Evaluating TCP BBRv3 Performance in Wired Broadband Networks

Jose Gomez[a], Elie F. Kfoury[a], Jorge Crichigno[a], Gautam Srivastava[b,c,d]

[a] *College of Engineering and Computing, University of South Carolina, Columbia, U.S.A*
[b] *Department of Mathematics and Computer Science, Brandon University, Canada*
[c] *Research Centre for Interneural Computing, China Medical University, Taichung, Taiwan*
[d] *Department of Computer Science and Mathematics, Lebanese American University, Beirut, Lebanon*

Abstract

Introduced by Google in 2016, the first version of the Bottleneck Bandwidth and Round-Trip Time (BBRv1) congestion control algorithm (CCA) marked a significant advancement in network communication. Unlike traditional loss-based CCAs such as CUBIC and Reno, BBRv1 focused on balancing throughput and delay without relying on packet losses as a congestion signal. However, BBRv1 faced fairness issues due to its aggressiveness when interacting with loss-based CCAs. To address this issue, BBRv2 was developed, incorporating multiple metrics to improve fairness with loss-based CCAs. Despite improvements, BBRv2 encountered bugs and performance limitations, prompting the release of BBRv3 in 2023. BBRv3 addressed bugs found in BBRv2 and fine-tuned parameters to enhance flow coexistence.

This paper evaluates the performance of BBRv3 across diverse network scenarios by comparing it with CUBIC and further contrasts this comparison with those involving CUBIC against BBRv2 and BBRv1. The evaluation explores BBRv3's behavior under different conditions, considering variations in the number of flows, propagation delays, loss rates, and buffer sizes. Additionally, this paper explores the influence of Active Queue Management (AQM) algorithms in addressing the RTT unfairness issue. The results indicate that BBRv3's coexistence with CUBIC is comparable to that observed in BBRv2, and it depends on factors such as buffer size and the number of flows. BBRv3 maintains high throughput and lower retransmissions at 1% loss rates compared to its predecessors. Moreover, BBRv3 consistently keeps low queue occupancy and achieves low Flow Completion Times (FCTs) in scenarios with short and long flows, even with large buffer sizes.

Keywords: Bottleneck Bandwidth and Round-trip Time (BBR), BBRv3, Active Queue Management (AQM), Bandwidth-delay Product (BDP), congestion control algorithm (CCA), Controlled Delay (CoDel), CUBIC, Round-trip Time (RTT) unfairness, router's buffer size.

1. Introduction

The Transmission Control Protocol (TCP) [1] has formed the foundation of reliable end-to-end communication, supported by a variety of congestion control algorithms (CCAs). A CCA is a mechanism that governs how TCP will react to congestion to achieve fairness and use the available bandwidth efficiently. Over the last few decades, the TCP CCAs experienced a paradigm shift from early loss-based algorithms, such as Reno [2] and CUBIC [3], to the model-based Bottleneck Bandwidth and Round Trip Time (BBR) [4] algorithm. This transition was driven by the recognition of limitations observed in using loss-based CCAs to transfer data over high-speed links (i.e., 10-100Gbps) that involved paths with large Round-trip Times (RTTs).

Loss-based CCAs faced a series of challenges that reduced their ability to ensure efficient and fair utilization of network resources. One of the issues consisted of excessive buffering and delays. This issue is known as bufferbloat [5] and results in an increasing delay without any performance gain. Loss-based algorithms heavily relied on packet loss to detect network congestion, assuming it to be a clear sign. However, this approach often failed because packet loss alone wasn't always a reliable indicator of congestion. These algorithms were too sensitive to early losses, causing them to react to congestion prematurely.

Additionally, these algorithms struggled when dealing with high-bandwidth paths. Reno and CUBIC, for instance, required a considerable amount of time to increase their throughput on paths with higher bandwidths. For instance, using Reno in a 10Gbps link with a 100ms propagation delay needs more than an hour to fully utilize the bandwidth. Avoiding this issue demands a loss rate lower than 0.00000002% (2 out of 10 billion packets), which is impractical. In similar conditions, CUBIC needs more than 40 seconds between any losses to fully utilize the 10Gbps bandwidth considering a loss rate of less than 0.0000029% (i.e., 2.9 out of 100 million packets), which is also impractical.

BBRv1 [4] presented a solution to these problems by taking a different approach. Instead of relying on packet losses to indicate congestion, it estimated the bottleneck bandwidth to determine the sending rate. This strategy aimed to reduce delays in queues and prevent bufferbloat while maximizing the amount of data being sent. To establish the sending rate, BBRv1 uses pacing, which involves sending packets at measured intervals from the sender. This spread-out timing strategy helped manage data flow. This approach differs from the traditional loss-based algorithms, where the sending rate is conditioned

Email addresses: gomezgaj@email.sc.edu (Jose Gomez), ekfoury@email.sc.edu (Elie F. Kfoury), jcrichigno@cec.sc.edu (Jorge Crichigno), srivastavag@brandonu.ca (Gautam Srivastava)

by the size of the congestion window, and the sender node may send packets in bursts.

However, the literature reported that BBRv1 presented poor coexistence with loss-based CCAs leading to unfairness issues [6–11]. Therefore, BBRv2 [12] was proposed to overcome these limitations. BBRv2 considers packet losses, and ECN signals in addition to the bandwidth and RTT already considered in BBRv1 to build a network model. As a result, evaluations showed that BBRv2 presented a better coexistence with loss-based CCAs and reduced the RTT unfairness problem observed in BBRv1. Nevertheless, BBRv2 presented convergence issues when sharing a link with competing BBRv2 flows. This issue did not allow BBRv2 flows to converge to a fair link share after a certain period. The convergence issue was observed in networks with and without packet losses.

In 2023, the release of BBRv3 [13] aimed to fix bugs and optimize performance parameters to resolve the convergence issues observed in BBRv2. BBRv3's goal is to attain fair bandwidth sharing among competing flows and enhance its coexistence with loss-based CCAs. This paper presents a performance evaluation of BBRv3 across various network conditions. These conditions encompass examining BBRv3's performance under different router buffer sizes, loss rates, and propagation delays. This paper also assesses BBRv3's performance in a context replicating RTT unfairness conditions. Additionally, the accuracy of BBRv3's bottleneck bandwidth estimation as a function of RTT is evaluated through both emulated topology and real hardware settings.

To the best of the authors' knowledge, the literature is still missing a performance evaluation of BBRv3. Therefore, this paper presents an experimental evaluation of BBRv3 considering routers with different buffer sizes, Active Queue Management (AQM) algorithms, links with different propagation delays and packet loss rates, and routers with changing bottleneck bandwidth and delays. The contributions of this paper can be listed as follows:

1. *Coexistence and loss resilience*: The coexistence of BBRv3 with CUBIC is influenced by buffer size and the number of flows, showing fairness issues initially but improving with more BBRv3 flows. The optimal coexistence is observed with buffer sizes between BDP and 10BDP. Furthermore, BBRv3 demonstrates remarkable resilience to packet losses, maintaining high throughput and lower retransmissions even at loss rates of around 1%, which is a key factor contributing to its unfairness to loss-based CCAs such as CUBIC.

2. *Queue occupancy*: BBRv3 maintains low queue occupancy, while CUBIC leads to queue saturation. This difference contributes to lower end-to-end latency for BBRv3.

3. *Bottleneck bandwidth estimation*: BBRv3 effectively estimates bottleneck bandwidth across varying propagation delays, maintaining stability in emulated bottleneck scenarios. This suggests its robustness across network conditions.

4. *Flow completion time*: In scenarios where short flows compete with a long flow, BBRv3 maintains consistently low Flow Completion Times (FCTs) even with larger buffer sizes, outperforming CUBIC.

Furthermore, this paper provides access to the virtual machine (VM) and scripts employed for executing the experiments within Mininet. The VM and scripts are available at [14]. The rest of the paper is organized as follows: Section 2 provides background on BBRv1, BBRv2, and BBRv3. Section 3 summarizes the related work. Section 4 describes the experimental setup. Section 5 presents the results and evaluations and section 6 concludes the paper.

2. Background

This section describes the principles of BBRv1, its shortcomings and the evolution of the CCA. Then, it delves into the origins of BBRv2, discussing its limitations, which paved the way for the development of BBRv3. This section also describes the lifecycle of BBRv3.

2.1. Principles of BBRv1

BBRv1 was introduced to overcome a design limitation in loss-based CCAs, which considered packet loss as a sign of congestion [15]. As Network Interface Cards (NICs) progressed from megabits per second (Mbps) to gigabits per second (Gbps) and memory chips advanced from kilobytes (KB) to gigabytes (GB), the correlation between packet loss and congestion diminished. Currently, in scenarios with large bottleneck buffers, loss-based CCAs maintain the bottleneck buffers at full capacity, resulting in bufferbloat [5]. Conversely, in situations where bottleneck buffers are small, loss-based congestion control interprets loss as a signal of congestion, leading to reduced throughput. BBRv1 aimed to overcome these issues proposing a new approach to congestion control that aligns with the evolved network landscape.

BBRv1 aimed to optimize data transmission over networks by estimating the bottleneck bandwidth and the round-trip propagation (RTprop). BBRv1's main goal is to maximize link utilization by dynamically adjusting the sending rate based on its estimation of the available bottleneck bandwidth. This real-time adaptation allows BBRv1 to respond effectively to varying network conditions, ensuring optimal data transfer performance. A distinctive feature of BBRv1 lies in its approach to congestion control, specifically in its avoidance of bufferbloat. Unlike traditional loss-based algorithms that rely on detecting packet losses as a signal of congestion, BBRv1 prioritizes preventing the accumulation of excessive queues in network buffers. This emphasis on buffer management contributes to more responsive and low-latency data transfers. BBRv1 also incorporates an aggressive startup mechanism, allowing it to quickly probe the available bandwidth and converge to the optimal sending rate. This aggressive approach facilitates rapid data transfer during the initial stages of a connection and allows it to dynamically respond to changes in network conditions.

