applied sciences

MDPI

Article

Performance Evaluation of TCP BBRv3 in Networks with Multiple Round Trip Times

Agnieszka Piotrowska

Department of Computer Networks and Systems, Silesian University of Technology; 44-100 Gliwice, Poland; agnieszka.piotrowska@polsl.pl

Abstract: The Transmission Control Protocol (TCP) serves as a cornerstone mechanism for implementing Congestion Control (CC) across the Internet. Designing a solution that provides high bandwidth utilization and mitigates the phenomenon of bufferbloat across a spectrum of diverse scenarios poses a considerable challenge. The introduction of Bottleneck Bandwidth and Round Trip propagation time (BBR) in 2016 marked a significant shift in congestion control methodology. Its improved performance and adaptability contributed to the initial acclaim and widespread interest that it received.. Unlike most currently used CCs, it operates around Kleinrock's optimal point, thus offering high throughput even in lossy networks while preventing buffer saturation. Unfortunately, it quickly became evident that BBR was unable to fairly share bandwidth with flows characterized by different path delays, as well as loss-based CCs. In response, Google recently introduced a third iteration to address these shortcomings. This study explores the performance of BBRv3 across a wide range of scenarios, thereby considering different buffer sizes and paths with varying Round Trip Times (RTTs), and it evaluates its superiority over its predecessors. Through extensive simulations, this work assesses whether BBRv3 can finally play fair with other bandwidth contenders, which is a critical consideration given the widespread deployment of BBR. The framework is publicly available to facilitate additional validation and ensure the reproducibility of the study's findings. The results indicate that while BBRv3 demonstrates enhanced fairness towards loss-based CC algorithms, it struggles when competing against other BBR flows, especially in multi-RTT networks, thus falling short even when compared to the initial version.

Keywords: TCP; BBRv3; BBRv2; BBRv1; CUBIC; congestion control; performance evaluation; simulation; fairness

Citation: Piotrowska, A. Performance Evaluation of TCP BBRv3 in Networks with Multiple Round Trip Times. *Appl. Sci.* **2024**, *14*, 5053. https://doi.org/10.3390/app14125053

Academic Editor: Inmaculada Ayala Viñas and Laura Panizo

Received: 10 May 2024
Revised: 4 June 2024
Accepted: 4 June 2024
Published: 10 June 2024

1. Introduction

The primary role of congestion control, implemented in the TCP, involves managing transmission rates to optimize bandwidth utilization, respond efficiently to dynamic congestion scenarios, ensure dependable data delivery, and fairly allocate bandwidth among competing flows. The TCP protects the Internet against data flooding and network collapse.

As link speeds increase, one might expect seamless and rapid data delivery. However, despite high bandwidth capacities, the actual transmission rate lags, thus resulting in significant delays across numerous scenarios. This phenomenon is attributed to loss-based TCPs, e.g., CUBIC, which frequently misinterpret packet losses as a signal of congestion, thus unnecessarily reducing transmission rates.

In 1979, Kleinrock proved that the most optimal operational point, ensuring maximal throughput and minimal delays, corresponds to the location denoted as (A) in Figure 1. To operate around point (A), the inflight data should saturate the pipeline without overflowing the buffer. Ideally, the inflight data should approximate the Bandwidth Delay Product (BDP), which is defined as $RTT_{min} \times BW_{max}$, where BW_{max} represents the available bottleneck bandwidth, and RTT_{min} is the minimum propagation delay along the respective path, without any additional queuing delay. If the estimated inflight volume is below the

BDP, the connection will not be able to fully utilize the link. Conversely, surpassing the (A) point leads to queue formation and subsequent performance degradation.

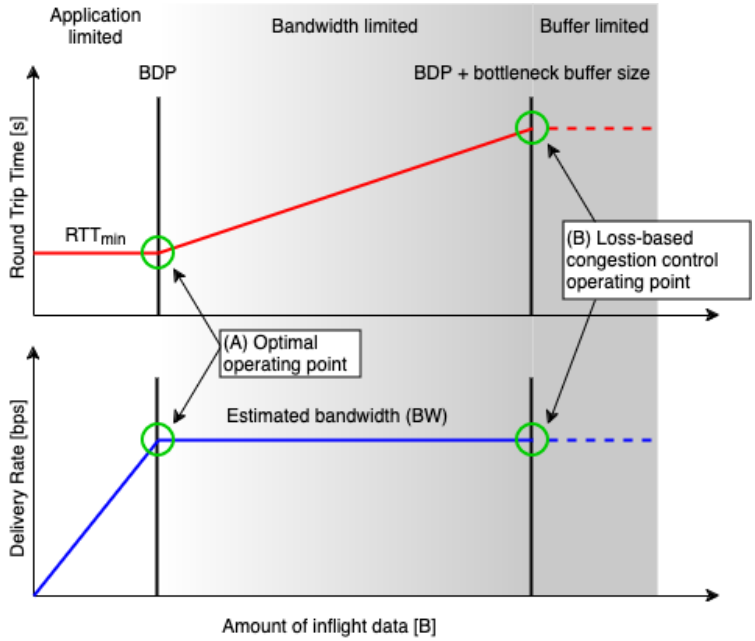


Figure 1. Influence of increasing inflight data on RTT and delivery rate based on [1].

At the same time, Jaffe presented a paper that proved it was impossible to design a distributed algorithm that would converge at Kleinrock's optimal point [2]. These results motivated research in other directions for congestion control design. Most common congestion control mechanisms follow the loss-based approach, which moves the operational point to the right towards point (B). This leads to several undesired issues such as buffer overflow and bufferbloat, i.e., "persistently full buffer", as described in [3]. A full buffer cannot compensate for packet bursts, which inherently occur due to the best-effort nature of the network. This results in substantial delays, thus impeding the real-time interactions necessary for numerous applications and consequently leading to poor user experiences. Moreover, it hinders latency-sensitive applications such as web services and financial transactions, as well as the low-latency communication required in Internet of Things (IoT) networks. Full buffers additionally degrade transmission performance in mobile and wireless networks, which are already subject to highly variable latency.

Most commonly used TCP variants, e.g., CUBIC, follow the loss-based approach. CUBIC demonstrates high performance across various scenarios, particularly when coupled with Active Queue Management (AQM), thus resulting in efficient bandwidth utilization, fair bandwidth sharing, and a relatively low drop ratio [4,5]. However, its effectiveness is limited as link capacities scale up to 10 or 100 Gbps, especially when coupled with long RTTs. For instance, to fully utilize a 10 Gbps link with a 100 ms RTT, CUBIC requires a packet loss ratio below 0.000003% and intervals exceeding 40 s between any instances of packet loss—criteria that are impractical given typical link characteristics. In the case of a 1% packet loss, a common occurrence in links exceeding 100 ms, CUBIC fails to achieve a throughput exceeding 3 Mbps [6]. Random losses are also common in wireless networks, which represent another battlefield where CUBIC is defeated.

In 2016, Google introduced a novel TCP algorithm known as Bottleneck Bandwidth and Round Trip propagation time (BBR) [7]. BBR follows a model-based, rate-driven approach with the primary objective of maintaining high bandwidth utilization, thereby achieving high throughput while mitigating buffer congestion to minimize delays. Unlike loss-based algorithms, BBR ignores packet losses and estimates both the available

bandwidth (BW_{max}) and the minimum round trip propagation time (RTT_{min}) along a transmission path to calculate the path's Bandwidth Delay Product (BDP).

BBR boasts several advantages that have earned it considerable attention. Unlike the majority of commonly used congestion control mechanisms (CCs), it is resilient to random losses (up to 15%) while consistently maintaining short queues regardless of buffer size [7]. Moreover, it is easy to deploy, since it requires changes only on the sender side, and it does not require any network functionality such as Explicit Congestion Notification (ECN) or Active Queue Management (AQM).

Soon after its initial release, it became evident that BBR, despite its numerous advantages, faced challenges related to fair bandwidth sharing and excessive loss ratios across various scenarios [8–10]. Its performance and the fairness it provided relied heavily on factors such as the buffer depth and the the RTTs of paths taken by competing flows. Furthermore, it tended to deprive loss-based connections of bandwidth [11,12].

In 2019, Google released the second iteration of BBR [13]. BBRv2 was designed to adjust more swiftly and foster better convergence to fair bandwidth sharing by allocating sufficient headroom for both loss-based and newly emerging flows. Additionally, BBRv2 actively monitors packet losses or ECN rates, thus preventing excessive loss ratios. The second iteration solved the problem of high loss and retransmission ratios; unfortunately, it could not provide fair bandwidth sharing in the presence of other BBR flows with different RTTs and loss-based TCP flows [14,15].

The most recent iteration, BBRv3 [16], is expected to improve convergence with other BBR flows and coexist more effectively with loss-based CCs while reducing queuing delays, irrespective of buffer size . Given that Google has replaced CUBIC for all TCP flows on Google's B4 WAN, thus resulting in BBR accounting for up to 14% of all traffic, as well as the recent announcement regarding the adoption of BBRv3 for all internal traffic and all Google.com public Internet traffic, this underscores the necessity of verifying the optimal performance of BBRv3 [17].

