JCBI
Journal of Computing & Biomedical Informatics
ISSN: 2710 - 1606
Volume 09 Issue 02
2025

Research Article
https://doi.org/10.56979/902/2025

Performance Evaluation of BBR-v3 with Cubic and Reno in a Ubiquitous Wired/Wi-Fi Channel

Muhammad Ahsan[1*], Muhammad Nabeel[1], Muhammad Mustansar Ali Khan[1], Muhammad Zulkifl Hasan[2], and Waqar Ashiq[1]

[1]Department of Software Engineering, University of Management and Technology, Lahore, Punjab, Pakistan.
[2]Department of Computer Science, University of Central Punjab, Lahore, Pakistan.
[*]Corresponding author Muhammad Ahsan. Email: mahsan@alumni.usc.edu

Received: June 02, 2025 **Accepted:** August 08, 2025

Abstract: Google has proposed a revision of the TCP congestion control algorithm (CCA), Bottleneck Bandwidth, and Round-trip propagation time (BBR). BBR v3 has proven to be a game-changer in the field of congestion control algorithms. BBR is a hybrid and a model-based congestion control algorithm, unlike pure loss-based CCAs such as Cubic and Reno. BBR builds a model of the network pipe by measuring the bottleneck bandwidth, round-trip propagation delays, and packet loss rate, and using newer explicit congestion notification (ECN) mechanisms. This model helps BBR to operate at Kleinrock's optimal operating point to achieve maximum bandwidth and lower latency. BBR v3 is a minor update over BBR v2, addressing the issues of fairness with Cubic and Reno flows. It also tries to revise the startup and congestion window (CWND) gains, along with performance tunings of the probing for bandwidth phases (ProbeBW). In this paper, we have evaluated the performance of BBR v3 with popular loss-based algorithms such as Cubic and Reno in a typical wired and wireless environment. A real-time physical testbed is used to test the performance of the three popular CCAs. Various tests have been performed to measure the throughput, latency, pacing rate, and CWND statistics. From the results of our thorough testing, we can conclude that BBR v3 gives superior performance both in terms of throughput as well as latency in the wireless scenario and with appreciably decreased latency in the wired Ethernet scenario when compared with Cubic/Reno.

Keywords: BBR; Congestion Control; AQM; Latency; Fairness; Pacing Rate; RTT; TSO; TSQ

1. Introduction

TCP (Transmission Control Protocol) and the congestion control algorithms were there before the launch of the World Wide Web (WWW) in 1990. TCP on the transport layer is responsible for providing reliable, in-order, byte-streams to the applications. Congestion control is an important part of the TCP stack. The Congestion Control Algorithms (CCA) have been evolving since 1998. We had TCP Tahoe [1], TCP Reno/NewReno, TCP Vegas [2], Compound TCP (CTCP) [3] from Microsoft corporation for its Windows-based operating system (OS), TCP Cubic [4] was introduced

for Linux in 2008 and gained a lot of popularity because of its better performance. It is still being used widely in Windows and Linux endpoints. These CCAs were mostly based on detecting congestion based on packet loss and duplicate acknowledgments. Hence, it they come under the category of pure loss-based algorithms.

In 2016, came up with a new algorithm known as Bottleneck Bandwidth and Round-Trip-Time (BBR). It was different from pure loss-based CCAs in the way that it didn't rely on packet loss only to detect congestion. The Google team proved that packet loss alone is not a good proxy to detect sustained congestion. BBR tries to work near Kleinrock's operating point [5], which is an optimal point where maximum bandwidth and lowest latency can be achieved. Improvements in BBR continued. BBR v2 was a major improvement in BBR, and it resolved a lot of issues with the previous versions, such as fairness issues with Cubic and Reno flows, excessive re-transmission, and Round-Trip-Time (RTT) fairness issues. BBR is different from loss-based algorithms as it is model-based. It tries to build a model of a network pipe by trying to access the bottleneck bandwidth (BtlBW) and RTT. The latest version of BBR is BBR v3, which is a minor upgrade over BBR v2.

"Fig 1" shows the relation between the delivery rate and RTT. As we can see, at the start, the delivery rate ramps up to point A (Kleinrock's operating point), and the RTT remains constant. From point A onwards, the delivery rate is constant, and the link's RTT starts to increase. This is the time when the buffer is getting filled. The larger the buffer, the more the increase in RTT. Now, the problem with pure loss-based algorithms such as Cubic and Reno is that they work at point B. They will react only when the packet loss occurs at point B. On the other hand, BBR, due to its model-based approach, has a better picture of the link's bandwidth and RTT, and it shifts its operating point to C. This results in minimum delays and an optimum delivery rate.

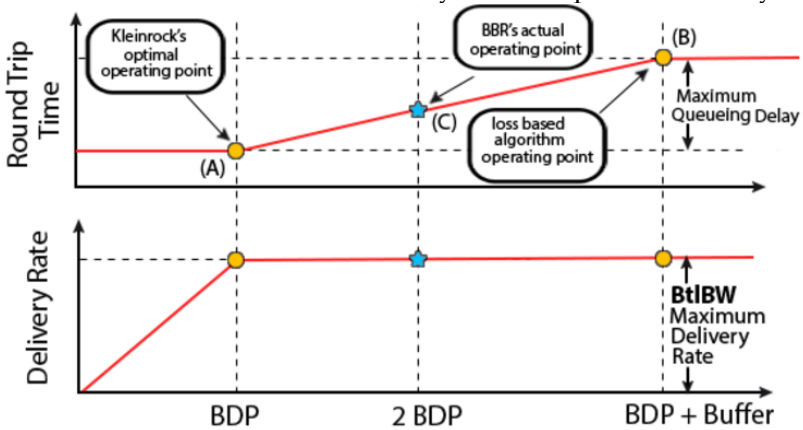

Figure 1. Bottleneck Buffer size > 1 BDP [6]

The increased buffer size has certain consequences, such as increased latency, no significant improvement in throughput, and the building up of large queues in the network buffers, resulting in a phenomenon known as bufferbloat [7]. The "Fig 2" represents a scenario of a shallow buffer. In networks with shallow buffers, a sudden spike or burst of data can temporarily fill the network buffer, and the loss-based algorithm reacts to it, cutting down its delivery rate appreciably. BBR, with its state-of-the-art model-based approach, responds more intelligently and shifts its operating point to point C.

In this work, we have evaluated the performance of BBR v3 (which is referred to as bbr3 throughout the rest of this paper) with pure loss-based algorithms such as TCP Cubic and Reno. The Reno version being used here is NewReno, but for the sake of simplicity, we will refer to it as Reno. We have extensively tested the three algorithms in both wired and wireless environments. It

is pertinent to mention here that BBR v3 is a hybrid model-based CCA; it not only takes into account bottleneck bandwidth and RTT but also considers packet loss and ECN markers. This makes it more robust than the pure loss-based algorithms, as BBR v3 also reacts when a 2% packet loss is observed in a round trip or an ECN marker rate threshold of 50% is reached in 3 to 5 continuous round trips.

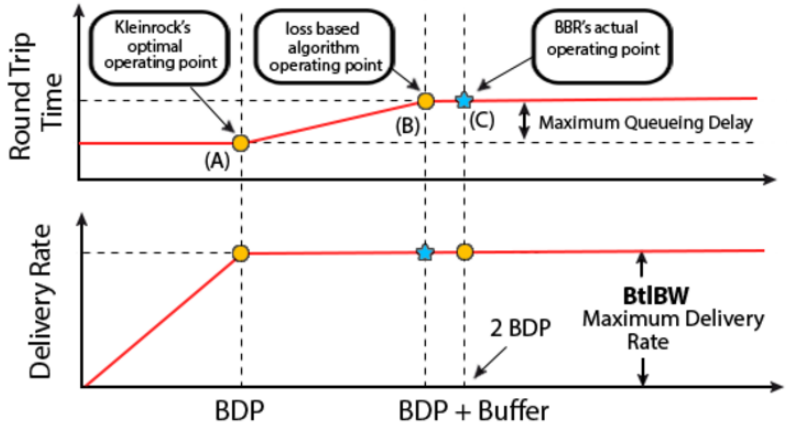

Figure 2. Bottleneck Buffer size < 1 BDP [6]

In this paper, we empirically evaluate the performance of BBRv3 with popular pure loss-based algorithms under both wired and wireless real-time setups. The matrices used are throughput, Internet Control Messaging Protocol (ICMP) ping delay, congestion window (CWND) statistics, and RTT. We tested in real-time bbr3, Cubic, and Reno flow pairs in turn in our RTT fairness tests. An accurate and real-time evaluation of the fairness claims of bbr3 is important, as bbr3 has already been deployed in all Google.com public internet traffic and its internal WAN. That is why we have evaluated bbr3 alongside Cubic and Reno flows. The TCP Upload tests further confirm that bbr3 gives more throughput as compared to Cubic and Reno. The latencies observed showed that bbr3 exhibited the minimum ping delay. The Flexible Network Tester (Flent) was used to perform a wide variety of tests using our atomic physical testbed.