2.2. Shortcomings of BBRv1 and foundations of BBRv2

While BBRv1 has improved the throughput of a TCP connection, the literature has reported some behavioral

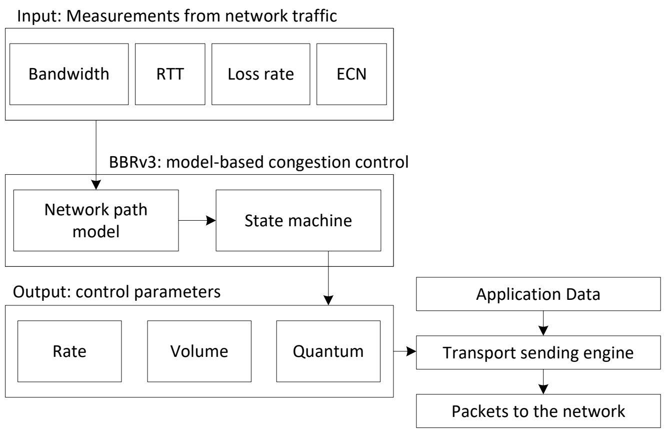


Figure 1: High-level architecture of BBRv3.

issues, such as unfairness with loss-based CCAs and high retransmission rates [16–18]. BBRv2 adopts a network traffic modeling approach, estimating bandwidth, monitoring RTT, measuring loss rates, and incorporating Explicit Congestion Notification (ECN) capabilities. Studies indicate that BBRv2 tolerates higher packet loss rates compared to loss-based CCAs, experiences lower retransmission rates than BBRv1, demonstrates improved coexistence with CUBIC, mitigates the impact of RTT unfairness, and exhibits reduced queueing delays [6–8, 11]. However, evaluations of BBRv2 revealed issues that caused interruptions in bandwidth probing following a packet loss event or ECN signal. Additionally, convergence problems were observed for buffer sizes exceeding 1.5 BDP. These challenges led to the development of BBRv3, a minor update to BBRv2 designed to address the identified bugs and fine-tune performance parameters.

2.3. Principles of BBRv3

BBRv3 presents similar behavior to the one of BBRv2. BBRv3 is designed to replace the previous versions of BBR (i.e., BBRv1 and BBRv2) [13]. BBRv3 represents an evolution of CCAs and operates on a model-based, rate-driven approach. The high-level architectural framework is illustrated in Figure 1. The essential metrics under consideration include bandwidth, RTT, packet loss ratio, and ECN marking frequency. These metrics collectively inform the computation of the Bandwidth-delay Product (BDP), facilitating the characterization of the network's end-to-end path, denoted as the network path model. Through an assessment of the current network path model, the algorithm navigates through various states within a finite-state machine. This dynamic process encompasses actions such as bandwidth and RTT probing. The resulting finite state machine generates three key control parameters: rate, volume, and quantum. These control parameters play an essential role in regulating the sending rate. The rate parameter defines the pacing rate for the sender's transmissions. The volume parameter dictates the allowed volume of data or bits within the path as they traverse from the sender to the receiver. This concept aligns with the inflight volume. Meanwhile, the quantum parameter establishes the maximum burst size achievable from the sender. The transport protocol sending engine evaluates these variables to regulate the amount of data that is injected into the network. Additionally, the sending engine segments application data into bursts of size determined by the quantum before injecting the data into the network as discrete packets.

BBRv3 implements an estimation strategy involving both short-term and long-term approximations of the bottleneck bandwidth and maximum inflight data volume. This approach is similar to the principles observed in CUBIC, where short-term slow start threshold estimates (`ssthresh`) and long-term maximum congestion window (`W_max`) play key roles. During the major part of the connection duration, BBRv3 operates within a period in which it swiftly achieves equilibrium in flow management. This involves adjusting the sending rate to match the updated bottleneck bandwidth or BDP. Simultaneously, it aims to retain spare capacity within the bottleneck link. This approach allows incoming flows to effectively share the bandwidth. To ensure regulated behavior, BBRv3 upholds short-term `bw_lo` and `inflight_lo` approximations, which establish a limit based on the latest delivery processes (e.g., loss, ECN). This strategy aims at keeping bounded queueing levels at the bottleneck link by considering previous delivery processes.

2.4. Lifecycle of a BBRv3 Flow

The state machine of BBRv3 closely resembles that of BBRv2, as depicted in Figure 2. It operates through cycles, alternating between bandwidth and RTT probing. Initially, a flow begins in the STARTUP phase (a) similar to the traditional slow start. During this phase, the algorithm tries to quickly identify the bottleneck bandwidth by doubling the sending rate of each RTT. If the packet loss rate or ECN mark rate surpasses their respective thresholds, the `inflight_hi` is established as an approximation of the maximum inflight capacity.

Upon reaching a stable value (plateau) during continuous bandwidth probes or setting `inflight_hi`, the flow departs the `STARTUP` phase and enters the `DRAIN` phase (b). Here, the aim is to deplete excess inflight bits and any queue formed at the bottleneck link in the previous phase. This is achieved by adopting a low pacing (sending) rate. The `DRAIN` phase concludes once the inflight volume falls to or below the estimated BDP.

The majority of the flow's lifespan is spent in the `CRUISE` phase (c), where the sending rate dynamically adapts to regulate queue levels. The {`bw, inflight`}`_lo` tuple is updated every RTT, utilizing information from packet loss and ECN signals. Subsequently, the `PROBE_BW:REFILL` phase (d) is initiated to probe for additional bandwidth and inflight capacity. The objective is to elevate the inflight volume (reduced earlier) by sending at the estimated capacity (bottleneck bandwidth) for one RTT. It's important to note that the anticipation here is that queues won't form, given that the sending rate doesn't exceed the estimated bottleneck bandwidth. However, in cases where routers have limited buffers, non-congestion-related packet losses might occur. To prevent reducing the sending rate and negatively impacting throughput, the algorithm tolerates up to `loss_thresh` losses each RTT.

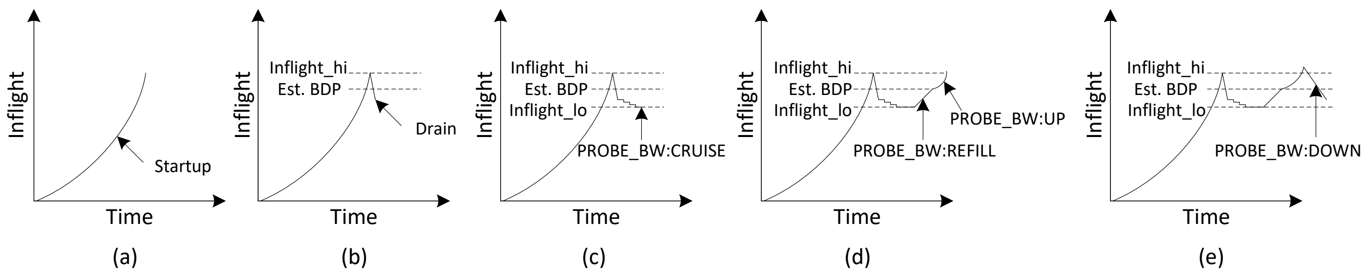


Figure 2: Life cycle of a BBRv3 flow and its five phases: (a) STARTUP, (b) DRAIN, (c) PROBE_BW:CRUISE, (d) PROBE_BW:REFILL and PROBE_BW:UP, and (e) PROBE_BW:DOWN.

During the bandwidth probing PROBE_BW:UP phase, if inflight_hi is fully utilized, it is incrementally increased exponentially each round (1, 2, 4, 8, ..., 2^n packets). Conversely, if the loss rate or ECN mark rate becomes excessive, the inflight_hi is reduced to the estimated maximum safe inflight volume. The exit criteria for the PROBE_BW:UP phase include setting inflight_hi or if the estimated queue surpasses a certain threshold (inflight volume $>1.25\times$estimated BDP).

Lastly, the flow transitions into the PROBE_BW:DOWN phase (e) to deplete the recently formed queue and make use of the available headroom. This phase concludes when two conditions are met: 1) the inflight volume is slightly below a headroom margin from inflight_hi, and 2) the inflight volume is at or below the estimated BDP.

2.5. Improvements of BBRv3 over BBRv1, and BBRv2

The major updates of BBRv3 consist of 1) fixing bugs that reduced bandwidth convergence and 2) tuning performance parameters that involved the pacing gain and the congestion window gain.

2.5.1. Bugs observed in BBRv2 and fixed in BBRv3

BBRv3 fixes bugs and mitigates performance issues presented by its predecessors, particularly addressing a critical bug that affected BBRv2. The first bug caused the interruption of bandwidth probing shortly after loss or Explicit Congestion Notification (ECN) was set in the inflight_hi parameter. This problem was the result of a circular dependency between maximum bandwidth and maximum inflight data. The consequences of this issue lead to BBRv2 flows struggling to achieve a fair share when competing against CUBIC or Reno flows. Moreover, these affected flows experienced prolonged periods to achieve full utilization after congestion events. BBRv3 addresses this issue through persistent bandwidth probing. It continues probing until either the loss rate or ECN mark rate surpasses predefined tolerance thresholds of 1% [19]. The bandwidth probe also stops when the inflight_hi parameter has not recently limited the amount of data being sent, and the available bandwidth reaches its maximum capacity.