This study undertakes a comprehensive analysis of the BBR protocol and extends it with BBRv3 to evaluate its performance across a wide range of scenarios. The objective is to discern the fundamental characteristics of the newly proposed version of BBR concerning transmission efficiency in multi-RTT networks and its coexistence with the popular loss-based CUBIC algorithm. This study establishes a simulation environment based on ns-3 to assess the performance of all available BBR versions. Through extensive analysis of the BBR protocol, the work demonstrates the capabilities of the simulation framework. This environment facilitates the easy manipulation of environmental conditions, thereby enabling the exploration of scenarios that may be challenging to replicate in an emulator environment or real-world configuration. This approach helps identify weaknesses and points for improvement. To the best of the author's knowledge, this study represents the first simulation-based comparison of all BBR versions. The scope discussed in this manuscript has not been comprehensively analyzed in published works using both emulator and real-world approaches.

The main contributions are as follows:

- This study adapted the implementations of BBRv2 and BBRv3 for integration into the ns-3 simulator and presents results confirming their intended behavior for single-flow scenarios. This study has released the simulation framework as open-source to encourage further experimentation and refinement within the research community.
- This study performed a comprehensive evaluation of all versions of BBR, thus analyzing network performance metrics, including throughput, loss ratio, and intra- and interprotocol fairness, in scenarios with various distributions of RTTs, including a representative distribution of RTTs at the central link, as well as a mix of varying RTT ratios for different links.
- This study conducted simulations on a 100 Mbps network, with buffer sizes ranging from 0.1 to 10 times BDP. The interprotocol fairness was analyzed in comparison

to CUBIC CC. Routers implemented a drop-tail strategy; neither ECN nor AQM was evaluated.
- Through detailed analysis, this study demonstrates that BBRv3 addresses certain limitations observed in its predecessors. However, issues persist regarding its fairness towards loss-based flows and those with varying RTT characteristics. Additionally, this study examines whether streams with longer RTT consistently achieve greater throughput.

The remainder of this paper is organized as follows: the related work is reviewed in Section 2. Section 3 provides details of BBR behavior and the distinctions among different versions of BBR. The simulation framework is described in Section 4, followed by the presentation of evaluation results in Section 5. Section 6 concludes and summarizes the findings of this evaluation.

2. Related Work

BBRv1 and BBRv2 have been thoroughly investigated in the existing literature. However, given the recent release of BBRv3, there are not many papers addressing its performance.

Empirical results detailing the performance of BBRv3 across a spectrum of network scenarios are presented in [18]. The authors explored various buffer sizes, RTT times, packet losses, and flow size distributions. The findings presented in this study contradict the design principles of BBRv3 and reveal its disappointing performance compared to the initial version.

The coexistence of BBRv3 with CUBIC and other BBRv3 flows was examined in [19] across various conditions. The authors evaluated the interactions between CUBIC and BBRv3 in scenarios with uniform RTTs across the network using the Mininet emulator and demonstrated that both BBRv2 and BBRv3 enhance fairness towards loss-based CCs. Additionally, they assessed interfairness as the number of flows was increased to 100. The intrafairness scenario, involving experiments with two different RTTs on the network, confirmed that the RTT unfairness issue persists in the latest iteration.

The challenge of ensuring fairness in a network with varying RTTs is not exclusive to BBR. Numerous studies highlight this as a well-documented issue, thus noting that most CC algorithms tend to favor streams with shorter RTTs [20]; conversely, BBR favors long-RTT streams [11,18,19]. Nevertheless, in this paper, it is posited that this correlation is not as straightforward, and its dynamic is more intricate.

The problem of BBR unfairness and lossiness has been widely studied in the literature [8–11,15]. Soon after releasing the first version, it became clear that BBR was responsible for excessive packet loss, especially in shallow buffers, and high queuing variation in large buffers. In multi-RTT networks, it favored flows with higher RTT; this was particularly evident if the ratio of RTT among flows was higher than 2. It was also apparent that there was a significant inequality in bandwidth distribution in the presence of loss-based algorithms.

Scholz et al. [9] presented a detailed analysis of the behavior of BBRv1. They confirmed that it is highly susceptible to shallow buffers, as it overestimates the available bandwidth and results in excessive packet loss. Cardwell et al. demonstrated that this holds true when the buffer cannot accumulate an additional 1.5*BDP [6] shallow buffers induce numerous retransmissions.

Numerous research papers have underscored that several BBRv1 flows tend to overestimate the available bandwidth. Proper RTT estimation is based on the assumption that during the PROBE RTT phase, the queue is completely drained. Therefore, the RTT cannot be properly assessed if other flows are building the queue. As explained by [11], when only BBR flows exist, the nature of BBR should ensure that these flows enter the PROBE RTT phase simultaneously, thus resulting in the proper behavior of BBR. However, since BBR uses the observed maximum filtered delivery rate as its sending rate, it tends to overestimate the available bandwidth, thus resulting in the buildup of larger queues (which impairs the PROBE RTT phase) or an excessive loss ratio, especially with small buffers.

The superiority of BBRv2 was studied in [15]. The authors confirmed that BBRv2 significantly better controls the amount of inflight data, thus reducing the number of packet losses and retransmissions compared to BBRv1 in various scenarios. However, it was quickly established that, although BBRv2 mitigates the major problem of its predecessor, it is not able to provide fair bandwidth sharing in the presence of other BBR flows with different RTTs and loss-based TCP flows. The authors showed that the convergence between two BBRv2 flows relies on the buffer size. Intraprotocol fairness is only achieved when the buffer is below 0.3*BDP. With larger buffers, staggering flows cannot achieve a fair bandwidth share. BBRv2 allows for better RTT fairness with small buffers, but with large buffers, the problem persists. With buffers below 2*BDP, BBRv2 provides better fairness towards CUBIC.

In [10], Nandagiri et al. presented a detailed analysis of both versions and provided results of comparison in terms of intraprotocol fairness and interprotocol fairness towards CUBIC. They indicated ECN as the most important factor that helps the second iteration better control the queue in shallow buffers. They confirmed that without ECN, BBRv2 is unfair to CUBIC in deep buffers. CUBIC also benefits from ECN (as from any other configurationless AQM).

3. BBR Operation Principles

BBR is designed to maximize the delivery rate without building the standup queue, thus minimizing delay and converging to Kleinrock's optimal operating point. BBR estimates both the available bandwidth (BW) and the minimum round trip propagation time (RTT_{min}) on a transmission path, and it calculates the path's Bandwidth Delay Product (BDP) as

$$BDP = BW * RTT_{min}. \quad (1)$$

However, these measures cannot be obtained simultaneously due to the necessity of operating in different regions. Measuring RTT_{min} entails moving to the left from operational point (A), thereby reducing the transmission rate. Conversely, assessing the maximum available bandwidth requires temporarily overloading the network, thus shifting to the right of operational point (A). Probing for bandwidth increases the queue, whereas probing for minimum propagation time requires draining the queue.

BBR spends most of its time transmitting at a constant, estimated delivery rate, although it periodically initiates bandwidth probing and refreshes the minimum Round Trip Time. When BBR probes for bandwidth, it increases the sending rate and immediately depletes any resulting queue. The BBR sender monitors the RTT with every reception of ACK; however, periodically, it enters the PROBE RTT phase, during which the transmission rate is substantially reduced to drain the queue. This mechanism guarantees the synchronization of the PROBE RTT phase across all long-lived BBR flows over time.

The sending rate is determined purely by the bandwidth estimator (BW), while the RTT_{min} measure determines the calculated BDP, thus setting the inflight cap. The maximum inflight data are limited to 2*BDP, thus serving to accommodate delayed and aggregated ACKs [7]. Additionally, BBR maintains control variables, including the `pacing_gain`, the parameter used to multiply the observed delivery rate (BW), thus reflecting the current `sending rate`

$$\texttt{sending rate} = \texttt{pacing\_gain} * BW, \quad (2)$$

and the windows increase factor (`cwnd_gain`), which regulates BBR inflight data volume to adhere to the desired limit.

$$\texttt{current inflight} = \texttt{cwnd\_gain} * BW * RTT_{min}. \quad (3)$$

3.1. Detailed Description of BBR v1

BBR encompasses four distinct control states within a finite state machine, with each dedicated to a different purpose:

- STARTUP—initiating a search for maximum bandwidth BW;
- DRAIN—draining the built-up queue;
- PROBE BW—exploring additional bandwidth, draining the queue, and sending data using the estimated BW,
- PROBE RTT—refreshing the RTT_{min}.

The BBRv1 state machine and the conditions for transitioning between states are presented in Figure 2. The STARTUP phase is similar to the slow start phase of the TCP CUBIC. BBR seeks the maximum available bandwidth by exponentially increasing the congestion window. During this phase, the `pacing_gain` and `cwnd_gain` are set to 2.89, which restricts BBR inflight almost to 3*BDP. The STARTUP phase concludes if the estimated bandwidth fails to exceed 1.25 times the previous bandwidth for three consecutive RTTs. Upon meeting this criterion, BBR transitions to the DRAIN phase, which allows for emptying of the queue that built up during the STARTUP phase. The `pacing_gain` is set inversely to the STARTUP phase. BBR transitions from DRAIN to PROBE BW when the inflight data drop below the estimated BDP.

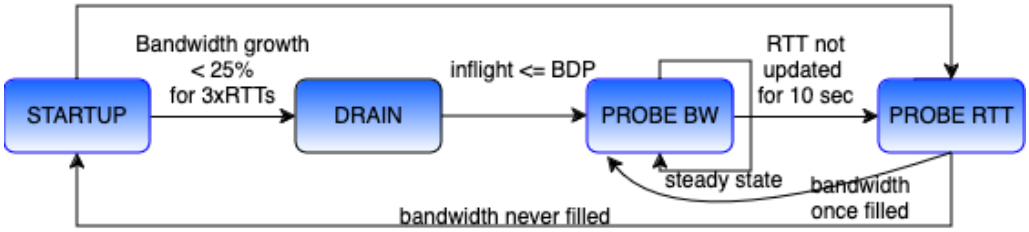


Figure 2. BBRv1 state machine and transition conditions.