The rest of this paper is structured as follows: Section 2 explores the relevant literature in this domain and examines the TCP architecture within the Linux system, including key details on the default BBR v3. Section 3 outlines the physical testbed used for generating results, which are subsequently analyzed in Section 4. Lastly, Section 5 presents the paper's conclusions along with potential directions for future research.

2. Related Literature

TCP congestion control algorithms are in a continuous evolution phase, and the scientific community is rigorously working to achieve the maximum efficient congestion control in a wide range of networking scenarios. TCP Cubic and BBR are currently the topmost CCAs. Cubic is the default congestion control not only in the Linux mainline kernel but also in the latest version of Microsoft Windows 11. The reason behind this is that Cubic has been optimized for today's high-bandwidth and high-latency networks. It uses a cubic function to adjust the window size and, hence, gauges the available bandwidth more quickly compared to Reno. BBR, on the other hand, is not lagging and is available in Linux as well as Windows latest versions, but it must be invoked to be the default congestion control. Development work on BBR is going at a good pace, and Google software engineers are working day and night to improve its code so that it works optimally in the broadest range of networking scenarios.

There has been a lot of research and development work going on at BBR. Analysis of the performance of BBR v1 has been performed in cellular, satellite, and Ethernet-wired environments

[8-10]. Tomoakai et al. [11] explored it in a 5G environment using real 5G devices. The comparison was made with Cubic and BBR v1 showed degraded performance as compared to Cubic. The reason was the fairness issues with Cubic and Reno flows that were identified in this early version of BBR. Yeong-Jun Song et al. [12] proposed a congestion window scaling method in its flavor known as BBR-CWS and tried to reduce fairness and re-transmission issues. With the arrival of BBR v2 in 2019, a new life was injected into BBR, and it started behaving well. Thanks to the refinements done not only in the congestion control algorithm [13] but also in the delivery rate algorithm by Google's team. Google started to use it in its external TCP traffic and also in its data centers. Although BBR v2 was a major leap towards stable congestion control, it still failed to reach the Linux mainline. Cubic continued to flourish in the Linux mainline as it gave better performance [14].

Active work on BBR v2 continued both in wired and wireless environments. Ahsan et al. [6] in their work on a wireless testbed showed that it still had some issues when deployed in a Wi-Fi 4/5 network scenario. BBR v2 performed poorly due to lesser frame aggregation and an optimized transmission/generic segmentation offload (TSO/GSO) budget. BBR-n was proposed to handle these issues in Wi-Fi 4/5. Wansu Pan et al.[15] In his paper tackled the BBR v2 issues by proposing its version with flow-aware ECN. Wansu Pan's flow-aware ECN tried to resolve the RTT fairness issues with multiple BBR flows sharing the same link. His work was also done using Network Simulator 3 (NS3). Work on BBR v3 is actively in progress. Jose et al. [16] evaluated it in a wired broadband scenario using Mininet [17]. It lacked a wireless scenario, and no physical setup was involved. The trend of using simulation has been following for many years, but the issue with simulations is that they don't bring the true picture of the network to the outside world. Moreover, simulations are not up to date enough to deal with new congestion controls and new modules; for example, TCP Small Queues (TSQ) and latest Active Queuing Management (AQM) [18] schemes are not implemented at all in any of the available simulators of the literature e.g. Network Simulator ver.3 (NS3), Optimum Network Performance Simulator (OpNet), Objective Modular Network Testbed (OMNet). Only a cutting-edge testbed like this can reflect the contribution of novel Linux modules under congestion.

2.1. Bottleneck Bandwidth and Round-trip propagation time (BBR) v3

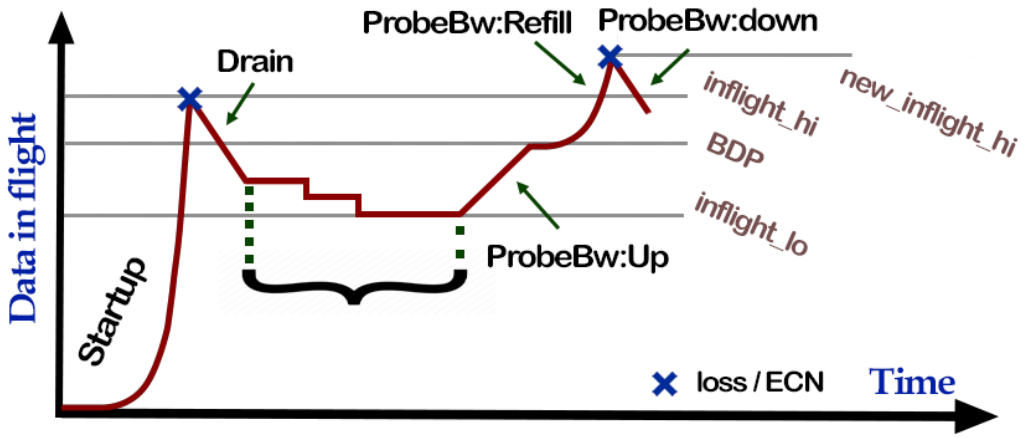

Figure 3. Congestion control phases of BBR v3

Bottleneck Bandwidth and Round-trip propagation time (BBR) v3 [19] is a model-based congestion control algorithm that takes a different approach to handling congestion as compared to pure loss-based algorithms such as Cubic and Reno. What it does differently from a pure loss-based algorithm is that it gets recent measurements of a TCP connection's delivery rate, round-trip time, and packet loss rate. It then uses these vital metrics to build an explicit model of the TCP

connection (a typical socket). It gets a true picture of the available estimated bandwidth, the bandwidth-delay product (BDP), and the resulting maximum data in-flight that should not cause excessive queue pressure. The state machine of BBRgen is driven by four key metrics: bandwidth, RTT, loss rate, and ECN. "Fig 3" shows its congestion control phases [20].

BBR v3 operates using a state machine, as shown in "Fig 4" that governs its behavior based on network conditions. Here are the core components and states involved:

In the Startup state, BBR aims to increase its sending rate aggressively to find the available bandwidth. It utilizes packet pacing to minimize queuing delays. After detecting packet loss or excessive queue build-up, BBR enters the Drain state to reduce the sending rate, allowing the network buffer to clear. The purpose of the ProbeBW state is to periodically test the bandwidth of the network to adjust the sending rate accordingly. It uses a model of the current network conditions to estimate the optimal sending rate. In the ProbeRTT state, BBR attempts to measure the round-trip time (RTT) by sending probe packets. The goal here is to optimize throughput based on current delay characteristics.

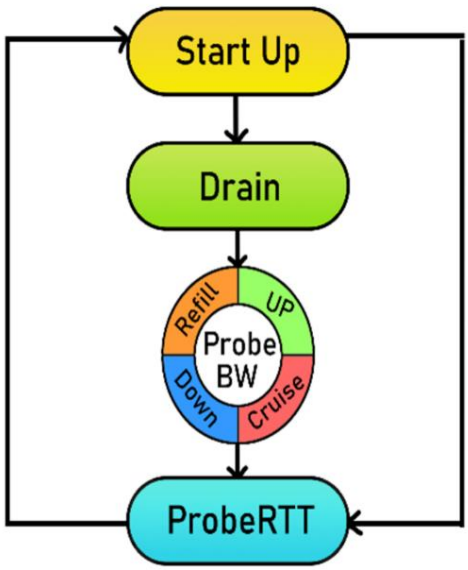

Figure 4. BBR v3 state machine [6]

BBR incorporates careful pacing of packets to avoid overwhelming the network and causing buffer bloat. BBR dynamically controls its sending rate based on estimated bandwidth and round-trip time. BBR implements loss recovery mechanisms to handle packet loss efficiently without resorting to traditional TCP back-off strategies. TCP BBR v3's state machine is designed to adapt to varying network conditions intelligently. By effectively managing sending rates, probing bandwidth, and minimizing latency, it aims to provide a more efficient and responsive TCP performance. Understanding these states and transitions is crucial for network administrators and systems programmers looking to implement or optimize BBR in their environments.

