# BBR vs. BBRv2: A Performance Evaluation

Rebecca Drucker
Stony Brook University, Furman University
rdrucker@cs.stonybrook.edu

Gauri Baraskar, Aruna Balasubramanian, Anshul Gandhi
Stony Brook University
{gbaraskar,arunab,anshul}@cs.stonybrook.edu

***Abstract*—Google's new congestion control algorithm, BBR, has seen wide adoption. However, there are concerns about its unfairness to legacy congestion control algorithms and the high retransmissions experienced under lossy conditions. In response, Google designed BBRv2 to address these concerns. However, the performance of BBRv2 with respect to BBR or Cubic has not been studied systematically. This paper presents a fine-grained performance study of BBRv2 under a variety of RTTs, bandwidths, buffer sizes, and loss conditions, in both LAN and WAN environments. We construct a decision tree to determine whether BBR or BBRv2 performs better under different conditions. We observe that BBRv2's goodput is significantly low compared to BBR's under induced loss and bursty losses, and the effect is magnified under large BDP. Our results suggest that BBRv2 trades performance for better fairness under losses. To bridge this gap, we investigate the workings of BBRv2 and find that BBRv2 employs a long-term upper bound on sending rate that is not robust to losses. This upper bound is continually decremented in the presence of persistent losses, thereby depressing goodput. We show that by aligning BBRv2's upper bound with its maximum bandwidth estimation, BBRv2's performance can be greatly improved while maintaining its fairness.**

## I. Introduction

In the past few years, the Bottleneck Bandwidth and Round-Trip Propagation Time (BBR) congestion control algorithm [1] has seen widespread adoption throughout the Internet, and now accounts for over 40% of web traffic [2]. BBR's popularity is largely driven by its ability to achieve high goodput while incurring low latency [3]. However, recent studies have uncovered significant drawbacks of BBR including its unfairness to other flows and its high retransmission rate [4], [5].

Consequently, BBRv2 [6] is being actively developed (with an alpha release of 2019) to address the drawbacks of BBR. Google has already begun to use BBRv2 internally [7], and once it is added to the Linux kernel, it is likely to be widely deployed in the Internet. Recent studies conducted in specialized networks [8] or using Mininet [9] have noted the improvements afforded by BBRv2 over BBR. However, a detailed study of the performance of BBRv2, especially when compared to BBR, in generic networks and under realistic conditions is lacking.

In this paper, we conduct an empirical measurement study to analyze the performance of BBRv2 versus BBR. We focus on goodput and fairness, as well as video quality, and experiment with diverse network conditions over a LAN and a WAN setting. In addition to buffer losses and induced random losses, we also experiment with bursty losses experienced in cellular networks using real cellular traces [10]. We construct a decision tree to characterize the conditions under which BBR and BBRv2 performance considerably vary.

Our key finding is that under both persistent random losses and bursty losses, BBRv2 performs significantly poorly compared to BBR. This is not surprising given BBRv2 has been explicitly designed to react to losses, while the original BBR does not react to losses. What is surprising is the extent of the poor performance. For example, under a 500mbps connection, when a 2% loss is induced, across all buffer conditions, BBRv2 performs 8×–17× worse than BBR. This poor performance holds in the WAN and in cellular networks that experience burst losses. While not as pronounced, we also see a reduction in video quality for BBRv2 under 2% loss when streaming a DASH video. However, BBRv2 does improve fairness compared to BBR, in terms of the recently introduced harm metric [5]. Further, under only buffer losses (i.e, no induced random loss or burst loss), BBR and BBRv2 perform similarly in terms of goodput.

To better understand this shortcoming of BBRv2, we investigate the BBRv2 congestion control logic under losses to identify potential culprits. BBRv2 uses two mechanisms to react to losses: (i) an upper bound on inflight packets that is determined based on a long-term loss rate estimate, and (ii) a lower bound on the inflight packets that is determined based on a single loss event. BBRv2 sets the packet in flight between this upper bound and lower bound values.

Based on a fine-grained investigation of these two parameters, we find that the lower bound parameter is effective in the presence of buffer losses, reducing the sending rate and subsequent congestion. However, when losses are persistent or bursty, the upper bound parameter is continually decremented. Since the packets in flight cannot exceed the upper bound, the goodput reduces significantly. However, completely removing the upper bound parameter affects fairness.

To improve BBRv2 performance under losses while maintaining fairness, we propose an adaptive version of BBRv2. In the adaptive version, the long-term upper bound is reset to BBRv2's maximum bandwidth estimate in each cycle rather than allowing it to decrement continuously under loss. The lower bound parameter remains unchanged. We implemented this adaptive BBRv2 version in the Linux kernel by modifying existing BBRv2. We evaluate the adaptive BBRv2 implementation under diverse network conditions and find that it is able to achieve goodput 8× greater than unmodified BBRv2 (and only 8% less than BBR) while being fairer to Cubic than BBR in the LAN and reducing the retransmission rate by 70%. Our results suggest that, with some modifications, BBRv2 can address the drawbacks of BBR without impacting performance, and replace BBR in production environments.

## II. Background

**BBR:** Loss-based congestion control algorithms such as Cubic use loss as their only congestion signal, and are known to

| RTTs (ms) | 10, 100 |
|---|---|
| BWs (mbps) | 25, 50, 100, 300, 500, 1000 |
| Buffer Sizes | 100KB, 2MB, 10MB, 50MB, 100MB |
| Loss Rates | 0%, 1%, 2% |

TABLE I: Network settings employed in our experiments.

suffer from bufferbloat. To alleviate these issues, the Bottleneck Bandwidth and Round-Trip Propagation Time (BBR) congestion control algorithm [1] was introduced. BBR does not react to loss. Instead, BBR aims to send at a rate that matches the available network capacity, called the bandwidth-delay product (BDP). BBR measures the available bandwidth and the minimum RTT over a period of time and computes the BDP as the product of bandwidth and minimum RTT. Since BBR will keep sending even in the presence of losses as long as the delivery rate can be maintained, BBR tends to be unfair to other flows and has high retransmission counts [3].