Long-lived BBR flows spend most of their time in the PROBE BW phase. At every eighth RTT, the `pacing_gain` is increased to 1.25 for one RTT interval only to promptly revert to 0.75 for the subsequent RTT interval. For the remaining 6 RTTs, it maintains a consistent sending rate equivalent to the windowed maximum filter of the delivery rate from the previous 10 RTTs (`pacing_gain` = 1). Meanwhile, the `cwnd_gain` is fixed at 2 to ensure sufficient inflight data.

If RTT_{min} has not been updated for at least 10 s, BBR transitions into the PROBE RTT phase. During this phase, the congestion window is reduced to four segments, and BBR attempts to measure the shortest RTT. This phase persists for 200 milliseconds. The significant reduction in the transmission rate allows for the draining of the queue to refresh the minimum RTT. Following the PROBE RTT phase, BBR reverts to the PROBE BW phase or to STARTUP if it failed to reach the bandwidth plateau earlier.

The initial results were very promising; however, the shortcomings of BBRv1 soon became apparent. Its insensitivity to packet loss contributed to elevated drop ratios and retransmission rates, particularly in scenarios involving shallow buffers. It also became evident that this protocol exhibits a more aggressive behavior than CUBIC and fails to ensure equitable bandwidth allocation across numerous scenarios.

3.2. Updates in BBRv2

In 2019, Google announced the second iteration of BBR. BBRv2 adopts a new model to control its behavior regarding both bandwidth utilization and inflight data management. The parameters governing these adaptations are determined by congestion signals, which encompass a blend of loss and ECN signals.

BBRv2 governs the sending rate over both short and long terms. Short-term boundaries (`bw_LO, inflight_LO`) are determined using the latest delivery and loss signals and remain unchanged when BBRv2 is probing for bandwidth. Long-term values (`bw_HI, inflight_HI`) represent the maximum safe bandwidth and inflight data volume before congestion signals. BBRv2 maintains headroom, thus providing space for new flows to enter. It employs a probing approach that occurs on a time scale to facilitate coexistence

with loss-based CCs and prevents overshooting by initiating probing at a tolerable inflight level. When entering the PROBE RTT phase, BBRv2 reduces the congestion window gain to 50% of the current inflight, thereby minimizing throughput oscillations.

The key modification introduced in BBRv2 is the monitoring of the loss and/or ECN ratio. The STARTUP phase terminates if either the estimated bandwidth fails to grow beyond 1.25 times the previous bandwidth for three consecutive RTTs or if the predefined loss or ECN threshold is surpassed. Surpassing the loss/ECN threshold results in setting `inflight_HI` to the maximum safe volume of inflight data. Throughout the DRAIN phase, we decrease the sending rate until the inflight data drops below the estimated BDP, provided that adequate headroom was left if we set `inflight_HI` during STARTUP. Additional headroom comprises 15% of the `inflight_HI`.

The PROBEBW phase is now split in four additional phases: *PROBE DOWN, PROBE CRUISE, PROBE REFILL, and PROBE UP*. The state machine for the new PROBE BW phase is presented in Figure 3.

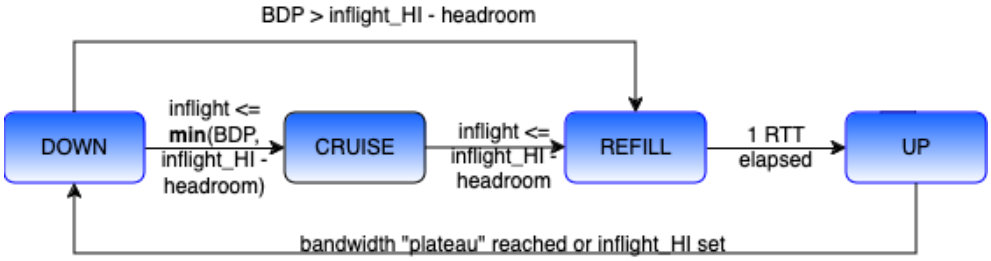


Figure 3. BBRv2 state machine of PROBE BW phase and transition conditions.

During the *PROBE DOWN* phase, the queue is emptied and unused headroom is retained, with the pacing gain set below 1 to drain excess inflight data. The transition condition remains the same as in the DRAIN phase. Long-lived BBRv2 flows spend the majority of their time in the *CRUISE* phase. The *CRUISE* phase aims to adapt to maintain a low and stable queue. The inflight data are initially set at the estimated BDP, with additional headroom left if the last probe was limited by `inflight_HI`. During this phase, the lower bounds are refreshed per RTT using loss/ECN signals.

During the *REFILL* phase, the objective is to replenish the pipeline without filling the queue. Therefore, BBR sends data at the estimated bandwidth for one RTT. We refrain from sending faster to prevent queue buildup. In the *PROBE UP* phase, we probe for additional bandwidth and increment the amount of sent data above the inflight level exponentially for each round. If the loss/ECN ratio exceeds the defined threshold, we set the ceiling of `inflight_HI` to the estimated safe volume of the inflight data. Probing concludes under the same conditions as in the STARTUP phase.

3.3. BBRv3

BBRv3 was introduced in July 2023 [16] and it officially obsoleted both previous versions, although it is still not available in any Linux kernel by default. The third version aims to reduce queuing delay and act more fairly towards loss-based CCs. The major updates of BBRv3 include fixing bugs that hindered bandwidth convergence in the presence of shallow and deep buffers and tuning the performance parameters involving gains and loss threshold.

The first major bug observed in BBRv2 interrupted bandwidth probing shortly after a loss or ECN signal was detected and the `inflight_HI` parameter was set. This issue stemmed from a circular dependency between the maximum bandwidth and maximum inflight data. Consequently, BBRv2 struggled to achieve fair bandwidth share when competing with loss-based CCs. It also hindered achieving high bandwidth utilization after congestion events. BBRv3 addresses this issue through persistent bandwidth probing, thereby continuing until either the loss rate or ECN mark rate exceeds predefined tolerance thresholds of 1%.

The second bug was observed when the buffer size exceeded 1.5*BDP, and there were no loss or congestion signals. BBRv2 encountered difficulty in achieving equitable bandwidth allocation due to a fixed `cwnd_gain`, which hindered slower flows from increasing their sending rate. BBRv3 addressed these issues by modifying the "gains" during the PROBE BW phase. During the *UP* phase, the `cwnd_gain` was raised from 2.0 to 2.25, thus enabling more effective adjustments in sending rates. Secondly, the `pacing_gain` was adjusted from 0.75 to 0.9 during the *DOWN* phase. This adjustment aids slower flows in consistently utilizing adequate bandwidth, thus improving convergence to an equitable share more swiftly.

BBRv3 has tuned several configuration parameters. During the STARTUP phase, the `cwnd_gain` was lowered from 2.89 to 2.0. This parameter significantly influences the inflight data limit during this phase. This adjustment has been motivated by [21]. Similarly, the `pacing_gain` was reduced to 2.77 based on the analytical reasoning presented in [22]. Upon exiting the STARTUP phase, `inflight_HI` is determined by selecting the maximum value between the estimated BDP and the highest number of packets successfully transmitted during the last RTT. Moreover, the transition condition has been altered, and the threshold for consecutive packet losses has been adjusted from eight to six. These modifications have led to a reduction in the experienced drop ratio and queuing delays.

4. Simulation Framework

4.1. Details of ns-3 Implementation

The full source code of the modified ns-3 simulation environment is available online [23]. The ns-3 was forked from the main branch of ns-3-dev. We adopted BBRv2 implementation from the repository (https://github.com/lanigan23/BBRv2-Eval-ns-3 (accessed on 6 May 2024) built on top of Vivek Jain's BBR adaption [24]. All simulation scripts used to produce the results are available online. The BBRv2 and BBRv3 implementations are based on the Linux kernel code from the official repository on GitHub: BBRv2 (https://github.com/google/bbr (accessed on 6 May 2024) and BBRv3 (https://github.com/google/bbr/blob/v3 (accessed on 6 May 2024).

It was also decided to address reported bugs that were found to be incompatible with the Linux implementation, thus causing BBR to inaccurately estimate the delivery rate in certain scenarios. The initial fix involved modifying the delivery rate update mechanism as described here (https://tinyurl.com/yeymtvcx (accessed on 6 May 2024). Furthermore, adjustments were made to the implementations of BBRv2 and BBRv3, thus incorporating conditions to verify rate samples in alignment with those in Linux kernel. Additionally, variables were introduced to enable the implementation of short-term and long-term inflight and bandwidth estimates, thus enhancing the original implementation, which solely relies on the windowed maximum filter of the delivery rate.

Limitations: The simulation model always entails simplifications compared to the actual implementation. In the Linux implementation, Core TCP stack informs us that the given skb was just marked lost and triggers function, which adapts the `inflight_HI` event that cannot be replicated in ns-3. Consequently, upper limits may be set too optimistically in some cases, thus leading to delayed responses to link degradation induced by either another stream or a change in conditions. Moreover, ECN support is not implemented in the current version.

4.2. Simulation Setup and Configuration

The standard dumbbell topology with two routers and six nodes (N1–N6) was utilized for all scenarios (see Figure 4). Simulation parameters common to all scenarios (S1–S4) are compiled in Table 1, with an asterisk indicating that detailed descriptions of each simulation group are provided preceding individual experiment descriptions.

