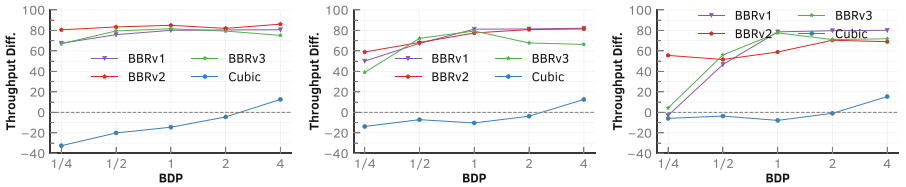
Fig. 9. (a) Cubic offers improved fairness as the number of flows increases, and (b) when all five flows compete at the bottleneck with a deep (32×BDP) buffer, BBRv3 has the worst fairness than any other CCA.

Figure 7 shows the timelines of flows in the staggered-flow experiment in a deep-buffer (i.e., 32×BDP) scenario for all three BBR versions. Per this figure, BBRv3 struggles to achieve a fair share between the flows; when all five flows compete at the bottleneck, BBRv3 performs the worst compared to both the previous versions. BBRv1 flows converge quickly to fair share both when new flows arrive and old ones leave; its quick convergence to fair share is perhaps largely due to its aggressive probing behavior, which can efficiently discover bandwidth changes. The average JFI across three runs (Fig. 8) shows that BBRv3 performs better than BBRv2, but poorer than BBRv1. Cubic, in contrast to BBRv3, performs poorly with a small numbers of flows, although it improves with increasing number of flows (Fig. 9a). Unlike Cubic, BBRv3's performance degrades with increasing number of flows. If we analyze fairness during the period when all five flows are competing for bandwidth (Fig. 9b), BBR typically performs poorer than Cubic on average, exhibiting large variations in performance. Particular in deep-buffer settings, BBRv3 performs worst compared to both Cubic and earlier versions of BBR, perhaps because of its less aggressive choice of probing parameters.

Takeaways. While BBRv3's intra-protocol fairness in deep buffer scenarios is better than that of BBRv2, its fairness is poorer than that of BBRv1 and seems to worsen as the number of flows increases (in deep-buffer scenarios). BBRv3 choice of less aggressive probing parameters compared to BBRv1 seems to be the reason behind the observed behavior.

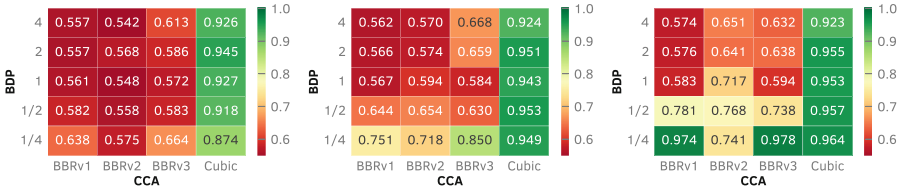
4.4 Dissimilar Flows: RTT Fairness

Evaluations concerning fairness of CCAs in prior work (e.g., [7,50]) usually consider flows with similar RTTs. Flows in the Internet typically have different RTTs, which make it challenging in terms of fairly sharing bandwidth when such flows interact at a bottleneck link, even if they all use the same CCA. The RTT dictates a CCA's reaction to bandwidth limitations or to other flows, since



(a) *TD; 160 ms vs 10 ms* (b) *TD; 160 ms vs 20 ms* (c) *TD; 160 ms vs 40 ms*

Fig. 10. Average throughput difference (TD); 160 ms flow starts first, then 15 s later the (10 ms, 20 ms, 40 ms) flow starts.



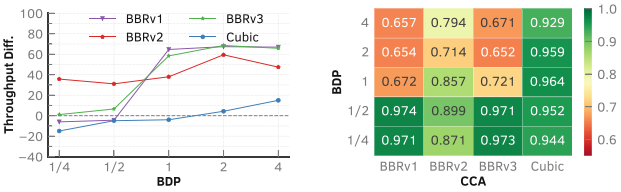
(a) *JFI; 160 ms vs 10 ms* (b) *JFI; 160 ms vs 20 ms* (c) *JFI; 160 ms vs 40 ms*

Fig. 11. Average Jain's fairness index (JFI); 160 ms flow starts first, then 15 s later the (10 ms, 20 ms, 40 ms) flow starts.

the signals typically used by a sender (e.g., packet loss or delay) for determining how much it can send and/or how fast, must traverse the path from the bottleneck to the receiver, which then echoes it back to the sender.