**BBRv2:** To address the poor fairness and high retransmissions of BBR, the designers of BBR made some key changes and released a new version, BBRv2, in 2019 [6]. BBRv2 also determines its sending rate based on estimated BDP; however, unlike BBR, BBRv2 does react to loss. BBRv2 introduces new upper and lower bounds on the number of packets in flight. The upper bound reacts to loss over the long term, and the lower bound reacts to short-term losses. We discuss these values further in Section IV-A.

BBRv2 operates in different phases (shown later in Figure 4). In the startup phase, BBRv2 increases its sending rate quickly. When BBRv2 exits the startup phase, it enters a set of four phases—PROBE_DOWN, PROBE_CRUISE, PROBE_REFILL, and PROBE_UP, which are part of the steady state. The number of packets in flight in steady state is between the long-term upper and short-term lower bound values. Periodically (every 5 secs), BBRv2 reduces its sending rate to half. This phase is called PROBE_RTT, and it eliminates queues in bottleneck routers.

## III. Comparing BBRv2 with BBR

### A. Experimental Setup, Metrics, and Methodology

We consider three types of networks in our experiments: LAN, WAN, and emulated cellular network using real cellular traces. We evaluate BBR and BBRv2 using iPerf3 and measure peak goodput under the network conditions. We also conduct DASH video experiments and evaluate video quality.

*a) LAN and WAN:* In our LAN setup, we connected a client and a server, both running Ubuntu 18.04, through a Linksys WRT1900ACS router with OpenWRT 19.07.1 installed. This setup allows us to abstract away all effects of the network other than those of the bottleneck router, where we set RTT, bandwidth, buffer size, and loss rate. The kernel has been modified to include BBRv2, which is not included in the Linux kernel by default. We also modify BBRv2's module parameters to allow us to collect debug logs of BBRv2's internal variables.

In our WAN experiments, the client is located in South Korea while the iPerf3 sender is located in the Northeast US, resulting in an RTT of around 200 ms and maximum goodput of approximately 100 mbps. In both LAN and WAN experiments, we induce up to 2% packet loss at the sender. Other recent studies induced or observed packet loss at similar rates [11], [12].

*b) Cellular:* The cellular experiments are also run on the aforementioned LAN with cellular conditions emulated using cell-emulation-util [13]that replays cellular traces collected by Akamai from a cellular ISP [10].

To understand the difference in performance between BBR and BBRv2, we ran experiments under a variety of network conditions, including varied bandwidth, RTT, buffer size, and random loss rates. The network conditions used in these experiments are shown in Table I. All experiments were run for five minutes at least five times per condition.

We measure performance using goodput and fairness using the harm metric. Goodput is total bytes received by the iperf server divided by the experiment length. Harm is a recently proposed metric [4] as an alternative to the classical Jain's fairness index. Jain's fairness index is an appropriate fairness metric when all flows are able to reach their full potential. However, it does not work well when one or more flows is unable to use its full share of the bandwidth due to factors not caused by other flows. To calculate the harm done by a new congestion control $x$ to a legacy congestion control $y$, we need to know the throughput of $x$ and $y$ in competition, as well as the throughput of $y$ alone. Then the harm done by $x$ to $y$ is:

$$\frac{throughput_{(y\ alone)} - throughput_{(y\ with\ x)}}{throughput_{(y\ alone)}}.$$

For example, if legacy congestion control $y$'s throughput is 2 mbps alone and 1 mbps in competition with new congestion control $x$, the harm done by $x$ to $y$ is 0.5, meaning that $y$'s throughput decreases by 50% when it competes with a flow using $x$. While there are many thresholds for the amount of harm considered acceptable, we reason that a new congestion control's harm to legacy congestion control is acceptable if it is less than or equal to the harm the legacy congestion control would do to itself. Therefore, to evaluate the harm inflicted by BBR and BBRv2 on Cubic, we calculate their harm values, as well as Cubic's harm to itself as a baseline.

For DASH experiments, we evaluate video quality between 0 (1 mbps bitrate) and 10 (25 mbps bitrate).

### B. Decision Tree for BBRv2 vs. BBR

We present all our experimental results using the conditions in Table I as a decision tree in Figure 1. We create the decision tree using scikit-learn's decision tree classifier. The target class for each pair of experiments is BBR or BBRv2. The decision tree returns BBR, if BBR achieves better throughput than BBRv2 in more experimental runs under a set of conditions, and returns BBRv2 otherwise. The BBR version that provides better throughput under each set of conditions is indicated in each node. To better illustrate the difference in throughput, we color code the nodes such that darker shades indicate larger

