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

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

A Real-World Testbed Evaluation of BBR-n+ with Other Major TCP Congestion Control Algorithms

Muhammad Ahsan[1*], Muhammad Nabeel[1], Ashfaq Ahmad[2], Iqra Javed[3], and Waqar Ashiq[1]

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

Received: June 09, 2025 **Accepted:** August 11, 2025

Abstract: The demand for a robust congestion control algorithm (CCA) to balance high throughput, low latency, and fairness between flows is on a high rise with evolving networks and tremendous increases in the network's bandwidth. While traditional CCAs like Cubic, Reno, and Data Center TCP (DCTCP) struggle with inefficiencies in dynamic environments, the Bottleneck Bandwidth and Round-trip propagation time (BBR) model-based approach has shown promise. This paper introduces BBR-n+ (BBR new, enhanced), which is a novel variant of BBR v3. While pure-loss-based algorithms are reactive to packet loss only, BBR-n+ uses a hybrid approach. It not only considers Bottleneck Bandwidth and Round-Trip-Time but also uses packet loss rate and an advanced Explicit Congestion Notification (ECN) technique known as Low Latency, Low Loss, and Scalable Throughput (L4S) to build a model of the network pipe. As a result, it provides better throughput, lower latency, and alleviates queue pressure in the router's buffers to reduce bufferbloat. This paper focuses on evaluating BBR-n+ with BBR v3, Cubic, Reno, Vegas, Nevada, Veno, Yeah, and DCTCP in various bottleneck scenarios. Rigorous testing using our physical testbeds for both wired and wireless scenarios has been performed. Performance matrices such as throughput, latency, and RTT fairness when sharing bandwidth with non-BBR flows were used to evaluate the CCAs. The Flent network tester was used to perform various tests on our physical testbeds. The results of our tests demonstrate that BBR-n+ provides superior performance, making it a leading candidate for modern congestion control.

Keywords: BBR; Congestion Control; AQM; Fairness; Pacing Rate; RTT

1. Introduction

The Transmission Control Protocol (TCP) [1] has been the main protocol at the transport layer for the last four decades. Dealing with process-to-process errors and flow control and providing reliable communications. Congestion control has been an important part of it since the infamous Internet congestion collapse [2]. Starting with TCP Tahoe and Reno [3], many congestion control algorithms (CCAs) have been proposed. TCP BIC (Binary Increase Congestion Control) [4], Compound TCP (CTCP) [5], TCP Cubic [3], Data Center TCP (DCTCP) [6], Vegas [7], Veno [8], TCP

YeAH (Yeh Another HighSpeed) [9] and the current one Bottleneck Bandwidth, and Round-trip propagation time (BBR) [10]. Loss-based congestion controls (CC), especially Reno and Cubic, were being used mainly, but with the arrival of BBR, the game has changed. It is because of BBR's better performance over its loss-based rivals. The network paths with large bandwidths and high latencies, termed Long Fat Networks (LFNs), made Cubic a popular choice over Reno for quite some time. As the internet evolved, the buffers in the internet also increased in size, not only in the core but also at the edge network, resulting in a phenomenon known as bufferbloat [11] with Reno and Cubic algorithms. When the buffer is shallow, a few bursts of data or spikes can lead to buffer overflow with loss-based congestion control reacting to it seeing it as an onset of congestion. On the other hand, the larger buffers resulted in loss-based algorithms continuing to fill them resulting in bufferbloat.

In a 10 Gbps link with a 100 ms round-trip time (RTT), Reno's congestion window growth is relatively sluggish. Since TCP Reno increases its congestion window by one segment per RTT, it takes a significant amount of time to reach the bandwidth capacity, which explains why it might need over an hour to fully utilize the link. This is a major reason why more advanced congestion control algorithms, such as CUBIC (used in Linux) and BBR (developed by Google and in the finalization phase to be included in Linux's latest mainline kernel), are preferred for high-speed networks, as they can ramp up bandwidth usage more efficiently. In the same scenario, Cubic will also require a very low loss rate to quickly achieve the link's full capacity. BBR's initial version, BBR v1 in 2016 [12], takes a different approach as it is based on a theory that packet loss alone is not a good proxy to detect sustained congestion. It periodically measures the Bottleneck Bandwidth and round-trip time to build a model of the network path and uses this model to set the congestion window (CWND) size. It intelligently sets its pacing rate and delivery rates to make sure the network stays at optimum utilization and the bursts are avoided. However, this initial version has fairness issues with Cubic and Reno flows going along. BBR has evolved in the last decade as its code has gradually improved with the input of the scientific community, which is actively researching it using various testbed setups and simulators.

BBR v2 [13] was a major release in 2019, which tackled the fairness issue with loss-based algorithms. The vital change in the algorithm was the inclusion of packet loss rate and Explicit Congestion Notification (ECN) mechanisms, along with the already available Bottleneck Bandwidth and Round-trip-time estimation mechanisms. This made BBR a hybrid model-based congestion control algorithm. It is pertinent to mention that here that Google borrowed an idea long ago presented by Leonard Kleinrock in his papers [14, 15], in which he talked about the optimal operating point for a flow and how the delivery rate and RTT go along. His famous saying, "Keep the pipe just full, but no fuller" [16], paved the way for Google's innovative congestion control algorithm. BBR v2 suffered from bandwidth convergence issues that prompted the release of its latest version, BBR v3 [17].

In this paper, we evaluate the performance of BBR v3 not only with pure loss-based algorithms such as Cubic and Reno, but we also compare it with Vegas, Yeah, Veno, DCTCP, and, notably, our proposed BBR-n+. We have extended our work done on BBR-n [18] to the latest BBR v3 and have ported it to Linux kernel 6.13.7 as a loadable module named BBR-n+. The testing has been performed on both wired and wireless physical testbeds. To the best of our knowledge, there is very little work done on the performance evaluation of BBR v3 in both environments, i.e., wired and wireless, together in the same paper, and that too using physical testbeds. Moreover, the evaluation of eight CCAs together using a wide variety of tests is also something that has never been done before in TCP congestion control algorithms research. Flent [19], which is a flexible network tester and comes with a wide variety of tests, has been used to evaluate and test these

seven algorithms. Various network performance measuring matrices, such as throughput, Internet Control Messaging Protocol (ICMP) ping latency, bandwidth-delay product (BDP), queue backlog, and TCP congestion window statistics, are used to evaluate the performance of the CCAs.

Rigorous tests such as the TCP Upload test, Real-time Response Under Load test (RRUL), RTT fairness tests, and Queue backlog tests have been performed. Moreover, the Active Queue Management (AQM) role to enhance performance in the case of BBR has also been explored. We also present a statistical analysis of our results using throughput and latency tests performed on these eight algorithms under consideration. The BBR-n+ code, the scripts used, and the tests performed with metadata for authenticity and reproducibility are available at our GitHub repository [20].

The remainder of this paper is structured as follows: Section 2 introduces the related works done in the context of BBR-v2 and BBR v3, along with BBR v3 architectural details. Section 3 outlines the experimental setup and methodology involved, while Section 4 discusses the results and evaluations. Finally, Section 5 presents the conclusion.

2. Related Works

The scientific community is actively working on TCP congestion control algorithms, and currently, BBR v3, which is the third iteration of BBR from Google, is gaining more interest due to its robust performance not only in enterprise environments but also at the edge on consumer endpoints. On the other hand, loss-based TCP Cubic is giving tough competition and is still the most deployed congestion control in today's most popular operating systems, Microsoft Windows and Linux. The legacy congestion controls, such as Reno, Vegas, Nevada, Veno, Yeah, and DCTCP, are comparatively less deployed when compared with BBR v3 and especially Cubic.

Danesh et al. [21] empirically evaluated BBR v3 with Cubic using Network Simulator 3 (NS-3) [22] using various performance metrics, including throughput, queuing delay, congestion window size, and packet loss. They tested BBR v3 in isolation and with other competing flows. Although the paper explored that BBR v3 still gives fairness issues with Cubic and convergence issues with other BBR v3, it does not propose specific algorithmic improvements or solutions to this issue. Moreover, as the study is simulation-based with NS-3, the results might differ from real-world network dynamics or more complex network environments. Gomez et al. [23] provide a thorough analysis of BBR v3 under different conditions, including variations in buffer sizes, loss rates, propagation delays, and flow counts, offering a broad perspective on performance in wired broadband networks in both emulated and real hardware experiments. As this study is based mainly on wired broadband, its conclusions may not generalize to other network environments such as wireless. It also discussed Active Queue Management (AQM) schemes such as Flow Queuing with Controlled Delay (FQ_CoDel) [24] and how it can mitigate RTT unfairness.