Another bug observed in BBRv2 occurred when the buffer size was more than 1.5 times the BDP. In this scenario, BBRv2 flows had trouble reaching a fair share. Two main reasons caused this: First, a fixed congestion window gain prevented slower flows from increasing their sending rates effectively. Second, slower flows using 0.75 times the estimated bandwidth allocated too much of the bandwidth to faster flows. This led to an imbalance in resource sharing, especially with larger buffer sizes and no explicit loss or congestion signals. BBRv3 mitigated these problems by making two key changes. First, during bandwidth probing (PROBE_BW:UP), the congestion window (cwnd) gain was increased from 2.0 to 2.25, allowing better adjustments in sending rates. Second, the pacing gain was adjusted from 0.75 times to 0.9 times (PROBE_BW:DOWN). This adjustment helps slower flows consistently use sufficient bandwidth, facilitating faster flows to converge to a fair share faster.

2.5.2. Performance tuning implemented in BBRv3

BBRv3 also implemented changes in performance parameters. These changes were made to the STARTUP phase, where the congestion window gain has been recalibrated from 2.89 to 2.0, a change grounded in analytical derivations [20]. Similarly, the STARTUP pacing gain has undergone a shift from 2.89 to 2.77, again guided by analytic reasoning [21]. When leaving the STARTUP phase, BBRv3 now sets the inflight_hi parameter based on the maximum value between the estimated BDP and the highest number of packets successfully transmitted during the last RTT. The criteria for transitioning out of the STARTUP phase due to packet loss have been also revised. Currently, BBRv3 requires a smaller number of loss events within a single round trip (6 instead of 8). These changes result in reduced queuing delays and lower rates of packet loss during the STARTUP phase and in the following stages.

3. Related Work

As far as the authors are aware, there has been a gap in the existing literature regarding the assessment of BBRv3. However, prior researchers have conducted evaluations of BBRv2 and BBRv1.

Hock *et al.* [18] conducted a comprehensive evaluation of the BBRv1, focusing on bottleneck links of 10 Gbps and 1 Gbps, multiple flows, and varying RTTs. Their findings reveal that while BBRv1 works effectively for a single flow at a bottleneck, its performance with multiple flows diverges from its original objectives. BBRv1's mechanisms unintentionally lead to sustained overload at the bottleneck, resulting in increasing inflight data and queuing at the bottleneck buffer. The authors noted that the absence of a mechanism to drain this queue leads to issues such as increased queuing delays, RTT unfairness in large buffers, and significant packet losses and unfairness in smaller buffers. The authors observed that

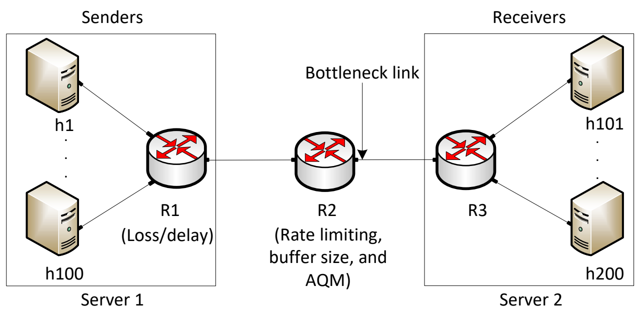


Figure 3: Topology used for the evaluations.

BBRv1 lacks explicit congestion detection and reaction mechanisms, leading to challenges in handling congestion and fairness among multiple flows. The paper emphasizes that BBRv1's approach and insights into its mechanisms are valuable for congestion control research. Additionally, the authors suggest further investigation of BBRv1's behavior with AQM mechanisms.

Hurtig *et al.* [16] evaluated the interaction of CUBIC and BBR. The authors provide a comprehensive understanding of BBRv1's effects, considering the heterogeneity of Internet traffic sizes. Unlike previous evaluations that primarily focused on steady-state performance with large flows, this work explores BBRv1's behavior during the startup phase and its impact on different-sized flows. The experiments reveal that, under specific conditions, BBRv1's startup phase can significantly reduce throughput for competing large CUBIC flows. Additionally, the steady-state operation of BBRv1 may negatively affect the performance of bursty flows using loss-based CCAs over bottlenecks with larger buffer sizes. The experiments demonstrate that an all-BBRv1, mixed-size traffic consistently performs well across various bandwidths and buffer sizes. However, in scenarios where a small buffer is shared by a large CUBIC flow and successive BBRv1 bursts, the aggressive BBRv1 startup mechanism leads to a significant reduction in throughput for the large flow and bottleneck utilization.

Nandagiri *et al.* [17] conducted an experimental evaluation to compare the performance differences between BBRv1 and BBRv2. Their work focuses on assessing how these versions perform in terms of throughput, delay, and fairness under varying network conditions. The experiments highlight the potential issues when BBRv2 coexists with CUBIC in routers with deep buffers. Furthermore, the performance of BBRv2 with ECN is compared to CUBIC with an AQM and ECN, indicating similar fairness and performance in terms of throughput, latency, and link utilization. The study emphasizes the need for careful consideration of BBRv2's adoption of Data Center TCP (DCTCP) style ECN, particularly in WAN deployments.

Kfoury *et al.* [6] used Mininet for assessing the performance of BBRv2. The study tested the alpha version of BBRv2 through experiments involving both small and large buffer sizes. The researchers replicated scenarios of RTT unfairness, examined BBRv2's interaction with other TCP flows, and analyzed the influence of AQM algorithms on the performance of both BBRv1 and BBRv2.

Gomez *et al.* [7] evaluated BBRv2 through Mininet emulation. The study highlighted improved coexistence of BBRv2 and CUBIC compared to BBRv1 and CUBIC. The research illustrated BBRv2's effectiveness in addressing RTT unfairness and achieving a better bandwidth distribution over BBRv1, particularly in dynamic network conditions.

Song *et al.* [8] conducted an evaluation and comparison of BBRv1 and BBRv2 using Mininet and a physical testbed. The study illustrated BBRv2's ability to mitigate the fairness concerns seen in BBRv1. The evaluations indicated that BBRv2 exhibited improved fairness when competing with other flows, particularly in scenarios involving routers with limited buffer capacities. The research underscored the synchronization and coexistence challenges faced by BBRv1 flows when interacting with loss-based CCAs.

Tierney *et al.* [9] evaluated the applicability of BBRv2 on Data Transfer Nodes (DTNs). The researchers conducted tests with BBRv2 in both a live network setting and a controlled experimental setup. Their assessments underscored the enhanced performance of BBRv2, particularly during large data transfers. Additionally, the study emphasized the potential of BBRv2 as a viable choice within high-speed, short-queue network environments. Furthermore, the authors confirmed that the outcomes observed with Mininet hold validity when applied to real-world networks.

Scherrer *et al.* [10] introduced a fluid model of both BBRv1 and BBRv2. Through simulations using ns-3 across diverse network configurations, the researchers presented analytical assessments, including stability analysis. Their findings indicated the model's capacity for accurately forecasting the behaviors of BBRv2. The study additionally validated BBRv2's effectiveness in mitigating the unfair behavior of BBRv1 while identifying the specific scenarios where BBRv2 could lead to issues like bufferbloat and unfairness.

Gomez *et al.* [11] evaluated the performance of BBRv2 using FABRIC [22], a large-scale testbed used to test the next generation of Internet applications. The authors used FABRIC to reproduce Wide Area Network (WAN) conditions. BBRv2 is compared with various CCAs in terms of throughput relative to the RTT, queue occupancy, RTT fairness, and packet loss rate as a function of the router's buffer size. The evaluation also investigates the impact of AQM algorithms in mitigating performance issues arising from RTT unfairness and the interactions between different CCAs when sharing a bottleneck link.

4. Experimental Setup

Figure 3 shows the topology used to conduct the experiments. At any time, a TCP connection has one slowest link or bottleneck link that determines the location of queue formation. The choice of topology is based on the necessity of creating a bottleneck link. This condition is crucial to evaluate CCAs because a bottleneck determines the maximum data delivery rate of the connection and serves as the location where persistent queues are established. In future evaluations of BBRv3 the authors plan to incorporate topologies with multiple routes, wireless paths, and asymmetric link conditions as suggested in [23]. The topology consists of 100 senders (h1, h2, ..., h100), each opening a TCP connection to a corresponding receiver (h101, h102, ..., h200). This topology is implemented using Mininet [24] and real hardware. The Operating System (OS) used for the experiment is Ubuntu 22.04 running a custom kernel compiled to implement BBRv3 [25]. The scripts used for the experiments and the link to the VM are available in the following GitHub repository [14].

Mininet is a network emulator that uses network namespaces to isolate computing resources. Mininet hosts use the real protocol stack in Linux and produce real traffic. These hosts are interconnected using Open Virtual Switches (OVS) [26], which are open-source network switches designed to operate within virtualized environments such as data centers, cloud infrastructure, and network virtualization platforms. OVS functions as a software-based switch that facilitates network communication between VMs and physical machines (servers), allowing them to efficiently exchange data across different network segments.

The topology implemented with Mininet is hosted on a server allocated with enough computing resources so that the CPU and Memory are not limiting factors in the tests. Specifically, the VM is allocated 16 vCPUs operating at 3.00GHz and 32GB of memory. A similar topology is implemented using a hardware switch, which is the Juniper MX-204 [27]. The experiments involving the hardware router replaced the role of the OVS switch, which was acting as router R2, with a physical Juniper MX-204 hardware router. This router also allows limiting the rate and modifying the buffer size.

Loss/delay emulation:. Router R1 is used to emulate delay and inject packet losses using NetEm [28]. All tests are configured with a 20ms propagation delay, unless otherwise specified.

