BBR Bufferbloat in DASH Video

Santiago Vargas*
Stony Brook University
USA
savargas@cs.stonybrook.edu

Rebecca Drucker*
Stony Brook University
USA
rdrucker@cs.stonybrook.edu

Aiswarya Renganathan
Stony Brook University
USA
arenganathan@cs.stonybrook.edu

Aruna Balasubramanian
Stony Brook University
USA
arunab@cs.stonybrook.edu

Anshul Gandhi
Stony Brook University
USA
anshul@cs.stonybrook.edu

ABSTRACT

BBR is a new congestion control algorithm and is seeing increased adoption especially for video traffic. BBR solves the bufferbloat problem in legacy loss-based congestion control algorithms where application performance drops considerably when router buffers are deep. BBR regulates traffic such that router queues don't build up to avoid the bufferbloat problem while still maintaining high throughput. However, our analysis shows that video applications experience significantly poor performance when using BBR under deep buffers. In fact, we find that video traffic sees inflated latencies because of long queues at the router, ultimately degrading video performance. To understand this dichotomy, we study the interaction between BBR and DASH video. Our investigation reveals that BBR under deep buffers and high network burstiness severely overestimates available bandwidth and does not converge to steady state, both of which results in BBR sending substantially more data into the network, causing a queue buildup. This elevated packet sending rate under BBR is ultimately caused by the router's ability to absorb bursts in traffic, which destabilizes BBR's bandwidth estimation and overrides BBR's expected logic for exiting the startup phase. We design a new bandwidth estimation algorithm and apply it to BBR (and a still-unreleased, newer version of BBR called BBR2). Our modified BBR and BBR2 both see significantly improved video QoE even under deep buffers.

CCS CONCEPTS

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

ACM Reference Format:
Santiago Vargas[1], Rebecca Drucker[1], Aiswarya Renganathan, Aruna Balasubramanian, and Anshul Gandhi. 2021. BBR Bufferbloat in DASH Video. In *Proceedings of the Web Conference 2021 (WWW '21), April 19–23, 2021, Ljubljana, Slovenia.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3442381.3450061

WWW '21, April 19–23, 2021, Ljubljana, Slovenia

ACM ISBN 978-1-4503-8312-7/21/04.
https://doi.org/10.1145/3442381.3450061

1 INTRODUCTION

BBR (Bottleneck Bandwidth and RTT) [8] is a relatively new congestion control algorithm which is seeing surging adoption in practice. Recent reports suggest that 40% of Internet traffic now uses BBR [33]. BBR is especially popular for video applications and is adopted by popular video streaming services like YouTube [7]. Given the prevalence of video traffic (accounting for 60.6% of total downstream traffic worldwide [44] in 2019) and the adoption of BBR as the primary congestion control algorithm for video, it is critical to understand the impact of BBR on video Quality of Experience (QoE). Unfortunately, given its recency, BBR is not as well-studied as other congestion control algorithms, especially in the context of video traffic.

In this paper, we provide an in-depth characterization of video QoE under BBR. To this end, we study how video performance is affected by the combination of congestion control algorithms and router configurations for different HTTP versions, video streaming algorithms, and network conditions. We especially focus on router configurations because the end hosts do not have control over how routers are configured, but the router configuration can severely affect the performance of congestion control algorithms, and in-turn the end application.

Congestion control algorithms critically impact application performance by regulating the amount of traffic to be sent on the network without causing congestion at the routers. The most popular congestion control algorithm, Cubic [33], uses packet loss to infer that routers are congested and regulates traffic accordingly. However, under deep buffers (since packet losses are limited), Cubic fails to infer congestion [6], resulting in inflated packet latencies and poor application performance, a condition known as *bufferbloat* [13]. BBR is designed specifically to address the problem of bufferbloat by not using losses to infer congestion but instead estimating the network's bandwidth delay product, *BDP*, and regulating its sending rate to maximize throughput while attempting to maintain BDP worth of packets in flight, *irrespective* of the size of the buffer.

Surprisingly, we find that DASH video under BBR performs poorly when router buffers are large, significantly affecting video QoE, including quality and stall rate. For example, even for moderate router buffer sizes of 3MB, video QoE drops by over 50% when using BBR, compared to shallow buffers. This buffer size corresponds to 9× BDP in our experiments, and studies show that router buffers are typically 10× to 30× BDP [27]. We find, via controlled and

*Primary authors with equal contribution.

WAN (Wide Area Networks) experiments, that this poor video performance for BBR persists under large buffers across various network configurations, HTTP versions, and video algorithms.

We also study the video performance of BBR2 [37–39], a newer version of BBR that is still unreleased but is designed to address some of the drawbacks of BBR (see Section 2). We find that BBR2 performs slightly better than BBR, but video QoE still drops sharply as buffer size increases. Cubic also performs poorly under deep buffers, but this is a well known, Cubic-specific problem [24].

The question, then, is why does BBR, designed specifically to overcome bufferbloat and be agnostic to router buffer sizes, perform poorly when buffer sizes are large? We make three observations.

(1) We find that BBR's latencies are elevated, showing that it is indeed bufferbloat that is causing poor video QoE. In other words, even though BBR is designed to only maintain BDP number of outstanding packets, more packets are filling up in the buffer and inflating latencies.
(2) We find that BBR sends many more packets into the network because BBR severely *overestimates* the bandwidth when streaming video under deep buffers. In our experiments with deep buffers, BBR overestimates bandwidth by over 3×, resulting in a much larger estimated BDP, and consequently sending excessive packets into the network.
(3) Under deep buffers, BBR and BBR2 spend a larger fraction of time in the initial *STARTUP* phase where packets are sent more indiscriminately compared to the steady state. For example, BBR2 spends 32% of its time in the STARTUP phase under deep buffers compared to less than 1% when the router buffers are shallow.

To identify the root cause of this bandwidth overestimation and extended stay in the STARTUP phase, we carefully investigate the impact of various configuration settings on video QoE. We find that BBR's elevated packet sending rate is ultimately caused by the router's ability to absorb bursts in traffic, which destabilizes BBR's bandwidth estimation and overrides BBR's expected logic for exiting the STARTUP phase. The bursty video traffic pattern of DASH further exacerbates this issue.

We address this root cause by modifying the BBR source code to improve its bandwidth estimation in the presence of temporary but artificially high bandwidth availability at routers. By accounting for potential outliers in BBR's moving window of bandwidth observations, we provide a more faithful bandwidth estimate. We modify both BBR and BBR2's bandwidth estimation. Using our new bandwidth estimation, video QoE improves significantly—the quality increases from under 4 to over 9 where max quality is 10, and the stall rate reduces from over 0.7 to 0. Our new bandwidth estimation is simple, easy-to-implement, and has the potential to solve the critical problem of bufferbloat, while retaining the benefits of BBR.

2 BACKGROUND

This section provides an overview of adaptive video streaming, followed by a brief description of BBR and BBR2 congestion control algorithms, which are the focus of our study.

Video Streaming: Video is the dominant traffic type throughout the Internet. In 2019, video accounted for 60.6% of total downstream volume worldwide [44]. The number of US users watching live video on Facebook has increased 50% since January 2020 [47]. Due to the lockdown effects of COVID-19, the popularity of video streaming has only gone up [50]. It is estimated that by the end of 2020, the number of online streaming users in US will surpass 240 million [54].

Much of this video traffic is delivered by HTTP-based adaptive streaming protocols such as Dynamic Adaptive Streaming over HTTP, or *DASH* [51]. In particular, DASH is an international Internet video streaming standard that is supported by large video streaming providers, including YouTube and Netflix [21]. In DASH, the web server hosts video files, which are segments of equal length encoded in multiple bitrates, along with a manifest file describing the segments and their bitrates. In general, a higher bitrate provides better video quality, but results in a higher file size. When a streaming session is started on the client, a DASH video client first learns about the available bitrates by downloading the manifest file. The client then downloads video segments with the goal of providing the best QoE for the end-user. The video QoE is often defined in terms of features such as video quality and rebuffering time [5].

Adaptive Bitrate (ABR) algorithms: To choose between the different segment bitrates, a client uses an ABR algorithm that dynamically decides the bitrate for subsequent segments with the goal of providing the highest Quality-of-Experience (QoE) to the end-user [5]. These ABR algorithms use network and/or player signals like available bandwidth estimates and player buffer occupancy to select the quality of future segments.

To maximize QoE under DASH, various ABR algorithms have been proposed, such as DYNAMIC [48], BOLA [49], Festive [25], Rate-Based (RB), and BBA [22]. Each of these algorithms uses different signals to choose the next segment quality—BOLA runs online optimizations using buffer occupancy, Festive uses the throughput of the last 5 segments, Rate-Based uses throughput estimation from the most recent segment download, DYNAMIC uses both buffer occupancy and throughput estimation, and BBA aims to keep a minimum buffer occupancy range, gradating quality based on this minimum.