In satellite networks, Claypool et al. [25] discuss BBR performance with Cubic and satellite-optimized for startup (Hybla), and also discuss TCP YeAH. It highlights how feedback delays and long RTTs weaken traditional TCP congestion mechanisms, leading to reduced link utilization, and evaluates alternative methods such as Explicit Congestion Notification (ECN) or optimized congestion control strategies that can better handle these challenges. As in most cases, this study was also based on emulation and simulations that might not capture all real-world complexities of satellite links. Wenjia et al. [26] proposed an integrated approach to improve Multipath TCP (MPTCP) performance in heterogeneous wireless networks by addressing bottleneck fairness issues using BBR, but their study was based on BBR v1. They proposed a novel coupled congestion control algorithm called MPTCP-BBR that dynamically detects shared bottlenecks among TCP flows. The work had limitations as it may not directly apply to single-path TCP. The coupled algorithm and

the scheduling involved added more computational overhead. Their scheme may have added delays when used with BBR v3, although it is not explored yet.

Zeynali et al. [27] in their work on BBR v3 explored it with Explicit Congestion Notification (ECN) enabled. It quantifies fairness using Jain's Fairness Index [28] and thoroughly compares BBR v3's bandwidth share against Cubic flows, analyzing it with and without ECN. Their evaluation is limited to controlled testbed conditions, which may not capture all complexities and dynamics of varied real-world Internet environments. It does not provide any fix or a revised algorithmic solution to the issues probed.

The works discussed in this section on the evaluation of BBR v3 are the only works available, and to the best of our knowledge, there is no evaluation of BBR v3 with Cubic and legacy congestion controls together in one consolidated work. Moreover, all of the works on BBR v3 discussed are via emulation or simulation, with no work on a real-time test bed that involves both wired and wireless network setups. Along with it, our algorithm BBR-n+, which we have built upon our prior works [18, 29], is empirically evaluated with BBR v3. Statistical analysis has been performed using ANOVA/MANOVA [30], confirming that our results are statistically significant. The paper contributes practical experimental resources by providing access to virtual machines and scripts used for the experiments, supporting reproducibility and further research.

2.1. BBR-n+ (BBR new, enhanced)

BBR-n+ is a revised version as a result of thorough testing on BBR-n [18]. We have custom compiled BBR-n to BBR v3 code, as it was initially based on BBR v2. For that purpose, BBR v3 was sought from GitHub and compiled over a Linux Ubuntu kernel 6.13.7. The revisions made for BBR-n were incorporated into a loadable module BBR-n+ compiled under this new kernel environment. The updated BBR v3 code incorporates patches for the revised startup gain, pacing gain values, and the quantum settings used for Transmission and Generic Segmentation Offloads (TSO/GSO). Based on our comprehensive analysis [29], CAKE AQM, identified as the most suitable active queue management algorithm was selected as the default AQM for all experiments presented in results and discussion section. Currently, fewer experimental papers exclusively focus on BBR v3 in diverse environments published publicly, as BBRv3 is relatively new and is not available directly in Linux main kernel 6.x. It needs meticulous efforts to have a BBR v3 congestion control enabled Linux kernel by custom compiling it as we did in this case. The BBR-n+ was further compiled over this custom kernel to perform the tests shared in results and discussion.

3. Methodology

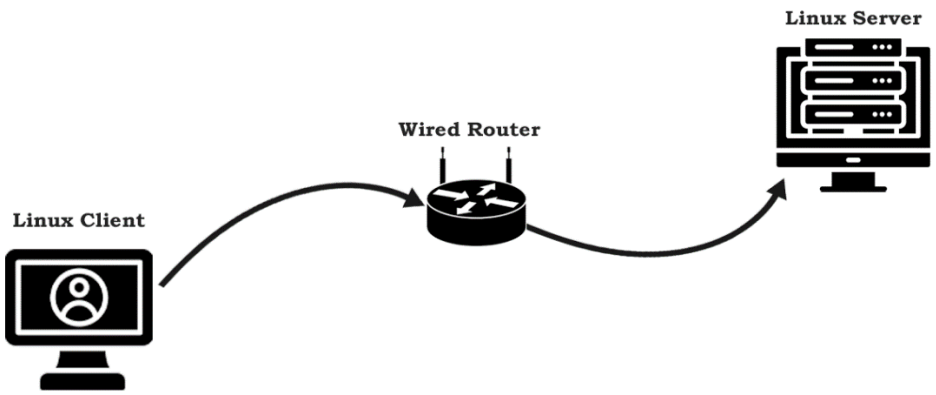

Figure 1. Linux-Ubuntu client via router to a Linux Debian server

To assess the real-time performance of BBR v3, BBR-n+, DCTCP, Veno, Yeah, Vegas, Nevada, Cubic, and Reno congestion control algorithms, we established dedicated physical testbeds for both wired and wireless network configurations. Our client machine, running Linux Ubuntu 21 with a custom Linux kernel 6.13.7, was specifically configured to load a BBR v3 module compiled from its latest GitHub branch. It connects via a Gigabit RealTek USB-based Ethernet adapter for wired tests and a Qualcomm QCA9377 PCIe-based adapter (supporting Wi-Fi 4/5) for wireless. The wireless access point is a Huawei EchoLife EG8143A5 GPON terminal, which also functions as an ONT. The wired server, an AMD Ryzen 7-based machine running Linux kernel 6.1.0-17-amd64, hosts Netperf 3 (listening on port 50,000) and is connected via a Realtek RTL8411 PCI Express Gigabit Ethernet Controller. These testbeds, **Figs 1 and 2**, simulate typical home and office network environments where clients can be wired or wireless, and servers are generally wired.

Experiments were conducted using the Flent tool, which facilitates automated test execution, throughput, and latency measurement, and robust metadata collection for result authenticity and reproducibility. Our tests included TCP upload tests with 1, 2, 4, 8, 12, and 16 streams across both wired and wireless setups. We also performed an RTT fairness test on the wireless testbed. Performance was evaluated by testing different CCA pairings, specifically BBR v3/CUBIC, BBR v3/Reno, and CUBIC/Reno. The specific testbed configurations are provided in "Table 1".

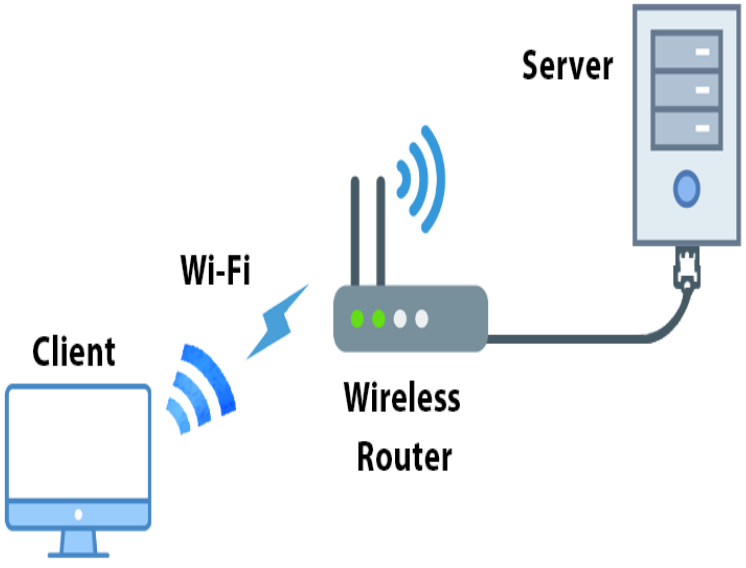

Figure 2. Linux-Ubuntu wireless client via Wi-Fi 4/5 router to a wired Linux Debian server

Table 1. Testbed 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), BBR-n+, Cubic, Reno, Dctcp, Vegas, Veno, Nevada, YeAH

TCP Small Queues (TSQ)	Standard TSQ
GSO quantum	4 MSS
Queueing Algorithms	FQ, CAKE, 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, RTT Fairness, RRUL, and RRUL_BE tests.
Key Metrics	ICMP Latency (ping RTT) TCP Throughput

4. Results and discussion

This part of our paper will provide the results that we have gathered after thorough testing on our physical testbeds. It is divided into four sub-sections. The test results gathered via Flent for Real-Time Response Under Load (RRUL) tests, TCP Upload tests for 1/2/4/8/16 streams test, RTT fairness test, and RRUL Best Effort (RRUL_BE) test will be shared and discussed. It is pertinent to mention that each experiment has been performed ten times to get the most accurate results. Furthermore, statistical analysis results using Analysis of Variance (ANOVA) [31] will be provided in a tabular form in this section wherever they are deemed necessary.