Rate limiting, buffer size and AQM:. In the emulated topology, the egress interface of router R2 limits the rate using Token Bucket Filter (TBF) and implements AQM policies [29] such as FQ-CoDel [30]. The default AQM policy used in routers is Tail Drop (TD) unless another policy is explicitly stated. Similarly, in the hardware switch, the rate is limited to 1Gbps by the Physical Interface Controller (PIC). Additionally, a Class of Service (CoS) [31] scheduler is used to set the buffer size.

Metrics Collection:. The tool used to generate traffic between senders and receivers is iPerf3 [32]. Performance metrics and variables include throughput, retransmission rate, size of congestion window, RTT, and others. The queue occupancy on the interface connecting to the bottleneck link is measured using Linux's traffic control (`tc`). The estimated bottleneck bandwidth in BBRv3 and other internal variables are measured using Linux's `ss` command [33]. The retransmission rate is calculated using the relationship between the number of sent packets and the number of retransmitted packets present in the iPerf3 report. The Jain's fairness index is computed to measure fairness, as described in RFC 5166 [34]:

$$\mathcal{F} = \frac{\left(\sum_{i=1}^{n} T_i\right)^2}{n \cdot \sum_{i=1}^{n} (T_i)^2} \tag{1}$$

where T_i is the throughput of flow i. For example, in a scenario with 100 simultaneous flows, $n = 100$ and $i = 1, 2, ..., 100$. The results report the fairness index in percentage, which is given by multiplying Eq. 1 by 100. In the experimental evaluations, each test was executed for 120 seconds, unless otherwise specified. The experiments were repeated 10 times and the results averaged. The size of the TCP send and receive buffers (`net.ipv4.tcp_wmem` and `net.ipv4.tcp_rmem`) on the end hosts (senders and receivers) was set to 10 times the BDP.

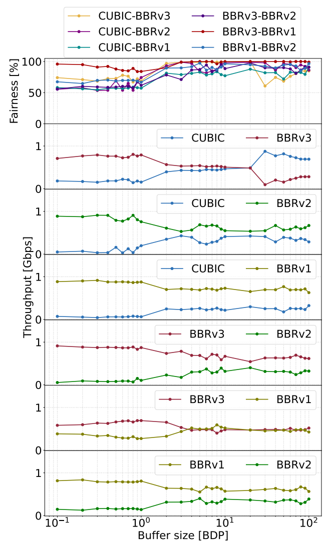
5. Results and Evaluations

This section presents the results and evaluations of BBRv3 obtained by running tests in different network conditions. Every experiment was repeated 10 times and results were averaged for more accuracy.

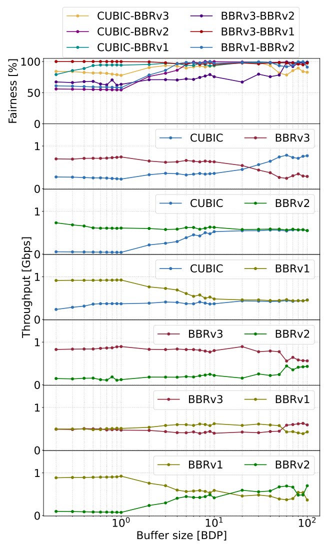
5.1. Inter-protocol Fairness

This experiment evaluates how effectively BBRv3, BBRv2, BBRv1, and CUBIC coexist. The evaluation involves observing the throughput as a function of the buffer size. Four distinct setups are examined: 1) Two competing flows without losses, 2) in a scenario without packet losses, 100 competing flows divided in a 50:50 ratio, 3) a scenario with 0.025% packet losses involving two competing flows, and 4) in a scenario with 0.025% packet losses, 100 competing flows divided in a 50:50 ratio. The aggregated throughput of these flows is analyzed both with and without packet losses. The bottleneck link's bandwidth for these experiments is set at 1Gbps, and the RTT is 20ms. The buffer sizes span from 0.1 times the BDP to 100 times the BDP. Specifically, the range starts from 0.1BDP and increments in steps of 0.1BDP up to BDP. Then, from BDP to 10BDP, the increment is equal to BDP. Finally, for buffer sizes ranging from 10BDP to 100BDP, the increment is 10BDP.

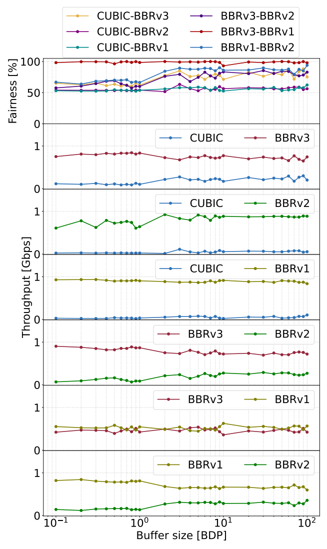
Figure 4(a) shows in the upper graph the fairness index, while the lower graphs illustrate the throughput of CUBIC against BBRv3, BBRv2, and BBRv1 flows. Additionally, the throughput between BBRv3, BBRv2, and BBRv1 is analyzed. The horizontal axis of the figures is shown in logarithmic scales to enhance readability. For buffer sizes less than BDP, BBRv3 achieves higher throughput, leading to lower fairness with CUBIC flows. In this range, BBRv3 affects CUBIC's throughput due to the large number of packet retransmissions, which CUBIC identifies as congestion. This behavior is also observed in BBRv2 and BBRv1 where the fairness index is around 60%. BBRv3 presents an improvement in the fairness in this range. This behavior was also observed in previous evaluations of BBRv2 and BBRv1 [6, 7, 11]. As buffer sizes grow from BDP to 10BDP, both CUBIC and BBRv3 exhibit improved convergence towards better fairness. Similarly, BBRv2 and CUBIC present a fairness index between 80% and 95%. On the other hand, BBRv1 exhibits lower fairness, as it acquires a larger share of



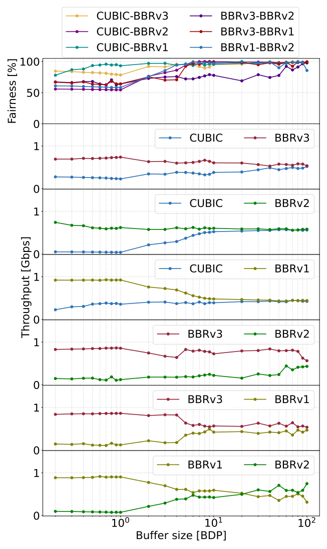
(a) 2 flows with no loss.



(b) 100 flows with no loss.



(c) 2 flows with 0.025% loss.



(d) 100 flows with 0.025% loss.

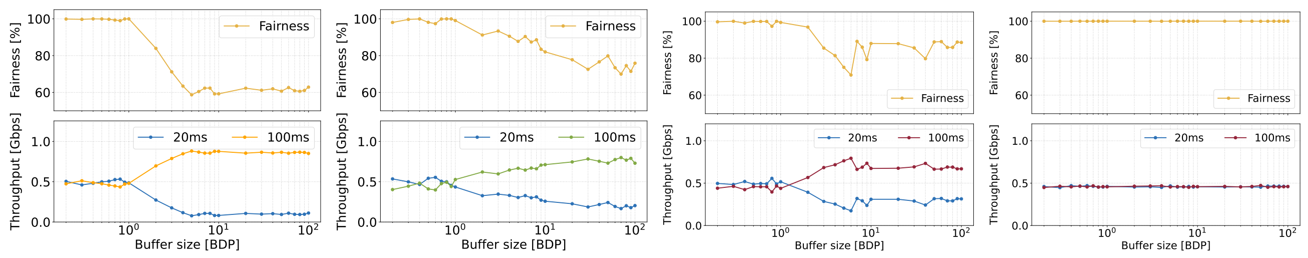
Figure 4: Fairness index and throughput with respect to the buffer size in four different scenarios: (a) Two competing flows without packet losses. (b) One hundred flows equally divided in a 50:50 ratio without packet losses. (c) Two flows with 0.025% packet losses. (d) One hundred flows equally divided in a 50:50 ratio with 0.025% packet losses.

the bandwidth compared to the CUBIC flow. This performance is attributed to the fact that BBRv1 does not reduce the amount of inflight data in response to packet losses caused by small buffers. In this range, BBRv3 achieves fair bandwidth distribution by accurately estimating the inflight data to the BDP without inducing excessive packet losses. This approach ensures sufficient time for the CUBIC flow to recover from losses.

Larger buffer sizes (>10 BDP) lead to less fairness, favoring CUBIC flows. This happens because CUBIC tends to fill up the bottleneck buffer completely, while BBRv3 keeps its inflight data around the BDP. A larger buffer tends to give more bandwidth share to CUBIC. However, because CUBIC creates persistent queues, the competing BBRv3 flow cannot estimate the bottleneck bandwidth accurately. This behavior is not observed in BBRv1 and BBRv2. In a scenario where a BBRv3 flow competes with a BBRv2 flow, it is noticed that the BBRv3 flow consistently obtains a larger portion of the available bandwidth across multiple buffer sizes. This behavior is due to the increased congestion window gain in BBRv3, which allows for occupying additional headroom relative to BBRv2. Conversely, when a bottleneck link is shared between a BBRv1 and a BBRv3 flow, better fairness is achieved. This pattern contrasts with the scenario of a BBRv1 flow competing against a BBRv2 flow, where fairness tends to be lower.