While we experiment with all five of the aforementioned ABR algorithms in this paper, we primarily employ the recent *DYNAMIC* [48] ABR algorithm, which is the default algorithm used in the dash.js reference client [11]. DYNAMIC uses a simple throughput-based ABR algorithm, THROUGHPUT, when the client buffer level is low (including at startup) and dynamically switches to BOLA, another ABR algorithm, when the buffer level is high. THROUGHPUT [48] first estimates the network throughput using a sliding-window approach, and then picks the highest bitrate that can be accommodated (with a small safety factor) by the estimated bandwidth. BOLA [49] is an online ABR algorithm with provable optimality guarantees that makes ABR decisions based on buffer levels to maximize QoE. The reasoning here is that throughput-based ABR has superior performance when the buffer is low or empty, whereas BOLA performs better when the buffer levels are large.

BBR: Online video streaming requires both high throughput and low latency for optimal QoE. While ABR algorithms can adjust to underlying network conditions, the network performance is also dictated by the lower-level TCP protocol.

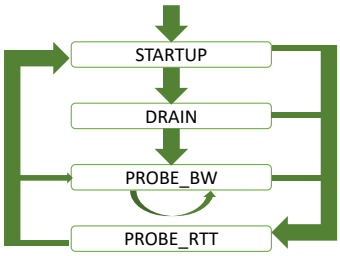

Figure 1: The state diagram for BBR's congestion control algorithm showing the four modes of operation.

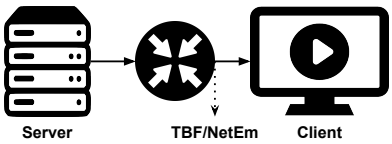

Figure 2: Illustration of our LAN testbed topology.

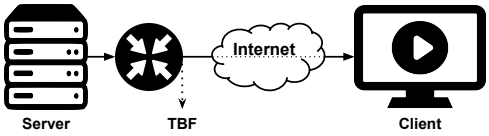

Figure 3: Illustration of our WAN testbed topology.

TCP aims to achieve the maximum throughput by having a send buffer whose size is approximately the bandwidth delay product (BDP) between the sender and the receiver. To deal with network congestion, traditional TCP algorithms enforce congestion control based on packet losses; when there is packet loss, TCP assumes that there is congestion and backs off to a lower transmission rate. However, when there is a large buffer at a network device, the packet loss detection can be significantly delayed, leading to *bufferbloat* – TCP overestimating the BDP and sending larger bursts of packets that fill up the deep buffers, resulting in high delays [13, 59].

In 2016, Google released the BBR (Bottleneck Bandwidth and Round-trip propagation time) congestion control algorithm [8], which attempts to operate at the network's optimal operating point [26] by maximizing achieved bandwidth while minimizing delay. To avoid bufferbloat, the amount of in-flight packets under BBR is a small multiple of the bandwidth-delay product (BDP); ideally, this should result in low buffer occupancy and, consequently, low delays. Since its release, BBR has been widely adopted by many online services around the world [33], and is in use in Google's B4 network as well as in YouTube video servers [8].

BBR works by operating in 4 modes, as shown in Figure 1: STARTUP, DRAIN, PROBE_BW, and PROBE_RTT. BBR starts in the STARTUP mode, where the goal is to quickly estimate available bandwidth. As in prior congestion control algorithms, BBR doubles its sending rate with each round trip until the measured bandwidth does not increase further; at this point, BBR is assumed to have reached the bottleneck bandwidth and thus exits the STARTUP mode. To drain the queue created during STARTUP, BBR enters DRAIN mode and temporarily reduces its *pacing_gain*. BBR then enters the PROBE_BW mode where it maintains steady state but periodically probes for more available bandwidth. To estimate available bandwidth, in PROBE_BW, BBR continuously samples the achieved bandwidth and uses the maximum over a moving window as its *BWest* [36]. Finally, BBR occasionally enters the PROBE_RTT mode to check the propagation delay and update its estimates of RTT (*MinRTT*). In PROBE_RTT, the congestion window size (*cwnd*) is reduced to 4 packets to drain any possible bottleneck queue, to obtain accurate RTT measurements.

BBR2: Despite its success, BBR has a few well documented shortcomings [6, 18, 23, 45, 52], including its inability to actively respond to losses, its unfair exploitation of available bandwidth when coexisting with other flows, the high retransmission rate in shallow buffers, and the severely low throughput in the PROBE_RTT mode. To address these shortcomings, Google introduced BBR2, which is still being evolved with ongoing improvements [37–39]. While not yet in production, we do employ BBR2 in our experiments for the sake of completeness. The major changes from BBR to BBR2 include improved (fair) coexistence with loss-based TCPs, much lower loss rates under shallow buffers, and greatly minimized throughput reduction in PROBE_RTT mode. While BBR2 does not react to individual packet losses, BBR2 does attempt to maintain a maximum packet loss rate before reducing its sending rate.

3 EXPERIMENTAL SETUP

Our goal is to characterize the performance of video applications under different TCP variants and router configurations. To this end, we measure video QoE across different HTTP versions, video streaming/ABR algorithms, network conditions, and router configurations. We consider three TCP variants in our experiments: BBR, BBR2, and Cubic (the default TCP variant used in Linux). In all, our study involved around 700 experiments.

3.1 Device and Network Setup

There are a large number of confounding network and end-host parameters that affect video QoE. To isolate the effect of these different factors, we run controlled experiments. We validate our findings through WAN experiments.

Network topology. For the controlled experiments, we connect a video client and a video server through a router. The client and server are run on Ubuntu 18.04 host machines. The machines are connected in a dumbbell topology, as shown in Figure 2, through a Linksys WRT1900ACS router with OpenWRT 19.07.1 installed. We simulate the bottleneck link and vary router configurations directly at the router. Setting network conditions at the router, as opposed to at the client or server, avoids interfering with TCP Small Queues [2, 10] and allows for more accurate and realistic control of bandwidth and RTT constraints [6].

Network and router configurations. We experiment with two different network conditions: a short real world RTT of 10ms and a larger RTT of 100ms. In both cases, we set the bandwidth to 25Mbps, as this bandwidth is sufficient to support our highest-quality video segments. We use the Linux utility TC (traffic control) NetEm [17] to impose delays and regulate bandwidth. We note that we do *not* explicitly set a loss rate; losses occur due to buffer overflow at

the router based on our router configuration, as is the case in the real-world.

When sending or receiving packets, a typical end host application encounters various router buffer configurations (that it has no control over) that have different effects on the application performance. We study this by varying two of the most common router parameters: *Limit* or *Buffer Size*, and *Burst*. Limit/Buffer Size defines the amount of data that the router buffer can hold before dropping packets, and burst is the maximum amount of traffic a router can process above the connection bandwidth limit. Both of these parameters are typically regulated using a token bucket filter [1]. The token bucket controls the rate at which packets may be sent by generating tokens at the specified bandwidth. A packet size cannot be sent out until enough tokens have been generated, commensurate to the packet size, and excess tokens are stored up to the specified burst size for future packet transmissions. We vary the burst parameter from 8KB to 2MB, based on configurations reported in prior work [12], and we vary the buffer size from 10KB to 100MB. Modern technologies like DOCSIS 3.1 support bursts in excess of 30MB for 1Gbps links [55, 56]; this roughly translates to a 1MB burst for our 25Mbps link. Unless specified otherwise, we present results when using a burst size of 1MB.

WAN setup. To validate our findings from the controlled experiments, we also experiment with a long-distance WAN network (over commodity links) with the server hosted in Northeast US and the client located at least 2,000 miles away from Northeast US, resulting in average RTT of around 200ms. Our WAN network topology is illustrated in Figure 3. We set the bandwidth for this network to 25Mbps via our router (using TC); we verified that our router, with the 25Mbps limit, is indeed the bottleneck rate limiter in this network.

3.2 Video and Protocol Setup

Video player. We run our video experiments using the dash.js reference client [11], an open source JavaScript DASH video player, in Google Chrome v80. All experiments use the 10-minute 4K video Big Buck Bunny [43] as it is the reference video for dash.js. We encode this video using ffmpeg 2.7.6 to 11 video quality levels, 1mbps to 25mbps (4K) in intervals of 2.5mbps; we set the ffmpeg configuration parameters (minrate, maxrate, and bufsize) to ensure minimal segment size variance. GPAC MB4Box [3] creates the DASH metadata and video files from the encoded videos using 5-second segments (resulting in 127 total segments), similar to previous work and industry recommendations [9, 31].

We experiment with different popular ABR algorithms, including DYNAMIC [48], which is the default algorithm used in dash.js, BOLA [49], Festive [25], BBA [22], and RB which is a rate-based algorithm; see Section 2 for details on these ABR algorithms. We borrow the open source implementations of these ABR algorithms [30]. In the default case, we show results when using DYNAMIC.

Client, server, and network protocols. The client machine uses the dash.js reference client, but is enhanced to collect additional statistics. We serve video segments using an nginx static web server [40]. The web server is configured to use HTTP/1.1 or HTTP/2. We set congestion control on the server side, and experiment with Cubic, BBR, and BBR2. Since BBR2 is still unreleased in the Linux kernel, we install BBR2 as a congestion control by following Google's instructions to recompile the Linux kernel with BBR2 source code [14]. BBR is used by default across all experiments unless otherwise specified.