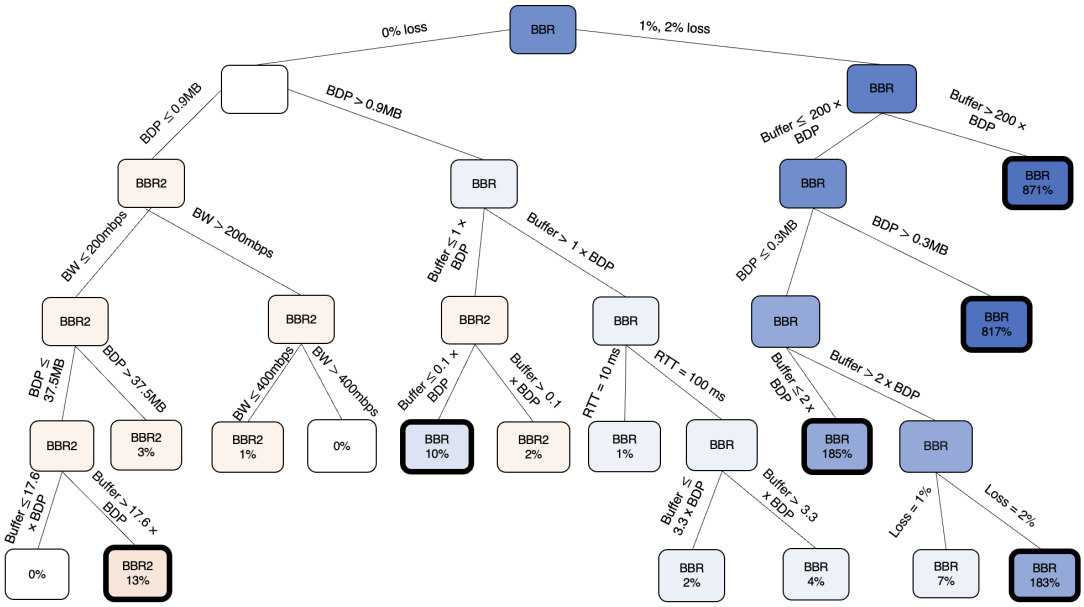



Fig. 1: Decision tree based on our experiments highlighting the choice between BBRv2 and BBR. Blue node indicate that BBR performs better than BBRv2 and orange node is vice versa (darker color indicate higher performance difference). Nodes in white indicate negligible goodput difference. Highlighted leaves are those where we see improvement of at least 10%.

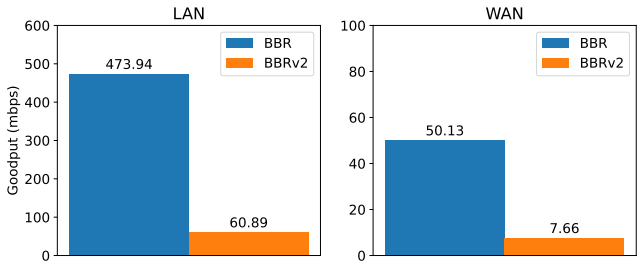



Fig. 2: The goodput for BBR and BBRv2, under 100 ms RTT, 500 mbps bandwidth, and a 2 MB buffer on LAN and in the WAN with 2% loss. BBRv2 experiences poor goodput under high loss

differences in throughput. If the difference is negligible (less than 1%), the node is colored white. For ease of readability, we highlight nodes (with a darker bounding box) where the difference is at least 10%.

*a) No induced loss:* BBR and BBRv2 largely perform similarly when no additional random loss is induced; see the left sub-tree under the root node in Figure 1. Both achieve goodput close to the full bandwidth, though BBR incurs more retransmissions in the process (results not shown). The only case in which the difference in goodput is greater than 10% is the smallest BDP and large buffer size (leaf node that is second from left), where BBRv2 outperforms BBR by 13%. This is because we use a burst size of 1MB which is proportionally large for the smallest BDP. Under these conditions, BBR is known to perform poorly due to bufferbloat [14].

*b) Induced loss and Large BDP:* Under induced loss, BBR consistently achieves higher goodput than BBRv2; see the right sub-tree under the root node in Figure 1. The difference in goodput ranges from 7% to 871%. Figure 2 takes a closer look at the goodput for BBR and BBRv2 under high bandwidth (500mbps), a 2 MB buffer, and 2% induced loss, which is the condition where we see the most difference between BBR and BBRv2. In this setting, BBRv2 has very low goodput—as much as 7.8× (with 2 MB buffer) – 21.2× (with 10 MB buffer) lower compared to BBR. This observation also holds in the WAN, as shown in the right plot of Figure 2; we see that BBRv2 achieves about 6.5× lower goodput compared to BBR under 2% induced loss. For video experiments, BBRv2 continues to suffer under high losses, though the performance drop is not as pronounced. Under 2% loss, we find that BBRv2 has 18% lower quality than BBR.

One of the reasons BBR has higher goodput compared to BBRv2 is that it does not react to losses. Consequently, BBR sends a large number of packets even under losses, resulting in as much as 3.3× higher retransmissions compared to BBRv2. To summarize, our analysis suggests that ***under high loss rate, BBR significantly outperforms BBRv2 in terms of goodput***.

### C. Evaluating fairness under BBRv2 and BBR

For a complete performance evaluation of BBRv2 vs. BBR, we now empirically contrast the fairness of BBRv2 and BBR when they (individually) coexist with Cubic flow(s). We use the same experimental methodology to run LAN experiments with iPerf3 as described in Section III-A.

We evaluate the harm fairness metric under a large BDP and small, medium, and large bottleneck buffers. We include results for no induced loss and 2% induced random loss. Table II shows the full set of results. Recall, from Section III-A, that a lower harm value is better, with Cubic Cubic Harm values being ideal. To understand how to interpret these values,

| Loss | Buffer Size | BBR Cubic Harm | BBRv2 Cubic Harm | Cubic Cubic Harm |
|---|---|---|---|---|
| 0% | Small | 0.98 | 0.61 | 0.48 |
| | Medium | 0.50 | 0.33 | 0.47 |
| | Large | 0.39 | 0.33 | 0.54 |
| 2% | Small | 0.38 | 0.03 | 0.05 |
| | Medium | 0.40 | 0.05 | 0.03 |
| | Large | 0.42 | 0.02 | 0.01 |

TABLE II: Harm results when BBR, BBRv2, and Cubic coexist with Cubic under large BDP setting (500mbps bandwidth and 100ms RTT) and various loss and buffer settings.

consider the following example. Under 0% induced loss and a small buffer, the Cubic Cubic Harm value is 0.48, meaning that a Cubic flow's goodput decreased by 48% when competing with another Cubic flow, compared to its goodput alone. The corresponding BBR Cubic Harm value is 0.98, meaning that the Cubic flow's goodput decreased by 98% when competing with a BBR flow, compared to its goodput alone.

Under no induced loss, we see that BBRv2 is more fair than BBR, especially for a small buffer. Because BBR does not react to losses, it is particularly unfair to Cubic in small buffers where losses occur frequently. BBRv2 reacts to loss and is therefore fairer to Cubic in this situation. Under a 2% induced loss, again BBRv2 is significantly more fair to Cubic than BBR; this is because, again, BBR does not react to losses and aggressively consumes available bandwidth, starving other flows. In summary, ***BBRv2 is consistently fairer than BBR, especially under losses***.