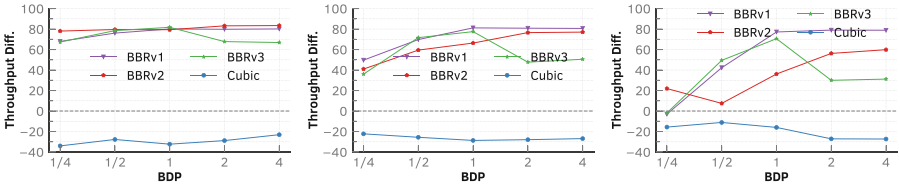
With this experiment, we evaluate the bandwidth share when flows using the same CCA traverse different routing paths, thus experiencing different RTTs, and sharing a common bottleneck along the way. We run two flows using the same CCA, where the first flow is our base flow with a fixed RTT of 160 ms and the second flows experiences a different RTT—one of 10, 20, 40, 80 ms, each constituting a different experiment. In the first experiment scenario, the base flow, 160 ms, starts first and the second flow joins after 15 s (Fig. 10, Fig. 11 and Fig. 12). In the second scenario, the base flow joins 15 s after the flow with the shorter RTT (Fig. 13, Fig. 14 and Fig. 15). These scenarios help us characterize how the different CCAs behave when the short (or long) RTT flow joins the network, while the other flow has already reached a steady state; we can safely assume a flow has finished slow start after 15 s.[3] Specifically, we calculate the throughput difference between the base flow and the second flow, as follows: $BaseFlow_{AvgTput} - SecondFlow_{AvgTput}$. A positive value implies that the base flow obtains the majority of the bandwidth, while a negative value indicates that the second flow receives higher bandwidth than the first. Additionally, we also compute the Jain's fairness index to quantify the CCA's fairness.

[3] We do not disturb a flow while in slow start to ensure that it does not exit slow start prematurely, which would result in link underutilization.



(a) *TD; 160 ms vs 80 ms* (b) *JFI; 160 ms vs 80 ms*

Fig. 12. Average throughput difference (TD) and Jain's fairness index (JFI); 160 ms flow starts first, then 15 s later the 80 ms flow starts.

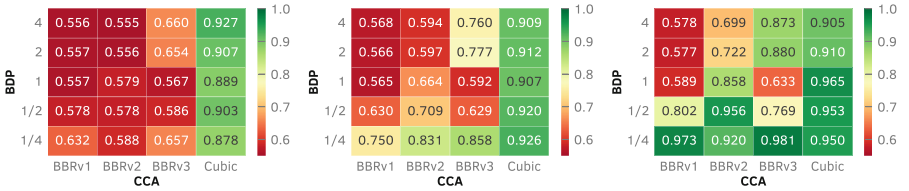


(a) *TD; 10 ms vs 160 ms* (b) *TD; 20 ms vs 160 ms* (c) *TD; 40 ms vs 160 ms*

Fig. 13. Average throughput difference (TD); short RTT flow (10 ms, 20 ms, 40 ms) starts first, then 15 s later the 160 ms flow starts.

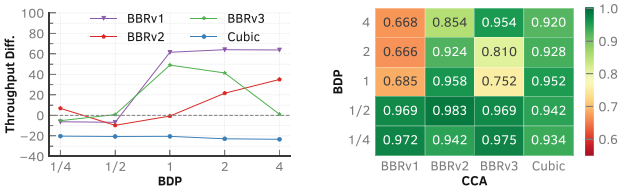
When we start the long-RTT flow first, all versions of BBR favor the long-RTT flow (Fig. 10); it receives much higher bandwidth than the second flow. This bias towards the long-RTT flow (starting first) diminishes a bit when the difference in flow RTTs decreases (Fig. 10c); BBRv2 and BBRv3 (which is largely similar to BBRv2) perform much better than BBRv1 when RTT differences between the base flow and second flow decreases (Fig. 12), perhaps owing to their less aggressive design. These observations also hold if we start the short-RTT flow first and then the long-RTT (or base) flow (Fig. 13 and Fig. 15). Our inferences are in agreement with prior work that demonstrated BBR favoring long-RTT flows over short-RTT flows [26,49], which is quite different from what has been observed in case of AIMD mechanisms [6,25]; our results confirm that BBRv3's behavior does not vary substantially from BBRv2 with respect to preferring long-RTT flows over short-RTT flows. Cubic, unlike BBR, favors the second (short-RTT) flow over the base (long-RTT) flow, even though the second flow starts after the base flow, across almost all buffer settings.

All BBR versions exhibit high RTT unfairness when the RTT difference between the base and second flow is significant, regardless of which flow starts first (Fig. 11, Fig. 12, Fig. 14, and Fig. 15). Unlike BBR, Cubic achieve high fairness across all settings, regardless of how we size the bottleneck buffer. BBR's bias towards long-RTT flows is less pronounced in shallow buffers (if the RTTs of the flows do not differ substantially), where the short-RTT flow is able to compete for a better bandwidth share with the long-RTT flow.



(a) *JFI; 10 ms vs 160 ms* (b) *JFI; 20 ms vs 160 ms* (c) *JFI; 40 ms vs 160 ms*

Fig. 14. Average Jain's fairness index (JFI); short RTT flows (10 ms, 20 ms, 40 ms) start first, then 15 s later the 160 ms flow starts.

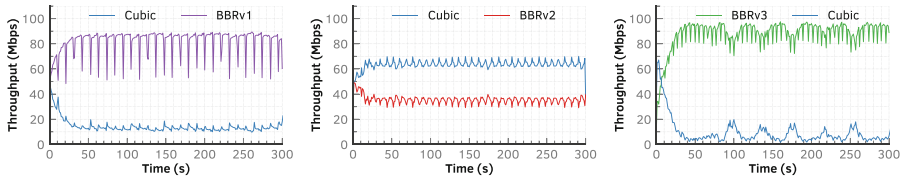


(a) *TD; 80 ms vs 160 ms* (b) *JFI; 80 ms vs 160 ms*

Fig. 15. Average throughput difference (TD), 80 ms flow starts first, then 15 s later the 160 ms flow starts.