3.3 Metrics

We measure video QoE using two metrics: average segment quality and stall rate. Average segment quality of downloaded segments ranges from 0 to 10 (11 bitrate levels). Stall rate is computed as follows:

$$Stall\ Rate = \frac{total\ playback\ time - video\ length}{video\ length}, \qquad (1)$$

where a stall rate larger than 0 implies that a user experiences rebuffering. Both metrics are critical to user experience since streaming high quality videos with stalls is not ideal, whereas streaming lower quality videos will result in poor user experience even if there are no stalls [5]. Note that the average segment quality for a video is averaged over the quality of all downloaded video segments (127 segments, in case of the reference video we employ in our experiments). Apart from the QoE metrics, for analysis, we also collect other video metrics, including the video buffer level and ABR estimated throughput.

At the server side, for analysis, we collect TCP statistics including the smoothed RTT (or sRTT), throughput, loss rate, and delivery rates. The smoothed RTT is a weighted moving average of the current RTT experienced by a connection [42]. We also collect BBR-specific statistics including BBR's bandwidth estimate and mode of operation, as described in Section 2.

4 VIDEO QOE UNDER DEEP BUFFERS

In this section, we present our experimental results characterizing the Quality of Experience (QoE) of video streaming under various router configurations and for the TCP variants of BBR, BBR2, and Cubic. We make the surprising observation that video QoE deteriorates substantially under deep buffers, even for BBR and BBR2 that are designed specifically to overcome the problems arising from deep buffers. This QoE deterioration extends across varying network conditions, HTTP variants, and ABR algorithms.

4.1 Video QoE under deep and shallow buffers

We start by empirically comparing the QoE under BBR and BBR2 with that of loss-based congestion control, specifically Cubic, in the LAN setting described above. We note that Cubic is the default congestion control algorithm in Linux and also the dominant TCP variant used in the Internet [33].

Figures 4 and 5 show the QoE of Cubic, BBR, and BBR2 for varying bottleneck buffer sizes under a 100ms RTT; these experiments used HTTP/2 and are averaged over 10 runs.

Under deeper buffers (corresponding to 9 BDP and above), we find that the video QoE suffers significantly. For example, for a 10MB buffer size, the average segment quality is lower than 5 (compared to the highest achievable quality of 10) for all TCP variants. The corresponding stall rate for BBR is at least 0.75, implying that the video took 75% longer to play back compared to its size. Studies show that router buffers are typically 10x to 30x BDP [27] for similar

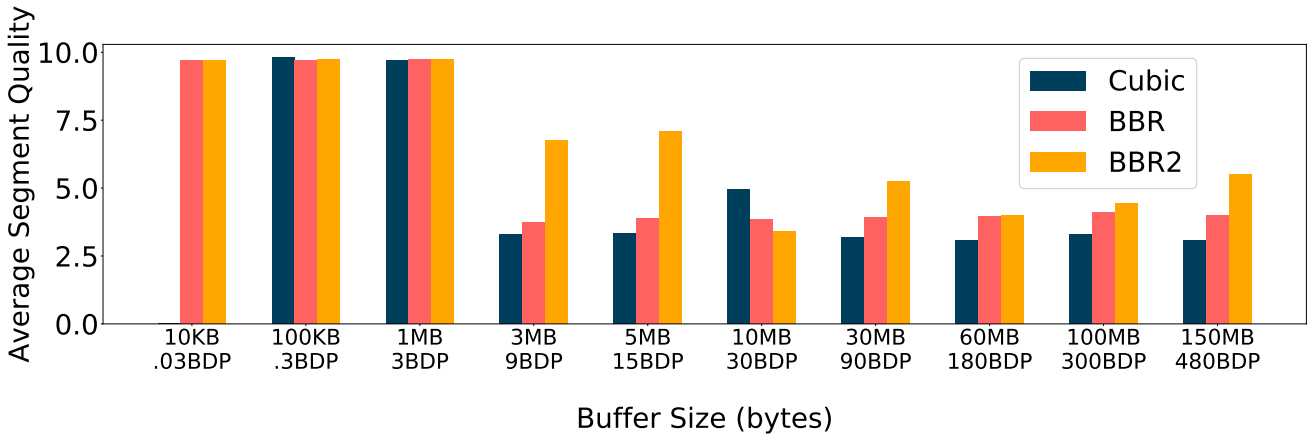

Figure 4: Average Segment Quality across bottleneck buffer sizes for Cubic, BBR, and BBR2. As the bottleneck buffer size increases, the video quality reduces even though BBR and BBR2 are explicitly designed to work well under bufferbloat.

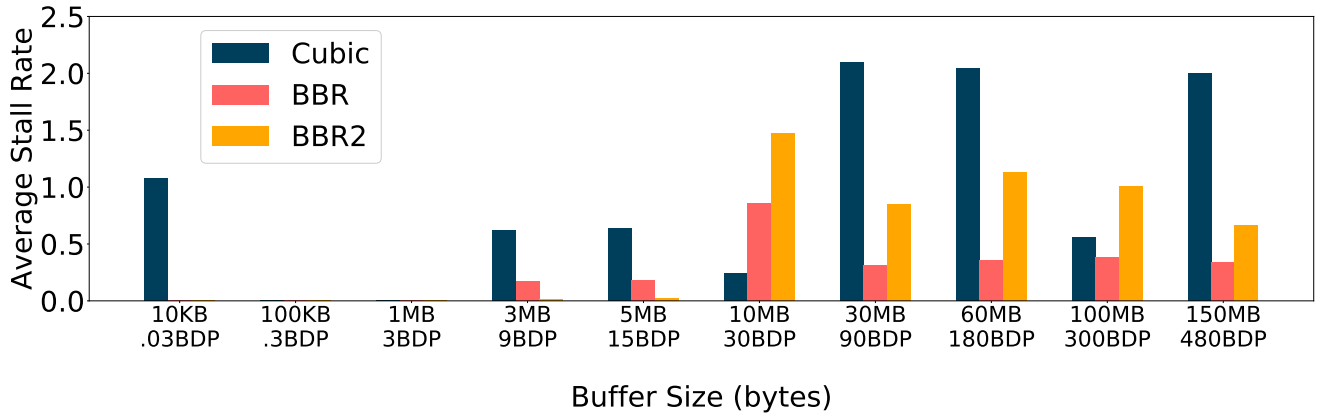

Figure 5: Average Stall Rate across bottleneck buffer sizes for Cubic, BBR, and BBR2. Similarly to average quality (Figure 4), under deep buffers, the stall rate increases.

100ms RTT conditions, so these setups are fairly common in the Internet.

While BBR2 performs better than BBR in terms of quality under 3MB and 5MB buffers, the attained segment quality is still significantly lower than the achievable quality of 10, as is the case for smaller buffer sizes. We discuss possible reasons for the improved performance of BBR2, compared to BBR, in Section 7.2.

Poor performance under deep buffers may be expected from legacy congestion control algorithms (such as Cubic) due to the downsides of using loss as a congestion signal. However, BBR and BBR2 are designed to keep router queues small with their mechanisms of estimating BDP at runtime and using packet pacing. However, they still experience poor performance under deep buffers.

When the bottleneck buffer is very small (10KB buffer, corresponding to 0.01BDP), Figure 4 shows that the segment quality is close to zero under Cubic with a very high stall rate of 1 (as seen in Figure 5). This is because the shallow buffer induces severe losses, resulting in low throughput for Cubic [6]. In contrast, BBR and BBR2, as intended, are both able to perform well since they do not directly react to individual losses. Slightly larger bottleneck buffers (100KB and 1MB) allow for improved QoE for all congestion controls (quality near 10 and a 0 stall rate) due to connections not being buffer limited.

4.2 Video QoE under different protocols and network conditions

Our experimental results in the previous subsection were obtained under HTTP/2 as the application layer protocol with a 100 ms RTT. For completeness, we also experiment with HTTP/1.1 and a 10 ms RTT in addition to a 100 ms RTT. Our results under these conditions are tabulated in Table 1; we report the QoE values averaged across five runs.

Our results are consistent across protocols and network conditions; video QoE continues to be poor under deep buffers (10MB). Under deep buffers, the quality is roughly 1 point higher for the 10 ms RTT than the 100 ms RTT for both BBR and BBR2. The QoE under shallow buffers (100KB) continues to be near-ideal.

RTT	Buffer Size	BBR Version	Avg. Quality	Stall Rate
10	100KB	BBR	9.86	0
10	100KB	BBR2	9.86	0
10	10MB	BBR	4.51	0.11
10	10MB	BBR2	5.25	0.08
100	100KB	BBR	9.85	0
100	100KB	BBR2	9.85	0
100	10MB	BBR	3.75	0.14
100	10MB	BBR2	3.68	0.24

Table 1: Average quality and stall rate for HTTP/1.1 video runs under 10ms and 100ms RTTs, shallow and deep buffers, and BBR and BBR2. QoE is low under deep buffers regardless of HTTP version or RTT.

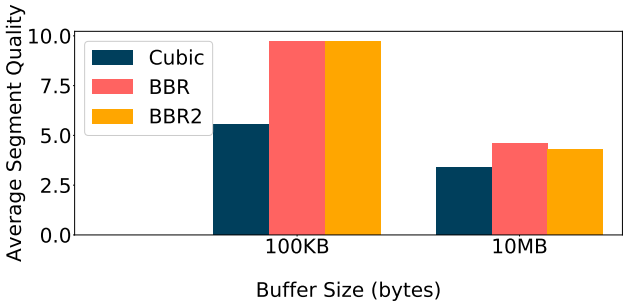

Figure 6: Video QoE under deep and shallow buffers in a WAN. The client and server are located at two different continents with roughly 200ms RTT. As before, we see the video QoE in terms of quality is poor under deep buffer conditions compared to shallow buffer condition.