### D. Cellular

All experiments described thus far have been under controlled conditions, as well as persistent (induced) random loss. To strengthen our findings about BBRv2's poor performance under loss, we ran experiments using cellular network traces. In cellular networks, losses do not occur at uniform random intervals. Instead they come in short bursts of high loss, and the timing and duration of these bursts is not deterministic.

| Condition | BW (mbps) | Loss (%) | Buffer (KB) |
|---|---|---|---|
| Good | 3 | 1.5 | 9, 40, 128 KB |
| Median | 1.7 | 5 | 9, 20, 128 KB |
| Poor | 1.4 | 10 | 9, 18, 128 KB |

TABLE III: Average bandwidth, loss rate for the good, median, and poor cellular conditions from traces [10]. The latency is 50ms for all traces.

We use existing cellular traces [10] where the traces are divided into good, median, and poor conditions. The characteristics of the three traces are shown in Table III. We emulate these conditions in our LAN and set different buffer sizes for the three conditions as shown in the table.

In all cases, BBR's average goodput is greater than BBRv2's average goodput. Figure 3 shows the results for the median network condition; results are similar for the poor and good network conditions. In these experiments, we use two bandwidth settings. One is the default bandwidth, which refers to the bandwidth seen by the cellular network. We also increase the bandwidth to 10mbps to simulate a high bandwidth cellular condition. In the small and medium buffers, BBR significantly

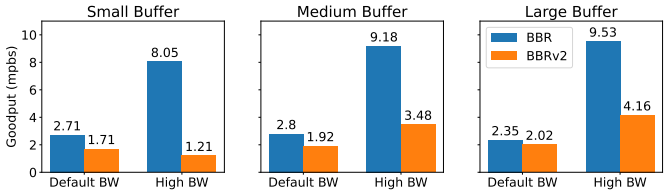



Fig. 3: Cellular: Goodput for BBR and BBRv2 under the median cellular condition.

outperforms BBRv2, by as much 6.6×; the difference is smaller in the large buffer case. The difference in goodput between BBR and BBRv2 is greater under the simulated high bandwidth condition; for all three buffer sizes, BBR outperforms BBRv2 by at least 2×. This is because BBR does not react to losses and so continues to aggressively use the available bandwidth. By contrast, under the burst losses experienced under cellular conditions, BBRv2 backs off its delivery rate substantially (as discussed later in Section IV-A).

We also analyze the fairness when BBR and BBRv2 coexist with Cubic under cellular conditions. Similarly to LAN conditions, BBRv2 is fair to Cubic while BBR is unfair. These results are omitted for brevity.

## IV. Examining BBRv2's Performance Under Loss

Our results above show that BBRv2 sees a huge performance hit (as much as 16.9×) under sustained and bursty losses. In this section, we analyze BBRv2 to understand why.

### A. BBRv2's Behavior Under Loss

Figure 4 shows how losses affect BBRv2's sending rate during different phases of its operation. Recall from Section II that BBRv2 maintains a short-term lower bound (called *inflight_lo*) and a long-term upper bound (called *inflight_hi*). The number of packets in flight (cwnd) is bound between these two values. Losses affect both these bounds, as we describe below

*a) STARTUP:* Both BBR and BBRv2 have a startup phase where sending rate increases quickly. In both cases, the protocol exits the startup phase when the delivery rate plateaus. In addition, BBRv2 can also exit the startup phase when the loss rate (calculated from the beginning of the startup) exceeds the loss threshold of 2%. When the loss threshold is exceeded, BBRv2 sets the long-term upper bound to the current cwnd. The short-term lower bound is not set in this phase.

*b) PROBE_DOWN, PROBE_CRUISE, PROBE_REFILL, PROBE_UP:* These four phases represent the steady state for BBRv2. Together, they last one cycle of 2–3 seconds from the start of the last PROBE_DOWN, plus one RTT

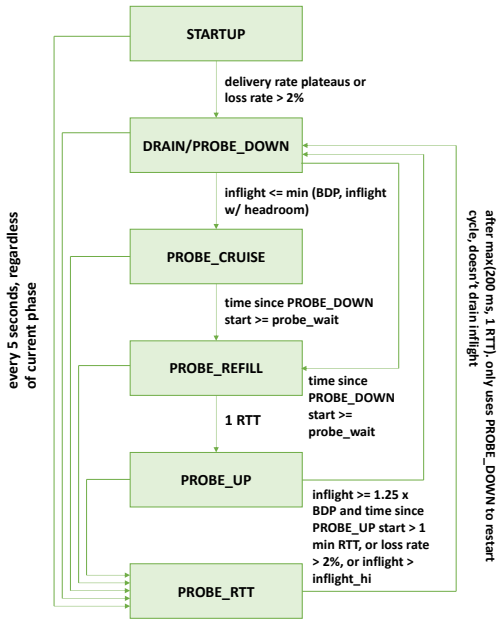



Fig. 4: Flowchart showing phases of BBRv2's cycle including transition conditions.

for PROBE_REFILL and a maximum of one RTT for PROBE_UP. The short-term lower bound is set to the current cwnd at the start of the cycle. Thereafter, each time a (single) loss occurs, the short-term lower bound is reduced by 30% or set to the maximum number of packets in flight in the previous roundtrip, whichever is greater. Since this is a short-term bound, it is reset once per cycle, before each PROBE_REFILL.

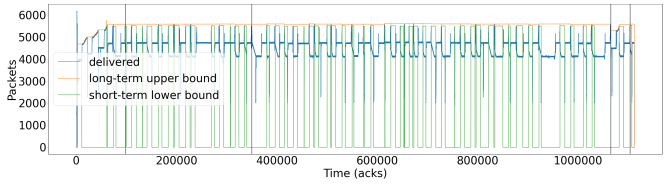
But the long-term upper bound is set differently in response to losses. If the long-term upper bound is not set in the startup phase, it is set to infinity. In the steady state, the long-term upper bound is decreased when the loss threshold is exceeded (rather than reacting to a single loss). Specifically, the upper bound is set to 30% less than BDP or the packets in flight, whichever is greater. The loss threshold is estimated over the cycle, so it is a longer-term loss estimate. If the loss threshold is not exceeded, upper bound value is increased incrementally (there are some corner cases that we omit).