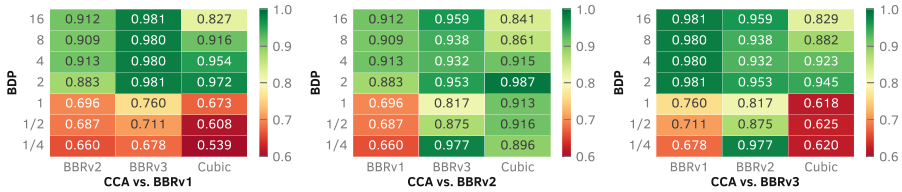
Fairness is improved for all BBR versions if the short RTT flow starts first, as it is able to grow its `cwnd` before the long RTT flow starts. Additionally, the smaller the gap between the two flow's RTT values, the better the fairness as both flows are able to estimate the bottleneck bandwidth and RTT at the same rate (Fig. 12 and Fig. 15). When BBR flows with different RTTs compete for bandwidth, the short-RTT flow is likely to observe the increase in queuing faster than the long-RTT flow (since the rate at which a sender can observe such signals depend on the RTT). Furthermore, since BBR determines the delivery rate based on such signals, the short-RTT flow may consistently slow down in response to any observed queuing before the long-RTT flow reacts. When flows eventually synchronize, short-RTT flows regain some of the bandwidth share, but the situation reverts to benefitting the long-RTT flow again as they probe for bandwidth.

Takeaways. BBR's bias towards long-RTT flows is well-known, and our evaluations confirm that BBRv3 improvements or optimizations do not change the status quo. BBR offers poor fairness across flows that differ substantially in RTT, which might be typical in the Internet. Cubic, the widely used CCA in the Internet, unlike BBR, offers higher fairness than BBR in nearly all our evaluations scenarios with flows with different RTTs.



(a) *BBRv1* (b) *BBRv2* (c) *BBRv3*

Fig. 16. Throughput of a BBR flow when competing for bandwidth with a Cubic flow on a bottleneck link with a 1×BDP buffer.



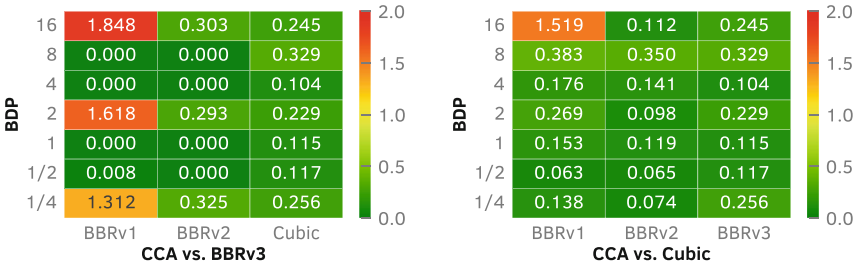
(a) *BBRv1 vs. others* (b) *BBRv2 vs. others* (c) *BBRv3 vs. others*

Fig. 17. Jain's fairness index (JFI) of two flows using different CCAs and competing on a bottleneck with a 1×BDPbuffer.

4.5 Dissimilar Flows: Inter-CCA Fairness

We now evaluate BBRv3's ability to share bandwidth equitably when competing with flows using CCAs other than BBRv3. With more than 40% of current Internet traffic estimated to be delivered using BBR [36], it is quite important to characterize how BBRv3 behaves when competing with non-BBRv3 flows already prevalent in the Internet. Such an inter-CCA characterization is also timely and crucial for network operators and generally the networking community, since Google intends to submit BBRv3 soon for inclusion in the Linux kernel [8]. To characterize BBRv3 flows' ability to co-exist with non-BBRv3 flows, we run two (co-existing) long-lived flows, each using a different CCA; we start these flows at the same time. We set the RTT to 100 ms and the bandwidth to 100 Mbps, and we vary the bottleneck buffer size as a function of BDP.

BBRv1 has long been reported to be extremely unfair to loss-based CCAs such as Cubic [7,26,49,50]. Figure 16b, hence, simply confirms this well-established claim. BBRv2 seems to fare well than BBRv1 (Fig. 16b), since unlike the former, the latter takes packet loss into consideration. BBRv2 improves upon BBRv1 with respect to its ability to share bandwidth with loss-based CCAs, even allowing Cubic to obtain more than its share of fair bandwidth when competing with BBRv2 flows. BBRv3, which Google claims will quickly converge to fair share, performs even slightly worse than BBRv1 (Fig. 16c). Even if flows experience same RTTs, BBRv3, in its current form, offers no room for Cubic flows to co-exist; any RTT differences between the flows may only exacerbate this situation.



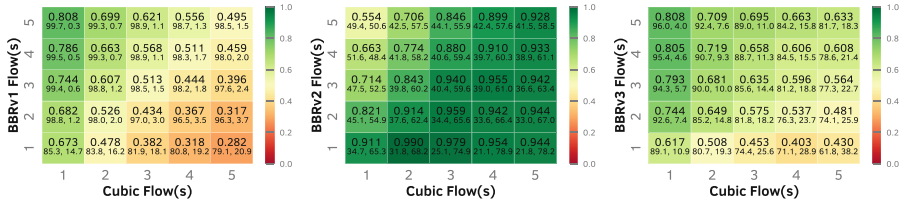
(a) *BBRv3* (b) *Cubic*

Fig. 18. Retransmissions when (a) a BBRv3 flow and (b) a Cubic flow compete against flows using other CCAs.