The escalating availability of bandwidth necessitates the continued development of advanced congestion control mechanisms. A core objective is to minimize the time required to detect available bandwidth, thereby maximizing link utilization. The integrated application of TCP at the transport layer and AQM at the network layer offers a promising pathway to achieve robust congestion control capable of effective operation across varying buffer depths and equitable resource allocation among competing flows. The contemporary landscape, characterized by the pervasive adoption of the Internet of Things (IoT) [21] and Cloud Computing, witnesses the generation of immense

volumes of internet traffic from countless connected devices. This heightened traffic volume inherently increases the vulnerability to malicious attacks, thus emphasizing the critical importance of Cybersecurity. A notable limitation is the absence of intrinsic security mechanisms within TCP to detect or mitigate Distributed Denial of Service (DDoS) or Low Data-Rate Denial of Service (LDDoS) attacks.

Fortunately, supplementary attack prevention and mitigation techniques are available externally to TCP. These include the deployment of firewalls and Intrusion Detection Systems (IDS) for malicious packet filtration, alongside rate limiting and traffic analysis for proactive attack prevention. The advent of Artificial Intelligence (AI), specifically machine learning algorithms, has enabled the dynamic detection of anomalies within network traffic. A pertinent example is the detection of Address Resolution Protocol (ARP) spoofing cyberattacks, prevalent in both IoT and general wired/wireless networks. Alsaaidah et al. [22] proposed an innovative machine learning-based approach, employing wrapper feature selection with an RF classifier for dataset optimization, which demonstrated a remarkable detection accuracy of 99.74% within 305 milliseconds.

3. Methodology

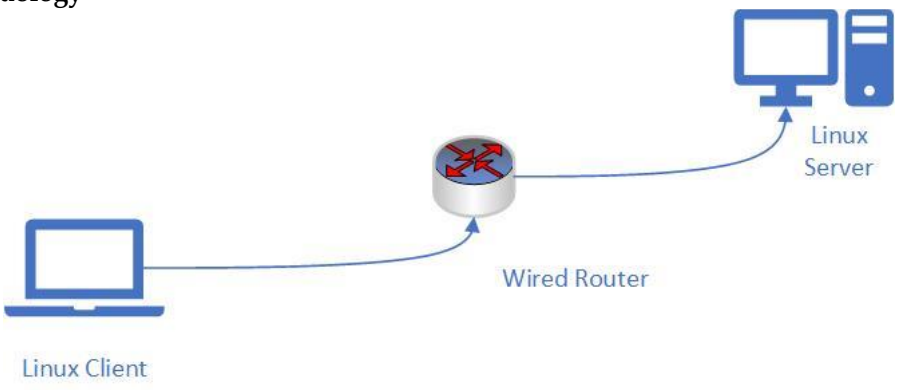

Figure 5. Linux-Ubuntu client via router to a Linux Debian server

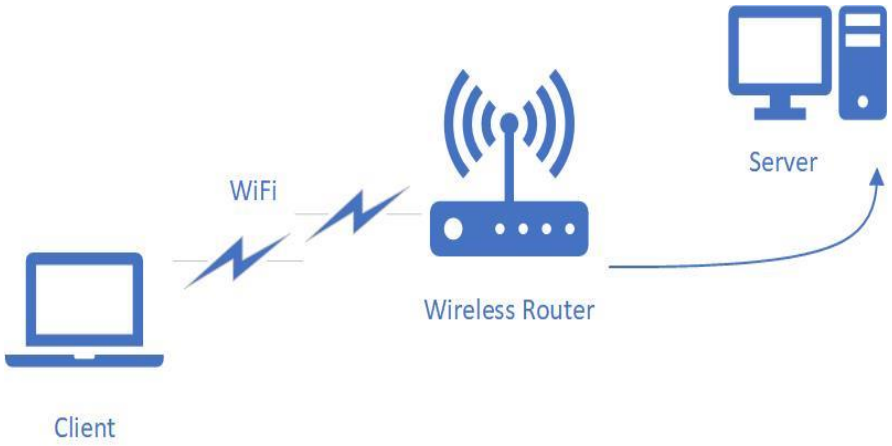

Figure 6. Linux Ubuntu wireless client via Wi-Fi 4/5 router to a wired Linux Debian server [6]

This section describes our physical testbeds deployed for real-time testing of both wired and wireless setups for the three CCAs (bbr3, cubic, and reno) under performance evaluation. The client and servers are running Linux Ubuntu 21 and Debian (v12 bookworm), respectively. The client here is specifically set up with bbr3 compiled from its latest branch available at Github [23]. It has been

configured to be loaded in the Linux kernel 6.13.7 on demand as a loadable module named bbr3. The client machine is a Linux Ubuntu 21 machine with a custom-compiled kernel 6.13.7. The wired Ethernet adapter in it is a Gigabit RealTek USB-based controller, and the wireless is a Qualcomm QCA9377 PCIe-based adapter that supports both Wi-Fi 4/5. The wireless router is a Huawei EchoLife EG8143A5, a Gigabit Passive Optical Network (GPON) terminal that functions as both an Optical Network Terminal (ONT) and a Wi-Fi access point. The server in the wired scenario is an AMD Ryzen 7-based machine running Linux kernel version 6.1.0-17-amd64. Netperf 3 server has been set on this Linux server connected through a Realtek RTL8411 PCI Express Gigabit Ethernet Controller, the server process is set to listen at port 50,000 "Fig 5". Generic segmentation offload (GSO) quantum on the client side BBR is kept initially at 2 MSS, which controls the maximum size of data aggregate that will be passed on to TCP small queues (TSQ) [24], which is set to 8 msec corresponding to a pacing shift of 7 in the Linux kernel 6.x series. These real-time testbeds of "Figs 5 and 6" represent ubiquitous computer network setups found in today's homes and offices, where either the client is wired or wireless, and the server is usually wired. We will be using these testbeds to evaluate the performance of bbr3, Cubic, and Reno.

Experiments performed in this paper are with the help of the Flent [25] tool, which is a network tester that helps run tests with various flows and measures the throughputs and latencies involved. Flent not only offers automation of test runs, but it also collects relevant metadata to ensure the authenticity and aid reproducibility of results. We have performed tests such as the TCP Upload test with 1/2/4/8/12/16 streams for both wired and wireless testbeds. The RTT fairness test has been performed on the wireless testbed. Different pairs of CCAs were used to evaluate the performance using these tests, such as bbr3/cubic, bbr3/reno, and cubic/reno. The parameters we used to configure the testbed of our experiments are given in "Table 1".

Table 1. Testbeds Specifications

Related Parameters	Corresponding Values
Linux Client's Kernel ver.	6.13.7+v3
Linux Server's Kernel ver.	6.11.0-25-generic (Ubuntu) 6.1.0-17-amd64 (Debian)
TCP CC Algorithms	BBR v3 (bbr3), Cubic, Reno.
TCP Small Queues (TSQ)	2 TSQ , 4 TSQ , 8TSQ
GSO quantum	1,2,4 MSS
Queueing Algorithms	FQ, FQ_Codel
WLAN adapters	Qualcomm QCA9377 Dlink 8812BU Realtek RTL8821CE
WLAN Driver Modules	rtl88x2bu,ath10k,e1000
Ethernet Adapter	Realtek RTL8411 Gigabit Ethernet Controller
Flent Tests	1/4/8/12 TCP Upload Test, RTT Fairness Test.
Key Metrics	ICMP Latency (ping RTT) TCP Throughput

4. Results and discussion

In this section, the results gathered from the physical testbed are shared. We will be dividing this section into three sub-sections. Each experiment has been performed ten times. The results are available at our GitHub repository [26]

4.1. TCP Upload Test (Wired scenario)

The TCP Upload test is one of the most important tests to evaluate the performance of any congestion control. The reason is that the logic of congestion control is mostly at the sender's side. Here, we perform this test on our physical testbed of "Fig 5"; the whole 100 Mbps Fast Ethernet bandwidth was presented to the CCAs. The test was performed for 60 seconds, and the data set consisted of 300 points taken at a step size of 0.2 seconds. The box and whisker plot is shown in the "Fig 7. On the right side of the plot, we have ICMP Ping latencies in ms, and on the left side, we have throughput in Mbits/sec. On a single dedicated stream test, all three CCAs performed well and gave the maximum throughput, reaching the channel's maximum capacity of 100 Mbps, but the latency results showed something else. Both Cubic and Reno flows exhibited a large latency, mainly due to bufferbloat at the client-side buffering, whereas BBR3, due to its improved and fine-tuned algorithm, performed exceptionally well with a minimum latency. It avoided the bufferbloat altogether.