In other words, *the short-term lower bound is affected by a single loss, but the value does not carry over beyond a cycle*. In contrast, *the long-term upper bound is set based on loss calculated over a longer period of time and the value persists.* The cwnd is set between the short-term lower bound and long-term upper bound, but is always 15% less than the upper bound, also referred to as the headroom.

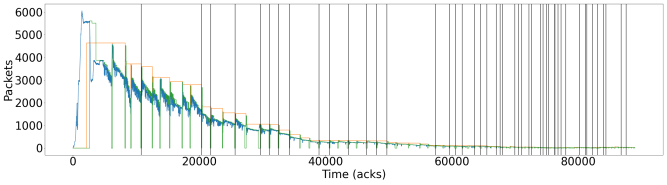
*c) PROBE_RTT:* The short-term lower bound is reset to infinity at the end of each PROBE_RTT to prevent the low sending rate from persisting.

### B. Importance of Upper and Lower Bounds

To illustrate the effects of the short-term lower bound and long-term upper bound on BBRv2's delivery rate, we show their values, along with the values of delivery rate and the




(a) 0% random loss




(b) 2% random loss

Fig. 5: BBRv2's delivery rate, the short-term lower bound, and the long-term upper bound during a five-minute run with 100 ms RTT, 500 mbps bandwidth, and a 2 MB buffer. Black vertical lines indicate times when loss threshold was exceeded.

times when the loss threshold was exceeded (indicated with a black vertical line).

Figure 5a shows a scenario with no induced loss. In this case, losses do occur due to relatively small buffers. But, every time a single loss occurs, the short-term lower bound value reduces the sending rate. By doing so, further losses are avoided and the long-term loss rate rarely exceeds the loss threshold. Consequently, the long-term upper bound persists at a high value. The delivery rate stays at 15% lower than the long-term upper bound value, resulting in high goodput.

In contrast, Figure 5b shows BBRv2's behavior under the same condition but with 2% induced random loss. When the loss threshold is exceeded repeatedly, the long-term upper bound and short-term lower bound decrease rapidly until cwnd converges to a very small value (by the end of the run). Because the long-term upper bound never gets reset, its value remains low and the sending rate is never given an opportunity to recover. At the beginning of each cycle, the short-term lower bound is set to the minimum of the short-term lower bound and long-term upper bound. Therefore, even though the short-term lower bound is reset and given a chance to increase every cycle, its value is influenced by the long-term upper bound, limiting its ability to increase goodput.

The main takeaway is that *the short-term lower bound reacts to short-term buffer losses and is able to control the sending rate. However, when the losses persist (either because we induce the loss or when the losses are bursty as in the case of cellular conditions), the long-term upper bound reduces significantly, resulting in poor goodput.*

## V. Making BBRv2 Robust to Losses

Our goal is to improve BBRv2's goodput under loss while maintaining most of its fairness improvements. Our approach is to modify the loss response of BBRv2.

### A. How important is the long-term upper bound?

A natural first step is to not set the long-term upper bound at all, essentially leaving it at infinity. To do this, we set the

*loss threshold* to an arbitrarily high value (1000%, in our experiments) so that the long-term upper bound will almost never be set. Recall that the upper bound value is set only when the loss estimate exceeds the loss threshold. By not setting the long-term upper bound, this version of BBRv2 will react to losses solely via the short-term lower bound value.

With this setting, the goodput dramatically improves, with BBRv2 (removing upper bound) performing similarly to BBR. However, it also results in poor fairness. The unfairness is not as bad as BBR, but is considerably worse than BBRv2.

The reason for the unfairness is that, without an upper bound value, the delivery rate can increase in an unbounded manner. Recall that the packets in flight is set to 15% less than the upper bound in the original BBRv2, providing an upper limit. Thus, removing the loss threshold and eliminating the long-term upper bound meets our performance goal (of high goodput), but at the expense of fairness.

### B. Adaptive long-term upper bound

Instead of removing the upper bound entirely, we propose to set it *adaptively*. The problem with the upper bound regulation algorithm in unmodified BBRv2 is that the value can drop to arbitrarily low numbers across cycles when loss persists. Our approach to stem this free fall of the upper bound value is to reset it periodically and adaptively.

*a) When should the upper bound be reset?:* Setting the long-term upper bound only once at the beginning of the flow means that the long-term upper bound will reflect the unsustainably high sending rate during the startup phase. Setting it once per cycle at the beginning of the steady state causes the long-term upper bound to reflect the low sending rates used during PROBE_RTT, which reduces goodput unnecessarily. We verified both of the above scenarios empirically.

Instead, we set the long-term upper bound after the steady state is reached, specifically, at the beginning of PROBE_REFILL (see Figure 4). When BBRv2 is in steady state, the number of packets sent is a near-accurate estimate of the capacity of the network. We update the value at every PROBE_REFILL, allowing the upper bound to reflect changes in bandwidth.

*b) What value should the upper bound be reset to?:* Rather than devising a complicated algorithm, we consider a simple heuristic that is easy to deploy: we set the long-term upper bound to the maximum number of packets in-flight during the last round trip. BBRv2 already tracks this value (as inflight_latest), making it easy to implement.

The intuition here is that, as stated above, the number of packets in flight during steady state is a good estimate of the capacity of the network. This value represents a high but safe sending rate, even in the presence of losses. Since BBRv2 must send at least 15% less than this upper bound, BBRv2 will send more conservatively and consequently be more fair.

## VI. Evaluation

We now evaluate the performance and fairness of our BBRv2 with adaptive long-term upper bound; we refer to this version of BBRv2 as "our solution" in the results.

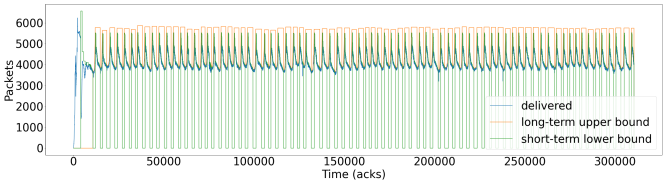



Fig. 6: Packets delivered, short-term lower bound, and long-term upper bound for our adaptive version of BBRv2. Experiment settings are the same as in Figure 5 (a) and (b). The loss threshold is never exceeded, but the long-term upper bound is reset every cycle.