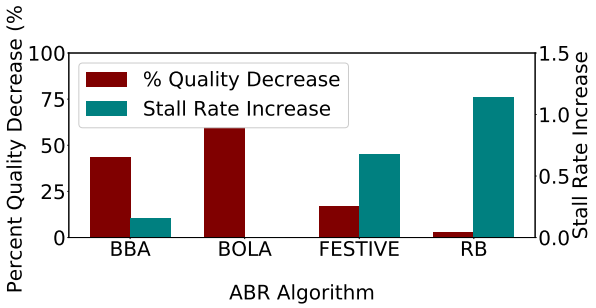

Figure 7: Video QoE deteriorates under deep buffers across ABR algorithms both in terms of video quality and stall rate.

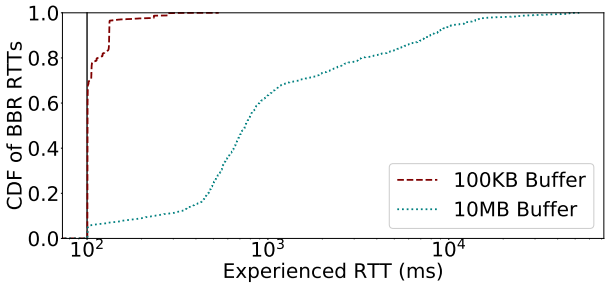

Figure 8: CDF of the RTTs estimated by BBR under shallow buffer (100KB) and deep buffer (10MB). Under deep buffers, RTTs are much higher than the propagation delay (100ms), indicating that packets experience large queueing delays. This is not the case in shallow buffers.

4.3 Video QoE in Wide Area Network

The aforementioned results were obtained under our controlled LAN environment. We now verify our findings under our WAN network that has larger RTTs (see Section 3.1 for details on our network setup).

Figure 6 shows our results for the WAN network. We see that video QoE is poor under deep buffers of 10MB (=16BDP) compared to bottleneck router buffer size of 100KB (= 0.16DP) for BBR and BBR2; the quality drops from around 10 to under 5. Interestingly, Cubic performs relatively poorly under both buffer conditions. Under shallow buffer, Cubic suffers large losses, and under deep buffer, Cubic sees bufferbloat due to RTT inflation. Thus, video QoE metrics continue to suffer under a deep buffer in the WAN network as well.

4.4 Video QoE under different ABR algorithms

A natural question that arises based on our observations from Section 4.1 is whether the drop in QoE under deep buffers is a result of specific artifacts in the default ABR algorithm used by the reference dash.js player. To explore this possibility, we repeat the experiments under different ABR algorithms (see Section 3.2 for details on the algorithms).

Figure 7 shows the difference in QoE between a smaller (100KB) and larger (10MB) bottleneck buffer under different ABR algorithms. The change in QoE is shown as a percentage difference for quality and as an absolute increase for stall rate. Overall, we see that video QoE continues to suffer under a large buffer. One group of ABRs, BBA and BOLA, experiences a near 50% decrease in quality along with a small increase in stall rate. The second group of ABRs, Festive and Rate-Based (RB), maintains high quality (less that 20% decrease in quality), but does so at the expense of a significant rise in stall rate. While all ABRs have virtually no rebuffering under 100KB buffers, Festive and RB suffer a huge increase (around 0.7 and 1.1 respectively) in stall rate under larger buffers.

We thus conclude that poor QoE under deep buffers cannot be attributed to the behavior of the ABR algorithm. We investigate possible root causes of poor QoE under deep buffers in the following section.

5 WHY IS VIDEO QOE POOR WHEN USING BBR UNDER DEEP BUFFERS?

We find that the poor video QoE when using BBR (or BBR2) is not only correlated with deep router buffers, it is in fact caused by deep router buffers. Under deep buffers, more packets are sent to the router buffers than can be processed, increasing queueing delays, inflating RTTs, and resulting in poor QoE. In this section, we show why BBR fills up router queues despite being designed to avoid bufferbloat. While a large part of the discussion focuses on BBR, the same phenomenon also occurs under BBR2.

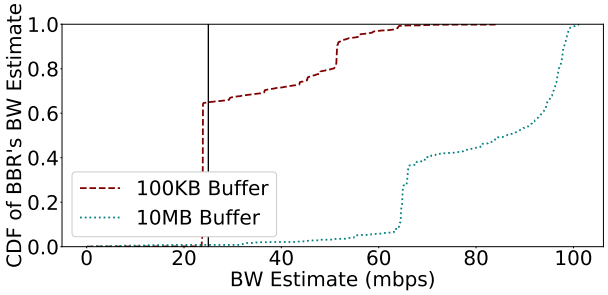

Figure 9: CDF of BBR's bandwidth estimates under shallow buffer (100KB) and deep buffer (10MB). Under deep buffers, BBR severely overestimates bandwidth. The available bandwidth is 25Mbps shown by the black vertical line.

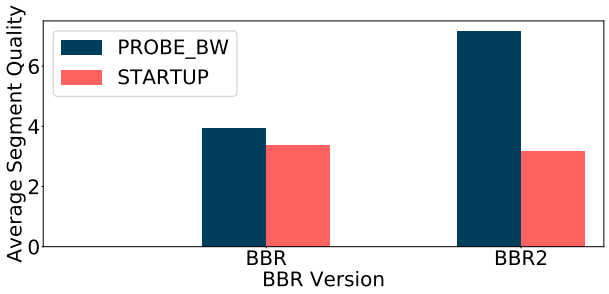

Figure 10: Average segment quality is lower for runs in STARTUP than runs in PROBE_BW. The quality difference is larger for BBR2 than BBR.

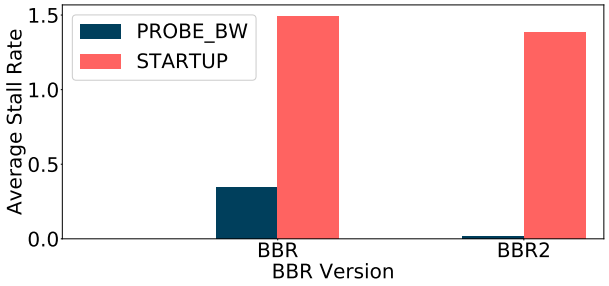

Figure 11: Average stall rate is higher for STARTUP runs than PROBE_BW runs. Both BBR and BBR2 have stall rates more than 3× higher for STARTUP than PROBE_BW.

5.1 Inflated RTTs

For video applications, packets are sent is terms of segments at a certain quality level. One reason for poor QoE is when the segment download experiences high latencies; this increases stall rates and the subsequent segments are requested at a lower quality level.

We find that the cause of poor QoE is in fact because packets experience high latencies when running BBR under deep buffers. Figure 8 shows the CDF of RTTs observed by BBR, for the network conditions of 100ms RTT and 25Mbps bandwidth. When the router buffer sizes are small (100KB in our experiments), in 70% of the cases, there are no queuing delays and the RTTs are equal to the propagation delay of the link. However, under deep buffers, the median RTT grows to 7× the propagation delay, severely inflating packet latencies.

Takeaway 1: BBR was designed to keep the router queues small using its novel RTT and bandwidth estimation and using packet pacing. However, BBR is not able to maintain these small queues for video traffic, resulting bufferbloat and inflated RTTs.

5.2 Bandwidth overestimation and lack of convergence

We next look at why BBR is not able to maintain small queues.

BBR severely overestimates available bandwidth: BBR is designed to ensure that buffer queues are small. It does so by estimating the bandwidth and minimum RTT, and caps the number of outstanding packets to 2 × bandwidth × min RTT, and uses packet pacing.

However, the problem is that for video traffic, BBR severely overestimates the *bandwidth*, in turn overestimating how much data can be sent. Figure 9 shows a CDF of BBR's bandwidth estimates across all our video runs under BBR. Note, the network rate is fixed at 25Mbps, denoted by the black line. Under the 100KB buffer, nearly 70% of BBR's bandwidth estimates are around the set network rate. However, when the buffer size increases to 10MB, BBR almost always overestimates available bandwidth, with the median overestimation being around 3× the set network rate of 25Mbps. This overestimation results in BBR sending excessive packets into the network.

Takeaway 2: BBR grossly overestimates bandwidth under deep buffers when streaming video, resulting in large queues at the router.

BBR is slower to converge to steady state: Recall from Section 2 that BBR operates in four modes. The steady state mode for BBR is the PROBE_BW mode where BBR sends at a target rate determined by its most recent bandwidth and min RTT estimates. BBR's STARTUP mode was designed to mimic the slow-start phase in loss-based congestion control. Similar to the slow start phase, when BBR is in the STARTUP mode, data is sent more than twice as fast compared to when in the PROBE_BW mode.