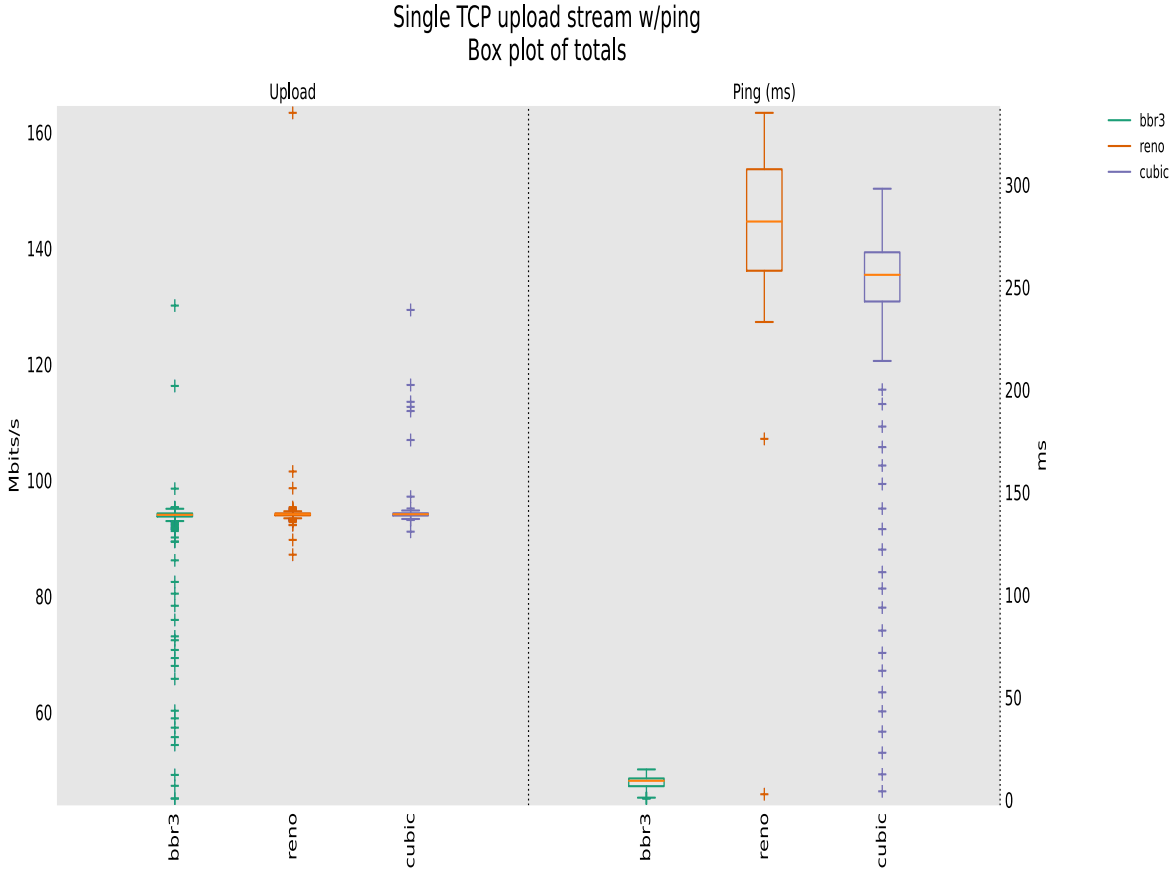

Figure 7. Congestion controls in upload (single stream)

Next, we performed the test with dual streams, and the results were nearly the same. The throughput was good, but Cubic and Reno suffered in providing a better latency.

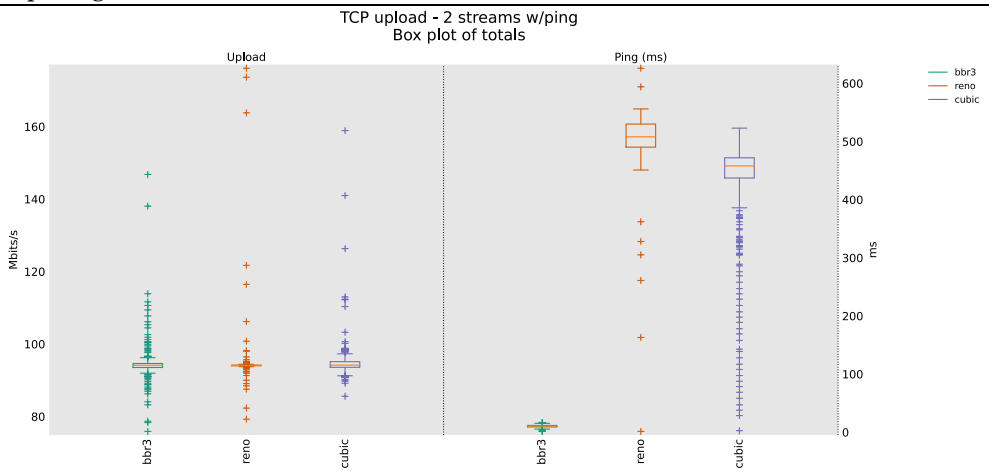

Figure 8. Congestion controls in dual streams

"Figs 9 and 10" give a very important observation as we increased the number of streams to twelve and sixteen. An appreciable decrease in throughputs for Cubic and Reno was observed, likely due to network congestion and inefficient congestion control. With more streams competing for bandwidth, the congestion increased, leading to higher packet loss and re-transmissions. Since both Cubic and Reno are pure loss-based algorithms, they reduced their congestion window aggressively, resulting in a lower throughput. As we are conducting these tests from the client side to the wired server, bufferbloat exacerbation occurred, and the queues filled up much more quickly than in the previous two tests with single and dual streams. Both cubic and Reno react reactively to congestion, and they keep on filling the buffer until they overflow, then only they react and largely reduce their congestion windows. On the other hand, bbr3 acts proactively; it adapts and maintains performance. From the tests in the upload, it is evident that bbr3 outperformed cubic and reno.