4.1. RTT Fair with RRUL test for wired scenario

The real-time response Under Load (RRUL) test is a network performance test designed to analyze how a network behaves under heavy workloads, particularly in detecting bufferbloat. It is implemented in Flent, a flexible network testing tool, and works by running RTT measurements using ICMP ping and UDP round-trip time while saturating the link with eight TCP streams (four downloads, four uploads). This helps expose latency issues when the network is under load. Moreover, the RRUL test uses Differentiated Services Code Points (DSCP) to classify traffic and analyze network fairness. It uses Best Effort (BE), Background (BK), Class Selector 5 (CS5), and Expedited Forwarding (EF) markings. These DSCP values help evaluate how different traffic classes behave under network load and whether prioritization mechanisms affect network performance.

Fig 3 shows all of the CCAs put under the test, and the Box Whisker plot shows the upload and download throughputs along with the ICMP Ping latencies. BBR-n+ provides better throughput in downloads as compared to BBR v3 with an appreciably low latency as well. Figs 4 and 5 provide a detailed breakup of the combined Fig 3, in terms of uplink and downlink throughputs, both for local and remote hosts at 10.103.108.9, 10.103.108.13, and at netperf-eu.bufferbloat.net and flent-fremont.bufferbloat.net.

Tables 2-4 provide us with the statistical analysis for Fig 3. The threshold value "α" of 0.05 was chosen. In the three tables, we get an F value which is greater than the F critical (F crit) value, and a Probability value (P-value) of zero suggests that the results are strongly statistically significant. We also performed MANOVA analysis and post-hoc tests such as Tukey's Honestly Significant Difference (Tukey HSD) test; for brevity, the results are shared at our online data repository at GitHub [20].

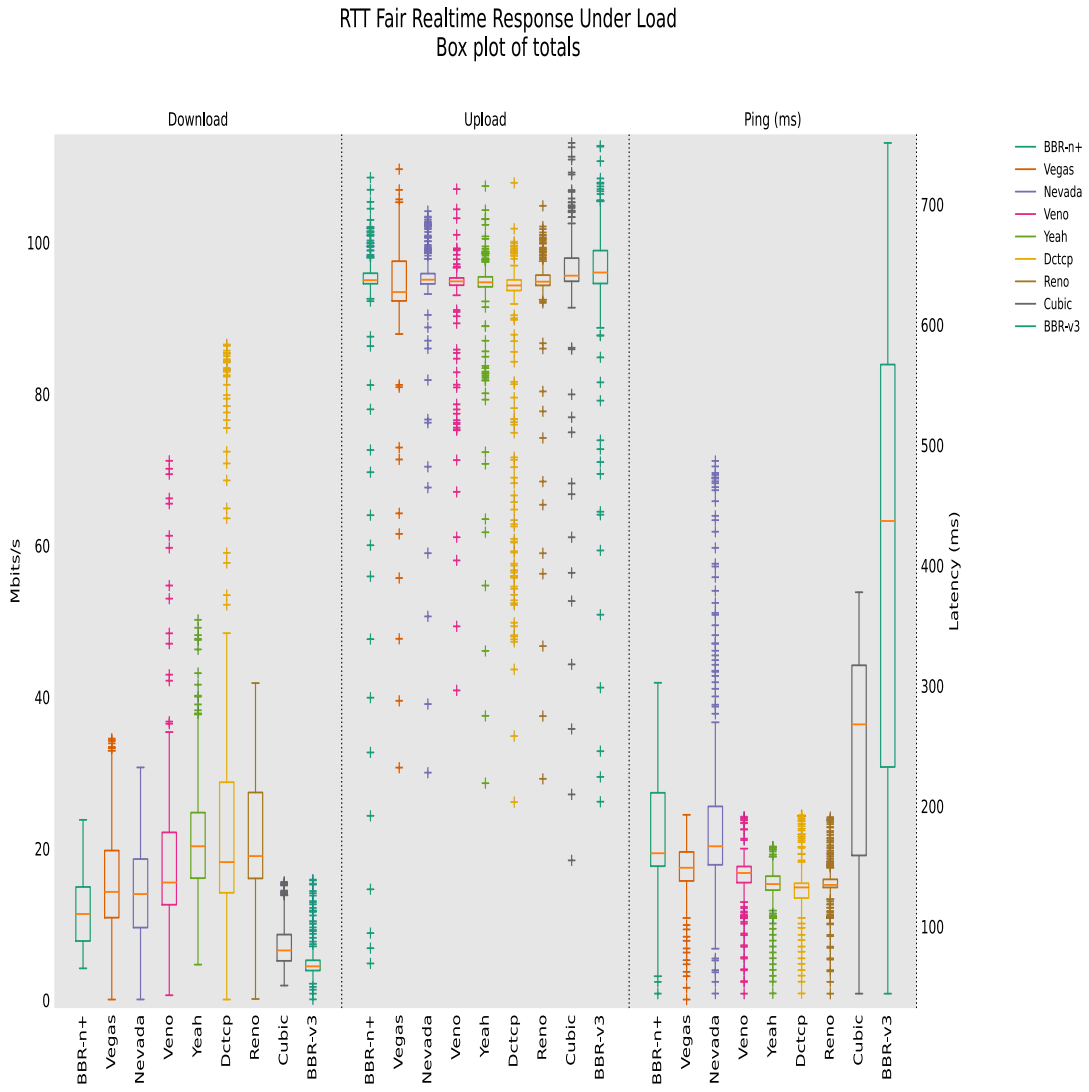

Figure 3. RRUL test for all CCAs

Table 2. ANOVA results for the throughputs achieved in the upload for all CCAs

Anova: Single Factor

SUMMARY

Groups	Count	Sum	Average	Variance
TCP upload sum - BBR v3	304	29488.7116	97.0023407	163.250985
TCP upload sum - BBR-n+	304	28904.4038	95.0802757	110.072966
TCP upload sum - Cubic	304	29312.9189	96.4240753	145.378116
TCP upload sum - Reno	304	28809.5814	94.7683597	92.0993107
TCP upload sum - Yeah	304	28645.912	94.2299736	106.311244
TCP upload sum - Veno	304	28674.6611	94.3245432	110.06105
TCP upload sum - Vegas	304	28736.3188	94.5273644	106.865826

TCP upload sum - Nevada	304	28997.6521	95.3870136	171.925233
TCP upload sum - Dctcp	304	27263.6533	89.6830699	275.954792

Table 2a. ANOVA results for the throughputs achieved in the upload for all CCAs

Anova						
Source of Variation	**SS**	**df**	**MS**	**F**	**P-value**	**F crit**
Between Groups	10448.8095	8	1306.10119	9.16977272	1.5825E-12	1.94179547
Within Groups	388421.616	2727	142.435503			
Total	398870.425	2735				

Table 3. ANOVA results for the throughputs achieved in downloads for all CCAs

Anova: Single Factor SUMMARY				
Groups	**Count**	**Sum**	**Average**	**Variance**
TCP download sum - BBR v3	303	1545.218824	5.099732091	8.675143397
TCP download sum - BBR-n+	303	3720.039257	12.27735728	35.45658604
TCP download sum - Cubic	303	2291.954174	7.564205194	16.12889992
TCP download sum - Reno	303	6663.096554	21.99041767	102.9503879
TCP download sum - Dctcp	303	8152.397218	26.90560138	528.677516
TCP download sum - Veno	303	5767.484526	19.0346024	177.0041863
TCP download sum - Vegas	303	4984.538874	16.45062335	69.6809185
TCP download sum - Nevada	303	4320.744624	14.25988325	51.05235306
TCP download sum - Yeah	303	6513.667868	21.49725369	140.6506195