In a scenario with multiple flows, as depicted in Figure 4(b), the fairness index corresponding to BBRv3 and CUBIC flows remains steady at approximately 80-90% from 0.1BDP to 10BDP. Buffer sizes ranging from 20BDP to 30BDP exhibit convergence to fairness. However, when buffer sizes surpass 30BDP, CUBIC gains a larger share of the bandwidth, lowering the fairness index to approximately 80%. On the other hand, for buffer sizes larger than 10BDP, CUBIC flows competing against BBRv1 and BBRv2 converge to fairness. When BBRv3 flows interact with BBRv2 flows, the major share favors BBRv3 flows similar to the scenario with two competing flows. It is noted that BBRv3 and BBRv1 present better fairness for a wide range of buffer sizes. Conversely, BBRv1 and BBRv2 show a lower fairness index for buffer sizes below 2BDP, whereas for large buffer sizes, the performance is similar to that of BBRv3 competing with BBRv1. When examining a scenario involving a single CUBIC flow and a single BBRv3 flow with a 0.025% packet loss rate (see Figure 4(c)), the fairness index is observed to settle around 60%, favoring the BBRv3 flow. Buffer sizes from BDP to 10BDP show an improving fairness index, peaking around 80% between 2BDP and 10BDP, favoring BBRv3. No substantial improvement is observed for buffer sizes ranging from 10 times the BDP to 100 times the BDP, as BBRv3 consistently retains a significant portion of the available bandwidth. On the other hand, BBRv2 and BBRv1 present a similar fairness index, which settles around 50%. This discrepancy arises due to the fact that CUBIC's throughput is inversely proportional to the square root of the RTT. The situation is further exacerbated by the presence of small buffers and the introduction of packet loss. When BBRv3 and BBRv1 share a bottleneck link with packet losses, the BBRv3 flow obtains a higher share of the available bandwidth. On the other hand, it is observed that BBRv3 and BBRv1 present better fairness for a wide range of buffer sizes even with packet losses. As reported in [6], BBRv1 and BBRv2 exhibit a low fairness index, with preference given to the BBRv1 flow.

Figure 4(d) shows a scenario with 100 flows. It is observed that the fairness index remains at around 80% for buffer sizes from 0.1BDP to BDP. Buffer sizes larger than BDP but less than 10BDP maintain a fairness index of about 90%. BBRv2 and BBRv1 present similar fairness for BDP values greater than 4 BDP. In this scenario, the impact of packet loss on throughput is shared among the different flows. Each flow can experience losses independently, and the combined effect is decreased by the parallel nature of these flows. Furthermore, the adaptive nature of the TCP flows helps to utilize the available



(a) BBRv1 with Tail Drop. (b) BBRv2 with Tail Drop. (c) BBRv3 with Tail Drop. (d) BBRv3 with FQ-CoDel.

Figure 5: Fairness index and throughput as a function of the buffer size for a pair of competing flows: one with a 20ms propagation delay and the other with a 100ms propagation delay.

bandwidth due to the decrease in throughput experienced by one of the flows. For buffer sizes greater than 10BDP, the aggregate throughput converges close to fairness. As in previous scenarios, it has been noted that when packet losses occur, BBRv3 flows tend to capture a larger share of the bandwidth compared to BBRv2 flows. Similarly, when comparing BBRv3 with BBRv1, it is observed a similar trend for buffer sizes below 3BDP. However, for larger buffer sizes, both BBRv3 and BBRv1 converge towards improved fairness, resembling the performance observed when BBRv1 and BBRv2 interact.

5.2. *RTT Unfairness*

BBRv3 considers both the RTT and the estimated bottleneck bandwidth to determine the optimal amount of data to introduce into the network. This quantity of data approaches the BDP, which reflects Kleinrock's optimal operating point [35]. This strategy aims to minimize delays and maximize throughput. Due to the relationship between the RTT and the bottleneck bandwidth (i.e., inflight_data = RTT×Btlbw), the higher the RTT, the higher the amount of data that BBRv3 injects into the network. Therefore, BBRv3 flows with higher RTTs tend to share a bottleneck bandwidth with BBRv3 unfairly compared to flows with lower RTTs. This problem is known as the RTT unfairness. This issue contradicts the typical behavior of traditional TCP CCAs, which generally favor flows with shorter RTTs, resulting in two significant consequences [36]. First, it introduces a tradeoff between achieving low latency and ensuring a high delivery rate, challenging decades of engineering efforts to reduce end-to-end latency. For instance, prioritizing a route with the minimum RTT through Open Shortest Path First (OSPF) may not be optimal, as flows on that route can be easily overwhelmed when competing with others on a suboptimal route with higher latency. Second, the preferential treatment of long RTT flows allows a malicious receiver to exploit this advantage, potentially stealing bandwidth from competing flows by artificially increasing the latency of a particular inbound traffic.

The following tests assess how well BBRv1, BBRv2, and BBRv3 perform in scenarios involving RTT unfairness. Specifically, the experiments focus on evaluating the performance of two competing flows relative to the buffer size. These flows experience RTTs of 20ms and 100ms, respectively. Additionally, the evaluations extend to a situation where a router implements the FQ-CoDel AQM [30]. Two scenarios are evaluated 1) two competing BBRv1, BBRv2, and BBRv3 flows, where the router creating the bottleneck link (referred to as router R2) uses a simple TD policy, and 2) two competing BBRv3 flows, but this time, the router (router R2) implements the FQ-CoDel mechanism to shape the bottleneck link.

In Figure 5(a), it is observed the results corresponding to BBRv1 flows. When the buffer sizes exceed BDP, the flow with a higher RTT cannot accurately estimate the bottleneck, leading to performance degradation for the flow with a lower RTT. Consequently, the fairness index drops to approximately 60%. On the other hand, figure 5(b) displays the results for BBRv2. Here, the fairness index improves compared to BBRv1. For buffer sizes below BDP, the fairness index remains similar to BBRv1. However, for buffer sizes greater than BDP, the fairness index does not fall below 80%.

Figure 5(c) represents the throughput of these competing BBRv3 flows. In this context, the router's queueing approach is based on the TD policy. The observation reveals that when the buffer sizes are below the BDP, the flows tend to achieve a state of fairness. However, when buffer sizes exceed BDP, the fairness index tends to settle around 80%, where the flow with the higher RTT receives the larger share of the available bandwidth. The RTT unfairness issue in BBRv3, similar to its predecessors, originates in two distinct phases. First, an excess queue forms and rapidly expands on the bottleneck when BBRv3 flows increase inflight to probe for additional bandwidth. Second, a long RTT flow introduces a larger volume of excess traffic (i.e., inflight_data = RTT×Btlbw) compared to a short RTT flow, dominating both the queue backlog and the delivery rate. Consequently, the short RTT flow observes a lower delivery rate, adjusts its speed to match the measurement, and becomes a disadvantaged participant against flows with higher RTTs. Additionally, the short RTT flow is prone to be limited by the congestion window, restricting its capacity for probing for more bandwidth. BBRv3 presents an improvement in the fairness for buffer sizes greater than BDP compared to BBRv2.

Figure 5(d) shows that the RTT unfairness issue is mitigated by implementing the FQ-CoDel AQM in the router. FQ-CoDel employs a fair queuing mechanism that separates TCP flows into different queues, preventing the dominance of a single flow over bottleneck bandwidth and ensuring a fair distribution of bandwidth shares. Moreover, FQ-CoDel incorporates the Controlled Delay (CoDel) mechanism in each queue, focusing on managing the delay experienced by packets rather than controlling the queue length. This dual mechanism highlights the efficacy of FQ-CoDel in addressing the

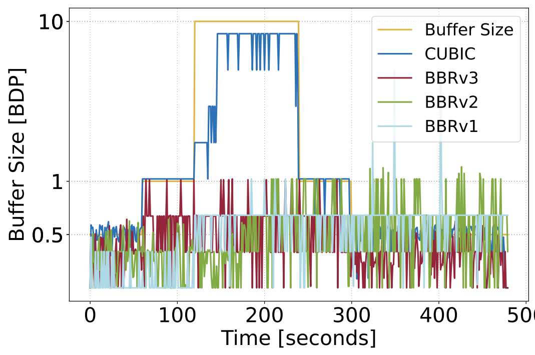


Figure 6: Queue occupancy for different buffer sizes for CUBIC BBRv1, BBRv2, and BBRv3.

challenge of RTT unfairness. In previous evaluations of BBRv1 and BBRv2 [6, 7, 11], the FQ-CoDel AQM effectively mitigated the RTT unfairness issue observed in competing flows.

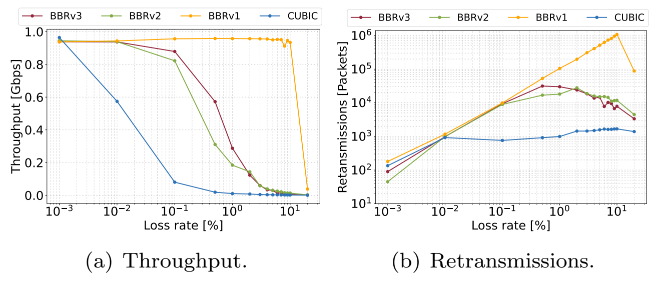
5.3. Queue Occupancy

This experiment examines the queue occupancy of CUBIC and BBRv3. The queue occupancy is obtained by parsing the number of bytes from the limit parameter of the TBF queueing discipline implemented in the egress interface of router R2. Various throughput tests are performed between one sender and one receiver and the queue is sampled every 0.5 seconds. Then, the results are aggregated and averaged. Each test lasts 500 seconds, and the propagation delay is 20ms. In this way, 5,000 samples are collected. The queue size undergoes variations, ranging from 0.5BDP to 10BDP, then reverses course from 10BDP down to 0.5BDP.