TCP sources were situated in nodes N1–N3, while nodes N4–N6 functioned as data recipients. All TCP flows utilized 1000-byte packets. The total number of TCP flows, link

delays, and buffer sizes varied for each scenario group. The routers employed the drop-tail policy. Simulations were conducted for 100 s, except for the first scenario.

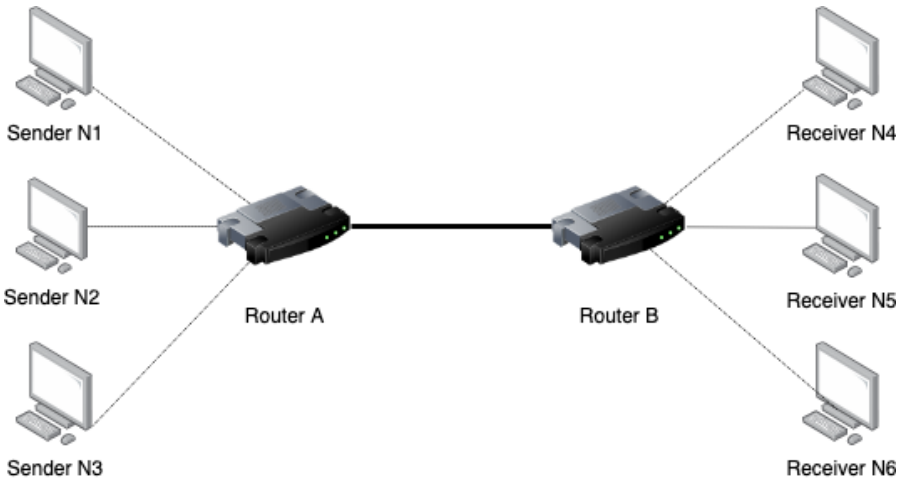


Figure 4. The dumbbell topology

Table 1. Configuration values used for simulation model, asterisk (*)—variable value, see scenario description for details.

Parameter	Value (S1/S2–S4)
Simulation time	200/100 s
Bottleneck bandwidth	10/100 Mbps
Bottleneck delay	10/*
Access link bandwidth	40/100 Mbps
Access link delay	1/*
Buffer size	2*BDP/*
Initial cwnd	10
MTU	1000
TCP algorithms	BBRv1, BBRv2, BBRv3, Cubic
Data limit	unlimited

To conduct a comprehensive comparison of the results, various metrics were collected. These included the overall throughput of the bottleneck link, the mean queue length, the standard deviation of the queue length, and the drop ratio at router A. The drop ratio refers to the proportion of dropped packets in relation to the total number of packets transmitted, and it is represented as a percentage. Additionally, the mean throughput for each individual flow was computed. To assess the fairness of link sharing among competing flows, Jain's index was calculated:

$$JI = \frac{(\sum_{i=1}^{n} x_i)^2}{n * \sum_{i=1}^{n} x_i^2}, \tag{4}$$

where x_i is the ith flow's throughput.

5. Performance Evaluation

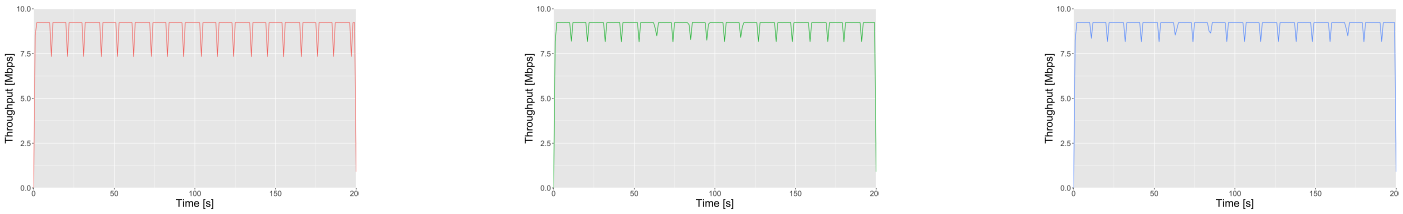
The primary objective of the simulations was to evaluate the performance of BBRv3 in terms of fair bandwidth sharing, bandwidth utilization, drop ratio, and buffer filling within multi-RTT environments. This section presents the outcomes derived from the simulations conducted across diverse network configurations.

5.1. BBR Intended Behavior

The intended behavior is observed through the transmission of a single stream. To achieve this objective, this study conducted two experiments, with each entailing the establishment of a single connection between node N1 and node N4. In both experiments,

access link delays were fixed at 2 ms, and the throughput was set at 40 Mbps, while the bottleneck exhibited a delay of 10 ms and a throughput of 10 Mbps. The buffer size was set to compensate 2*BDP of traffic. The data transmission spanned 200 s.

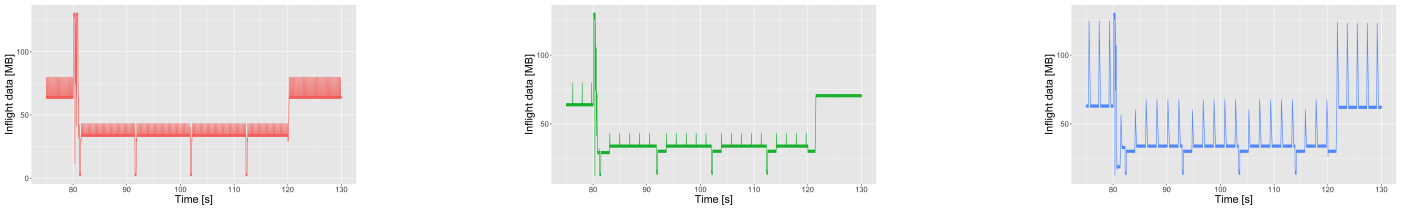
For the first experiment, the network conditions remained constant. The throughput trends over time are depicted in Figure 5. All versions of the BBR protocol are able to fully utilize the bottleneck link's capacity. A periodic reduction in throughput, occurring every 10 s, is attributed to the PROBE RTT phase. Versions 2 and 3 demonstrate significantly smaller throughput reductions, thus achieved by halving the window size relative to the current inflight.



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

Figure 5. The throughput trend over time and the steady state behavior.

In the subsequent scenario, the bottleneck bandwidth capacity alternated every 40 s, thus fluctuating between increments to 20 Mbps and decrements to 10 Mbps. Figure 6 illustrates the amount of inflight data spanning from the 75th to the 130th second, thus capturing the interval of bandwidth reduction from 20 Mbps to 10 Mbps at the 80th second and the subsequent bandwidth increase at the 120th second. Each version promptly adjusted its estimations to the altered conditions. The simulation results for BBRv1 mirror the findings outlined in [9].



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

Figure 6. The amount of inflight data for each BBR version while varying bandwidth capacity.

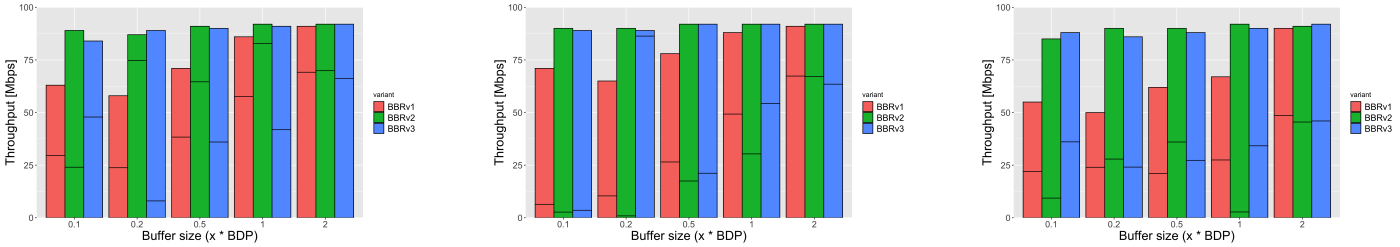
5.2. Intraprotocol Fairness

The purpose of this group of simulations was to verify if BBRv3 showed similarly strong dependency on the bottleneck buffer size, as well as unfairness towards flows with different RTTs. Our objective was to investigate whether the impact on results stems solely from the ratio of RTT times or if it was the relation of the RTT to buffer size that played a significant role for a part of the bandwidth that the flow achieves. Across both sets of simulations, the connection between node N1 and N4 consistently maintained an RTT of 40 ms (*baseRTT*), while the throughput of all the links remained fixed at 100 Mbps. Buffer sizes were determined as multiples of BDP, relative to the baseRTT, and were calculated as the $BDP = baseRTT * 100\ Mbps$.

This work varied the RTT between node N2 and N5, thus ensuring that the delay on the N2–N5 path spanned increments of 0.1, 0.2, 0.5, 0.8, 1, 2, and 5 times the baseRTT. It employed two distinct settings for the bottleneck delay: 2 ms and 10 ms, depending on the required RTT ratio. Simulations were conducted across a spectrum of buffer sizes ranging from 0.1 to 10 times BDP.