Table 3a. ANOVA results for the throughputs achieved in downloads for all CCAs

Source of Variation	**SS**	**df**	**MS**	**F**	**P-value**	**F crit**
Between Groups	121558.735	8	15194.841 87	120.9912472	1.2346E-17	1.941806675
Within Groups	341343.5364	2718	125.58629 01			
Total	462902.2714	2726				

Table 4. ANOVA results for Ping latency for all CCAs

Anova: Single Factor: Summary				
Groups	**Count**	**Sum**	**Average**	**Variance**
Ping (ms) avg - BBR v3	351	147117.3	419.1376	41683.95

Ping (ms) avg - BBR-n+	351	62894.09	179.1854	3011.238
Ping (ms) avg - Cubic	351	86076.03	245.2308	7106.763
Ping (ms) avg - Reno	351	47453.19	135.1943	359.1036
Ping (ms) avg - Vegas	351	51999.03	148.1454	628.0762
Ping (ms) avg - Veno	351	49481.96	140.9742	422.7777
Ping (ms) avg - Nevada	351	66654.2	189.898	7197.48
Ping (ms) avg - Yeah	351	47061.33	134.0779	287.4948
Ping (ms) avg - Dctcp	351	46185.05	131.5813	409.8849

Table 4a. ANOVA results for Ping latency for all CCAs

ANOVA

Source of Variation	SS	df	MS	F	P-value	F crit
Between Groups	24342005.5	8	3042751	448.146	0	1.941341
Within Groups	21387369.2	3150	6789.641			
Total	45729374.7	3158				