Usually, BBR spends most of its time in the PROBE_BW mode [8]. However, for video applications, when buffers are deep, we find that BBR spends a large fraction of its time in STARTUP mode, sending data indiscriminately and further increasing queue sizes. Table 2 shows the average percentage of time spent in each BBR state across video runs. For both BBR and BBR2, average time spent in STARTUP is higher in deep buffers than shallow buffers, and average time spent in PROBE_BW is lower. The difference is especially pronounced for BBR2: on average, a BBR2 video run in a deep buffer spends just over half its time in PROBE_BW, compared to nearly 100% when the buffer is shallow. BBR2 video runs often fail to converge to steady state at all, remaining in STARTUP for the duration of the video.

When BBR is in STARTUP state, video QoE is much worse compared to when it is in PROBE_BW. The effect is more pronounced

BBR Version	Avg % STARTUP		Avg % PROBE_BW		Avg % DRAIN		Avg % PROBE_RTT	
Buffer Size	100KB	10MB	100KB	10MB	100KB	10MB	100KB	10MB
BBR	0.23%	5.3%	98.4%	82.8%	0%	0.3%	1.2%	11.5%
BBR2	0.17%	**31.9%**	96.7%	**54.8%**	0.01%	0.16%	2.9%	13.1%

Table 2: Average percentage of video run spent in each BBR state. Runs spend more time in STARTUP for the deep 10MB buffer than the shallow 100KB buffer. The difference is more dramatic for BBR2.

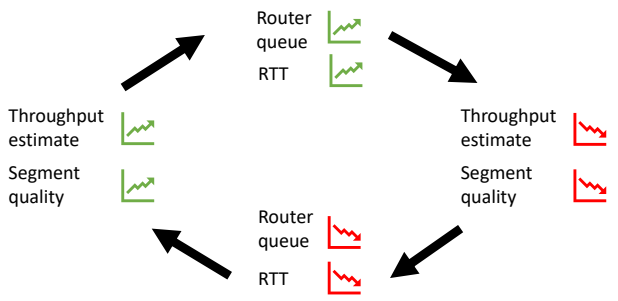

Figure 12: Under deep buffers and large burst, BBR and ABR both operate according to the control loop shown here. BBR's bandwidth overestimation and resulting excessive sending causes high RTTs due to queuing delay, resulting in poor throughput for the application. The application then sends lower-quality segments, which allows the router queue to drain and RTTs to return to normal. The cycle repeats when the application sends higher-quality segments at an unsustainable rate, filling the buffer again.

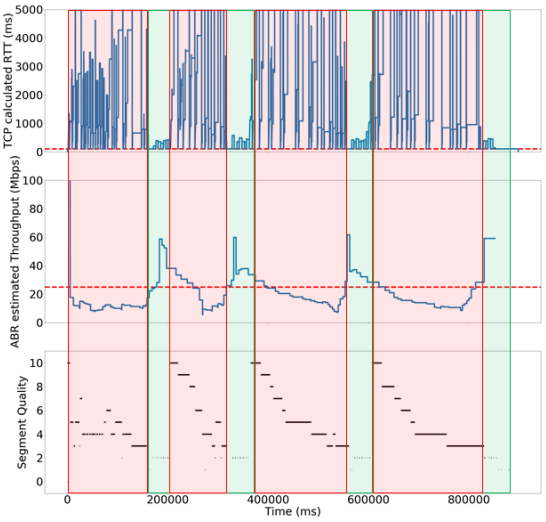

Figure 13: Illustration of how RTT, segment quality, and ABR estimated throughput cycle in response to one another over the course of a video run under a 10MB buffer and 1MB burst. Red dashed lines on the TCP calculated RTT plot and the ABR estimated throughput plot indicate the propagation delay (100ms) and set bandwidth (25Mbps) respectively.

for BBR2 than BBR. Figures 10 and 11 compare quality and stall, respectively, between runs whose primary state is STARTUP with runs whose primary state is PROBE_BW. We say that a run's primary state is the state in which it spends the plurality of its time. While BBR manages to maintain a similar average quality for runs stuck in STARTUP, the stall rate during STARTUP mode increases over 3×. For BBR2, the effect is more pronounced with quality decreasing by over 50% and stall rate jumping from virtually 0 under PROBE_BW to nearly 1.5 in STARTUP runs.

These figures show results from 60 video runs each for BBR and BBR2, where RTT is set to 100 ms, bandwidth is set to 25 Mbps, and buffer size is set to 10MB. All videos are served over HTTP/2. Note that videos served over HTTP/1.1 spend even more time in STARTUP because HTTP/1.1 opens multiple connections, all of which must meet BBR's conditions to exit STARTUP. As a result, HTTP/1.1 video QoE is even lower than HTTP/2 on average (not shown here).

Takeaway 3: For video traffic, BBR under deep buffers does not converge to steady state in several cases, instead remaining in the STARTUP mode. In the STARTUP mode, packets are sent indiscriminately, increasing queue sizes, and reducing video QoE.

5.3 Interaction between BBR and ABR

The sending rates under DASH video are controlled by both the congestion control algorithm and the ABR of the DASH player. We have a dual control loop problem where both ABR and BBR affect overall sending rates and can affect each other's sending rates as well. In this work, we focus on the BBR piece of the control loop issue by providing measurements of BBR. Accordingly, we will describe how the BBR algorithm affects ABR, leading to poor performance.

Our analysis suggests that it is BBR's overestimation of bandwidth that leads to unsustainably higher delivery rates under BBR, which in turn causes bufferbloat and results in poor video QoE. This bufferbloat, which manifests itself as high RTTs, reduces video QoE because ABR interprets high RTTs as low throughput. We illustrate the tight control loop between BBR and ABR in Figure 12, focusing on ABR operating in throughput mode (a similar control loop occurs in buffer mode as well):

(1) High RTTs cause the requested video segments to be sent to the client slower than expected.
(2) The throughput estimation indicates low throughput because high RTTs cause the requested segments to be sent

more slowly. The ABR selects lower quality segments due to low estimated throughput, resulting in poor video QoE.

(3) Sending smaller, low-quality segments allows the router queue to drain, lowering RTTs to normal levels.

(4) The throughput estimation indicates high throughput because multiple low-quality segments have been sent successfully over a short period of time. ABR then chooses a high-quality segment to match the perceived high throughput.

Steps (1) through (4) repeat throughout the video run. Figure 13 illustrates the cycling video quality over the course of one video run. Red areas correspond to steps (1) and (2) in the ABR control loop while green areas correspond to steps (3) and (4). The ABR and BBR control loops described above drive this quality cycle.

6 BURST IS THE CAUSE

We found in the previous section that bandwidth overestimation is the main culprit behind poor video QoE. In this section, we investigate the causes of BBR's bandwidth overestimation.

BBR estimates bandwidth by finding the *maximum* over throughput samples across ten RTTs. In Figure 14, we plot throughput samples (delivery rates are used as proxy for the samples) across 10 runs each for BBR and BBR2 under a 10MB deep buffer. Even though the maximum bandwidth in the link is only 25Mbps, in more than 20–30% of the samples, the bandwidth is above 25Mbps. In other words, the throughput for some samples is high, causing BBR to overestimate bandwidth.

We find that these spikes in throughput are caused by the router's ability to absorb bursts in traffic. Recall that routers are configured with a burst capacity [12] that allows the router to process a *burst* worth of packets above the connection rate (§3). Since BBR uses a max filter over throughput samples, BBR will incorrectly assume that the connection can sustain the burst rate. To verify this, we characterize video performance with burst sizes 8KB, 100KB, 1MB, and 2MB. Similar to previous works, we choose these burst sizes because they are commonly seen in the wild [12]. Figure 15 shows that under large burst sizes of greater than 1MB, video quality is poor. However, under smaller burst sizes of 8KB and 100KB bursts, average segment quality is 9, suggesting excellent video QoE. The results for stall rate are similar.

Circling back to bandwidth estimation, we find that indeed, under lower burst sizes, the overestimation problem in BBR disappears, as shown in Table 3. Similarly, under smaller bursts, BBR converges to steady-state and does not remain in extended STARTUP, different from when the burst sizes were large (figure omitted for brevity). We believe that video runs are more likely to remain in STARTUP for extended periods in deep buffers due to the deep buffer's ability to absorb large bursts of packets without inducing losses. Bursts vary in size, but as long as one RTT out of every three has at least a 25% larger burst than the other two, the BBR flow will not exit STARTUP [8].

Overall, large burst sizes cause overestimation and convergence problems in BBR and BBR2, severely impacting video QoE.

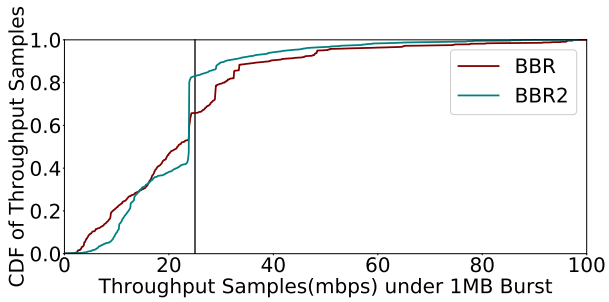

Figure 14: CDF of throughput samples obtained by BBR and BBR2 to estimate bandwidth. The bandwidth is estimated based on historical samples. For BBR, the samples exceed the set bandwidth of 25 Mbps in more than 30% of the cases; for BBR2, this happens in about 20% of the cases.