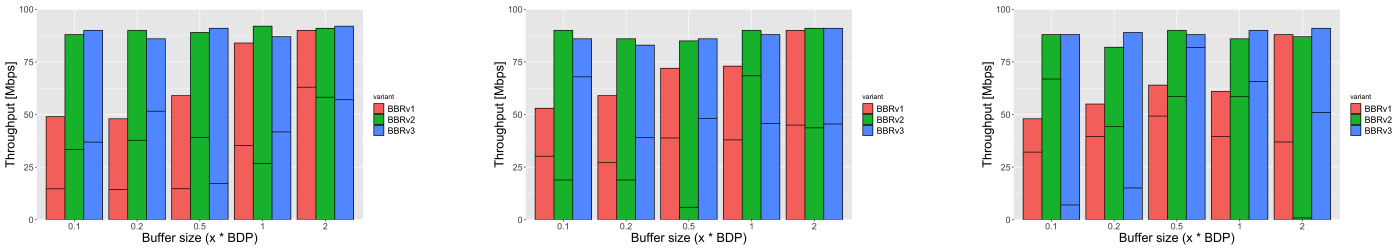
The overall throughput is depicted in Figures 7–9. Each bar is divided to represent the throughput experienced by individual flows, with the lower section consistently indicating the throughput of the flow with *baseRTT*. Given the substantial volume of results acquired,

this study restricted the presented results to configurations where differences were evident; in certain instances, the outermost buffer sizes did not yield significant alterations in outcomes. The Jain's index data is collated in Tables 2 and 3.



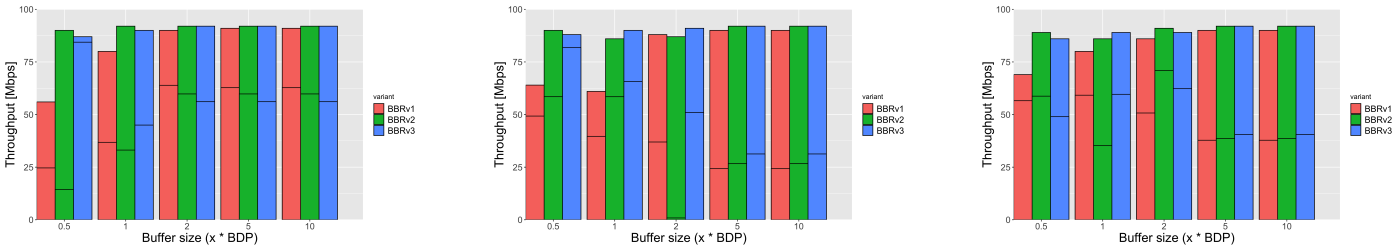
(**a**) 0.1*baseRTT (**b**) 0.2*baseRTT (**c**) 0.8*baseRTT

Figure 7. The flow throughputs (bottom—baseRTT, top—x*baseRTT) for different buffer sizes and different RTT ratios; bottleneck link delay = 2 ms.



(**a**) 0.5*baseRTT (**b**) 1*baseRTT (**c**) 2*baseRTT

Figure 8. The flow throughputs (bottom—baseRTT, top—x*baseRTT) for different buffer sizes and different RTT ratios; bottleneck link delay = 2 ms.



(**a**) 0.5*baseRTT (**b**) 2*baseRTT (**c**) 5*baseRTT

Figure 9. The flow throughputs (bottom—baseRTT, top—x*baseRTT) for different buffer sizes and different RTT ratios; bottleneck link delay = 10 ms.

For smaller buffers, BBRv1 failed to fully utilize the available bandwidth, thus requiring a buffer size of at least 2*BDP to achieve 90% bandwidth utilization. In contrast, both BBRv2 and BBRv3 did not exhibit these limitations and consistently delivered high throughputs across various buffer sizes. This is attributed to their ability to respond to packet losses by dynamically adjusting the sending rate. This work has not presented the results of the drop ratio, as they merely confirm previous findings that BBRv1 exhibited a packet loss ratio of up to 5% for shallow buffers. The observation of a high loss ratio (and consequently, a high retransmission rate) is in line with the findings of prior work [6,9,11].

When RTTs are equal, the fairness depends on the buffer size. BBRv1 exhibited the most aggressive probing behavior and facilitated rapid synchronization; thus, even with very shallow buffers, it provided high fairness. However, this came at the expense of bandwidth underutilization and a high loss/retransmission ratio. BBRv2 and BBRv3 required a buffer size of at least 1–2*BDP to ensure equitable operation. BBRv2 struggled the most to achieve fairness when the buffer was shallow. This issue may be attributed to a known bug in this iteration, which was fixed with BBRv3 (the details are provided in Section 3.3).

The buffer size and disparities in the RTT significantly influence bandwidth allocation between flows. In the case of BBRv1, buffers smaller than 1*BDP typically resulted in shorter RTT streams obtaining more bandwidth. However, it was only when the buffer exceeded 2*BDP that a notable disparity emerged in favor of streams with longer RTT times.

These findings present a partial contradiction to the assertions found in the existing literature. The results presented in [11,12,18] suggest that long RTT flows dominate over short RTT flows; however, in some of these findings, we can observe that the bias towards long RTT flows is less pronounced in shallow buffers. The problem lies in the lack of clarity regarding the precise buffer size, which is typically provided in terms of multiples of BDP. Do we calculate it using the shortest, the longest, or the mean RTT of all paths? With large differences between RTT times, in very shallow buffers, streams with shorter RTTs are indeed able to achieve better throughput. In particular, short RTT BBRv1 flows, which ignore packet loss signals, update their *BW* estimates faster and maintain a relatively higher sending rate.

This study would anticipate that scenarios with 0.5*baseRTT would demonstrate an inverse pattern compared to those with 2*baseRTT. However, this trend was only observable for sufficiently large buffers of 2*BDP and above. As the buffer size decreased, this correlation diminished, thus confirming that bandwidth allocation is influenced not only by the discrepancies in absolute RTTs but also by the relationship of the maximum RTT to the buffer size. The BBRv1 streams occupied a larger share of the bandwidth in shallow buffer scenarios. This dependency was less pronounced with BBRv3.

Table 2. Jain's index for different buffer settings and RTT ratios for all BBR versions; bottleneck link delay = 2 ms.

		0.1*bRTT	**0.2*bRTT**	**0.5*bRTT**	**0.8*bRTT**	**1*bRTT**	**2*bRTT**
0.1*BDP	BBRv1	1.00	0.60	0.86	0.96	0.98	0.90
	BBRv2	0.82	0.53	0.95	0.62	0.75	0.79
	BBRv3	0.98	0.54	0.97	0.97	0.74	0.59
0.2*BDP	BBRv1	0.97	0.69	0.87	1.00	0.99	0.84
	BBRv2	0.65	0.51	0.97	0.87	0.76	1.00
	BBRv3	0.60	0.53	0.96	0.84	1.00	0.70
0.5*BDP	BBRv1	0.99	0.90	0.80	0.90	0.99	0.77
	BBRv2	0.84	0.73	0.99	0.96	0.57	0.91
	BBRv3	0.96	0.77	0.72	0.87	0.99	0.58
1*BDP	BBRv1	0.90	0.99	0.98	0.97	1.00	0.92
	BBRv2	0.60	0.90	0.85	0.54	0.79	0.89
	BBRv3	0.99	0.97	1.00	0.94	1.00	0.82
2*BDP	BBRv1	0.79	0.81	0.86	0.99	1.00	0.97
	BBRv2	0.79	0.83	0.93	1.00	1.00	0.52
	BBRv3	0.84	0.87	0.95	1.00	1.00	0.98

Colors are used to emphesize the lowest and highest values.

Summary: With varying path lengths, bandwidth sharing relies less on the ratio of RTT times and more on the buffer size calculated in relation to the RTT of the path. This dependency was particularly evident with BBRv1, while BBRv3 partially mitigated this correlation. If the disparity in the RTT among flows is significant, a buffer size of at least 1–2*BDP calculated using the longest RTT should be provided to ensure high fairness.

Table 3. Jain's index for different buffer settings and RTT ratios for all BBR versions; bottleneck link delay = 10 ms.

		0.5*bRTT	1*bRTT	2*bRTT	5*bRTT
2*BDP	BBRv1	0.86	1.00	0.97	0.97
	BBRv2	0.92	1.00	0.52	0.76
	BBRv3	0.96	1.00	0.98	0.86
5*BDP	BBRv1	0.87	1.00	0.82	0.97
	BBRv2	0.92	1.00	0.85	0.97
	BBRv3	0.96	1.00	0.91	0.99
10*BDP	BBRv1	0.87	1.00	0.82	0.97
	BBRv2	0.92	1.00	0.85	0.97
	BBRv3	0.96	1.00	0.91	0.99

Colors are used to emphesize the lowest and highest values.

5.3. RTT Fairness and Access Link Scenario

In this scenario, each of the nine paths had a unique RTT. All link rates were set to 100 Mbps, while the propagation delay between routers A and B was set to 2 ms. The one-way propagation delays were distributed across the range from 0 ms to 75 ms. The RTT values for all connections are compiled in Table 4, thus aligning with the representative distribution of RTTs on the central link, as reported by [25]. Buffer sizes were determined as multiples of BDP and relative to the mean RTT, thus resulting in a size of 900 packets. The flows were initiated randomly within the first second of the simulation.

Table 4. Values of RTT [ms] between all node pairs.

N1–N4	8	N2–N4	32	N3–N4	58
N1–N5	78	N2–N5	102	N3–N5	128
N1–N6	154	N2–N6	178	N3–N6	204

The disparity in throughput among the various flows was considerable, and it mainly depended on both the RTT and the buffer size. The buffer size, denoted as a multiplication of the BDP, is calculated in reference to the average RTT, which was set at 100 ms. Therefore, for very short and very lengthy paths, it becomes considerably deeper or shallower.

The overall throughput, Jain's index, drop ratio, and average queue size with the standard deviation of the queue size for all simulation runs are presented in Table 5.