### A. Methodology

The evaluation setup is similar to the one described in Section III. In addition to BBR and BBRv2, we evaluate the performance of our adaptive variant of BBRv2. Unless stated otherwise, we report results averaged over at least 5 runs. We present results from four network environments:

- LAN, WAN, and cellular setups (see Section III-A)
- LAN with changing bandwidth: In this scenario, we change the bandwidth during the course of the experiment to study how well our solution can adapt its sending rate.

### B. Performance evaluation under LAN and WAN settings

Figure 6 shows the long-term upper bound, short-term lower bound, and the delivery rate for each ack during a five-minute run using a 2 MB buffer and 2% induced loss. Our modification causes the long-term upper bound to be set to the maximum number of packets in flight during the latest round trip at the beginning of each PROBE_REFILL, limiting the long-term upper bound to 85% of that value. The long-term upper bound remains steady throughout the flow, unlike unmodified BBRv2 (in Figure 5b). The experiment length in Figure 6 is the same as in Figure 5b, but the significantly increased delivery rate with our solution results in more acks on the x-axis in Figure 6.

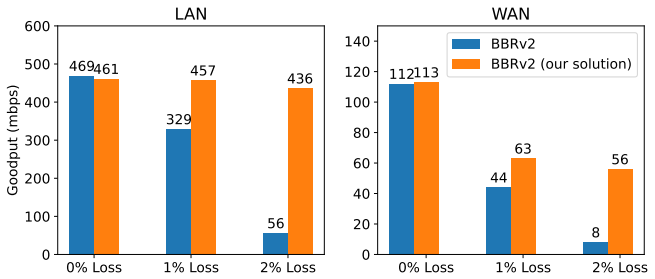



Fig. 7: Goodput for BBRv2 and our adaptive solution in the LAN, under a 100 ms RTT and 500 mbps bandwidth, and in the WAN with 0%, 1%, and 2% induced random loss.

Figure 7 shows goodput for our modified BBRv2 alongside unmodified BBRv2 in the LAN and in the WAN. When no additional loss is induced, the two BBRv2 versions perform similarly, achieving goodput comparable to BBR. Under 1% and 2% loss, the our modified BBRv2 vastly outperforms

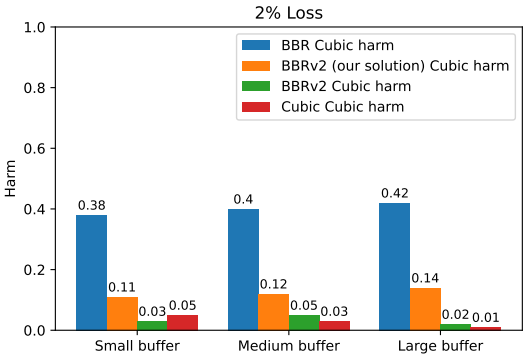



Fig. 8: LAN set up: Harm values for 2% loss, 500mpbs bandwidth, and 100ms RTT.

standard BBRv2 due to its reduced loss response. The retransmission rate (not shown) is similarly low for both BBRv2 variants, showing that unlike BBR, our modified BBRv2 does not incur excessive retransmissions to maximize goodput.

We also find that our modified BBRv2 achieves optimal video quality similar to that of BBR under 2% induced loss (results not shown).

Figure 8 shows the harm metric [5] for our LAN experiments under 2% induced loss. Under the 2% induced loss setting, our solution is significantly more fair than BBR, but more unfair than BBRv2. This is because BBRv2, with its default loss threshold of 2%, significantly reduces its delivery rate (resulting in its unusually low goodput in Figure 7), leaving more bandwidth for Cubic to leverage. By contrast, our solution maintains high goodput while significantly improving fairness over BBR.

In summary, our adaptive solution achieves high goodput compared to BBRv2 while improving fairness over BBR.

### C. Performance under changing bandwidth

We confirm that our adaptive BBRv2 is able to respond to changes in bandwidth. We consider a five-minute changing bandwidth scenario which starts with a bandwidth of 500 mbps, followed by 50 mbps decrements in available bandwidth every 20 secs until the bandwidth reaches 250 mbps. The bandwidth remains at 250 mbps for 50 secs before increasing by 50 mbps every 20 secs until reaching 500 mbps, where it stays for the remainder of the run. The buffer size is 10 MB. We include experiments with 0% and 2% induced loss.

Figure 9 shows that our adaptive BBRv2 responds to changes in bandwidth as quickly as BBR. Thus, setting the long-term upper bound once per cycle suffices to allow our version of BBRv2 to respond quickly to changes in bandwidth. BBR, BBRv2, and our adaptive BBRv2 achieve similar goodput under the changing bandwidth condition except for BBRv2 under 2% induced loss. In this case, BBRv2 has low goodput due to its loss response, rather than an inability to respond to changes in bandwidth. Thus, our solution responds well to changing bandwidth and maintains high goodput (on par with BBR) under these conditions as well, as shown in Table IV.

| | BBR | BBRv2 | BBRv2 (our solution) |
|---|---|---|---|
| 0% induced loss | 363 | 360 | 364 |
| 2% induced loss | 359 | 23 | 342 |

TABLE IV: Goodput (mbps) for BBR, BBRv2, and our adaptive BBRv2 under the changing bandwidth condition.

### D. Performance evaluation under cellular setting

We also analyze the performance of our modified BBRv2 under emulated cellular conditions.

Our adaptive BBRv2 provides significant goodput improvements over unmodified BBRv2 under cellular conditions. Figure 10 shows the goodput achieved by BBRv2 and our adaptive BBRv2 under the median cellular condition and 10mbps bandwidth. Our version provides, on average, 143% improvement across all cases, and as much as 191% for the small buffer setting. We see similar improvements under good and poor cellular conditions for 10mbps and 100mbps bandwidth (not shown here).

Our solution is much fairer than BBR under the small buffer. Our solution is slightly more fair than BBR under the medium and large; however, BBRv2 is much more fairer than our solution in all cases.