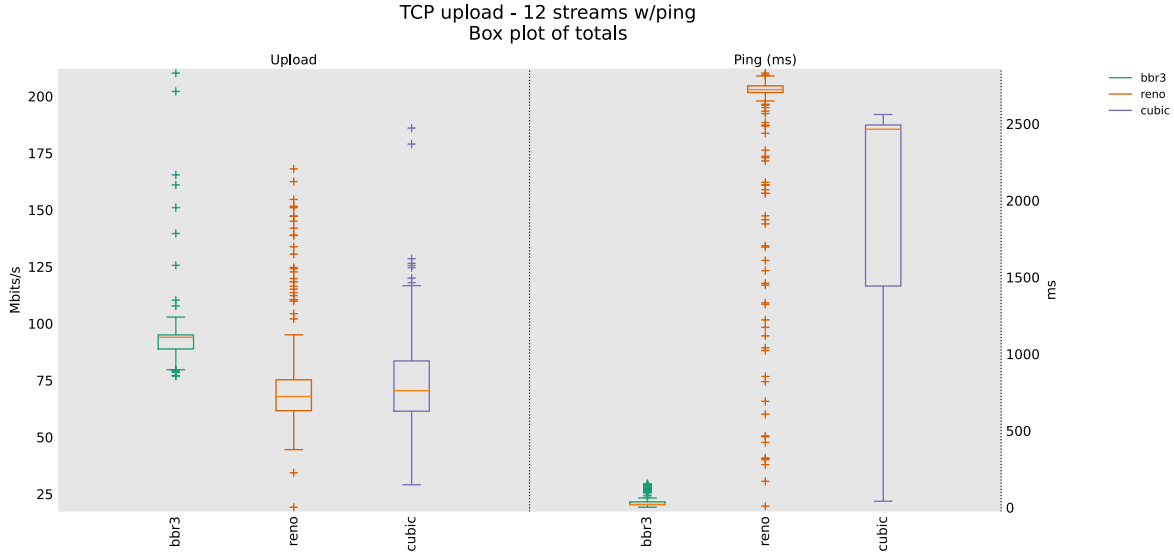

Figure 9. Congestion controls evaluated with twelve streams

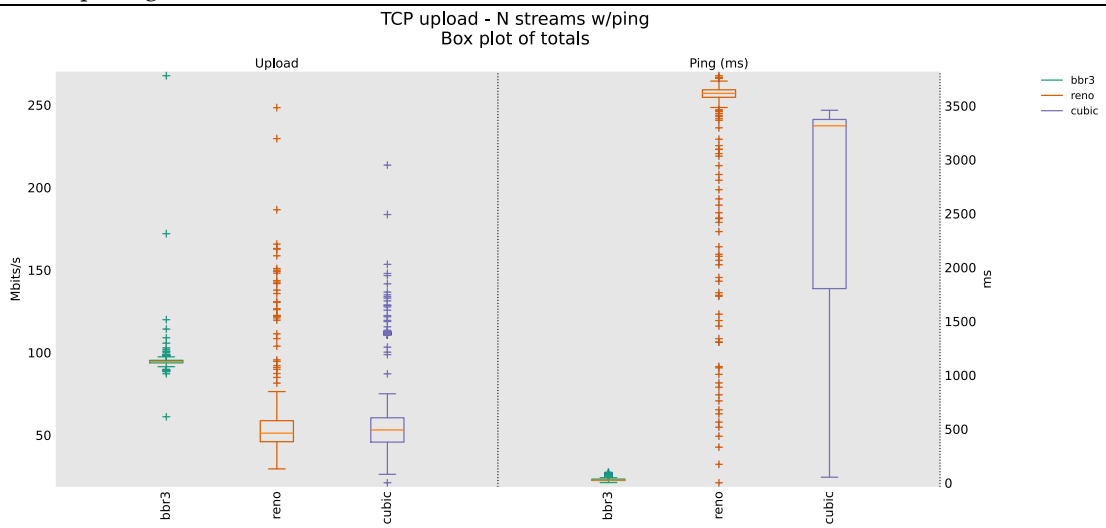

Figure 10. A strenuous 16 streams test for the three CCAs

Table 2. ANOVA results for the throughputs achieved in upload (wired)

Anova: Single Factor

SUMMARY

Groups	Count	Sum	Average	Variance
TCP upload sum - bbr3	301	28895.80409	95.99934912	189.1972895
TCP upload sum - cubic	301	19210.37913	63.82185757	1047.04273
TCP upload sum - reno	301	18434.63804	61.24464464	1005.274918

Table 2a. ANOVA results for the throughputs achieved in upload (wired)

ANOVA

Source of Variation	SS	df	MS	F	P-value	F crit
Between Groups	225742.221	2	112871.110	151.06449	2.69E-57	3.0057260
Within Groups	672454.481	900	747.171645			
Total	898196.702	902				

In "Table 3," three of the CCAs of "Fig 10" have been analyzed via the (Analysis of Variance) ANOVA test [27]. The top table provides the summary of the Sum, Average, and Variance for the three groups under study for 350 data points. The ANOVA results are at the bottom of the table. The sum of the Squares "SS", degree of freedom "df", and mean squared values "MS" are calculated between and within groups. The "F" factor is the ratio of MS between and within groups and is known to follow the F distribution. To infer a statistical conclusion, we compare this "F" value with the "F crit" value at a significance level "α" of 0.05 in the F table. Since this value of F of 151.06 is greater than the F crit value of 3.0, the results given in "Table 3" may be interpreted as statistically significant at a 0.5 % significance level. The P-value of 2.69E-57 is below the threshold value of 0.05, and it proves that our results are statistically significant.

Table 3. ANOVA results for ping for CCAs (wired)

ANOVA Single Factor: SUMMARY

Groups	Count	Sum	Average	Variance
Ping (ms) ICMP - bbr3	350	9635.222	27.52921	326.3074
Ping (ms) ICMP - cubic	350	829330.9	2369.517	1855440
Ping (ms) ICMP - reno	350	1041544	2975.84	1483448

Table 3a. ANOVA results for ping for CCAs (wired)

ANOVA

Source of Variation	SS	df	MS	F	P-value	F crit
Between Groups	1.7E+09	2	8.48E+08	762.2714	5.1E-205	3.00432
Within Groups	1.17E+09	1047	1113071			
Total	2.86E+09	1049				

"Table 4" shows the ANOVA results for ICMP ping latencies measured for bbr3, cubic, and reno for 350 data points. BBR v3 (bbr3) exhibits the lowest latency of 27.5 ms. The ANOVA results show that the F value is 762.27 and the F crit is 3.0, as F > F crit and the P-value of 5.1E-205 is lower than "α" of 0.05, the results are statistically significant.

4.2. TCP Upload Test (Wireless scenario)

In this sub-section, we performed tests using our physical testbed of "Fig 6". In these tests, TCP upload tests were carried over a Wi-Fi 4-based link. It is still the most common Wi-Fi technology being used in homes and offices. The reason is that its 2.4 GHz band provides better coverage as compared to the 5 GHz band. The signal strength during these tests was at -50 dBm.

From "Figs 11 to 14", we observe that even from the single stream test, as shown in "Fig 11", the decreasing trends in throughputs for cubic and reno were observed as compared to a stable throughput for bbr3 for Wi-Fi 4. This was because of Cubic and Reno filling the buffers over the Wi-Fi 4 link, resulting in packet losses and re-transmission. Their reactive algorithms aggressively reduce their congestion window and bring down the throughput. BBR v3 (bbr3), on the other hand, "Keeps the pipe just full, but no fuller" and provides not only optimum throughputs in all the tests as shown in "Figs 11 to 14", but the latency is also less as compared to huge variations in cubic and Reno.

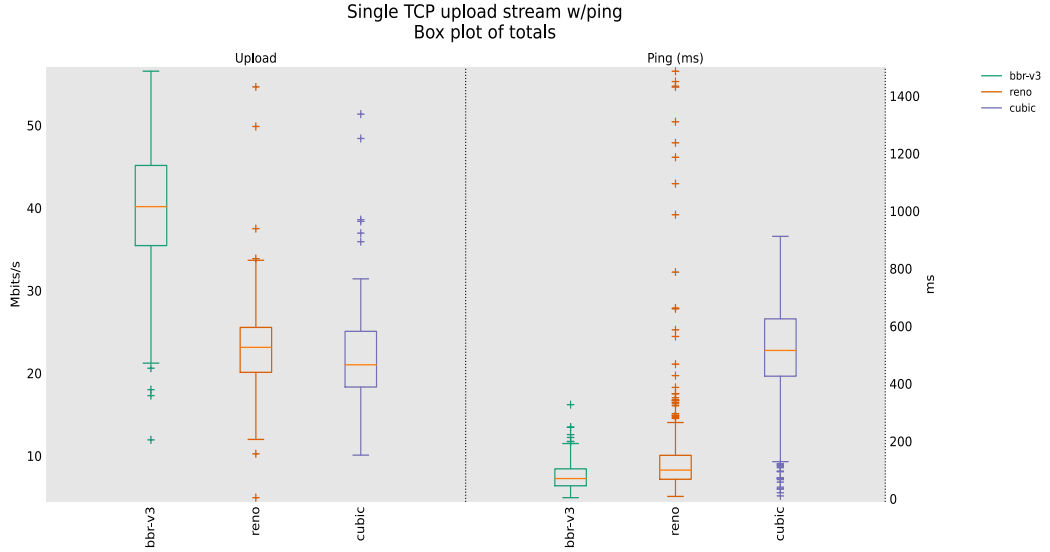

Figure 11. Wi-Fi 4 (single stream)