Figure 17 shows the Jain's fairness index for the competing flows to characterize BBR's inter-CCA fairness. In shallow-buffer scenarios, BBRv1 is quite unfair to all others. No other CCA, not even other versions of BBR, is able to obtain a fair share when competing with BBRv1 (Fig. 17a); increasing bottleneck buffer sizes, nevertheless, seems to alleviate this situation substantially. This behavior is presumably due to BBRv1 only backing off when its large in-flight data cap (3×BDP) is exceeded (refer Table 1), ignoring packet loss unlike BBRv2 and BBRv3. Unlike BBRv1, BBRv2 performs quite well (with high JFI values) across all bottleneck buffer settings (Fig. 17b) primarily because of its ability to reach to packet loss. It can equitably share bandwidth with Cubic, making it safer than BBRv1 for deployment in the public Internet where the latter is quite prevalent. BBRv3, despite being only a minor revision of BBRv2, performs quite poorly, especially against Cubic (Fig. 17c). In deep-buffer settings, Cubic is able to send more data into the network before experiencing congestion (i.e., packet loss); naturally, it obtains a higher bandwidth compared to shallow-buffer settings and, as a result, the fairness index improves.

Overall, while the design of BBRv3 (which it inherits from BBRv2) represents a substantial progress (in creating a fair and safe CCA), the performance tweaks [8] or optimizations (refer Table 1) that Google has introduced in BBRv3 are exactly the opposite. With regards to Google's claims about reduced packet loss, we observe frequent retransmissions (Fig. 18) when BBRv3 competes against Cubic, comparable almost to those when BBRv1 competes against Cubic in both deep and shallow-buffer settings.

Thus far we focused on a scenario where a single BBR flow competes with a single Cubic flow for bandwidth on a bottleneck. Previous studies showed that even when competing with several Reno or Cubic flows a single BBRv1 flow was able to grab most of the link bandwidth [40,50]. We revisit this evaluation where we pit a single BBR flow against several Cubic flows. More specifically, we conduct a series of experiments where we vary the number of BBR flows as well as competing Cubic flows from 1 through 5, and characterize the fairness (measured via JFI) across the competing flows in each experiment (Fig. 19). As



(a) *BBRv1 vs. Cubic* (b) *BBRv2 vs. Cubic* (c) *BBRv3 vs. Cubic*

Fig. 19. JFI when 1–5 Cubic flows compete against 1–5 (a) BBRv1, (b) BBRv2 and (c) BBRv3 flows.

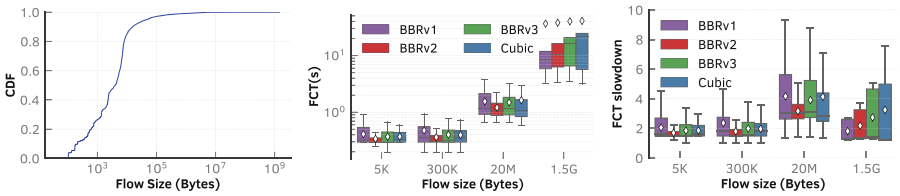
before, we set the link bandwidth to 100 Mbps, the RTT to 100 ms, and the buffer size to 1×BDP(i.e., 1250 KB). We start all the flows at the same time and run them for 360 s. Figure 19 shows the JFI of the experiments with the three BBR versions competing against Cubic. The top value in each cell is the mean JFI value over time, concatenated over three experiment runs, while the bottom value show the mean throughput shares of BBR and Cubic.

In Fig. 19a, JFI values become smaller (i.e., fairness worsens) as we move from top to bottom or right to left. This observation confirms the observations from prior work that even a single BBRv1 flow can outcompete multiple Cubic flows [40,50]. BBRv2's design completely reverses this behavior (Fig. 19b); when we increase the number of Cubic flows fairness improves, emphasizing once again that we can safely deploy BBRv2 in the public Internet. BBRv3, in contrast, resurrects BBRv1's aggressive behavior (Fig. 19c); it offers lower fairness than BBRv2, although the Cubic flows seem to be able to obtain a higher bandwidth when competing against BBRv3 than when competing against BBRv1.

Takeaways. While BBRv2 makes significant improvements towards achieving equitable bandwidth sharing when competing against loss-based CCAs, BBRv3 seems to undo most, if not all, such improvements. BBRv3 does not equitably share bandwidth with loss-based CCAs such as Cubic, and its behavior in some instances is even worse than that of BBRv1. Despite its ability to react to packet loss, it remains highly unfair to Cubic flows, in shallow buffers.