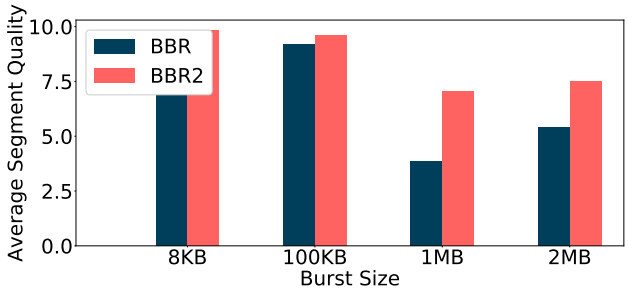

Figure 15: Average video segment quality for BBR and BBR2 under four burst sizes. BBR's average quality is 60% under a 1MB burst than an 8KB burst, while BBR2's quality is 30% lower under a 1MB burst than an 8KB burst.

	Avg. BBR est. BW	
Burst	**BBR**	**BBR2**
8KB	24.1	24.2
100KB	28.3	29.4
1MB	91.8	65.7
2MB	162.9	107.9

Table 3: Average BBR estimated bandwidth for BBR and BBR2 across burst sizes. The link bandwidth is set to 25Mbps. For large burst sizes of 1MB and 2MB, BBR and BBR severely ovestimate the bandwidth as also seen in Section 5.2. But for smaller burst sizes, BBR and BBR2 are able to fairly estimate the bandwidth.

7 NEW ALGORITHM FOR BANDWIDTH ESTIMATION

While we identify *burst* in traffic policers/shapers as a crucial parameter that affects BBR video QoE, the solution is not as simple as ISPs decreasing burst parameters across their networks. ISPs use wide ranges of different network hardware and technologies, all of which have varying buffer sizes and recommended burst sizes. For example, most DOCSIS 3.0 implementations use a burst between 1 and 1.5 MB in size, while the DOCSIS 3.1 standard [16] suggests using a burst between 10 MB and 30 MB depending on the

speed of the connection [55]. Further, changing the burst parameter to suit BBR video can produce unwanted effects across other applications/network settings.

7.1 BBR-S: BBR with Sampled BW Estimation

The key problem with BBR and BBR2, as noted in the preceding sections, is the overestimation of bandwidth. The bandwidth estimation is done over 10 RTTs, which is a long period of time; it takes 10 RTTs to phase out old data. Further, the estimate picks the maximum bandwidth from the samples, which results in overestimation even if one sample experiences high bandwidth. However, if we underestimate bandwidth, then BBR will not be able to efficiently use the available bandwidth, resulting in poor throughput.

Instead, we implement a bandwidth estimation mechanism that balances over- and under-estimation. Our new algorithm, BBR-Sampled (BBR-S for short), derives the estimated bandwidth from a list of sampled estimates. The main differences between BBR and BBR-S are outlined below:

(1) Instead of storing a single, window-based max bandwidth filter, BBR-S stores a history of bandwidth samples every time an ACK for a non-app-limited transmission is received.
(2) Instead of using the maximum bandwidth sample from the history, BBR-S treats the history as a distribution of samples and sets its bandwidth estimate based on a *hist_percentile* parameter, representing a high percentile of the observed samples.

The key idea behind this approach is that since network bursts are short by definition, we can remove bandwidth estimates that have been amplified by these short bursts. In our implementation, we use a sample history size of 100. Recall from Figure 14 that the throughput samples seen by BBR and BBR2 have a long tail. At the 85th percentile, the overestimation is modest, so we set *hist_percentile = 85%ile*. On further experimentation (not shown here), we find that modest overestimation of bandwidth does not affect video QoE, which indicates that there is some flexibility in the choice of *hist_percentile*. While the above solution validates BBR's over-estimation of bandwidth with DASH video workloads, we do note that it is not intended to be a general purpose solution for all types of traffic. Our solution can serve as a starting point for further improving BBR. Analytically finding the optimal *hist_percentile* under this solution is also left for future work.

7.2 Bandwidth estimation for BBR2

BBR2 uses a slightly different bandwidth estimation compared to BBR. Specifically, BBR2 does not use the maximum bandwidth across a window, and instead estimates bandwidth to be 85% of the maximum bandwidth across a window. This is done to provide headroom for competing flows to grow quickly, because BBR can be unfair to other flows [52]. This naturally will result in BBR2 not overestimating bandwidth quite as much as BBR, and is perhaps responsible for the slight improvements in QoE between BBR and BBR2 under some buffer sizes seen in Figures 4 and 5. However, BBR2 still overestimates bandwidth (as shown in Figure 14) and video QoE under BBR2 is still poor for deep buffers.

We modify BBR2's bandwidth estimation as follows: we use the distribution of samples and pick the *hist_percentile* value, after which we set the estimated bandwidth to be 85% of the value. We call this estimation algorithm BBR2-S. We find that *hist_percentile = 85%ile* works well for BBR2-S as well.

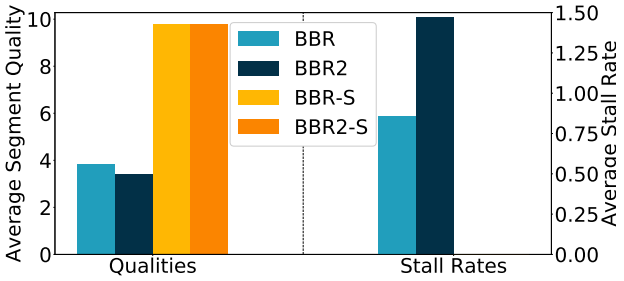

Figure 16: Our new bandwidth estimation algorithms BBR-S and BBR2-S outperform BBR and BBR2 respectively in terms of both quality and stall rate. Experimental setup is similar to that of Figure 4 and 5 with 100ms RTT, 25Mbps bandwidth, 10MB router buffer size, 1MB router burst size, over HTTP/2.

7.3 Performance of BBR-S and BBR2-S

Figure 16 shows the QoE of BBR-S and BBR2-S compared to BBR and BBR2, respectively. We use the same experimental setup as that used in Figure 4 and 5. The results show that our sampled approach significantly improves video QoE. In terms of average quality, BBR-S and BBR2-S improve quality from below 4 to near 10. In terms of stall rates, BBR-S and BBR2-S see no stalls at all, while the stall rates are greater that 0.75 for BBR and BBR2. In effect, our sampled-based estimation technique, which is easy to implement, addresses the bufferbloat problem for video applications for both BBR and BBR2, resulting in much improved video QoE.

8 RELATED WORK

This section discusses related work on QoE evaluation of video streaming and performance analysis of BBR.

ABR algorithms. Much of the work in improving the QoE of video applications has been focused on developing new ABR algorithms to better adapt to changing network conditions. The most closely related ABR work is Adaptive CoDel [32], that aims to mitigate bufferbloat by dropping packets at the router if the queue delay goes above a target value, thus improving the QoE real-time video streaming. LDM [28] also focuses on frame dropping to improve QoE for live video streaming over HTTP/2. Our efforts in this paper are directed at addressing bufferbloat-related issues in the TCP congestion control algorithms, specifically BBR and BBR2, without dropping any packets at the router.

There are also orthogonal ABR that use of machine learning techniques to better predict the available bandwidth and adapt bitrate accordingly [20, 46, 57] or focus on improving QoE when coexisting with other video streaming applications [24, 35]. Our focus in this paper is on mitigating the bandwidth overestimation problem under deep buffers in BBR and BBR2, irrespective of the ABR in use.

Recent work on the design of BBR. Recent work [6] has demonstrated that BBR outperforms other loss-based congestion control algorithms for shallow buffers, but performs poorly under deep buffers. In the case of deep buffers, the authors find that Cubic has higher throughput than BBR when using iPerf, because BBR fails to utilize the deep buffers to queue up enough packets. By contrast, we find that, for video traffic, BBR overestimates bandwidth and causes bufferbloat.

Ware et al. [53] investigate BBR's fairness properties when coexisting with other competing loss-based flows. The authors find that in such cases, BBR's inflight cap dictates the sending rate, which does not adapt to the number of competing flows, thus causing unfairness with legacy TCPs.

There have also been efforts to tailor BBR for specific applications to obtain improved performance. For example, RCP-BBR [34], designed for VoIP, works by removing the in-order delivery and packet retransmission features of BBR to achieve better throughput. Zhang et al. [58] propose a delay response BBR (Delay-BBR) algorithm specifically for real-time video transmission which reduces the sending rate when the link delay exceeds a threshold. While our work focuses on static video streaming, we believe that our BBR-S and BBR2-S modification can improve the performance of any bursty traffic type under deep buffers and large router burst sizes. Further, we do not modify the core TCP features of BBR, such as in-order packet delivery and retransmissions.

Network bandwidth variability. Flach et al. [12] show that loss rates are six times higher on average with traffic policing, which severely impacts video playback quality. The authors evaluate the impact of burst sizes in the range of 8KB – 2MB and find that rebuffering reduces with smaller burst sizes but larger burst sizes provide increased throughput. The choice of burst sizes in our experiments was based on the values employed by the authors. We note that our sampled bandwidth estimation solution reduces the sending rate at the server, thus avoiding excessive policing, while still providing high QoE to end-users.