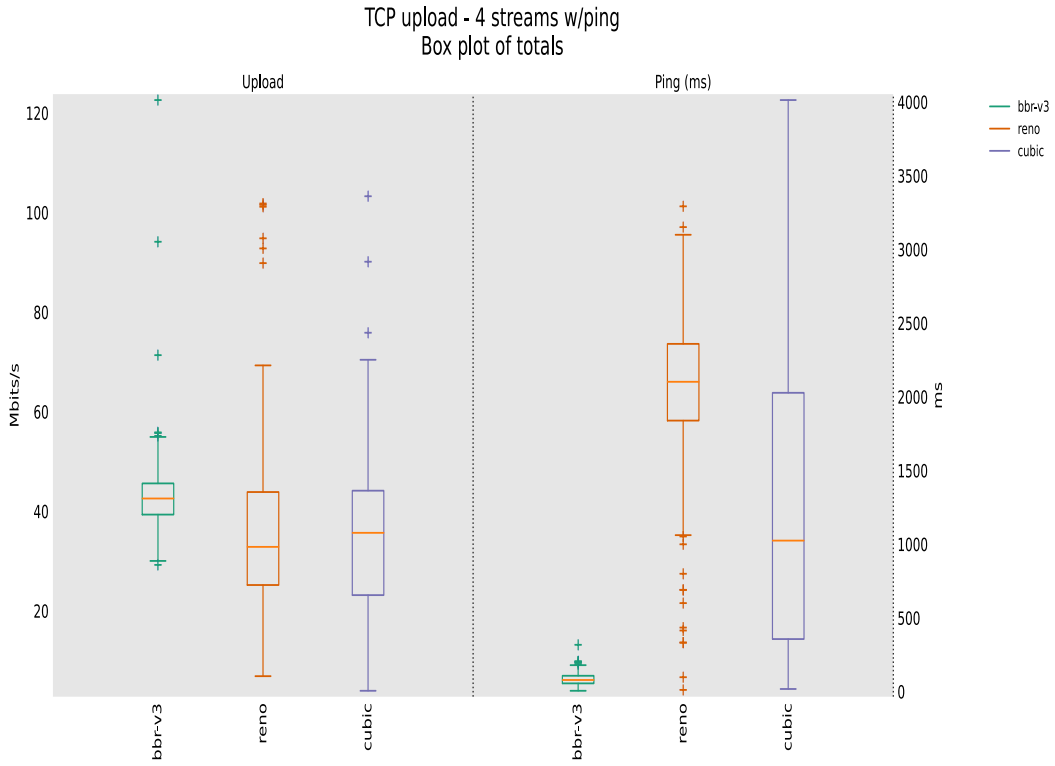

Figure 12. Wi-Fi 4 (four streams in upload)

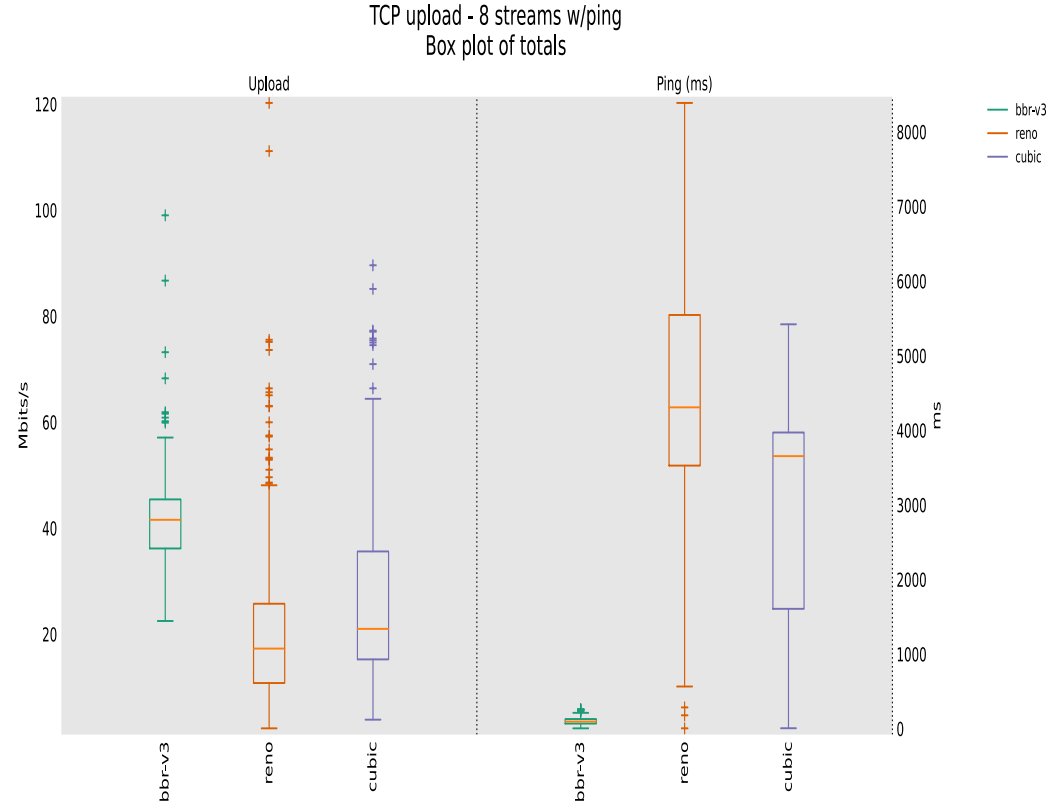

Figure 13. Wi-Fi 4 (eight streams in upload)

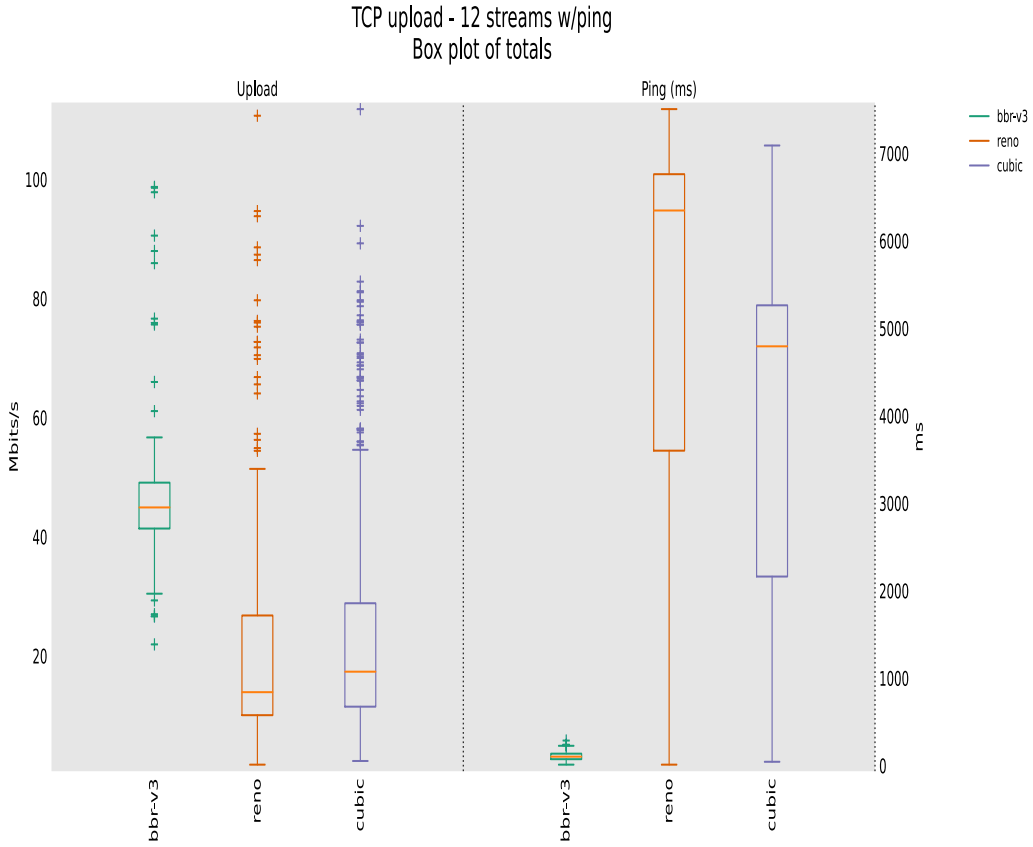

Figure 14. Wi-Fi 4 (with a strenuous load of twelve streams)

Table 4. ANOVA results for the throughputs achieved in upload (wireless)

Anova: Single Factor				
SUMMARY				
Groups	**Count**	**Sum**	**Average**	**Variance**
TCP upload sum – bbr3	302	13868.34	45.92164	91.17581
TCP upload sum - cubic	302	7754.156	25.67601	516.2004
TCP upload sum - reno	302	6562.44	21.72994	377.9039

Table 4a. ANOVA results for the throughputs achieved in upload (wireless)

ANOVA						
Source of Variation	**SS**	**df**	**MS**	**F**	**P-value**	**F crit**
Between Groups	101743.4	2	50871.68	154.895	1.46E-58	3.005693
Within Groups	296569.3	903	328.4267			
Total	398312.7	905				

4.3. RTT Fairness Test (Wireless scenario)

The RTT Fair Realtime Response Under Load test in Flent is designed to evaluate network performance under load conditions. It measures round-trip time (RTT) while simultaneously generating traffic to stress the network. This test helps identify issues like bufferbloat, which can

cause excessive latency when a network is congested. It provides valuable insights into the fairness between bbr3 and cubic congestion control algorithms. It helps analyze how these two TCP congestion control mechanisms share bandwidth and handle network congestion. BBR v3 (bbr3) aims to optimize throughput while maintaining low latency, whereas cubic/reno is a loss-based congestion control algorithms that adjust its sending rate based on packet loss. This test allows users to compare how bbr3 and cubic/reno flows coexist, particularly in scenarios with varying RTTs and network conditions.