4.6 Dissimilar Flows: Real-World Workloads

Until now, we used a small, fixed number of (long-lived) flows in our evaluations. Now, we turn to using realistic traffic workloads to quantify BBR's performance in real-world network conditions. Specifically, we use a distribution of flows comprising both short-lived (i.e., mouse) and long-lived (i.e., elephant) flows. We sample the size of these flows from an empirical distribution of flow sizes observed in the Internet (Fig. 20a). The CDF in Fig. 20a depicts the flow size distribution from a MAWI trace (refer Sect. 3). We use the Harpoon [44] traffic generator to sample the flow sizes and connection times from this trace data. We fixed the random seed used in the sampling to ensure that the sampled



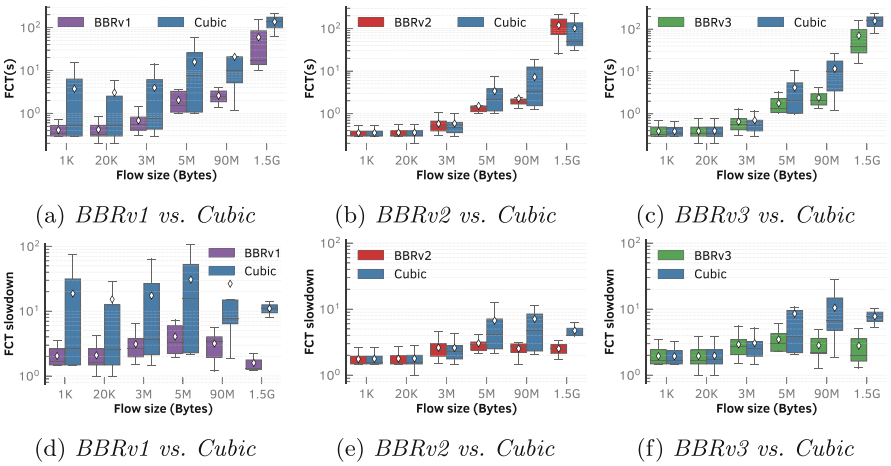
(a) *flows in MAWI trace* (b) *FCTs* (c) *FCT slowdowns*

Fig. 20. (a) Distribution of flow sizes in the MAWI trace; Flow completion times (FCTs) and FCT "slowdown" for various flow sizes across different CCAs.

distribution of flow sizes remains the same across the evaluations of all CCAs; we can then compare the performance of the same set of flows across different CCAs and determine which CCA performs the best. We conduct two distinct experiments using the topology shown in Fig. 2 with the bottleneck bandwidth set to 100 Mbps, RTT to 100 ms, and the buffer size to `BDP`. In the first, we set all flows to use the same CCA, and in the second, we configure half of the flows to use one of the BBR versions and the other half to use Cubic. We then analyze the flow completion times (FCTs) as well as the FCT "slowdowns" [53] computed by normalizing the measured FCT by the theoretically optimal FCT obtained by taking into account the flow size, the bandwidth, and the RTT.

When All Flows Use the Same CCA. In case of BBRv1, the FCTs of short flows, i.e., flows smaller than 300 KB, are comparatively longer than those of the long flows (Fig. 20b). The FCT slowdowns for the longest flows in Fig. 20c when compared to those of the smaller flows clearly show that BBRv1 prioritizes long flows over their shorter counterparts. BBRv2, in contrast, offers lower FCT for short flows than for long flows. Lastly, BBRv3 offers smaller FCT slowdowns for short flows than for long flows, but its FCT slowdowns are consistently higher than those of BBRv2.

When Flows are Split Equally Between Two CCAs. To evaluate how BBR behaves when competing with Cubic (similar to the earlier evaluations in Sect. 4.5) in realistic traffic conditions, we configure half of the flows in the workload to use one of the BBR versions and the other half to use Cubic. As before, we plot the FCTs and FCT slowdowns of the flows grouped into buckets of different sizes in Fig. 21. The FCTs and FCT slowdown values corresponding to the experiment where BBRv1 flows compete with Cubic (Fig. 21a and Fig. 21d) are not surprising; BBRv1 is quite unfair to Cubic, as amply demonstrated in Sect. 4.5, and its ability to seize bandwidth aggressively from Cubic flows result overall in substantially smaller FCTs and FCT slowdowns for BBRv1 flows than Cubic flows, regardless of flow size. BBRv2 is far less aggressive to Cubic than BBRv1 (Fig. 21b and Fig. 21e). Short flows of both CCAs experience similar FCT slowdowns, while long flows using BBRv2 fare better than those using Cubic. BBRv3's performance is similar to that of BBRv2, except in case



(a) *BBRv1 vs. Cubic* (b) *BBRv2 vs. Cubic* (c) *BBRv3 vs. Cubic*

(d) *BBRv1 vs. Cubic* (e) *BBRv2 vs. Cubic* (f) *BBRv3 vs. Cubic*

Fig. 21. Flow completion times (FCTs) and FCT "slowdowns" for flows of varying sizes split between two CCAs.

of long flows. (Fig. 21c and Fig. 21f) Long flows using Cubic experience higher FCTs and FCT slowdowns when competing against BBRv3 than against BBRv1.

Takeaways. The evaluations confirm the highly aggressive and unfair behavior of BBRv1 towards loss-based CCAs such as Cubic. BBRv2's design significantly improves that status quo, making Cubic flows achieve nearly their fair share when competing with BBRv2 flows. While BBRv3 behaves similar to BBRv2 for short flows allowing both Cubic and BBRv3 flows to experience similar FCT slowdowns, long flows using BBRv3 experience much smaller FCT slowdowns than those using Cubic.