Figure 6 illustrates that BBRv3 effectively maintains a controlled queue length, preventing it from exceeding 0.5BDP. This is due to the estimation mechanism that BBRv3 implements to approximate the optimal operation point where the inflight data is maximized and the propagation delay is minimized. This result is similar to the standing queue observed in BBRv1 and BBRv2, which also settles around 0.5 BDP [11]. In contrast, CUBIC tends to fill the queue to its maximum capacity, resulting in bufferbloat [5]. As a result, the end-to-end latency of the connection is around 200ms.

5.4. Throughput, Retransmisions, and Packet Losses

This experiment evaluates the performance of a BBRv3 flow as a function of the loss rate. Unlike loss-based CCAs, BBRv3 and BBRv2 take a different approach, considering factors such as bandwidth, RTT, loss rate, and ECN to model the network path. On the other hand, BBRv1 only considers the bottleneck bandwidth estimation and the RTT. This experiment considers a loss rate ranging from 0.001% (i.e., 10 out of 1,000,000 packets) to 20% (200 out of 1,000 packets). The metrics evaluated include throughput and the number of retransmitted packets. Each test employs a 20ms propagation delay, with buffer sizes adjusted to the BDP. Ten evaluations are performed for each loss rate, and the results are averaged. The tests have a duration of 120 seconds each. Figure 7(a) illustrates that the throughput of BBRv3 and BBRv2 starts to decline noticeably when the loss rate surpasses 0.1%. At a loss



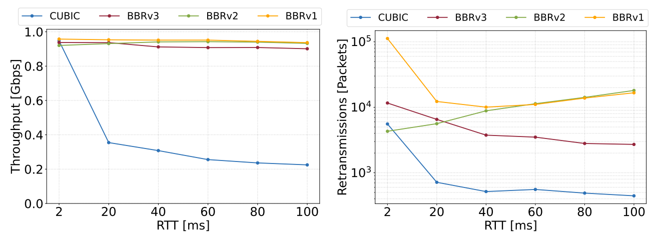
(a) Throughput. (b) Retransmissions.

Figure 7: Throughput and retransmissions as a function of the packet loss rate.

rate of 0.5%, BBRv3 and BBRv2 achieve around 60% of their maximum performance, coinciding with the point of the highest retransmissions, as observed in Figure 7(b). A similar trend was reported in [11], where BBRv2 achieves its maximum retransmission rate with a 0.1% loss rate before the throughput decreases below 60%. This observation indicates that while BBRv3 exhibits a proactive response with packet retransmissions similar to BBRv2, the retransmission rate is significantly reduced compared to BBRv1. This behavior occurs because BBRv3 and BBRv2 calculate the loss rate to adjust the sending rate, with the goal of preventing an excessive impact on competing flows. On the other hand, CUBIC's performance dips to 60% with just 0.01% packet losses. It's worth noting that at 0.001% and 0.01% loss rates, CUBIC, BBRv3, and BBRv2 exhibit a similar number of retransmissions. However, when the loss rate reaches 0.1%, CUBIC's performance drops below 20% of the available bandwidth, and its retransmission rate settles around 1000 packets for higher loss rates. In contrast, BBRv3 and BBRv2 maintain a performance above 20% until the loss rate exceeds 1%. BBRv1 exhibits a throughput of around 1Gbps even under loss rates up to 20%, which is also observed in [4] with a 100Mbps link. However, this performance gain comes at the cost of a significantly higher retransmission rate than BBRv3 and BBRv2. Specifically, when facing loss rates ranging from 1% to 20%, BBRv1 results in a retransmission count of up to 10 million packets, a significantly higher number than BBRv3 and BBRv2. This behavior leads to unfairness with loss-based CCAs, as observed in section 5.1.

5.5. Performance with Different RTTs and Packet Loss

Packet losses can significantly affect TCP performance, and they can occur due to various network components such as routers, switches, and firewalls [37, 38]. Loss-based TCP CCAs such as CUBIC respond to packet losses by reducing their sending rate, which leads to a decrease in the overall performance. This impact becomes more pronounced as the RTT of the network path increases. In contrast, BBRv3 takes a different approach. It considers multiple metrics, including bandwidth, RTT, loss rate, and ECN, to model the network path and maximize throughput. This experiment aims to evaluate how packet losses affect the performance of BBRv3, BBRv2, and BBRv1 across different RTTs and compare it to the performance of a CUBIC flow under the same conditions. The RTTs vary from 2ms to 100ms, with a bottleneck link of 1Gbps,



(a) Throughput. (b) Retransmissions.

Figure 8: Throughput and retransmission of CUBIC, BBRv1, BBRv2, and BBRv3 as a function of the RTT.

and the buffer size is adjusted to the BDP for each RTT. Additionally, the number of retransmissions as a function of the RTT is reported.

Figure 8 shows the throughput and retransmissions of data transfers across a 1Gbps path. The emulated packet loss rate is 0.025% (i.e., 1 out of 4000 packets). Figure 8(a) shows that BBRv3, BBRv2, and BBRv1 maintain high throughput over RTTs ranging from 2ms to 100ms compared to CUBIC, which with RTTs higher than 2ms and packet losses presents lower performance. Similar observations are reported in [8, 9, 11]. This performance observed in BBRv3 makes it suitable for environments such as a Science DMZ [37] to enable large data transfers. Figure 8(b) illustrates that BBRv3 decreases the number of retransmissions as the RTT increases, contrasting with the behavior of BBRv2 [11], which exhibits an increase in retransmissions with growing RTT. BBRv3 employs adaptive rate control mechanisms, adjusting the sending rate based on the available bandwidth and the RTT. By updating the gain of the performance parameters [20, 21], BBRv3 achieves and maintains high throughput while minimizing retransmission rates in contrast to BBRv2. This behavior enables a better coexistence with loss-based CCAs such as CUBIC. On the other hand, for RTTs of 2ms and 20ms, BBRv1 exhibits a higher retransmission rate while demonstrating similar performance to BBRv2 for RTTs higher than 20ms.

5.6. *Fairness of BBRv3, BBRv2, and BBRv1 Flows with a CUBIC flow*

This experiment evaluates the fairness between multiple BBRv3, BBRv2, and BBRv1 flows with a CUBIC flow. In this scenario, five of each BBRv3, BBRv2, and BBRv1 flows compete against one CUBIC flow to observe the evolution of the fairness index and the throughput as a function of time. Initially, the CUBIC flow begins, and then, at intervals of 60 seconds, additional BBRv3, BBRv2, and BBRv1 flows join. The scenario considers a bottleneck link of 1Gbps and an RTT of 20ms. The buffer size of the bottleneck link is adjusted to BDP. The test duration is 480 seconds.

Figure 9 shows the fairness index and the throughput of a CUBIC flow and multiple subsequent BBRv3, BBRv3, and BBRv1 flows. It is observed at t=60, that one CUBIC flow and one BBRv3 flow do not converge to fairness. In this case, the fairness index remains around 80%. However, as additional BBRv3 flows join, they share the available bandwidth equitably. When the last BBRv3 flow joins, fairness is achieved among them.

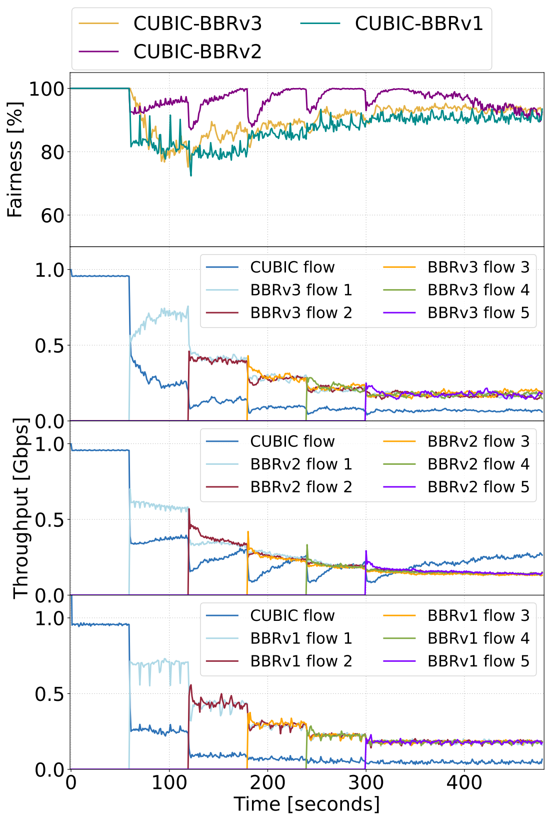


Figure 9: Throughput and fairness index over time as a CUBIC flow interacts with successive BBRv3, BBRv2, and BBRv1 flows.

Conversely, the CUBIC flow retains a smaller share, resulting in a fairness index of approximately 90% that favors the BBRv3 flows. When BBRv2 flows compete against a CUBIC flow, it's noted that a new BBRv2 flow initially decreases the bandwidth share of the CUBIC flow. However, after approximately 60 seconds of adding the BBRv2 flow, the flows converge towards fairness. After all BBRv2 flows start, the CUBIC flow experiences an increase in bandwidth share, reducing the overall fairness among the flows. This behavior aligns with previous observations in evaluations of BBRv2 [7]. In contrast, BBRv1 exhibits a lower fairness index than BBRv3. Across all three experiments, BBR flows eventually converge to fairness, whereas the CUBIC flow achieves fairness only after 60 seconds when competing against BBRv2 flows.

5.7. *Coexistence with CUBIC*

This test examines how effectively different quantities of BBRv3 flows coexist with CUBIC flows. Specifically, the experiment assesses the coexistence of 1, 2, 3, 5, and 10 BBRv3 flows with various numbers of CUBIC flows, ranging from 1 to 10. The test setup includes a 1Gbps bottleneck link with a 20ms delay. Each test is repeated ten times for improved accuracy and the test duration is 120 seconds.