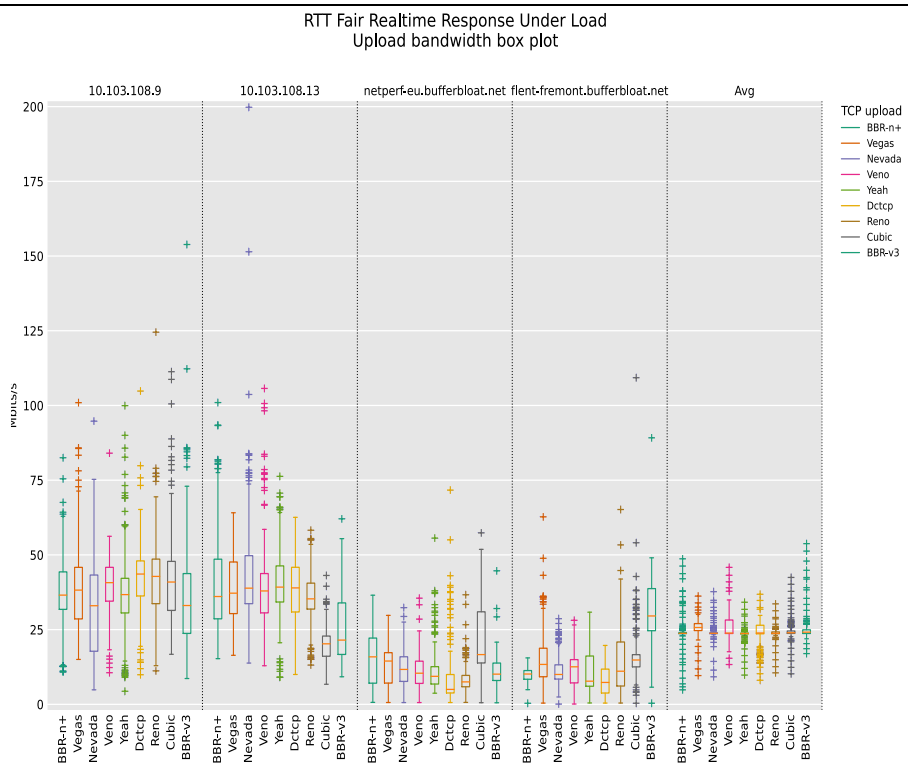

Figure 4. Upload the plot with local/remote hosts for all CCAs

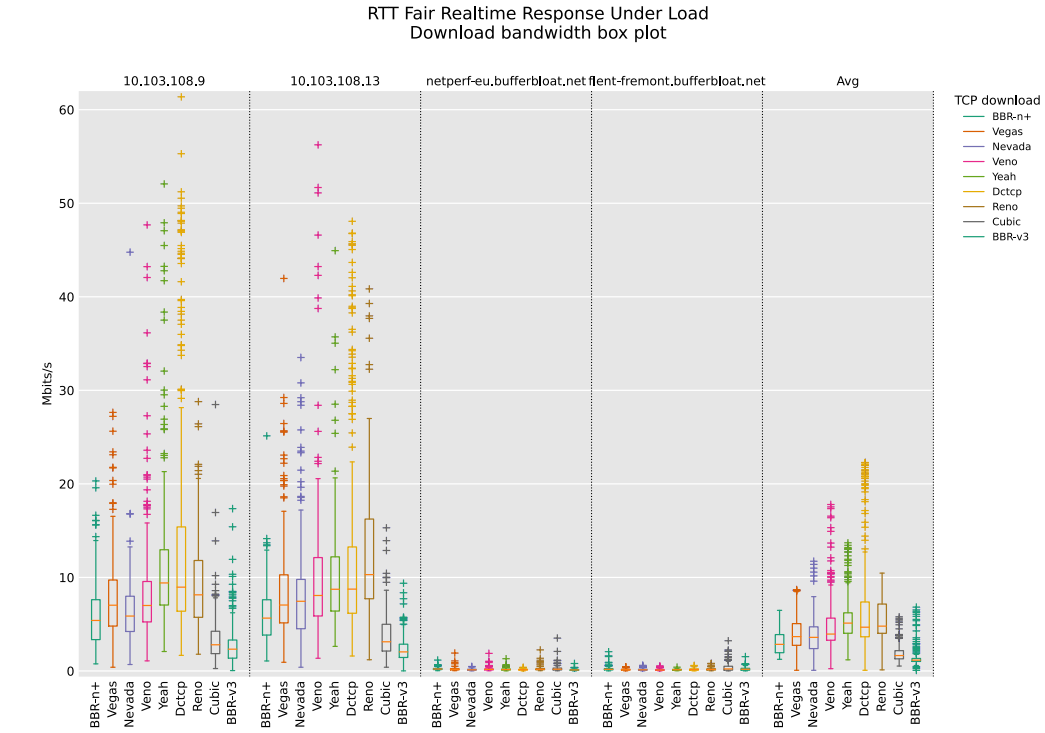

Figure 5. Download the plot with local/remote hosts for all CCAs

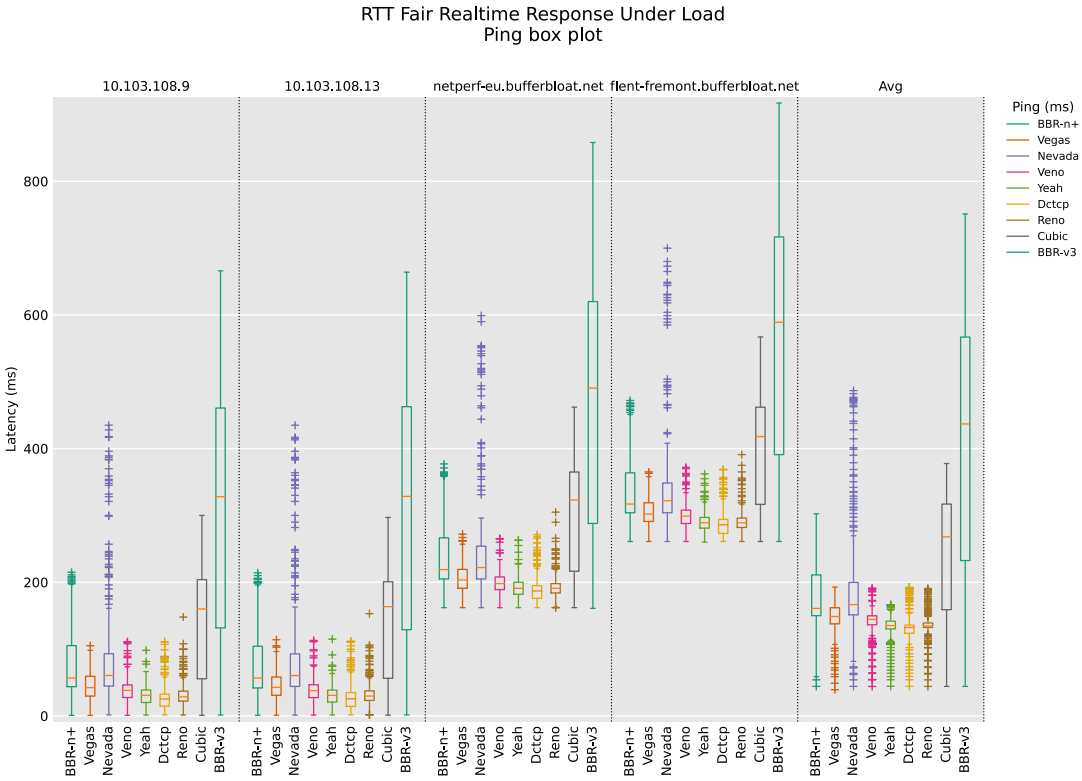

Figure 6. ICMP Ping latencies for all CCAs

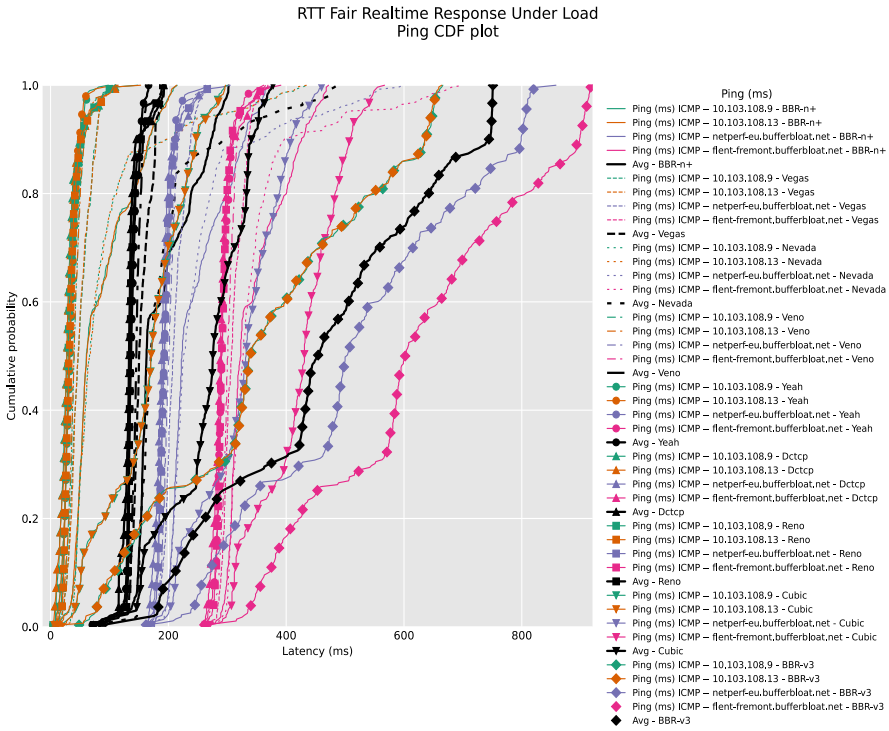

Figure 7. Ping CDF plot for all CCAs

The cumulative distribution function (CDF) plot of **Fig 6** and the Ping plot of **Fig 7** show that the latency of BBR-n+ is lower than BBR v3.