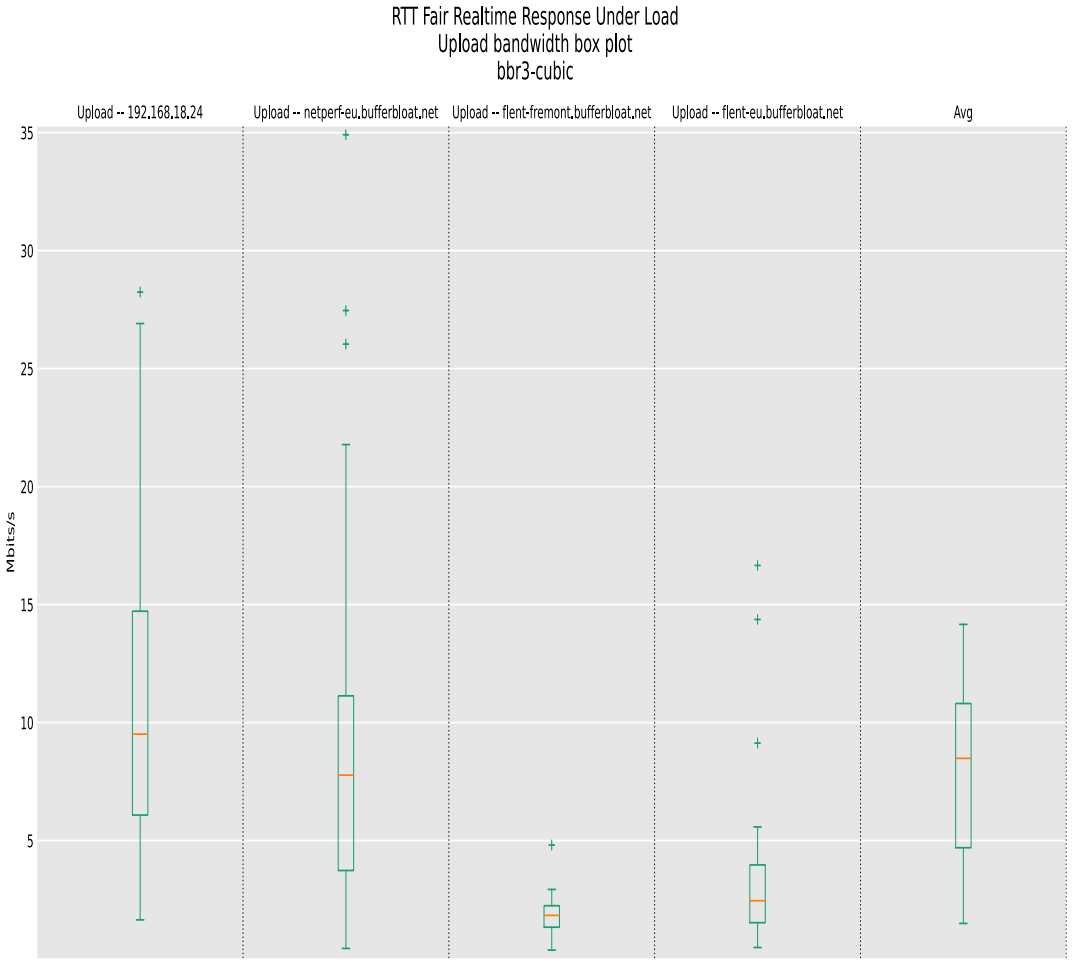

Figure 15. RTT fairness test (bbr3-cubic)

In "Fig 15", we had a local Netperf server at 192.168.18.24, and the other three servers were netperf-eu.bufferbloat.net, flent-fremont.bufferbloat.net, and flent-eu.bufferbloat.net. The first bbr3 flow was targeted to 192.168.18.24. Simultaneously, cubic flow was sending traffic to netperf-eu.bufferbloat.net, and the remaining two servers were also getting bbr3 traffic. From "Fig 15", we see that the first bbr3 flow and second cubic flow are getting a nice share of the bandwidth, and the third and fourth bbr3 flows are at the low side. It is not that all bbr3 flows are using the whole bandwidth; rather, cubic gets its fair share. In "Fig 16" below, the second box-whisker plot is of Reno, and the first, third, and fourth are the bbr3 box-whisker plots. It is evident from the plot that bbr3 gives Reno a fair share of bandwidth so that it can co-exist with bbr3 flows. "Fig 17" is the

fairness comparison between cubic/Reno flows, in which we can see that cubic is a bit unfair with Reno flow and is eating away more bandwidth.

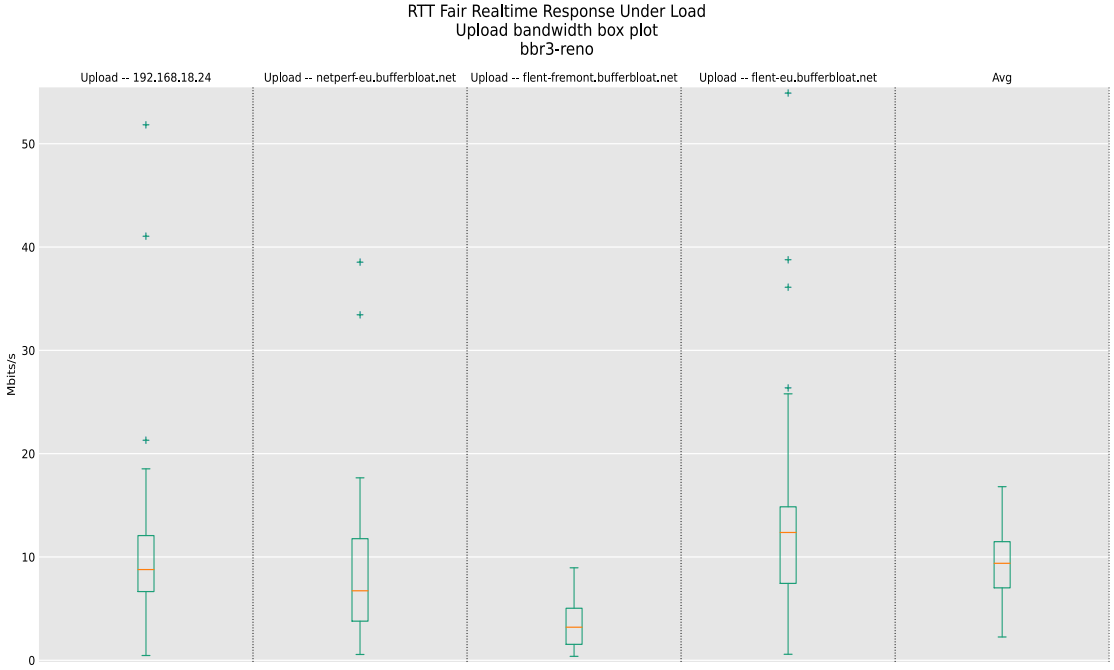

Figure 16. RTT fairness test (bbr3-reno)