5 Related Work

There is an extensive body of prior work on congestion control in the Internet. Google's BBR is a relatively new addition and its design intelligently combines several good ideas from the literature. Several independent studies have examined BBR's performance in various scenarios and compared it with that of other CCAs, partly perhaps because of how quickly Google has been transitioning large volumes of traffic to use BBR [8]. Below, we briefly review these prior work.

Many prior work such as [7,26,42,49,50] conducted comprehensive evaluations of BBRv1, focusing on its interactions with other well-known and widely used CCAs, e.g., NewReno, Cubic, and Vegas. Past evaluations of BBRv1 considered both similar and dissimilar flows (i.e., flows with varying RTTs or using different CCAs). Some investigated BBRv1's performance in both shallow and deep buffer scenarios [7,26,50], and some others analyzed the impact of extremely long delays, which are typical in satellite networks [18]. Prior studies identified several shortcomings with BBRv1, including unfairness to loss-based CCAs in

shallow buffers, frequent retransmissions, high RTT unfairness as well as high queue occupancy. The scope of virtually all such evaluations was, however, limited to long-lived flows. Hurtig et al. tested BBRv1 with various buffer sizes and link bandwidths and used a workload comprising a mix of bulk as well as short transfers [27], but they used fixed sizes (instead of sampling them from observed traffic distributions) for long-lived and short-lived flows.

Active queue management (AQM) techniques are designed for mitigating congestion [3], and several prior work have studied the impact of deploying AQMs on the performance of different CCAs (e.g., [24]). In the case of BBR, studies have shown that AQMs such as FQ-CoDel improve the performance of both BBRv1 and BBRv2 in deep buffer scenarios [21,33,55]. We consider the analyses of AQM deployments on BBRv3 as orthogonal to this work.

BBRv2 uses ECN signals to detect queuing, and Tahiliani et al. [48] showed that BBRv2 reduces queue occupancy at the bottleneck, when ECN was enabled. Multiple studies demonstrated that BBRv2 has lower link utilization than BBRv1 owing to its conservative behavior during bandwidth probing and that both versions were unfair to loss-based CCAs in deep buffer scenarios [30,37,45,52]. We did not evaluate the benefits of ECN for BBRv3 in this study, but we leave that for future work.

Some recent work also attempted to address the BBR's shortcomings. Bi et al. [5], for instance, focused on BBRv1's frequent retransmissions issue. They proposed dynamically adjusting the in-flight data cap based on the estimate bottleneck buffer size from RTT samples as well as packet losses. BBRv2+ used path delay information for tuning the aggressiveness of bandwidth probing [52]. They used Mahimahi [39] for evaluating BBRv2+ in emulated WiFi and LTE networks. While we do not propose solutions for addressing the shortcomings in BBRv3, we hope our comprehensive evaluations and detailed analyses will pave the way for designing and validating different solutions.

6 Concluding Remarks

The bottleneck bandwidth and round-trip propagation time (BBR) algorithm is a relatively new congestion control algorithm (CCA). BBR's design eschews the typical congestion signals (e.g., transient queue delays and losses) used in the vast majority of prior work. It measures instead bottleneck bandwidth and round-trip times on each `ACK` and paces the sender to retain the delivery rate close to the bottleneck bandwidth and avoid any queue build up. Several studies, however, demonstrated this initial design to be highly unfair to loss-based CCAs such as Cubic. Google responded to the criticisms by evolving the design and releasing newer versions of BBR. BBRv3, released in July 2023, is the most recent version in the evolution of BBR.

Given the increasing volume of traffic that Google has been transitioning to use BBR and its current efforts to include BBRv3 in the Linux kernel, we turned our attention to performing a systematic and rigorous evaluation of BBRv3 in a range of realistic network conditions and traffic scenarios. Specifically, we

checked whether Google's claims of BBRv3's fairness towards other CCAs hold water in our evaluations. While BBRv3 has evolved substantially compared to the earlier versions, in our evaluations BBRv3 struggles to achieve an equitable bandwidth sharing with competing flows. Even when competing with similar flows (i.e., flows using BBRv3) a small difference in arrival times causes BBRv3 to incur substantial delays in bandwidth convergence. Despite the revisions, optimizations, and bug fixes, BBRv3's highly competitive behavior stifles Cubic on a bottleneck link, particularly in shallow buffer settings. These results have crucial implications for BBRv3's adoption and deployment in the public Internet. We release our testbed configuration and scripts for running and analyzing the experiments as open source artifacts [54], and we hope our efforts encourage the community to think about an objective template or framework for evaluating modern CCAs.