Table 5. (1) Overall throughput [Mbps]. (2) Jain's index. (3) Drop ratio [%]. (4) Queue size [pkts] and standard deviation of queue size for different buffer settings.

		(1)	(2)	(3)	(4)
0.5*BDP	BBRv1	90	0.84	4.55	279 [185]
	BBRv2	92	0.51	0.29	347 [132]
	BBRv3	92	0.70	0.70	238 [114]
1*BDP	BBRv1	92	0.95	4.22	637 [310]
	BBRv2	93	0.54	0.27	601 [316]
	BBRv3	93	0.62	0.29	612 [255]
2*BDP	BBRv1	93	0.98	2.56	1218 [615]
	BBRv2	93	0.70	0.21	1073 [492]
	BBRv3	93	0.81	0.11	521 [251]
10*BDP	BBRv1	91	0.49	0.00	2448 [883]
	BBRv2	93	0.80	0.48	2534 [1512]
	BBRv3	94	0.86	0.04	5308 [1993]

Colors are used to emphesize the lowest and highest values.

The throughput results for all BBRv1 flows are presented in Table 6. BBRv1 provided very high fairness when the buffer size ranged between 1 and 2*BDP. Consistently across

these scenarios, the packet loss ratio remained notably high. Analysis of both the average queue length and the standard deviation reveals considerable oscillations induced by BBRv1, thus contributing to a lack of queue stability.

In scenarios with a buffer size of 0.5*BDP, streams with the shortest RTTs tend to monopolize the majority of available bandwidth—three flows utilized almost 50% of the available capacity. This phenomenon occurs because these streams possess relatively larger buffer shares. With an increase in buffer size, the distribution of bandwidth shifts towards connections characterized by longer RTTs. For a buffer size of 10*BDP, streams with extended RTTS emerged as dominant, with the three longest-duration streams collectively occupying over 70% of the available bandwidth. This outcome aligns with findings from prior studies, thus affirming that in scenarios featuring sufficiently large buffers, streams with longer RTTs tend to consume the majority of bandwidth, which is attributable to an upper limit of inflight data calculated at twice the estimated BDP.

Table 6. Throughput of BBRv1 flows [Mbps] and overall Jain's index (JI) for access link scenario.

Buffer Size	0.5*BDP	BDP	2*BDP	10*BDP
RTT [ms] | JI	0.84	0.95	0.98	0.49
8	19	15	12	5
32	14	12	10	2
58	11	12	12	2
78	10	11	10	2
102	9	10	9	3
128	8	9	12	8
154	7	10	10	13
178	5	6	9	32
204	5	8	9	26

The throughput results for BBRv2 are presented in Table 7. The performance of BBRv2 significantly degraded in terms of fairness. In scenarios involving smaller buffers, there persistently existed a single connection with an RTT below the average, thus consuming approximately 30% of the available bandwidth. Unlike BBRv1, no evident trend emerged regarding the excessive allocation of bandwidth by streams with shorter RTTs for small buffer sizes. This discrepancy can be attributed to the absence of a limiting factor calculated using the estimated bandwidth and propagation time.

Table 7. Throughput of BBRv2 flows [Mbps] and overall Jain's index (JI) for access link scenario.

Buffer Size	0.5*BDP	BDP	2*BDP	10*BDP
RTT [ms] | JI	0.51	0.54	0.70	0.80
8	34	10	14	5
32	6	11	10	10
58	12	8	8	7
78	8	36	24	1
102	4	6	7	13
128	1	6	9	14
154	23	12	19	16
178	5	1	2	10
204	2	4	2	19

As the buffer size increased, there was an observable trend towards a fairer distribution of bandwidth. However, in scenarios with the largest buffer, BBRv2 exhibited the same behavior as the previous version: the three streams with the longest RTTs collectively consumed a large part of the bandwidth. It is also worth mentioning that in an analogous scenario, CUBIC achieved a JI of 0.7 when the buffer size was equal to the BDP [5].

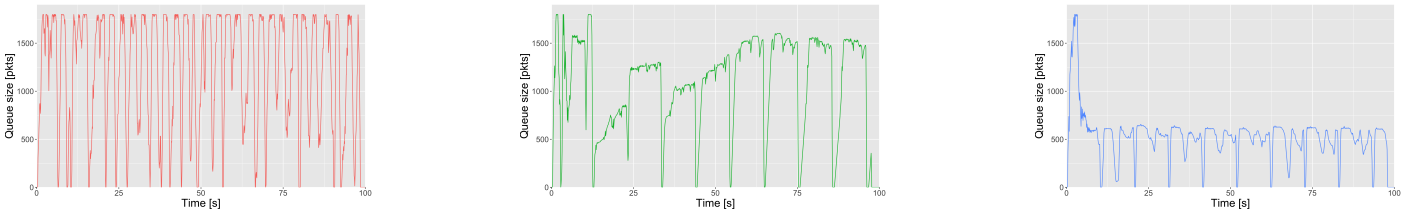
The results for BBRv3 (see Table 8) are the most inconclusive. This version demonstrated optimal performance when the buffer size equaled 2*BDP, although the flow with

the shortest RTT allocated 25% of the bandwidth, and the Jain's index was much lower than for the first version. Worse fairness can result from less aggressive probing parameters compared to the first version. Notably, only the average queue size and standard deviation exhibited significantly lower values compared to previous iterations. On the other hand, we can observe that for the deepest buffer, the average queue size became notably large.

To delve deeper into this phenomenon, this study presents the queue length trends across different iterations of the BBR algorithm for the two largest buffer settings. The results of this analysis are presented in the Figures 10 and 11.

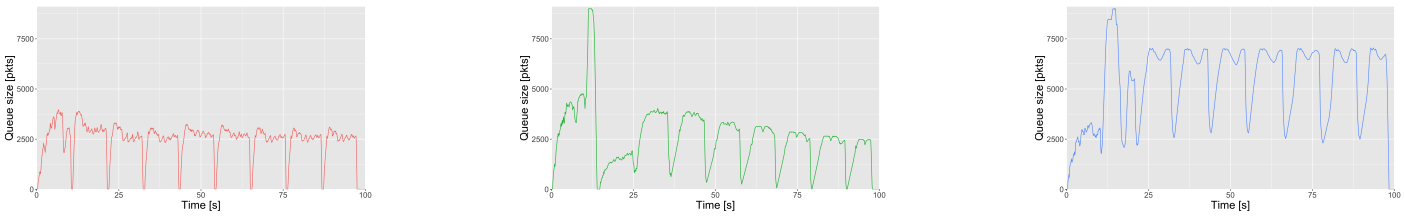
Table 8. Throughput of BBRv3 flows [Mbps] and overall Jain's index (JI) for access link scenario.

Buffer Size	0.5*BDP	BDP	2*BDP	10*BDP
RTT [ms] | JI	0.70	0.62	0.81	0.86
8	18	20	24	10
32	10	10	11	13
58	2	6	11	18
78	23	2	9	12
102	2	3	10	12
128	3	21	8	10
154	12	23	8	5
178	11	3	9	3
204	12	5	5	13



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

Figure 10. The queue length for all BBR versions in access link scenario; buffer size = 2*BDP.



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

Figure 11. The queue length for all BBR versions in access Link scenario, buffer size = 10*BDP.

BBRv1, despite providing a high fairness index, induced significant queue oscillations. Only for a very deep buffer did the queue achieve stability, thus remaining below an occupancy threshold of 5*BDP. During the PROBE RTT phases of BBR, synchronization occurs, thus leading to buffer depletion during these intervals. Conversely, BBRv2 and BBRv3 offered improved queue stability, and periodic buffer emptying demonstrated the synchronization of flows during the PROBE RTT phases, thus accurately estimating the minimum RTT values of their respective paths.

BBRv3 with a buffer size of 2*BDP maintained a stable queue at approximately one-fourth of its nominal size following the initial STARTUP phase, which is the expected and desired behavior. However, for a buffer size of 10*BDP, after the STARTUP and DRAIN phases, the remaining queue was too long. Subsequent PROBE UP phases aimed at probing for more bandwidth resulted in the buffer remaining persistently full, thereby impairing the effectiveness of the PROBE RTT phase and the overall algorithm operation.

BBRv3's performance in deep buffers should be further investigated. It is conceivable that the issue stems from a simplified simulation model; however, in [18], we can observe that in very deep buffers, the fairness provided by BBRv3 dropped, which may confirm the problem of BBRv3 depleting the deep queue.

Summary: In scenarios featuring multiple paths with varying RTT times, BBRv3 maintained a stable and low queue when the available buffer size was equal to 2*BDP. Unfortunately, the protocol fell short in ensuring equitable bandwidth distribution among active flows, thus leading to a fairness lower than that of the initial iteration.