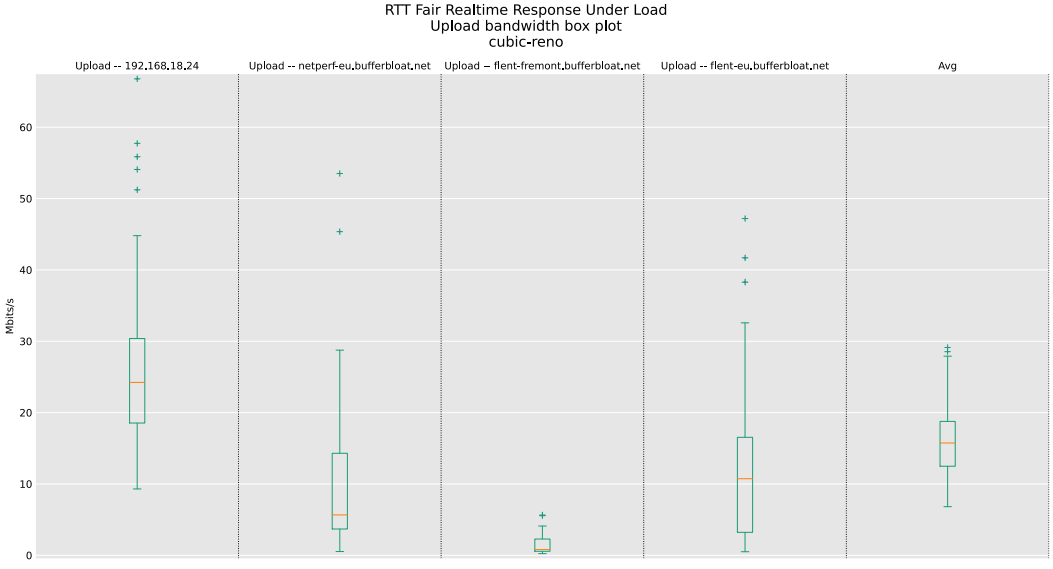

Figure 17. RTT fairness test (cubic-reno)

5. Conclusion and future directions

This paper has evaluated BBR v3 (bbr3) with Cubic and Reno CCAs in wired as well as wireless environments. Most of the research done on BBR v3 is based on a wired model, and the experimentation is performed via simulators only. Research on BBR v3 in both wired and wireless scenarios, and that too using a real-time physical testbed, is being conducted for the first time. Moreover, it is performed to give confidence to the home/office user community, which constitutes millions and millions of people, that BBR has now improved enough that it should not only be in

the Linux mainline kernel but also Microsoft Windows 11. It is pertinent to mention here that bbr2 is already included in Windows 11, so it is a matter of time before bbr3 will also be available there for an end user. Our experiments confirmed that bbr3 outperforms Cubic and Reno in the majority of the experiments done, especially in a scenario when the network load is high and with multiple streams. The TCP Upload test done with various streams, both wired and wireless, shows that bbr3 is indeed the winner. BBR v3's 95 Mbps throughout, as compared to Cubic's 63.8 Mbps and Reno's 61.2 Mbps, speaks for itself. It is a 48.3 % increase when compared with Cubic and a 55.2 % increase when compared with Reno. RTT fairness evaluation with Real-Time Response Under Load test further proved that bbr3 can now co-exist with cubic and reno flows more peacefully than it was doing a few years back. The ICMP Ping latency results clearly show that bbr3 exhibits far fewer delays than its rivals, Cubic and Reno. Thus, it is evident from the performance evaluations that BBR v3 (bbr3) is indeed one of the best congestion control algorithms available these days for the most ubiquitous network setups available in today's networks.

Based on the provided conclusion, future research directions should expand on the real-world evaluation of BBR v3. While this study successfully demonstrated its superior performance in wired and wireless environments with TCP uploads, future work could test the algorithm with a wider variety of real-world network conditions and different types of traffic, such as video streaming and online gaming. Additionally, researchers could investigate the practical challenges and long-term stability of integrating BBR v3 into major operating systems like Microsoft Windows 11 and in mainline Linux kernel, further solidifying the case for its widespread adoption by millions of home and office users.

References

1. Fall K, Floyd S. Simulation-based comparisons of Tahoe, Reno and SACK TCP. ACM SIGCOMM Computer Communication Review. 1996;26(3):5-21.
2. Mo J, La RJ, Anantharam V, Walrand J, editors. Analysis and comparison of TCP Reno and Vegas. IEEE INFOCOM'99 Conference on Computer Communications Proceedings Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies The Future is Now (Cat No 99CH36320); 1999: IEEE.
3. Tan K, Song J, Zhang Q, Sridharan M, editors. A compound TCP approach for high-speed and long distance networks. Proceedings-IEEE INFOCOM; 2006.
4. Ha S, Rhee I, Xu LJASosr. CUBIC: a new TCP-friendly high-speed TCP variant. 2008;42(5):64-74.
5. Kleinrock L. Internet congestion control using the power metric: Keep the pipe just full, but no fuller. Ad hoc networks. 2018;80:142-57.
6. Ahsan M, Muhammad SS. TCP BBR-n: Increased throughput for wireless-AC networks. PLoS One. 2023;18(12):e0295576.
7. Gettys J, Nichols K. Bufferbloat: Dark Buffers in the Internet: Networks without effective AQM may again be vulnerable to congestion collapse. Queue. 2011;9(11):40-54.
8. Bruhn P, Kuehlewind M, Muehleisen M. Performance and improvements of TCP CUBIC in low-delay cellular networks. Computer Networks. 2023;224:109609.
9. Claypool S, Chung J, Claypool M, editors. Measurements comparing TCP cubic and TCP BBR over a satellite network. 2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC); 2021: IEEE.
10. Grazia CA, Klapez M, Casoni M. Bbrp: improving tcp bbr performance over wlan. IEEE Access. 2020;8:43344-54.
11. Kanaya T, Tabata N, Yamaguchi S, editors. A study on performance of CUBIC TCP and TCP BBR in 5G environment. 2020 IEEE 3rd 5G World Forum (5GWF); 2020: IEEE.
12. Song Y-J, Kim G-H, Cho Y-Z. BBR-CWS: Improving the inter-protocol fairness of BBR. Electronics. 2020;9(5):862.
13. Cardwell N, Cheng Y, Yeganeh SH, Jha P, Seung Y, Swett I, et al., editors. BBR v2: a model-based congestion control IETF 105 update. Presentation at IETF105; 2019; Montreal.
14. Ahsan M, Awan MJ, Yasin A, Bahaj SA, Shehzad HMF. Performace evaluation of TCP cubic, compound TCP and NewReno under Windows 20H1, via 802.11 n Link to LTE Core Network. Annals of the Romanian Society for Cell Biology. 2021;25(6):5357-69.
15. Pan W, Tan H, Li X, Xu J, Li X. Improvement of BBRv2 Congestion Control Algorithm Based on Flow-aware ECN. Security communication networks. 2022;2022(1):1218245.
16. Gomez J, Kfoury EF, Crichigno J, Srivastava G. Evaluating TCP BBRv3 performance in wired broadband networks. Computer Communications. 2024;222:198-208.
17. Pryimachenko M. Simulation tools for network technology research. NATURAL AND HUMANITARIAN SCIENCES. 2024:109-10.
18. Ahsan M, Muhammad SS. TCP BBR-n interplay with modern AQM in Wireless-N/AC networks: Quest for the golden pair. Plos one. 2024;19(9):e0304609.
19. Piotrowska A. Performance Evaluation of TCP BBRv3 in Networks with Multiple Round Trip Times. Applied Sciences. 2024;14(12):5053.
20. Zeynali D, Weyulu EN, Fathalli S, Chandrasekaran B, Feldmann A, editors. Promises and Potential of BBRv3. International Conference on Passive and Active Network Measurement; 2024: Springer.
21. Malik J, Akhunzada A, Al-Shamayleh AS, Zeadally S, Almogren A. Hybrid deep learning based threat intelligence framework for industrial iot systems. Journal of Industrial Information Integration. 2025;45:100846.

22. Alsaaidah A, Almomani O, Abu-Shareha AA, Abualhaj MM, Achuthan A. ARP spoofing attack detection model in IoT network using machine learning: Complexity vs. accuracy. Journal of Applied Data Sciences. 2024;5(4):1850-60.
23. Cardwell N. TCP BBR v3 Release Github2025 https://github.com/google/bbr/blob/v3/README.md.
24. Grazia CA, Patriciello N, Høiland-Jørgensen T, Klapez M, Casoni M, Mangues-Bafalluy J, editors. Adapting TCP small queues for IEEE 802.11 networks. 2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC); 2018: IEEE.
25. Høiland-Jørgensen T, Grazia CA, Hurtig P, Brunstrom A, editors. Flent: The flexible network tester. Proceedings of the 11th EAI International Conference on Performance Evaluation Methodologies and Tools; 2017.
26. Ahsan M. BBR v3 tests results GitHub: Ahsan; 2025 [Flent Tests Archive].https://github.com/mahsan76/bbr3.
27. St L, Wold S. Analysis of variance (ANOVA). Chemometrics intelligent laboratory systems. 1989;6(4):259-72.