Figure 10 examines how the number of competing flows impacts the throughput share for each CCA. In the graph, dashed lines depict the theoretical fair share, while solid lines indicate the actual measured share. Overall, it's evident that when BBRv3 flows compete against CUBIC flows, BBRv3 tends to utilize more bandwidth than its fair share. However, as the number of BBRv3 flows

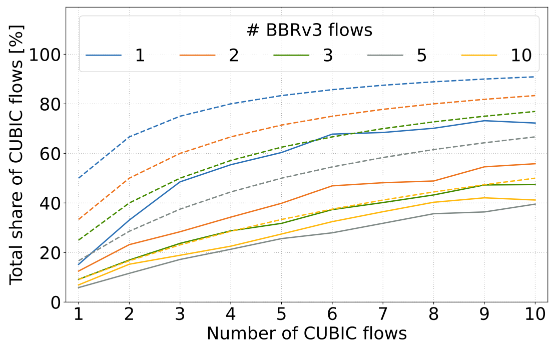


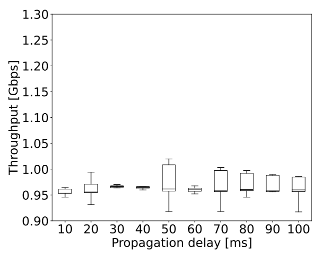
Figure 10: Bandwidth share for competing CUBIC and BBRv3 flows. The dashed lines show the ideal fair share and the solid lines show the measured share.

increases, CUBIC's portion of the bandwidth gets closer to the expected fair share. When there are ten BBRv3 flows and two CUBIC flows, the achieved share matches the fair share precisely. Previous evaluations of BBRv2 [6] present the same behavior, where the fair share is achieved with ten BBRv3 flows and two CUBIC flows. In this experiment is observed that having more BBRv3 flows diverts more from the fair share compared to using BBRv2. This behavior indicates that BBRv3 flows are more aggressive than BBRv2 flows when interacting with loss-based CCAs.

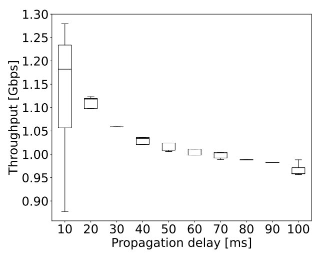
5.8. *Bottleneck Bandwidth Estimation*

This experiment compares how BBRv3, BBRv2, and BBRv1 estimate the bottleneck bandwidth in relation to the propagation delay. The bottleneck bandwidth estimation represents the maximum data rate that can be effectively transmitted through a specific path or segment of a network. The experimental setup considers a Mininet topology with an emulated 1Gbps bottleneck link. The experiments involve a BBRv3 flow from a sender to a receiver. The estimation of the bottleneck bandwidth is derived from Linux socket statistics, utilizing the `ss -tin` command [33], which probes for samples every 100ms.

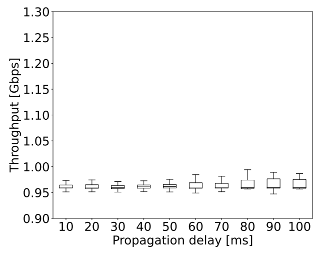
Figure 11 presents results acquired from an emulated bottleneck link. These results illustrate the distribution of bottleneck bandwidth estimations across propagation delays ranging from 10ms to 100ms. It is observed only a minor fluctuation in values across these varying delays. In BBRV3 and BBRv1, it is observed that the mean value of the estimation settles around 0.95Gbps. This behavior is observed for all the propagation delays. BBRv3, as its predecessors, relies on the sender's computed ACK rates to estimate the bottleneck bandwidth. However, Su *et al.* [39] identified an issue with the estimation mechanism in BBRv1, where it produces implausible ACK patterns, leading to inaccurate bottleneck estimations. This problem arises during the probing phase, where BBR's sending rate can exceed the bottleneck rate by 1.25 times. Even if the ACK rate does not surpass the sending rate, BBR adapts to and maintains the overestimated rate for approximately 10 RTTs, introducing additional queuing delays. This overestimation problem results in prolonged queuing delays, particularly in low-packet-loss environments, as the RTT increases. On the other hand, it is observed that BBRv2 overestimates the bottleneck bandwidth for lower propagation delays. Similar findings were



(a) BBRv3.



(b) BBRv2.



(c) BBRv1.

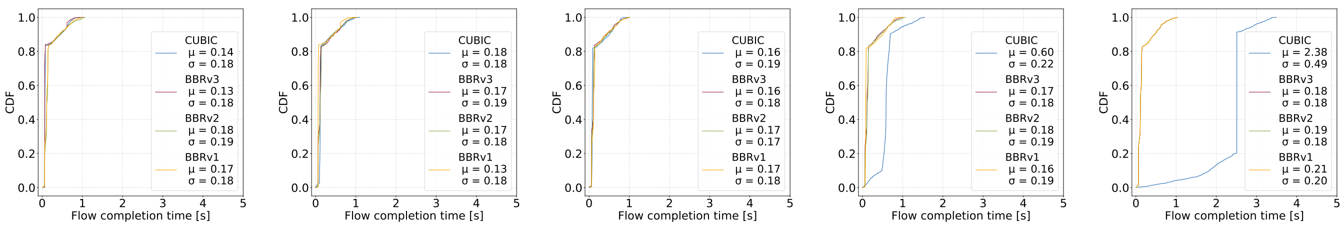
Figure 11: Boxplot of the bottleneck bandwidth estimation as a function of the RTT of (a) BBRv3, (b) BBRv2, and (c) BBRv1.

reported by Vargas et al. [40], indicating that the overestimation is attributed to the burstiness of the traffic. In response to this, the authors suggested an alternative estimation approach designed to filter out the effects of traffic burstiness.

5.9. *Flow Completion Time of Short Flows*

A major share of internet traffic involves web browsing. This experiment focuses on measuring the FCT of short BBRv3 flows, which closely resemble typical web browsing behavior. This test also includes the CDFs of BBRv1 and BBRv2 flows. The setup involves a client generating 5,000 HTTP requests to a web server. These requests are designed to retrieve small files stored on the web server, resulting in the creation of short-lived TCP flows. Concurrently, a long TCP flow shares the same network bottleneck as the short flows. The primary objective here is to examine how the completion time of these short flows is affected when they compete for network resources with a long flow. The experimental conditions consider a 1Gbps bottleneck link with a 20ms propagation delay. Multiple buffer sizes are considered and the Cumulative Distribution Function (CDF) of the FCT of short flows is reported.

The results, as depicted in Figure 12, illustrate the CDFs of the FCT for both CUBIC, BBRv1, BBRv2, and BBRv3 flows. It is observed when the buffer sizes are less than or equal to BDP, the FCT values for BBRv1, BBRv2, BBRv3, and CUBIC flows are similar. Within this range, the mean FCT values for both algorithms vary from 0.13 seconds to 0.19 seconds. However, as the buffer sizes exceed the BDP threshold, the FCT of CUBIC flows experiences a significant increase. For instance, with buffer sizes of 100BDP, the mean FCT for CUBIC reaches approximately 2.38 seconds. In contrast, BBRv1, BBRv2, and BBRv3 maintain lower FCT values, with a mean of around 0.18 seconds, even with larger buffer sizes. This behavior in CUBIC is explained because the CUBIC flows tend to fill the router's buffer. On the other



(a) 0.1BDP (b) 0.5BDP (c) BDP (d) 10BDP (e) 100BDP

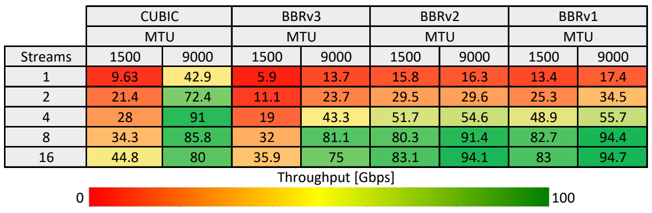
Figure 12: The Cumulative Distribution Function (CDF) for the Flow Completion Time (FCT) of short flows using CUBIC, BBRv1, BBRv2, and BBRv3 is analyzed while these flows compete with a long flow. It is important to note that the CDF curves for BBRv1, BBRv2, and BBRv3 overlap.

hand, BBRv1, BBRv2, BBRv3, try to send data at the highest possible rate without creating a queue.

5.10. Effects of Parallel Streams and Different MTUs

This experiment explores the performance variations achievable using different configurations of parallel data streams and Maximum Transmission Units (MTUs) with CUBIC, BBRv3, BBRv2, and BBRv1. The experimental setup involves two VMs connected via a high-speed 100Gbps link, characterized by a low propagation delay of less than 1 ms. There are no emulated bottlenecks or packet losses in this topology. Path MTU Discovery (PMTUD) is enabled (i.e., `net.ipv4.tcp_mtu_probing=1`) to prevent PMTU black holes [41, 42] and allow TCP flows to achieve higher performance. Each experiment has a duration of 120 seconds and is repeated ten times. The results from these repetitions are then averaged to enhance accuracy.

Figure 13 presents a heatmap illustrating the average throughput concerning the number of parallel streams and MTUs for CUBIC, BBRv3, BBRv2, and BBRv1. It is observed that CUBIC consistently achieves a higher throughput compared to BBRv3 across different configurations. CUBIC demonstrates its peak performance when utilizing four parallel streams with an MTU of 9000 bytes. In contrast, BBRv3 reaches its optimal performance when employing eight parallel streams and an MTU of 9000 bytes. On the other hand, for MTU sizes of 1500 bytes, both CUBIC and BBRv3 perform at their best with 16 parallel streams. CUBIC performance is higher than BBRv3 with low latency because it can quickly increase the window size after a congestion event and avoid oscillations around the optimal point [3]. However, BBRv3, while more aggressive in situations with higher RTTs, experiences limited startup gains in low-RTT scenarios. BBRv2 and BBRv1 present comparable performance when using the same MTUs [11].