Goyal et al. [15] explore the performance of iPerf under different congestion control algorithms, including BBR, and under time-varying links. They find that many congestion controls perform sub-optimally under time-varying links, which they evaluate using cellular traces. Specifically, they show that BBR experiences high delay under these links and BBR's rate does not match the true network rate. Our work evaluates the performance of BBR under DASH video and networks with burst configurations.

Addressing Bufferbloat. Mansy et al. [29] show that a single video stream can cause up to one second of queuing delay, which gets worse when the home link is congested in a typical residential setting. To address this problem, the authors present SABRE (Smooth Adaptive Bit RatE), a client based technique for video players that reduces delays by adjusting the flow control TCP window (rwnd) in a DASH client. Our paper focuses on bufferbloat caused by router settings, and is thus orthogonal to the client-centric work in SABRE.

PIE [41] is a latency-based enqueuing technique that limits the average queuing latency to a reference value by dropping packets based on queuing delay. Hohlfeld et al. [19] suggest using load-dependent buffer sizing schemes as they find that the level of competing network workloads, rather than the buffer size, is the main factor affecting end-user QoE. They also find that bufferbloat affects QoE only when the buffers are oversized and sustainably filled. Unlike the above works, we address the bufferbloat problem at the server, thereby avoiding filling up deep buffers and improving QoE. Further, we find that bufferbloat can impact QoE even under bursty traffic, as in video playback.

9 CONCLUSION

In this work, we present an in depth analysis into the performance of DASH Video under BBR, a recent and popular congestion control used by a large fraction of Internet traffic. We set out to understand the performance of BBR under varying network settings and router configurations.

We first find that, with BBR under deep buffers, video QoE is significantly worse even under different ABR algorithms, application protocols, and network conditions. On further investigation, we find the QoE drop occurs as a result of BBR bufferbloat, even though BBR was designed specifically to solve the problem of bufferbloat, reminiscent of loss-based congestion controls like Cubic operating in deep buffers. When analyzing BBR's behavior under deep buffers, two immediate observations emerge: (i) BBR severely overestimates the available bandwidth, thus pumping an excessive amount of packets, causing bufferbloat, and (ii) BBR fails to converge to its steady state, spending a long time in the startup phase indiscriminately sending packets, further exacerbating the bufferbloat problem. The root cause of BBR's overestimation is that BBR's bandwidth estimation does not account for network burstiness present in network shapers and policers, which are used by ISPs. Instead, BBR takes a greedy approach to bandwidth estimation by using the maximum observed bandwidth over a fixed time window as a proxy for available bandwidth.

To address this problem, we develop an alternative bandwidth estimation technique for BBR and BBR2 that filters out bursty bandwidth observations when estimating available bandwidth. We find that this change in estimation significantly improves the QoE of BBR and BBR2 under deep buffers.

While we examine BBR's response to network burstiness in the context of TBF burst, we believe that future work could look at BBR's bandwidth estimation in light of other network phenomena that may cause burstiness, such as cross-traffic, buffer sharing, and unstable network paths. We also make our experiment data and BBR-S code publicly available [4].

10 ACKNOWLEDGEMENTS

We thank the reviewers for their wonderful feedback. This work was supported in part by NSF grant CNS-1909356.

REFERENCES

[1] [n.d.]. TC-TBF - Linux man page. https://linux.die.net/man/8/tc-tbf.
[2] 2017. BBR evaluation with netem. https://groups.google.com/u/2/g/bbr-dev/c/8LYkNt17V_8.
[3] 2017. GPAC. https://github.com/gpac/gpac.
[4] 2021. BBR Bufferbloat in Video Project. https://github.com/SBUNetSys/BBR-Bufferbloat-Video-Project.
[5] Zahaib Akhtar, Yun Seong Nam, Ramesh Govindan, Sanjay Rao, Jessica Chen, Ethan Katz-Bassett, Bruno Ribeiro, Jibin Zhan, and Hui Zhang. 2018. Oboe: Auto-tuning Video ABR Algorithms to Network Conditions. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*. Budapest, Hungary, 44–58.

[6] Yi Cao, Arpit Jain, Kriti Sharma, Aruna Balasubramanian, and Anshul Gandhi. 2019. When to Use and When Not to Use BBR: An Empirical Analysis and Evaluation Study. In *Proceedings of the Internet Measurement Conference* (Amsterdam, Netherlands) *(IMC '19)*. Association for Computing Machinery, New York, NY, USA, 130–136. https://doi.org/10.1145/3355369.3355579

[7] Neil Cardwell. [n.d.]. Youtube and BBR. https://groups.google.com/u/1/g/bbr-dev/c/EipEH0Bq8J0.

[8] Neal Cardwell, Yuchung Cheng, C Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. 2016. BBR: Congestion-based congestion control. *Queue* 14, 5 (2016), 20–53.

[9] Cloudflare. [n.d.]. What is MPEG-DASH? https://www.cloudflare.com/learning/video/what-is-mpeg-dash/.

[10] Jonathan Corbet. 2012. TCP small queues. https://lwn.net/Articles/507065/.

[11] Dash-Industry-Forum. [n.d.]. Dash.js Github. https://github.com/Dash-Industry-Forum/dash.js.

[12] Tobias Flach, Pavlos Papageorge, Andreas Terzis, Luis Pedrosa, Yuchung Cheng, Tayeb Karim, Ethan Katz-Bassett, and Ramesh Govindan. 2016. An Internet-Wide Analysis of Traffic Policing. In *Proceedings of the 2016 ACM SIGCOMM Conference* (Florianopolis, Brazil) *(SIGCOMM '16)*. Association for Computing Machinery, New York, NY, USA, 468–482. https://doi.org/10.1145/2934872.2934873

[13] Jim Gettys and Kathleen Nichols. 2012. Bufferbloat: dark buffers in the internet. *Commun. ACM* 55, 1 (2012), 57–65.

[14] Google. 2019. TCP BBR v2 Alpha/Preview Release. https://github.com/google/bbr/blob/v2alpha/README.md.

[15] Prateesh Goyal, Anup Agarwal, Ravi Netravali, Mohammad Alizadeh, and Hari Balakrishnan. 2020. ABC: A Simple Explicit Congestion Controller for Wireless Networks. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. USENIX Association, Santa Clara, CA, 353–372. https://www.usenix.org/conference/nsdi20/presentation/goyal

[16] B. Hamzeh, M. Toy, Y. Fu, and J. Martin. 2015. DOCSIS 3.1: scaling broadband cable to Gigabit speeds. *IEEE Communications Magazine* 53, 3 (2015), 108–113.

[17] Stephen Hemminger. 2005. Network emulation with NetEm. *Linux Conf Au* (05 2005).

[18] Mario Hock, Roland Bless, and Martina Zitterbart. 2017. Experimental evaluation of BBR congestion control. In *2017 IEEE 25th International Conference on Network Protocols (ICNP)*. IEEE, 1–10.

[19] Oliver Hohlfeld, Enric Pujol, Florin Ciucu, Anja Feldmann, and Paul Barford. 2014. A QoE Perspective on Sizing Network Buffers. In *Proceedings of the 2014 Conference on Internet Measurement Conference* (Vancouver, BC, Canada) *(IMC '14)*. Association for Computing Machinery, New York, NY, USA, 333–346. https://doi.org/10.1145/2663716.2663730

[20] T. Huang and L. Sun. 2020. Deepmpc: A Mixture Abr Approach Via Deep Learning And Mpc. In *2020 IEEE International Conference on Image Processing (ICIP)*. 1231–1235.

[21] Te-Yuan Huang, Nikhil Handigol, Brandon Heller, Nick McKeown, and Ramesh Johari. 2012. Confused, Timid, and Unstable: Picking a Video Streaming Rate is Hard. In *Proceedings of the 2012 Internet Measurement Conference* (Boston, Massachusetts, USA) *(IMC '12)*. Association for Computing Machinery, New York, NY, USA, 225–238. https://doi.org/10.1145/2398776.2398800

[22] Te-Yuan Huang, Ramesh Johari, Nick McKeown, Matthew Trunnell, and Mark Watson. 2014. A Buffer-Based Approach to Rate Adaptation: Evidence from a Large Video Streaming Service. *SIGCOMM Comput. Commun. Rev.* 44, 4 (Aug. 2014), 187–198. https://doi.org/10.1145/2740070.2626296

[23] Per Hurtig, Habtegebreil Haile, Karl-Johan Grinnemo, Anna Brunstrom, Eneko Atxutegi, Fidel Liberal, and Åke Arvidsson. 2018. Impact of TCP BBR on CUBIC Traffic: A Mixed Workload Evaluation. In *2018 30th International Teletraffic Congress (ITC 30)*, Vol. 1. IEEE, 218–226.

[24] N. Iya, N. Kuhn, F. Verdicchio, and G. Fairhurst. 2015. Analyzing the impact of bufferbloat on latency-sensitive applications. In *2015 IEEE International Conference on Communications (ICC)*. 6098–6103.