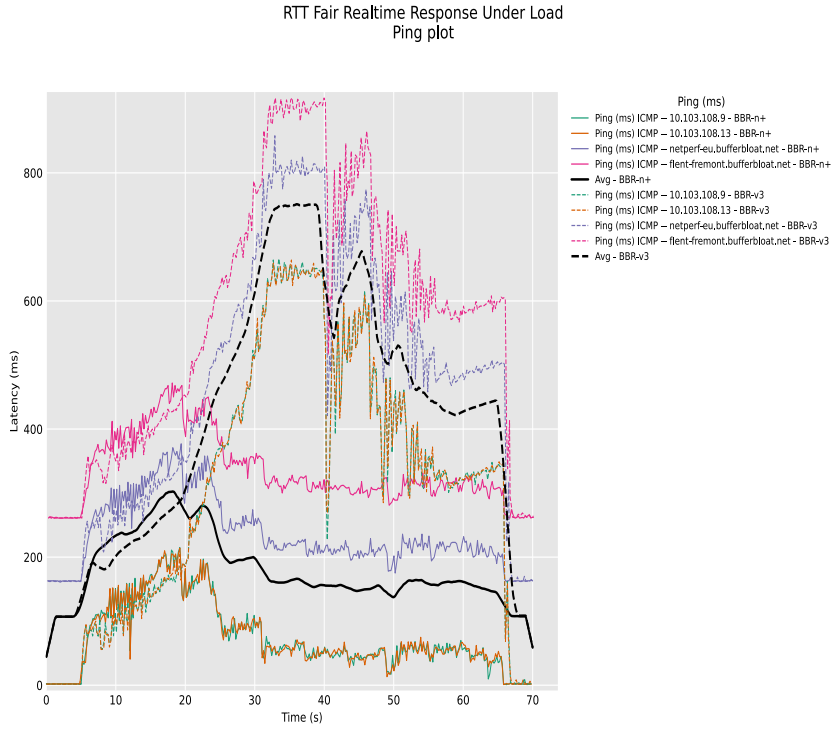

Figure 8. BBR-n+ vs BBR v3 Ping plot

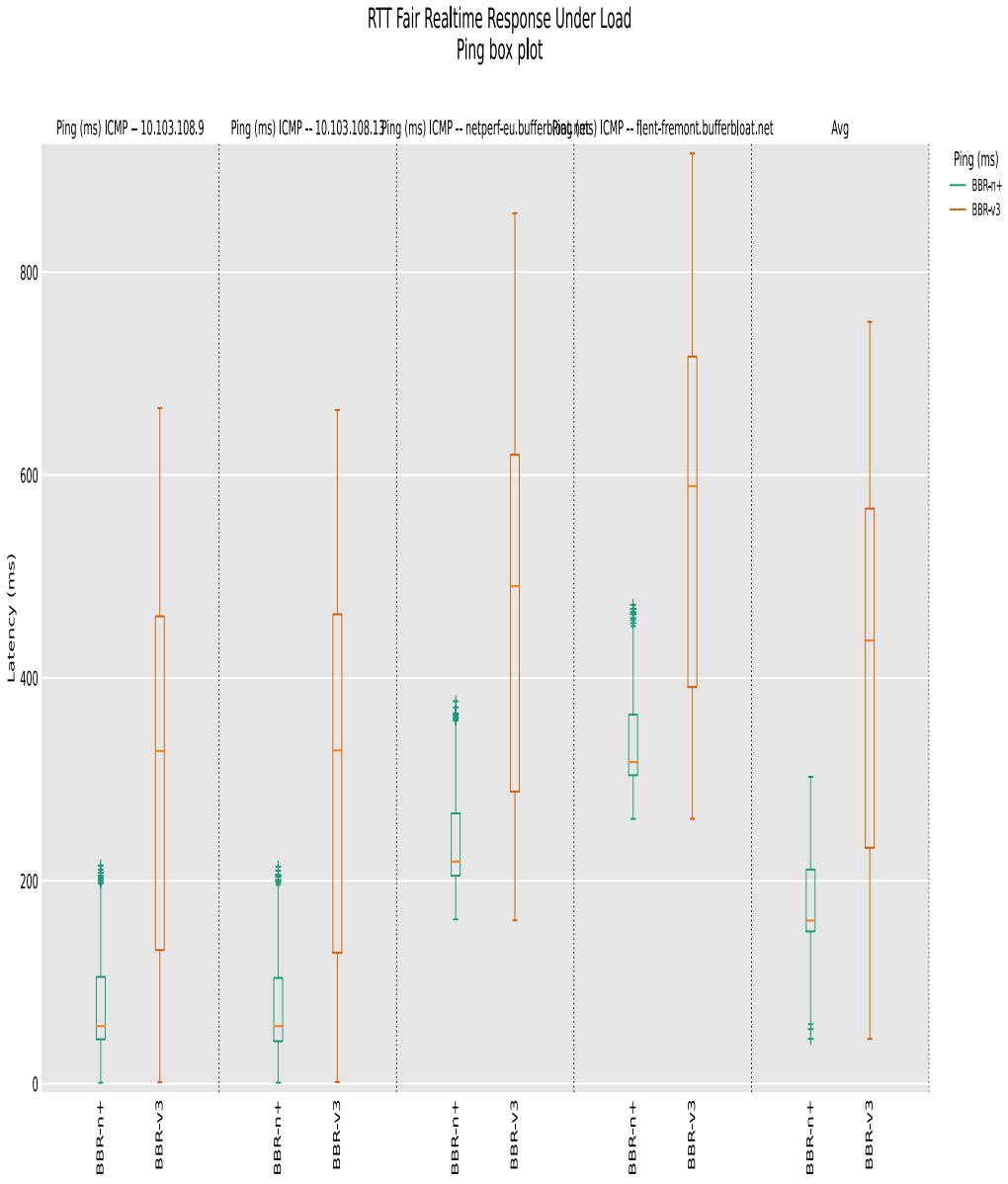

Figure 9. BBR-n+ vs BBR v3 Ping box plot

Figs 8 and 9 shows the comparison specifically between BBR-n+ and BBR v3, in which the latency of BBR-n+ is lower not only for local servers at 10.103.108.9 and 10.103.108.13 but also for remote servers “netperf-eu.bufferbloat.net” and “flent-fremont.bufferbloat.net”.

4.2. Upload streams test for wireless scenario

In this test, starting from a single stream, multiple streams were used to test the congestion control’s performance in upload, as the congestion control mainly works at the sender side (outgoing data). Figs. 10 and 11 show the results of such a test in which sixteen and twelve streams were sent in Upload using the physical testbed shown in Fig 2. We observe that the latest BBR, BBR v3, not only gives better throughput but also exhibits the lowest latency of all the CCA under test.