References

1. Kuznetsov, A.N., Hubert, B.: TC-TBF(8) - Linux man page (2023). https://linux.die.net/man/8/tc-tbf. Accessed 1 Oct 2023
2. Mishra, A., Zhang, J., Sim, M., Ng, S., Joshi, R., Leong, B.: Conjecture: existence of nash equilibria in modern internet congestion control. In: Proceedings of the 5th Asia-Pacific Workshop on Networking, APNet 2021 (2022)
3. Baker, F., Fairhurst, G.: IETF Recommendations Regarding Active Queue Management. RFC 7567 (2015)
4. Bauer, S., Jaeger, B., Helfert, F., Barias, P., Carle, G.: On the evolution of internet flow characteristics. In: Applied Networking Research Workshop (ANRW) (2021)
5. Bi, P., Xiao, M., Yu, D., Zhang, G.: oBBR: optimize retransmissions of BBR flows on the internet. In: USENIX Annual Technical Conference (ATC) (2023)
6. Brown, P.: Resource sharing of TCP connections with different round trip times. In: IEEE International Conference on Computer Communications (INFOCOM) (2000)
7. Cao, Y., Jain, A., Sharma, K., Balasubramanian, A., Gandhi, A.: When to use and when not to use BBR: an empirical analysis and evaluation study. In: ACM Internet Measurement Conference (IMC) (2019)
8. Cardwell, N., et al.: BBRv3: algorithm bug fixes and public internet deployment. Technical report, Internet Engineering Task Force (IETF) (2023). https://datatracker.ietf.org/meeting/117/materials/slides-117-ccwg-bbrv3-algorithm-bug-fixes-and-public-internet-deployment-00
9. Cardwell, N., Cheng, Y., Gunn, C.S., Yeganeh, S.H., Jacobson, V.: BBR: congestion-based congestion control. ACM Queue **14**(5), 20–53 (2016)
10. Cardwell, N., et al.: BBR congestion control: IETF 100 update: BBR in shallow buffers. Technical report, Internet Engineering Task Force (IETF) (2017). https://www.ietf.org/proceedings/100/slides/slides-100-iccrg-a-quick-bbr-update-bbr-in-shallow-buffers-00.pdf
11. Cardwell, N., et al.: BBR congestion control work at google IETF 101 update. Technical report, Internet Engineering Task Force (IETF) (2018). https://datatracker.ietf.org/meeting/101/materials/slides-101-iccrg-an-update-on-bbr-work-at-google-00

12. Cardwell, N., et al.: BBR Congestion Control: Fundamentals and Updates (2023). https://research.cec.sc.edu/files/cyberinfra/files/BBR%20-%20Fundamentals%20and%20Updates%202023-08-29.pdf
13. Cardwell, N., Cheng, Y., Yeganeh, S.H., Swett, I., Jacobson, V.: BBR Congestion Control. Internet-Draft draft-cardwell-iccrg-bbr-congestion-control-02, Internet Engineering Task Force (IETF) (2022). https://datatracker.ietf.org/doc/draft-cardwell-iccrg-bbr-congestion-control/02/
14. Cardwell, N., et al.: BBR v2: a model-based congestion control IETF 105 update. Technical report, Internet Engineering Task Force (IETF) (2019). https://datatracker.ietf.org/meeting/105/materials/slides-105-iccrg-bbr-v2-a-model-based-congestion-control-00
15. Cardwell, N., Yuchung, C.: TCP BBR congestion control comes to GCP - your Internet just got faster (2017). https://cloud.google.com/blog/products/networking/tcp-bbr-congestion-control-comes-to-gcp-your-internet-just-got-faster/. Accessed 04 Feb 2024
16. Chandrasekaran, B., Smaragdakis, G., Berger, A., Luckie, M., Ng, K.C.: A server-to-server view of the internet. In: ACM CoNEXT (2015)
17. Claypool, S.: Sharing but not Caring - Performance of TCP BBR and TCP CUBIC at the Network Bottleneck (2019)
18. Claypool, S., Chung, J., Claypool, M.: Comparison of TCP congestion control performance over a satellite network. In: Passive and Active Measurement (PAM) (2021)
19. Intel®: Intel® Tofino 6.4Tbps, 4 pipelines (2017). https://www.intel.com/content/www/us/en/products/sku/218643/intel-tofino-6-4-tbps-4-pipelines/specifications.html. Accessed 01 Oct 2023
20. Dong, M., et al.: PCC Vivace: online-learning congestion control. In: USENIX Symposium on Networked Systems Design and Implementation (NSDI) (2018)
21. Fejes, F., Gombos, G., Laki, S., Nádas, S.: Who will save the internet from the congestion control revolution? In: Workshop on Buffer Sizing (2019)
22. Google: TCP BBR v2 Alpha/Preview Release (2021). https://github.com/google/bbr/tree/v2alpha. Accessed 01 Oct 2023
23. Google: TCP BBR v3 Release (2023). https://github.com/google/bbr/tree/v3. Accessed 01 Oct 2023
24. Grazia, C.A., Patriciello, N., Klapez, M., Casoni, M.: A cross-comparison between TCP and AQM algorithms: which is the best couple for congestion control? In: International Conference on Telecommunications (2017)
25. Ha, S., Rhee, I., Xu, L.: CUBIC: a new TCP-friendly high-speed TCP variant. ACM SIGOPS Oper. Syst. Rev. **42**(5), 64–74 (2008)
26. Hock, M., Bless, R., Zitterbart, M.: Experimental evaluation of BBR congestion control. In: IEEE International Conference on Network Protocols (ICNP) (2017)
27. Hurtig, P., et al.: Impact of TCP BBR on CUBIC traffic: a mixed workload evaluation. In: International Teletraffic Congress (2018)
28. Huston, G.: BBR TCP (2017). https://labs.ripe.net/author/gih/bbr-tcp/
29. ICSI: Zeek: An Open Source Network Security Monitoring Tool (2022). http://www.zeek.org. Accessed 20 Sept 2023
30. Ivanov, A.: Evaluating BBRv2 on the dropbox edge network. In: Netdev, The Technical Conference on Linux Networking (2020)
31. Corbet, J.: TCP small queues (2012). https://lwn.net/Articles/507065/. Accessed 01 Oct 2023
32. Kenjiro, C., Koushirou, M., Akira, K.: Traffic data repository at the WIDE project. In: USENIX Annual Technical Conference (ATC) (2000)