5.4. Interprotocol Fairness

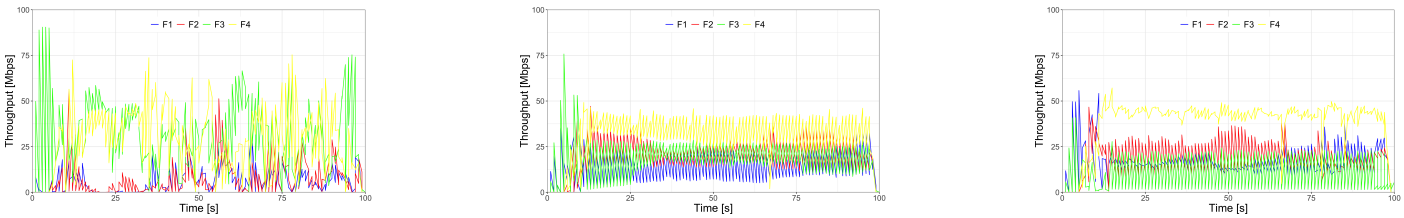
In the final scenario, this study investigated the interaction between BBR and the TCP CUBIC. A CUBIC sender was deployed on node N1, thus initiating two flows denoted as F1 and F2, while on node N2, a BBR sender transmitted data via two flows labeled as F3 and F4. The recipient for the data streams F1 and F2 was implemented at node N4, whereas the recipient for the F3 and F4 flows was located at node N5. The transmission performance was evaluated under two distinct scenarios: one where all paths were set to an identical RTT of 40 ms and another where the path N1–N4 had an RTT of 40ms, while the path N2–N5 had an RTT of 80 ms. The start times for flows F1 and F3 were randomized around the beginning of the simulation, while flows F2 and F4 were randomized to initiate transmission approximately at the 5th second. The simulations were executed with two different buffer sizes, namely 0.5 times and 5 times the BDP. The BDP was computed based on the *baseRTT* of the N1–N4 path.

The results are compiled in Table 9. The throughput for each TCP version was calculated for individual flows, thus considering their start and stop times. The throughput for individual flows is depicted in Figures 12 and 13 for flows sharing equal RTT, while for two different RTTs, the throughput is presented in Figures 14 and 15.

Table 9. (1) Overall throughputs for CUBIC and BBR flows [Mbps]. (2) Jain's index. (3) Drop ratio [%] for scenario with 4 flows; two buffer sizes: shallow (SB) and deep (DB) and the same or different RTT on paths.

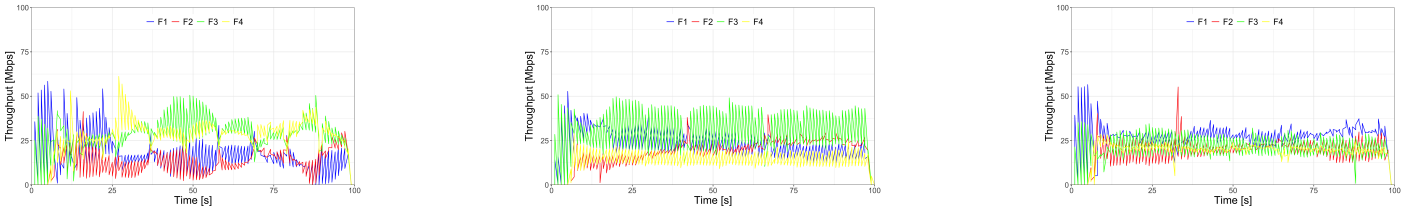
		RTT = 40 ms			**RTT = 40/80 ms**		
		(1)	**(2)**	**(3)**	**(1)**	**(2)**	**(3)**
SB	BBRv1	15/65	0.72	1.80	27/57	0.86	1.64
	BBRv2	52/43	0.95	0.31	61/35	0.86	0.24
	BBRv3	51/46	0.75	0.25	46/49	0.97	0.13
DB	BBRv1	30/65	0.87	0.60	34/60	0.92	0.84
	BBRv2	41/55	0.80	0.06	51/46	0.79	0.22
	BBRv3	46/50	0.95	0.12	41/55	0.69	0.18

Colors are used to emphesize the lowest and highest values.



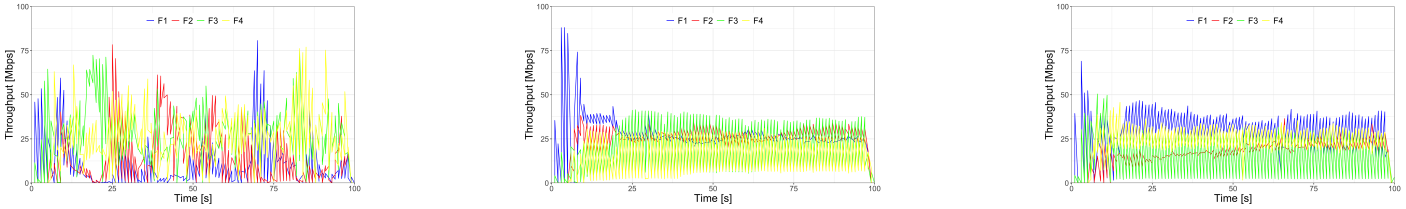
(**a**) BBRv1 (**b**) BBRv2 (**c**) BBRv3

Figure 12. The flow throughputs: F1 and F2—CUBIC; RTT = 40 ms; F3 and F4—BBR; RTT = 40 ms; shallow buffer (0.5*BDP). F1 and F3 start at 0; F2 and F4 start at 5 s.



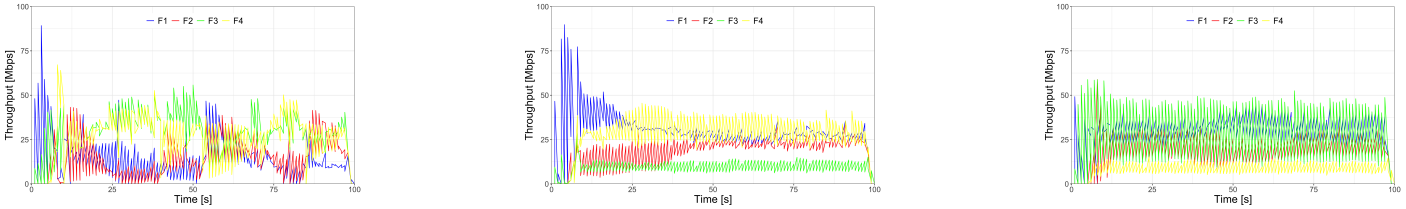
(**a**) BBRv1 (**b**) BBRv2 (**c**) BBRv3

Figure 13. The flow throughputs: F1 and F2—CUBIC; RTT = 40 ms; F3 and F4—BBR; RTT = 40 ms; deep buffer (5*BDP). F1 and F3 start at 0; F2 and F4 start at 5 s.



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

Figure 14. The flow throughputs: F1 and F2—CUBIC; RTT=40ms; F3 and F4—BBR; RTT = 80 ms; shallow buffer (0.5*BDP). F1 and F3 start at 0; F2 and F4 start at 5 s.



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

Figure 15. The flow throughputs: F1 and F2—CUBIC; RTT = 40 ms; F3 and F4—BBR; RTT = 80 ms; deep buffer (5*BDP). F1 and F3 start at 0; F2 and F4 start at 5 s.

Similarly to the previous scenarios, BBRv1, characterized by significant throughput oscillations, impaired the full utilization of the available bandwidth and induced the highest drop ratio in scenarios with shallow buffers. Notably, it monopolized nearly 80% of the bandwidth. The additional buffer capacity allowed CUBIC to claim more bandwidth; however, the BBRv1 flows continued to control over two-thirds of the bandwidth share. The results confirm the widely accepted assertion regarding BBRv1's unfairness toward CUBIC. As the buffer size increased, CUBIC transmitted more data before encountering loss, thus consequently achieving a higher throughput.

BBRv2 and BBRv3 significantly modified the allocation of bandwidth resources. BBRv3 exhibited a nearly equitable distribution of bandwidth capacity between congestion controls in almost all cases. In several experiments, even when the bandwidth was distributed fairly among the CUBIC and BBR flows, the Jain's Index (JI) indicated suboptimal fairness. The high or low values of the Jain's Index cannot be solely attributed to BBR, as CUBIC also played a significant role in this regard. However, upon analyzing the throughputs of the individual flows, it becomes apparent that BBR struggled more to equitably distribute its allocated bandwidth share, particularly in scenarios with deep buffers where earlier-starting flows consumed a disproportionate share of the bandwidth. These findings may suggest the prolonged convergence time of BBRv3 towards achieving fair bandwidth allocation.

For shallow buffers, BBRv1's indifference to packet losses provided it with an advantage over CUBIC. BBRv3 operated around the estimated BDP without causing additional losses, thereby facilitating the swift loss recovery performed by CUBIC. In the case of deep buffers, CUBIC tended to fill the entire available buffer space, whereas BBRv3 maintained

the volume of inflight data close to the BDP. The bufferbloat caused by CUBIC impaired the proper operation of BBR.

Summary: BBRv3 improved fair bandwidth sharing in the presence of CUBIC, especially when all flows shared identical RTTs. However, when confronted with varying RTTs and deep buffer configurations, BBRv3 exhibited an extended convergence time towards achieving a fair bandwidth distribution.

6. Conclusions and Future Work

This paper has presented the results of an extensive evaluation of all BBR iterations, thus analyzing network performance metrics, including throughput, loss ratio, and intra- and interprotocol fairness in scenarios with various distributions of RTTs. This study verified whether the introduced amendments in BBRv3 achieved their intended objectives, which namely include the following:

- Enhanced convergence in shallow buffers following packet losses;
- Improved convergence in deep buffers in the absence of loss signals;
- Reduced queuing delay irrespective of buffer size.

This study did not observe the anticipated enhanced convergence in the recent version. While BBRv3 did enhance fairness compared to BBRv2, in many cases, BBRv1 yielded results that were at least as satisfactory, and in numerous instances, it exhibited a 10–20% higher fairness index. The sole advantage of this version over the original is a more stable connection and a negligible loss ratio.