In the small buffer case, all variants of BBRv2 achieve low goodput compared to BBR. Our adaptive BBRv2 solution does increase goodput by nearly 3× compared to unmodified BBRv2, but this throughput is still lower than BBR (not shown). Thus, when run with Cubic, all versions of BBRv2 leave significant bandwidth for Cubic, decreasing their harm.

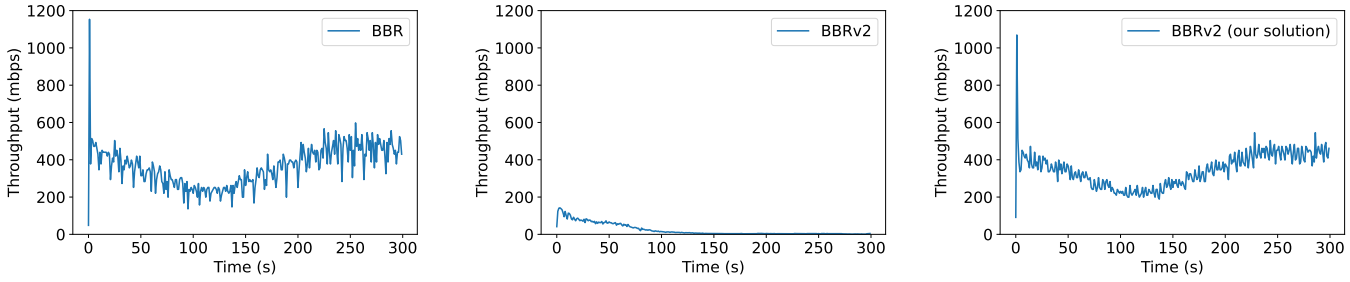
While our solution's fairness is better than that of BBR under cellular conditions, there is much room for improvement when comparing with BBRv2. In results not shown here, we find that under some conditions, goodput improves while fairness suffers, and under others, fairness is acceptable but goodput is still low. The problem here is the extreme loss. The median cellular condition in some cases sees a burst of loss where as much as 80% of packets are lost, with the average loss rate being about 5% overall. Our solution is currently unable to gracefully tradeoff fairness with Cubic at the expense of increased goodput under such large loss bursts.

### E. Performance evaluation for DASH video application

We compare the performance of our BBRv2 modification on a DASH video application to that of BBR and unmodified BBRv2. As shown in Table V, when no loss is induced, all three BBR variants achieve near-optimal video quality. However, under 2% loss, BBRv2's video quality decreases. Our modified BBRv2 improves video quality over standard BBRv2 in this case by 20%.

## VII. Related Work

Given that BBRv2 recently had its alpha release in 2019 [6], there is little published work on BBRv2's performance. The work by Ivanov at al. [8] study BBRv2 on the Dropbox edge




(a) BBR (b) BBRv2 (c) BBRv2 (our solution)

Fig. 9: Changing bandwidth set up: Throughput for BBR, BBRv2, and our adaptive BBRv2 under a changing bandwidth scenario with 2% induced loss.

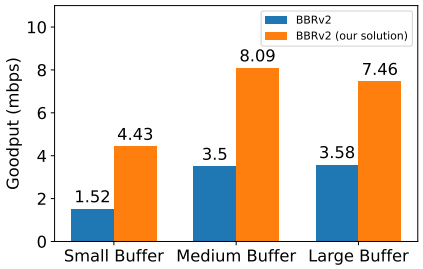



Fig. 10: Cellular set up: Goodput for BBRv2 and our BBRv2 version for median cellular condition with 10 mbps bandwidth.

| | BBR | BBRv2 | BBRv2 (our solution) |
|---|---|---|---|
| 0% induced loss | 9.614 | 9.43 | 9.542 |
| 2% induced loss | 9.754 | 8.002 | 9.746 |

TABLE V: Video quality, where 0 is the lowest bitrate and 10 is the highest, for BBR, BBRv2, and our adaptive BBRv2 under 100 ms RTT, 25 mbps bandwidth, and a 100 KB buffer.

network and show that the throughput of BBRv2 is comparable to BBR under high bandwidth and low buffer conditions. The study reports lower packet loss rates, less data in flight, lower RTTs, and higher fairness as compared to BBR. Yang et al. [9] evaluate BBRv2 using Mininet and find that it has slow receptivity to changing bandwidth and low resilience to random losses. To this end, they propose BBRv2+ which alters the bandwidth probe phase of BBRv2 to include a delay parameter based on the network conditions while providing more fair behavior.

Tierney et al. [15] study BBR and BBRv2 on Data Transfer Nodes (DTNs), which are characterized by high speed hosts, a large number of parallel flows, and high latency paths. The authors find that BBR performs much better on high latency and high loss paths, but at the expense of high retransmissions. They also state that long BBRv2 flows are unfair to Cubic flows on long latency and high loss paths. Nandagiri et al. [16] perform an experimental evaluation of BBR vs. BBRv2, focusing on fairness, queue delay, and link utilization. The authors confirm BBR's unfairness to Cubic due to its lack of response to losses. Our focus in this paper is primarily on BBRv2's poor performance under loss. Recently, there have been modeling works on BBRv2. Scherrer et al. [17] model the performance and fairness of BBR and BBRv2 and validate their model using Mininet. Mishra et al. [18] model BBR's performance when it competes with differing proportions of BBR and Cubic flows and conclude that BBR's throughput advantage over Cubic diminishes as more BBR flows join. Their model also gives accurate throughput predictions for BBRv2 when RTT is small.

However, to the best of our knowledge, *ours is the first measurement study that evaluates BBRv2's performance in LAN and WAN settings under diverse network conditions*.

In contrast, there have been several studies that evaluate the performance of BBR. For example, Cao et al. [3] study the performance of BBR under various network conditions and find that BBR is unfair to conventional TCP congestion algorithms and does not perform well under deep buffer conditions. Ware et al. [5] show that BBR is especially unfair when competing with multiple Cubic flows and Philip et al. [19] find that BBR can be unfair even to itself when experiments are not run at the edge but at the core.