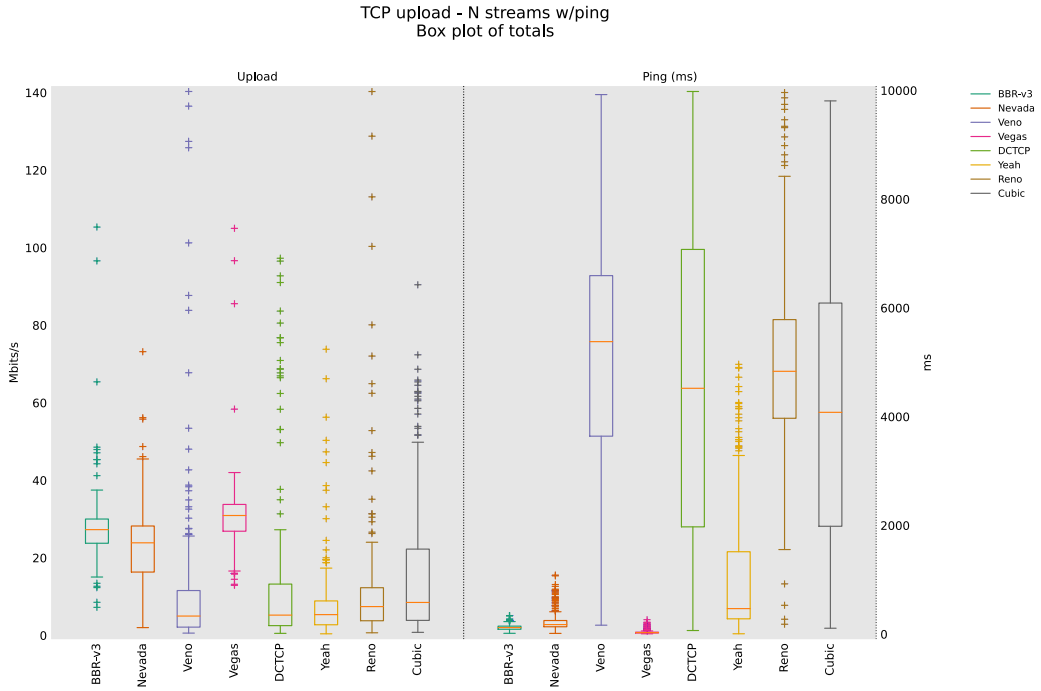

Figure 10. Sixteen streams in the Upload test for all CCAs

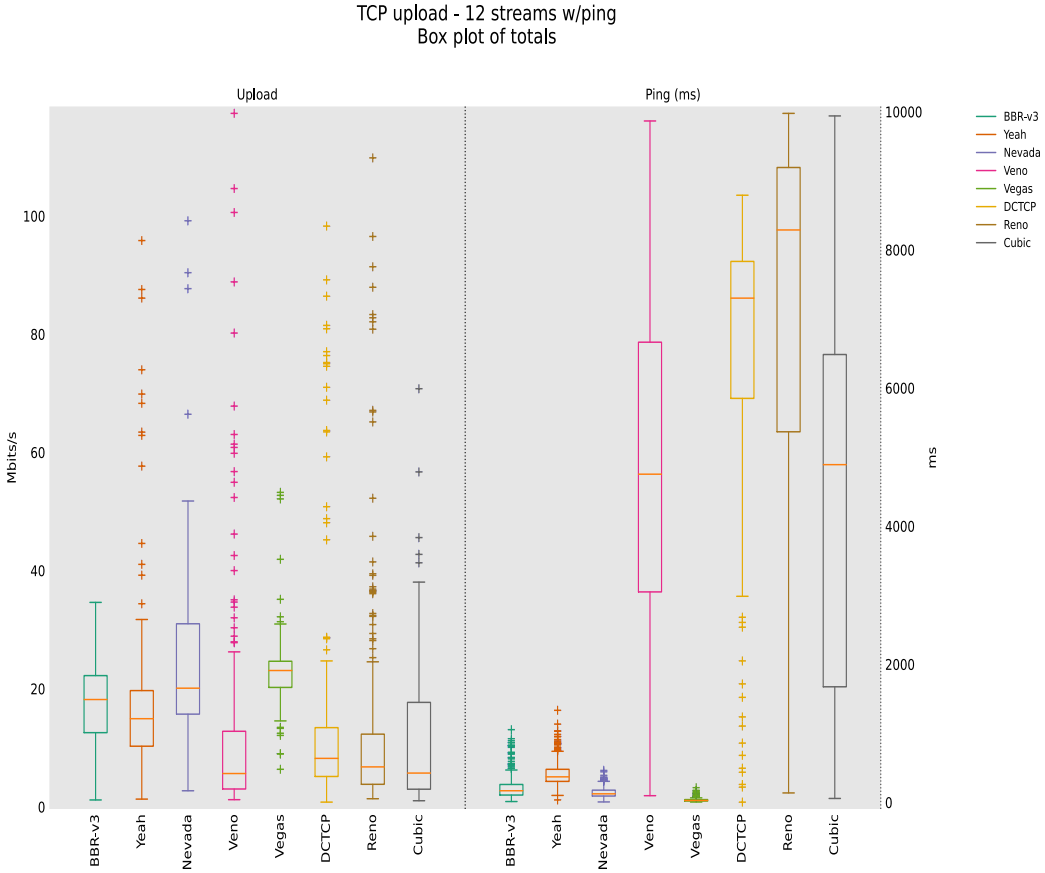

Figure 11. Twelve streams in the Upload test for all CCAs

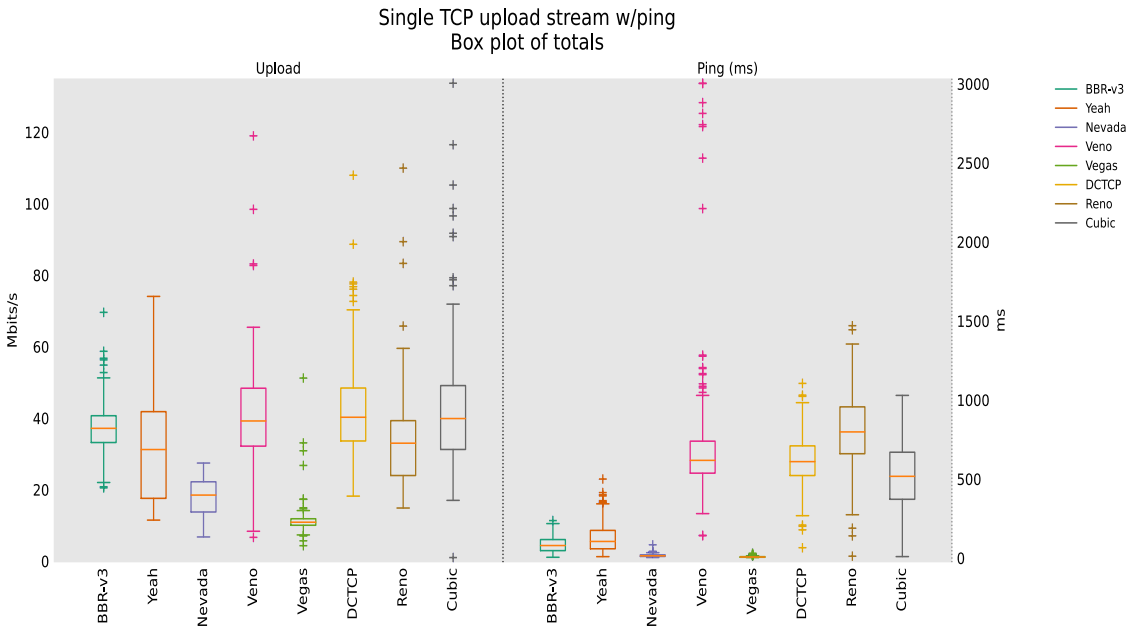

Figure 12. Single stream for all CCAs

Fig 12 uses a single stream in which all of the bandwidth of the channel, in this case, Wi-Fi 4, was presented to it, and BBR v3 and Cubic were at the top in terms of throughput, but the latency of BBR v3 was a lot lower than Cubic. Thus, making BBR v3 an overall winner.

4.3. Upload streams test for wired scenario

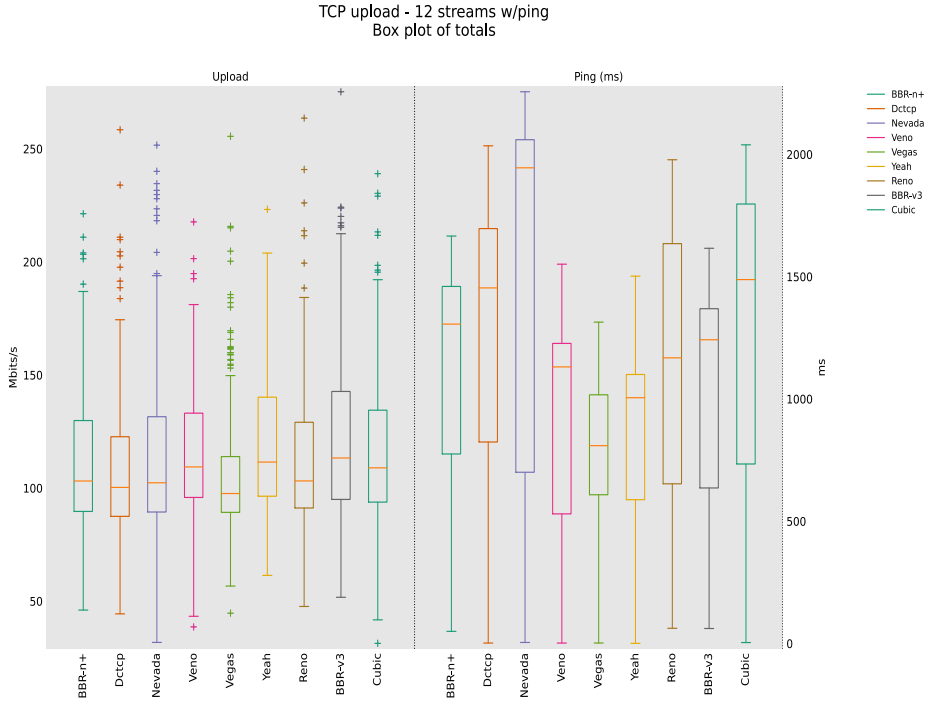

Figure 13. Twelve streams in Upload for all CCAs

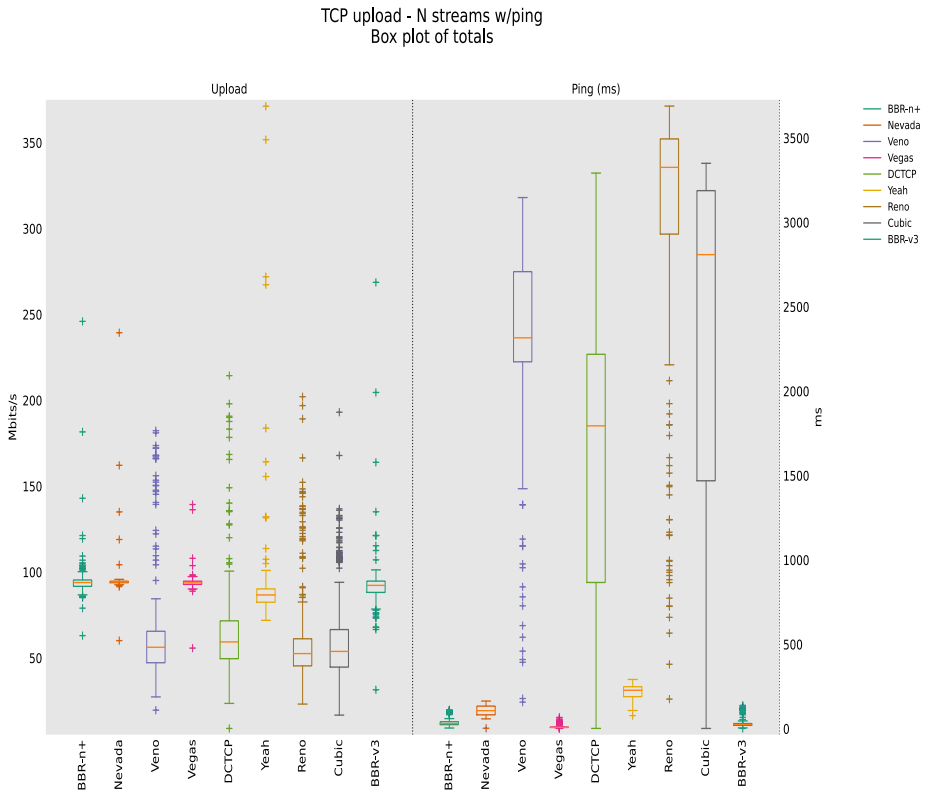

Figure 14. Sixteen streams in Upload for all CCAs

For the wired uplink results of Figs 13 and 14, BBR demonstrates strong performance with a clear emphasis on managing latency and avoiding bufferbloat, even if algorithms like Cubic and Reno can achieve slightly higher peak or median throughputs by being more aggressive and filling network buffers. The choice between BBR and these other algorithms often comes down to priority: raw throughput (Cubic/Reno) versus lower latency and more predictable performance (BBR).

4.4. Real-time Response Under Load: Exclusively Best Effort

(RRUL_BE) test for wired scenario

RRUL_BE is a variation of the Realtime Response Under Load (RRUL) test that runs without classification, meaning it does not apply Differentiated Services Code Point (DSCP) markings or Quality of Service (QoS) prioritization to the traffic streams. This makes it the fairest test because all data flows are treated equally, allowing for a pure measurement of network performance without interference from prioritization mechanisms.

BBR, as shown in Figs 15 and 16 (especially BBR-n+ and BBR v3), consistently delivers significantly lower latency across the board compared to all other algorithms in this best-effort scenario. This is a primary design goal of BBR. In downloading throughput, BBR is generally outperformed by Yeah, Nevada, and Veno, which achieve higher median throughputs by being more aggressive. In upload throughput, BBR is generally outperformed by Cubic and Reno. While BBR might not always achieve the highest raw throughput in a purely best-effort context, its superior latency performance makes it more suitable for applications sensitive to network delay, as it actively avoids bufferbloat, which is rampant with other algorithms in this test.