BBRv3 generally improved in maintaining stable and low queues. However, further investigation is required regarding buffer occupancy when the size equals or exceeds 10*BDP, particularly concerning performance in scenarios involving numerous flows across paths with significantly divergent RTT values. Nevertheless, a buffer size of at least 2*BDP is recommended when deploying BBRv3. With 2*BDP settings, BBRv3 demonstrates nearly a 60% improvement in the average queue size compared to the first version and approximately a 50% improvement compared to the second.

BBRv3 enhanced bandwidth availability for CUBIC streams, especially when the paths' RTTs were similar. In a multi-RTT network with shallow buffers, BBRv3 enhanced fairness by 12%. In deep buffer scenarios, the bandwidth sharing among combined CUBIC and BBR flows was improved by 5%. However, further optimization is necessary to ensure equitable bandwidth allocation between streams and to enhance the convergence speed. The fairness of BBRv3 depends solely on the buffer size. Moreover, additional optimization is needed to improve fairness in networks with varying RTTs. Unfortunately, this challenge is not unique to BBR; none of the currently employed CC algorithms yielded satisfactory results in this regard.

BBRv2 and BBRv3 significantly enhanced bandwidth utilization compared to the initial version, particularly in scenarios with very short buffers. Depending on the RTT ratio, the improvement was as high as 80%. Moreover, in these scenarios, both versions effectively mitigated the issue of excessive loss ratio, thus reducing it by a factor of 10–20.

A comprehensive evaluation of the TCP necessitates the analysis of results across a wide range of diverse scenarios. This paper specifically tackles the issue of fairness to other flows and the TCP CUBIC in networks characterized by varying RTTs. However, further research is necessary to explore the following areas: lossy networks, enhancing performance through ECN and/or AQM, assessing performance in networks with distinct conditions such as Wi-Fi and cellular networks, conducting additional scenarios involving CUBIC, varying the number of flows, and analyzing diverse traffic patterns. The author believes that the simulation environment provided with this paper will facilitate extensive research and subsequent enhancements in this domain.

Funding: The work was carried out within the statutory research project of the Department of Computer Networks and Systems.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original data presented in the study are openly available in GitLab at https://gitlab.com/agnieszka.brachman/ns-3-dev/-/blob/dev-bbrv3-fixed (accessed on 6 May 2024).

Conflicts of Interest: The author declares no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AQM	Active Queue Management
BBR	Bottleneck Bandwidth and Round Trip propagation time
BDP	Bandwidth Delay Product
CC	Congestion Control
ECN	Explicit Congestion Notification
RTT	Round Trip Time
TCP	Transmission Control Protocol

References

1. Kleinrock, L. Power and deterministic rules of thumb for probabilistic problems in computer communications. In Proceedings of the ICC 1979; International Conference on Communications, Boston, MA, USA, 10–14 June 1979; Volume 3, pp. 43–51.
2. Jaffe, J. Flow control power is nondecentralizable. *IEEE Trans. Commun.* **1981**, *29*, 1301–1306. [CrossRef]
3. Nichols, K.; Jacobson, V. Controlling Queue Delay. *Queue* **2012**, *10*, 20–34. [CrossRef]
4. Grazia, C.A.; Patriciello, N.; Klapez, M.; Casoni, M. Transmission Control Protocol and Active Queue Management together against congestion: Cross-comparison through simulations. *Simulation* **2019**, *95*, 979–993. [CrossRef]
5. Piotrowska, A. On Cross-Layer Interactions for Congestion Control in the Internet. *Appl. Sci.* **2021**, *11*, 7808. [CrossRef]
6. Cardwell, N.; Cheng, Y.; Gunn, C.S.; Yeganeh, S.H.; Swett, I.; Iyengar, J.; Vasiliev, V.; Jacobson, V. BBR Congestion Control: IETF 100 Update: BBR in shallow buffers. In Proceedings of the IETF-100. Internet Engineering Task Force, Singapore, 13 November 2017.
7. Cardwell, N.; Cheng, Y.; Gunn, C.S.; Yeganeh, S.H.; Jacobson, V. BBR: Congestion-Based Congestion Control: Measuring bottleneck bandwidth and round-trip propagation time. *Queue* **2016**, *14*, 20–53. [CrossRef]
8. Ma, S.; Jiang, J.; Wang, W.; Li, B. Towards RTT Fairness of Congestion-Based Congestion Control. *arXiv* **2017**, arXiv:1706.09115.
9. Scholz, D.; Jaeger, B.; Schwaighofer, L.; Raumer, D.; Geyer, F.; Carle, G. Towards a Deeper Understanding of TCP BBR Congestion Control. In Proceedings of the 2018 IFIP Networking Conference (IFIP Networking) and Workshops, Zurich, Switzerland, 14–16 May 2018; pp. 1–9. [CrossRef]
10. Nandagiri, A.; Tahiliani, M.P.; Misra, V.; Ramakrishnan, K.K. BBRvl vs BBRv2: Examining Performance Differences through Experimental Evaluation. In Proceedings of the 2020 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN), Orlando, FL, USA, 13–15 July 2020; pp. 1–6. [CrossRef]
11. Hock, M.; Bless, R.; Zitterbart, M. Experimental evaluation of BBR congestion control. In Proceedings of the 2017 IEEE 25th International Conference on Network Protocols (ICNP), Toronto, ON, Canada, 10–13 October 2017; pp. 1–10. [CrossRef]
12. Ware, R.; Mukerjee, M.K.; Seshan, S.; Sherry, J. Modeling BBR's Interactions with Loss-Based Congestion Control. In Proceedings of the Internet Measurement Conference—IMC '19, New York, NY, USA, 27–29 March 2019; pp. 137–143. [CrossRef]
13. Cardwell, N.; Cheng, Y.; Yeganeh, S.H.; Swett, I.; Vasiliev, V.; Jha, P.; Seung, Y.; Mathis, M.; Jacobson, V. BBRv2: A model-based congestion control. In Proceedings of the IETF 104th Meeting, Internet Engineering Task Force, Prague, Czech Republic, 23–29 March 2019.
14. Kfoury, E.F.; Gomez, J.; Crichigno, J.; Bou-Harb, E. An emulation-based evaluation of TCP BBRv2 alpha for wired broadband. *Comput. Commun.* **2020**, *161*, 212–224. [CrossRef]
15. Song, Y.J.; Kim, G.H.; Mahmud, I.; Seo, W.K.; Cho, Y.Z. Understanding of BBRv2: Evaluation and Comparison with BBRv1 Congestion Control Algorithm. *IEEE Access* **2021**, *9*, 37131–37145. [CrossRef]
16. Cardwell, N.; Cheng, Y.; Yang, K.; Morley, D.; Yeganeh, S.H.; Jha, P.; Seung, Y.; Jacobson, V.; Swett, I.; Wu, B.; et al. BBRv3: Algorithm Bug Fixes and Public Internet Deployment. In Proceedings of the IETF-117. Internet Engineering Task Force, San Francisco, CA, USA, 26 July 2023.
17. Cardwell, N.; Cheng, Y.; Yang, K.; Morley, D.; Yeganeh, S.H.; Jha, P.; Seung, Y.; Jacobson, V.; Swett, I.; Wu, B.; et al. BBR Congestion Control: Fundamentals and Updates. 2023. Available online: https://research.cec.sc.edu/files/cyberinfra/files/BBR%20-%20Fundamentals%20and%20Updates%202023-08-29.pdf (accessed on 6 May 2024).
18. Zeynali, D.; Weyulu, E.N.; Fathalli, S.; Chandrasekaran, B.; Feldmann, A. Promises and Potential of BBRv3. In Proceedings of the Passive and Active Measurement: 25th International Conference, PAM 2024, Virtual Event, 11–13 March 2024; Proceedings, Part II; Springer Nature: Cham, Switzerland, 2024; pp. 249–272. [CrossRef]
19. Gomez, J.; Kfoury, E.F.; Crichigno, J.; Srivastava, G. Evaluating TCP BBRv3 performance in wired broadband networks. *Comput. Commun.* **2024**, *222*, 198–208. [CrossRef]

20. Brown, P. Resource sharing of TCP connections with different round trip times. In Proceedings of the Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064), Tel Aviv, Israel, 26–30 March 2000; Volume 3, pp. 1734–1741. [CrossRef]
21. Swett, I.; Google BBR Team. BBR Startup Cwnd Gain: A Derivation. 2018. Available online: https://github.com/google/bbr/blob/master/Documentation/startup/gain/analysis/bbr_startup_cwnd_gain.pdf (accessed on 6 May 2024).
22. Google BBR Team. BBR Startup Pacing Gain: A Derivation. 2018. Available online: https://github.com/google/bbr/blob/master/Documentation/startup/gain/analysis/bbr_startup_gain.pdf (accessed on 6 May 2024).
23. Piotrowska, A. Simulation Framework Repository. 2024. Available online: https://gitlab.com/agnieszka.brachman/ns-3-dev/-/blob/dev-bbrv3-fixed (accessed on 6 May 2024).
24. Jain, V.; Mittal, V.; Tahiliani, M.P. Design and implementation of TCP BBR in ns-3. In Proceedings of the 2018 Workshop on Ns-3, New York, NY, USA, 13–14 June 2018; WNS3 '18; pp. 16–22. [CrossRef]
25. Andrew, L.; Marcondes, C.; Floyd, S.; Dunn, L.; Guillier, R.; Gang, W.; Eggert, L.; Ha, S.; Rhee, I. Towards a Common TCP Evaluation Suite. In Proceedings of the PFLDnet 2008, Manchester, UK, 5–7 March 2008.