[25] Junchen Jiang, Vyas Sekar, and Hui Zhang. 2012. Improving Fairness, Efficiency, and Stability in HTTP-Based Adaptive Video Streaming with FESTIVE. In *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies* (Nice, France) *(CoNEXT '12)*. Association for Computing Machinery, New York, NY, USA, 97–108. https://doi.org/10.1145/2413176.2413189

[26] Leonard Kleinrock. 1979. Power and deterministic rules of thumb for probabilistic problems in computer communications. In *Proceedings of the International Conference on Communications*, Vol. 43. 1–43.

[27] Christian Kreibich, Nicholas Weaver, Boris Nechaev, and Vern Paxson. 2010. Netalyzr: illuminating the edge network. In *ACM Internet Measurement Conference; Melbourne, Australia; November 2010*. 246–259.

[28] Y. Li, S. Wang, X. Zhang, C. Zhou, and S. Ma. 2020. High Efficiency Live Video Streaming With Frame Dropping. In *2020 IEEE International Conference on Image Processing (ICIP)*. 1226–1230.

[29] Ahmed Mansy, Bill Ver Steeg, and Mostafa Ammar. 2013. SABRE: A Client Based Technique for Mitigating the Buffer Bloat Effect of Adaptive Video Flows. In *Proceedings of the 4th ACM Multimedia Systems Conference* (Oslo, Norway) *(MMSys '13)*. Association for Computing Machinery, New York, NY, USA, 214–225. https://doi.org/10.1145/2483977.2484004

[30] Hongzi Mao. [n.d.]. Pensieve Open Source Code. https://github.com/hongzimao/pensieve.

[31] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. 2017. Neural Adaptive Video Streaming with Pensieve. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication* (Los Angeles, CA, USA) *(SIGCOMM '17)*. Association for Computing Machinery, New York, NY, USA, 197–210. https://doi.org/10.1145/3098822.3098843

[32] S Matilda, B Palaniappan, and P Thambidurai. 2013. Bufferbloat Mitigation for Real-time Video Streaming using Adaptive Controlled Delay Mechanism. *International Journal of Computer Applications* 63, 20 (2013).

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

[34] Sayed Najmuddin, Muhammad Asim, Kashif Munir, Thar Baker, Zehua Guo, and Rajiv Ranjan. 2020. A BBR-based congestion control for delay-sensitive real-time applications. *Computing* (2020), 1–23.

[35] Vikram Nathan, Vibhaalakshmi Sivaraman, Ravichandra Addanki, Mehrdad Khani, Prateesh Goyal, and Mohammad Alizadeh. 2019. End-to-End Transport for Video QoE Fairness. In *Proceedings of the ACM Special Interest Group on Data Communication* (Beijing, China) *(SIGCOMM '19)*. Association for Computing Machinery, New York, NY, USA, 408–423. https://doi.org/10.1145/3341302.3342077

[36] C. Stephen Gunn Soheil Hassas Yeganeh Van Jacobson Neal Cardwell, Yuchung Cheng. 2017. *Measuring bottleneck bandwidth and round-trip propagation time*. https://cacm.acm.org/magazines/2017/2/212428-bbr-congestion-based-congestion-control/fulltext

[37] Soheil Hassas Yeganeh Ian Swett Victor Vasiliev Priyaranjan Jha Yousuk Seung Matt Mathis Van Jacobson Neal Cardwell, Yuchung Cheng. IETF-104 : iccrg, Mar 2019. *BBR v2: A Model-based Congestion Control*. https://www.ietf.org/proceedings/104/slides/slides-104-iccrg-an-update-on-bbr-00

[38] Soheil Hassas Yeganeh Priyaranjan Jha Yousuk Seung Ian Swett Victor Vasiliev Bin Wu Matt Mathis Van Jacobson Neal Cardwell, Yuchung Cheng. IETF-105 : iccrg, Jul 2019. *BBR v2: A Model-based Congestion Control*. https://www.ietf.org/proceedings/105/slides/slides-105-iccrg-bbr-v2-a-model-based-congestion-control-00

[39] Soheil Hassas Yeganeh Priyaranjan Jha Yousuk Seung Kevin Yang Ian Swett Victor Vasiliev Bin Wu Luke Hsiao Matt Mathis Van Jacobson Neal Cardwell, Yuchung Cheng. IETF-106 : iccrg, Nov 2019. *BBR v2: A Model-based Congestion Control*. https://www.ietf.org/proceedings/106/slides/slides-106-iccrg-update-on-bbrv2-00

[40] Inc. Nginx. [n.d.]. nginx. http://nginx.org.

[41] R. Pan, P. Natarajan, C. Piglione, M. S. Prabhu, V. Subramanian, F. Baker, and B. VerSteeg. 2013. PIE: A lightweight control scheme to address the bufferbloat problem. In *2013 IEEE 14th International Conference on High Performance Switching and Routing (HPSR)*. 148–155.

[42] Vern Paxson and Mark Allman. 2000. RFC2988: computing TCP's retransmission timer.

[43] Ton Roosendaal. 2008. Big Buck Bunny. In *ACM SIGGRAPH ASIA 2008 Computer Animation Festival* (Singapore) *(SIGGRAPH Asia '08)*. Association for Computing Machinery, New York, NY, USA, 62. https://doi.org/10.1145/1504271.1504321

[44] Sandvine. 2019 (accessed October 16, 2020). *Netflix falls to second place in global internet traffic share as other streaming services grow*. https://www.sandvine.com/inthenews/netflix-falls-to-second-place-in-global-internet-traffic-share

[45] Dominik Scholz, Benedikt Jaeger, Lukas Schwaighofer, Daniel Raumer, Fabien Geyer, and Georg Carle. 2018. Towards a Deeper Understanding of TCP BBR Congestion Control. In *IFIP Networking*. 109–117.

[46] Usman Sharif, Adnan N. Qureshi, and Seemal Afza. 2021. ORTIA: An Algorithm to Improve Quality of Experience in HTTP Adaptive Bitrate Streaming Sessions. In *Intelligent Systems and Applications*, Kohei Arai, Supriya Kapoor, and Rahul Bhatia (Eds.). Springer International Publishing, Cham, 29–44.

[47] Matt Southern. 2020 (accessed October 16, 2020). *Facebook Focusing on Live Streaming As Usage Spikes During COVID-19 Lockdowns*. https://www.searchenginejournal.com/facebook-focusing-on-live-streaming-as-usage-spikes-during-covid-19-lockdowns/357884/

[48] Kevin Spiteri, Ramesh Sitaraman, and Daniel Sparacio. 2018. From Theory to Practice: Improving Bitrate Adaptation in the DASH Reference Player. In *Proceedings of the 9th ACM Multimedia Systems Conference* (Amsterdam, Netherlands) *(MMSys '18)*. Association for Computing Machinery, New York, NY, USA, 123–137. https://doi.org/10.1145/3204949.3204953

[49] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman. 2016. BOLA: Near-optimal bitrate adaptation for online videos. In *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*. 1–9.

[50] Bijan Stephen. 2020 (accessed October 16, 2020). *The lockdown live-streaming numbers are out, and they're huge*. https://www.theverge.com/2020/5/13/21257227/coronavirus-streamelements-arsenalgg-twitch-youtube-livestream-numbers

[51] Thomas Stockhammer. 2011. Dynamic Adaptive Streaming over HTTP –: Standards and Design Principles. In *Proceedings of the Second Annual ACM Conference on Multimedia Systems* (San Jose, CA, USA) *(MMSys '11)*. Association for Computing Machinery, New York, NY, USA, 133–144. https://doi.org/10.1145/1943552.1943572

[52] Ranysha Ware, Matthew Mukerjee, Srini Seshan, and Justine Sherry. 2019. Modeling BBR's Interactions with Loss-Based Congestion Control. In *Proceedings of the 2019 ACM SIGCOMM Internet Measurement Conference (IMC)* (Amsterdam, Netherlands) *(IMC '19)*. ACM, New York, NY, USA.

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

[54] Market Watch. 2020 (accessed October 16, 2020). *COVID-19 Impact on Video Streaming Market Primary Research, Secondary Research, Product Research, Trends and Forecast by 2027*. https://www.marketwatch.com/press-release/covid-19-impact-on-video-streaming-market-primary-research-secondary-research-product-research-trends-and-forecast-by-2027-2020-08-03

[55] Greg White and Rong Pan. 2016. A PIE-based AQM for DOCSIS cable modems. *IETF Internet draft* (2016).

[56] Greg White and D Rice. 2014. Active queue management in docsis 3. x cable modems. *Technical report, CableLabs* (2014).

[57] Eiko Yoneki, Harrison Brown, and Kai Fricke. 2020. World-Models for Bitrate Streaming. (2020).

[58] Songyang Zhang and Weimin Lei. 2019. An Optimized BBR for Multipath Real Time Video Streaming. arXiv:1901.09177 [cs.NI]

[59] Anfu Zhou, Huanhuan Zhang, Guangyuan Su, Leilei Wu, Ruoxuan Ma, Zhen Meng, Xinyu Zhang, Xiufeng Xie, Huadong Ma, and Xiaojiang Chen. 2019. Learning to Coordinate Video Codec with Transport Protocol for Mobile Video Telephony. In *The 25th Annual International Conference on Mobile Computing and Networking* (Los Cabos, Mexico) *(MobiCom '19)*. Association for Computing Machinery, New York, NY, USA, Article 29, 16 pages. https://doi.org/10.1145/3300061.3345430