	CUBIC		BBRv3		BBRv2		BBRv1	
	MTU		MTU		MTU		MTU	
Streams	1500	9000	1500	9000	1500	9000	1500	9000
1	9.63	42.9	5.9	13.7	15.8	16.3	13.4	17.4
2	21.4	72.4	11.1	23.7	29.5	29.6	25.3	34.5
4	28	91	19	43.3	51.7	54.6	48.9	55.7
8	34.3	85.8	32	81.1	80.3	91.4	82.7	94.4
16	44.8	80	35.9	75	83.1	94.1	83	94.7

Throughput [Gbps]

0 100

Figure 13: Average throughput of CUBIC, BBRv3, BBRv2, and BBRv1. The throughput is given as a function of the number of parallel streams and the Maximum Transmission Unit (MTU).

6. Conclusion

This paper presented an evaluation of BBRv3, using Mininet and real hardware. BBRv3 shows resilience to packet losses, compared to CUBIC in maintaining throughput at 1% loss rates. Fairness in BBRv3 is influenced by buffer size and propagation delay, with optimal fairness achieved between BDP and 10 BDP. The integration of FQ-CoDel AQM mitigates RTT unfairness across buffer sizes. BBRv3 exhibits low queue occupancy, contributing to lower latency compared to CUBIC. In scenarios with short flows against a long flow, BBRv3 maintains consistently low Flow Completion Times (FCT) with larger buffer sizes. While BBRv3 presents high performance in various scenarios, coexistence challenges with other BBRv3 flows and fairness issues with loss-based algorithms may require further optimization in specific network conditions. Future research could explore BBRv3 in higher-throughput scenarios and non-emulated propagation delays.

7. Acknowledgement

This work is supported by the National Science Foundation (NSF), Office of Advanced Cyberinfrastructure (OAC), under grant number 2118311.

References

[1] J. Postel, "RFC 793: Transmission control protocol (TCP)," September 1981.
[2] M. Allman, "RFC 2581: TCP congestion control," April 1999.
[3] S. Ha, I. Rhee, and L. Xu, "CUBIC: a new TCP-friendly high-speed TCP variant," *ACM SIGOPS operating systems review*, 2008.
[4] N. Cardwell, Y. Cheng, S. Gunn, S. Yeganeh, and V. Jacobson, "BBR: Congestion-based congestion control," *Queue*, 2016.
[5] J. Gettys, "Bufferbloat: Dark buffers in the Internet," *IEEE Internet Computing*, 2011.
[6] E. Kfoury, J. Gomez, J. Crichigno, and E. Bou-Harb, "An emulation-based evaluation of TCP BBRv2 alpha for wired broadband," *Computer Communications*, 2020.
[7] J. Gomez, E. Kfoury, J. Crichigno, E. Bou-Harb, and G. Srivastava, "A performance evaluation of TCP BBRv2 alpha," in *2020 43rd International Conference on Telecommunications and Signal Processing (TSP)*, 2020.
[8] Y. Song, G. Kim, I. Mahmud, W. Seo, and Y. Cho, "Understanding of BBRv2: evaluation and comparison with BBRv1 congestion control algorithm," *IEEE Access*, 2021.

[9] 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.

[10] 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*, 2022.

[11] J. Gomez, E. Kfoury, J. Crichigno, and G. Srivastava, "Understanding the performance of TCP BBRv2 using FABRIC," *2023 IEEE BlackSeaCom, Istanbul, Turkiye*, 2023.

[12] N. Cardwell, Y. Cheng, S. Yeganeh, I. Swett, V. Vasiliev, P. Jha, Y. Seung, M. Mathis, and V. Jacobson, "BBRv2: a model-based congestion control," *Presentation in the Internet Congestion Control Research Group (ICCRG) at IETF 105 Update, Montreal, Canada, July, 2019.*

[13] N. Cardwell, Y. Cheng, K. Yang, D. Morley, S. Y. Hassas, P. Jha, Y. Seung, V. Jacobson, I. Swett, B. Wu, and V. Vasiliev, "BBRv3: Algorithm bug fixes and public Internet deployment." [Online]. Available: `https://tinyurl.com/2p9x5fjn`, Accessed on 08-30-2023.

[14] J. Gomez, "BBRv3 experiments' scripts." [Online]. Available: `https://github.com/gomezgaona/bbr3`, Accessed on 08-14-2023.

[15] V. Jacobson, "Congestion avoidance and control," *ACM SIGCOMM computer communication review*, 1988.

[16] P. Hurtig, H. Haile, K.-J. Grinnemo, A. Brunstrom, E. Atxutegi, F. Liberal, and Å. Arvidsson, "Impact of TCP BBR on CUBIC traffic: A mixed workload evaluation," in *2018 30th International Teletraffic Congress (ITC 30)*, 2018.

[17] A. Nandagiri, M. Tahiliani, V. Misra, and K. Ramakrishnan, "BBRv1 vs BBRv2: Examining performance differences through experimental evaluation," in *2020 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN*, 2020.

[18] M. Hock, R. Bless, and M. Zitterbart, "Experimental evaluation of bbr congestion control," in *2017 IEEE 25th international conference on network protocols (ICNP)*, 2017.

[19] N. Cardwell, Y. Cheng, K. Yang, D. Morley, S. Hassas, P. Jha, and Y. Seung, "BBR congestion control: Fundamentals and updates." [Online]. Available: `http://tinyurl.com/tj2ts8a8`, Accessed on 08-29-2023.

[20] I. Swett and The Google BBR Team, "BBR startup cwnd gain: a derivation." [Online]. Available: `https://tinyurl.com/yyexzn3w`, Accessed on 08-14-2023.

[21] The Google BBR Team, "BBR startup pacing gain: a derivation." [Online]. Available: `https://tinyurl.com/mr2vbw3v`, Accessed on 08-14-2023.

[22] I. Baldin, A. Nikolich, J. Griffioen, I. Monga, K. Wang, T. Lehman, and P. Ruth, "FABRIC: A national-scale programmable experimental network infrastructure," *IEEE Internet Computing*, 2019.

[23] S. Floyd and M. Allman, "Specifying new congestion control algorithms," tech. rep., 2007.

[24] K. Kaur, J. Singh, and N. Ghumman, "Mininet as software defined networking testing platform," in *International conference on communication, computing & systems (ICCCS)*, 2014.

[25] N. Cardwell, "TCP BBRv3 Release." [Online]. Available: `https://tinyurl.com/mruajjvw`, Accessed on 08-14-2023.

[26] B. Pfaff, J. Pettit, T. Koponen, E. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, and P. Shelar, "The design and implementation of open vSwitch," in *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI15)*, 2015.

[27] Juniper Networks, "MX204 universal routing platform." [Online]. Available: `https://tinyurl.com/yz86p3vx`, Accessed on 08-14-2023.

[28] S. Hemminger, "Network emulation with NetEm," in *Linux conf au*, 2005.

[29] T. Høiland-Jørgensen, P. Hurtig, and A. Brunstrom, "The good, the bad and the WiFi: Modern AQMs in a residential setting," *Computer Networks*, 2015.

[30] T. Høiland-Jørgensen, P. McKenney, D. Taht, J. Gettys, and E. Dumazet, "RFC 8290: The flow queue CoDel packet scheduler and active queue management algorithm," tech. rep., January 2018.

[31] Juniper Networks, "Class of Service User Guide." [Online]. Available: `https://tinyurl.com/2swtshvj`, Accessed on 09-11-2023.

[32] J. Dugan, S. Elliott, B. Mah, J. Poskanzer, and K. Prabhu, "iPerf3, tool for active measurements of the maximum achievable bandwidth on IP networks," *URL: https://github.com/esnet/iperf*, 2014.

[33] M. Kerrisk, *The Linux programming interface: a Linux and UNIX system programming handbook.* 2010.

[34] S. Floyd, "RFC 5166: metrics for the evaluation of congestion control mechanisms," 2008.

[35] L. Kleinrock, "Internet congestion control using the power metric: Keep the pipe just full, but no fuller," *Ad hoc networks*, 2018.

[36] S. Ma, J. Jiang, W. Wang, and B. Li, "Towards RTT fairness of congestion-based congestion control," *Computer Science Archive (arXiv)*, 2017.

[37] J. Crichigno, E. Bou-Harb, and N. Ghani, "A comprehensive tutorial on science DMZ," *IEEE Communications Surveys & Tutorials*, 2018.

[38] E. Dart, L. Rotman, B. Tierney, M. Hester, and J. Zurawski, "The science DMZ: A network design pattern for data-intensive science," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, 2013.

[39] B. Su, X. Jiang, G. Jin, and H. Chen, "Rethinking the rate estimation of BBR congestion control," *Electronics Letters*, 2020.

[40] S. Vargas, R. Drucker, A. Renganathan, A. Balasubramanian, and A. Gandhi, "BBR bufferbloat in DASH video," in *Proceedings of the Web Conference 2021*, 2021.

[41] Energy Sciences Network (ESnet), "MTU issues." [Online]. Available: `https://tinyurl.com/5fypnknn`, Accessed on 09-08-2023.

[42] J. Bosma, B. Overeinder, and W. Toorop, "Discovering path MTU black holes on the internet using RIPE Atlas," 2012.