## VIII. Conclusion

This work presents the first performance evaluation of BBRv2 versus BBR under a diverse set of network and loss conditions. We construct a decision tree based on our empirical measurements to compare the performance of BBR and BBRv2. Our key finding is that, under induced random loss and bursty losses, BBRv2 experiences significant goodput degradation (as much as 871%) compared to BBR. This performance degradation extends to the WAN, cellular network conditions, and video quality. Our investigation of BBRv2's code reveals that the long-term upper bound on packets-in-flight employed by BBRv2 excessively suppresses the goodput under lossy conditions. To address this problem, we devise a modification to BBRv2 that sets the upper bound adaptively, making BBRv2 react to losses without reducing performance. Our adaptive BBRv2 achieves goodput comparable to BBR and achieves close to the fairness achieved by BBRv2. We conclude that, with our modifications, BBRv2 can be a suitable replacement for BBR, especially under losses.

## REFERENCES


[1] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "Bbr: Congestion-based congestion control," *Queue*, vol. 14, no. 5, pp. 20–53, 2016.

[2] A. Mishra, X. Sun, A. Jain, S. Pande, R. Joshi, and B. Leong, "The great internet tcp congestion control census," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 3, no. 3, Dec. 2019. [Online]. Available: https://doi.org/10.1145/3366693

[3] Y. Cao, A. Jain, K. Sharma, A. Balasubramanian, and A. Gandhi, "When to use and when not to use bbr: An empirical analysis and evaluation study," in *Proceedings of the Internet Measurement Conference*, ser. IMC '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 130–136. [Online]. Available: https://doi.org/10.1145/3355369.3355579

[4] R. Ware, M. Mukerjee, S. Seshan, and J. Sherry, "Beyond Jain's Fairness Index: Setting the Bar For The Deployment of Congestion Control Algorithms," in *Proceedings of the Eighteenth ACM Workshop on Hot Topics in Networks (HotNets)*. New York, NY, USA: ACM, 2019.

[5] ——, "Modeling bbr's interactions with loss-based congestion control," in *Proceedings of the 2019 ACM SIGCOMM Internet Measurement Conference (IMC)*, ser. IMC '19. New York, NY, USA: ACM, 2019.

[6] Google, "Tcp bbr v2 alpha/preview release," https://github.com/google/bbr/\newlineblob/v2alpha/README.md, Jul 2019.

[7] N. Cardwell, Y. Cheng, K. Yang, P. J. Soheil Hassas Yeganeh, Y. Seung, L. Hsiao, M. M. V. Jacobson, I. Swett, B. Wu, and V. Vasiliev, *BBRv2 Update: Internet Drafts & Deployment Inside Google*, IETF-112 : iccrg, Nov 8, 2021. [Online]. Available: https://datatracker.ietf.org/meeting/112/materials/slides-112-iccrg-bbrv2-update-00

[8] A. Ivanov, "Evaluating bbrv2 on the dropbox edge network," 2020. [Online]. Available: https://arxiv.org/abs/2008.07699

[9] F. Yang, Q. Wu, Z. Li, Y. Liu, G. Pau, and G. Xie, "Bbrv2+: Towards balancing aggressiveness and fairness with delay-based bandwidth probing," *Computer Networks*, vol. 206, p. 108789, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128622000226

[10] U. Goel, M. Steiner, M. P. Wittie, S. Ludin, and M. Flack, "Domain-sharding for faster HTTP/2 in lossy cellular networks," *CoRR*, vol. abs/1707.05836, 2017. [Online]. Available: http://arxiv.org/abs/1707.05836

[11] V. Arun and H. Balakrishnan, "Copa: Practical Delay-Based congestion control for the internet," in *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*. Renton, WA: USENIX Association, Apr. 2018, pp. 329–342. [Online]. Available: https://www.usenix.org/conference/nsdi18/presentation/arun

[12] M. Rudow, F. Y. Yan, A. Kumar, G. Ananthanarayanan, M. Ellis, and K. Rashmi, "Tambur: Efficient loss recovery for videoconferencing via streaming codes," in *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. Boston, MA: USENIX Association, Apr. 2023, pp. 953–971. [Online]. Available: https://www.usenix.org/conference/nsdi23/presentation/rudow

[13] Akamai, "cell-emulation-util github," https://github.com/akamai/cell-emulation-util.

[14] S. Vargas, R. Drucker, A. Renganathan, A. Balasubramanian, and A. Gandhi, "BBR Bufferbloat in DASH Video," in *Proceedings of the Web Conference 2021*, ser. WWW '21. Ljubljana, Slovenia: Association for Computing Machinery, 2021, p. 329–341.

[15] B. Tierney, E. Dart, E. Kissel, and E. Adhikarla, "Exploring the bbrv2 congestion control algorithm for use on data transfer nodes," in *2021 IEEE Workshop on Innovating the Network for Data-Intensive Science (INDIS)*, 2021, pp. 23–33.

[16] A. Nandagiri, M. P. Tahiliani, V. Misra, and K. K. Ramakrishnan, "Bbrvl vs bbrv2: Examining performance differences through experimental evaluation," in *2020 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN*, 2020, pp. 1–6.

[17] S. Scherrer, M. Legner, A. Perrig, and S. Schmid, "Model-based insights on the performance, fairness, and stability of bbr," in *Proceedings of the 22nd ACM Internet Measurement Conference*, ser. IMC '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 519–537. [Online]. Available: https://doi.org/10.1145/3517745.3561420

[18] A. Mishra, W. H. Tiu, and B. Leong, "Are we heading towards a bbr-dominant internet?" in *Proceedings of the 22nd ACM Internet Measurement Conference*, ser. IMC '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 538–550. [Online]. Available: https://doi.org/10.1145/3517745.3561429

[19] A. A. Philip, R. Ware, R. Athapathu, J. Sherry, and V. Sekar, "Revisiting tcp congestion control throughput models and fairness properties at scale," in *Proceedings of the 21st ACM Internet Measurement Conference*, ser. IMC '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 96–103. [Online]. Available: https://doi.org/10.1145/3487552.3487834