33. Kfoury, E.F., Gomez, J., Crichigno, J., Bou-Harb, E.: An emulation-based evaluation of TCP BBRv2 Alpha for wired broadband. Comput. Commun. **161**, 212–224 (2020)
34. Kleinrock, L.: Power and deterministic rules of thumb for probabilistic problems in computer communications. In: IEEE International Conference on Communications (1979)
35. Li, Y., et al.: HPCC: high precision congestion control. In: Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM) (2019)
36. Mishra, A., Sun, X., Jain, A., Pande, S., Joshi, R., Leong, B.: The great internet TCP congestion control census. Proc. ACM Meas. Anal. Comput. Syst. **3**(3), 1–24 (2019)
37. Nandagiri, A., Tahiliani, M.P., Misra, V., Ramakrishnan, K.K.: BBRvl vs BBRv2: examining performance differences through experimental evaluation. In: IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN) (2020)
38. Cardwell, N.: BBR evaluation with netem (2017). https://groups.google.com/g/bbr-dev/c/8LYkNt17V_8. Accessed 01 Oct 2023
39. Netravali, R., et al.: Mahimahi: accurate record-and-replay for HTTP. In: USENIX Annual Technical Conference (ATC) (2015)
40. Philip, A.A., Ware, R., Athapathu, R., Sherry, J., Sekar, V.: Revisiting TCP congestion control throughput models & fairness properties at scale. In: ACM Internet Measurement Conference (IMC) (2021)
41. Sandvine: The Global Internet Phenomena Report January 2023. Technical report, Sandvine Corporation (2023). https://www.sandvine.com/hubfs/Sandvine_Redesign_2019/Downloads/2023/reports/Sandvine%20GIPR%202023.pdf?hsCtaTracking=3cbc04da-fd44-481b-ad03-811d23b7b2c5%7C131df09f-dbdd-41a0-9dce-aac8879403ff
42. Scholz, D., Jaeger, B., Schwaighofer, L., Raumer, D., Geyer, F., Carle, G.: Towards a deeper understanding of TCP BBR congestion control. In: IFIP Networking (2018)
43. Shah, A.: BBR Evaluation at a Large CDN (2019). https://edg.io/tw/resources/technical-article/bbr-evaluation-at-a-large-cdn/
44. Sommers, J., Kim, H., Barford, P.: Harpoon: a flow-level traffic generator for router and network tests. In: ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Science, vol. 32, no. 1 (2004)
45. Song, Y.J., Kim, G.H., Mahmud, I., Seo, W.K., Cho, Y.Z.: Understanding of BBRv2: evaluation and comparison With BBRv1 congestion control algorithm. IEEE Access **9**, 37131–37145 (2021)
46. Spang, B., Walsh, B., Huang, T.Y., Rusnock, T., Lawrence, J., McKeown, N.: Buffer sizing and video QoE measurements at Netflix. In: Workshop on Buffer Sizing (2019)
47. Hemminger, S., Ludovici, F., Pfeifer, H.P.: tc-netem(8) - Linux manual page (2023). https://man7.org/linux/man-pages/man8/tc-netem.8.html. Accessed 01 Oct 2023
48. Tahiliani, M.P., Misra, V.: A principled look at the utility of feedback in congestion control. In: Workshop on Buffer Sizing (2019)
49. Turkovic, B., Kuipers, F.A., Uhlig, S.: Interactions between congestion control algorithms. In: Network Traffic Measurement and Analysis Conference (TMA) (2019)

50. Ware, R., Mukerjee, M., Seshan, S., Sherry, J.: Modeling BBR's interactions with loss-based congestion control. In: ACM Internet Measurement Conference (IMC) (2019)
51. Yan, F.Y., et al.: Pantheon: the training ground for Internet congestion-control research. In: USENIX Symposium on Networked Systems Design and Implementation (NSDI) (2018)
52. Yang, F., Wu, Q., Li, Z., Liu, Y., Pau, G., Xie, G.: BBRv2+: towards balancing aggressiveness and fairness with delay-based bandwidth probing. Comput. Netw. **206**, 108789 (2022)
53. Zapletal, A., Kuipers, F.: Slowdown as a metric for congestion control fairness. In: SIGCOMM Workshop on Hot Topics in Networking (HotNets), HotNets 2023 (2023)
54. Zeynali, D., Weyulu, E.N., Fathalli, S., Chandrasekaran, B., Feldmann, A.: Promises and Potential of BBRv3: Testbed configuration and evaluation artifacts (2024). https://dzeynali.gitlab.io/
55. Zhang, H., Zhu, H., Xia, Y., Zhang, L., Zhang, Y., Deng, Y.: Performance analysis of BBR congestion control protocol based on NS3. In: International Conference on Advanced Cloud and Big Data (CBD) (2019)
56. Zhang, Y., Breslau, L., Paxson, V., Shenker, S.: On the characteristics and origins of internet flow rates. In: Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM) (2002)