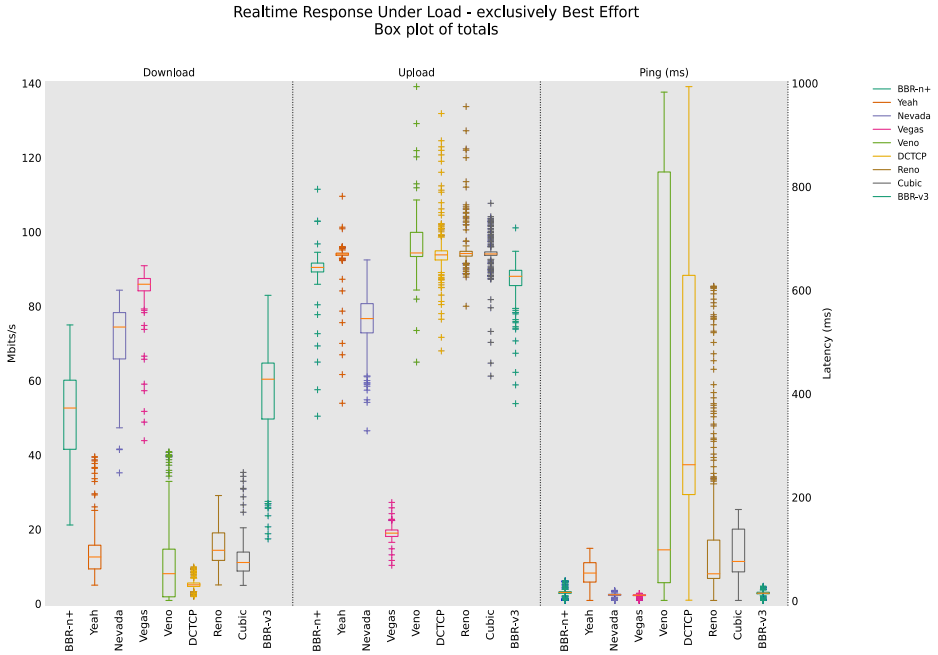

Figure 15. RRUL without classification for all CCAs

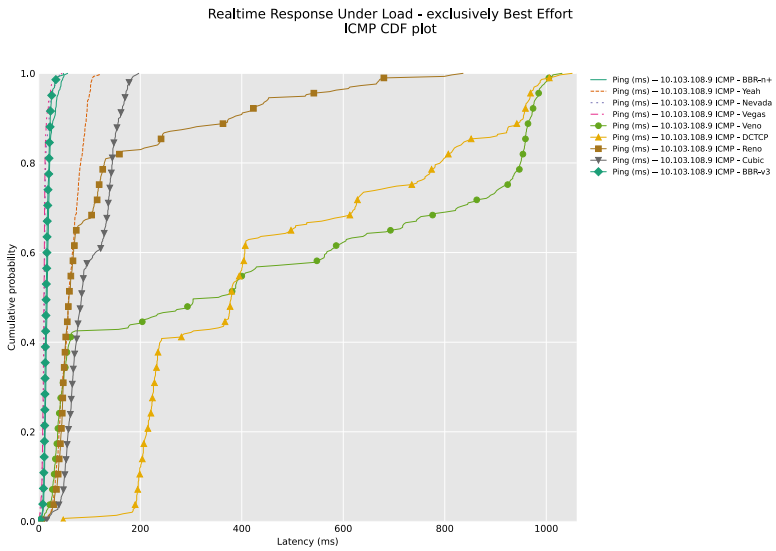

Figure 16. ICMP CDF plot

Table 5. ANOVA results for the throughputs achieved in the upload for all CCAs

Anova: Single Factor

SUMMARY

Groups	Count	Sum	Average	Variance
TCP upload sum - BBR v3	300	26448.84	88.1628	75.63685
TCP upload sum - BBR-n+	300	27355.68	91.18561	69.06302

TCP upload sum - Cubic	300	28558.21	95.19404	98.35445
TCP upload sum - Reno	300	29031.47	96.77156	507.7139
TCP upload sum - DCTCP	300	28722.5	95.74167	243.0529
TCP upload sum - Vegas	300	5812.471	19.3749	25.12146
TCP upload sum - Veno	300	29396.57	97.98857	319.7254
TCP upload sum - Nevada	300	23116.86	77.05619	103.6034
TCP upload sum - Yeah	300	28411.41	94.70471	52.00687

Table 5a. ANOVA results for the throughputs achieved in the upload for all CCAs

ANOVA

Source of Variation	SS	df	MS	F	P-value	F crit
Between Groups	1509041	8	188630.1	1136.114	0	1.941841
Within Groups	446789.2	2691	166.0309			
Total	1955830	2699				

Table 6. ANOVA results for the throughputs achieved in downloads for all CCAs

Anova: Single Factor

SUMMARY

Groups	Count	Sum	Average	Variance
TCP download sum - BBR n+	300	17225.59	57.41864	363.9073
TCP download sum - BBR-v3	300	15490.31	51.63435	359.1198
TCP download sum - Cubic	300	3666.575	12.22192	51.02462
TCP download sum - Reno	300	4596.609	15.32203	44.96165
TCP download sum - DCTCP	299	1551.392	5.188602	4.773101
TCP download sum - Vegas	300	25744.28	85.81425	31.29006
TCP download sum - Veno	300	3176.303	10.58768	114.1197
TCP download sum - Yeah	300	4144.279	13.81426	54.72315
TCP download sum - Nevada	300	21720.5	72.40168	158.7578

Table 6a. ANOVA results for the throughputs achieved in downloads for all CCAs

ANOVA

Source of Variation	SS	df	MS	F	P-value	F crit
Between Groups	2276055	8	284506.9	2164.28	0	1.941842
Within Groups	353615.7	2690	131.4557			
Total	2629671	2698				

Table 7. ANOVA results for ICMP Ping for all CCAs

Anova: Single Factor

SUMMARY

Groups	Count	Sum	Average	Variance
Ping (ms) ICMP - Cubic	350	30349.93	86.7141	2705.828

Ping (ms) ICMP - BBR n+	350	4837.034	13.8201	48.55731
Ping (ms) ICMP - BBR-v3	350	5616.921	16.04834	82.81009
Ping (ms) ICMP - Reno	350	38514.25	110.0407	23406.82
Ping (ms) ICMP - Yeah	350	18575.25	53.07215	814.257
Ping (ms) ICMP - DCTCP	350	139612.7	398.8935	90602.36
Ping (ms) ICMP - Vegas	350	3398.252	9.70929	30.84757
Ping (ms) ICMP - Veno	350	128374.2	366.7833	156912.8
Ping (ms) ICMP - Nevada	350	5790.871	10.83106	41.93828

Table 7a. ANOVA results for ICMP Ping for all CCAs

ANOVA

Source of Variation	SS	df	MS	F	P-value	F crit
Between Groups	66678509	8	8334814	273.1271	0	1.94135
Within Groups	95851515	3141	30516.24			
Total	1.63E+08	3149				

Tables 5-7 provide us with the statistical analysis for Fig 15. The threshold value "α" of 0.05 was chosen. In the three tables, we get an F value which is greater than the F critical (F crit) value, and a Probability value (P-value) of zero suggests that the results are strongly statistically significant. BBR-n+ provides better throughput than BBR-v3 and exhibits lower latency as compared to BBR-v3 and legacy congestion controls. TCP Cubic and Reno although provides an increased throughput but it is due to bufferbloat phenomenon as evident from the increased latency. DCTCP, Vegas, Veno and Nevada also struggle with latency issues. BBR-n+ overall performance in throughput and exceptional performance in reduced latency makes it a clear winner for todays real-time applications and games as well where latency is the key.

5. Conclusion

The Bottleneck Bandwidth and Round-trip propagation time (BBR) v3 is the latest iteration of BBR since its inception in 2016. Google has already shifted most of its internal WAN traffic to it. It is being used for all Google.com and YouTube public internet traffic. The importance of BBR is evident from the fact that Google is trying to shift almost all of its traffic to it and is also putting in its best efforts so that it gets included in the Linux mainline kernel. This motivated the research work in this paper, and we tested it in both typical wired and wireless environments (Wi-Fi 4 and Fast Ethernet) that are being used by millions of users at home and in the office.

We have also tested our ported version of BBR, the BBR-n+, along with the default BBR v3, using our real-time testbeds. From the thorough tests conducted via Flent, we saw that both BBR v3 and BBR-n+ gave throughputs of 97 Mbps and 95 Mbps, respectively, in Upload and 5 Mbps and 12 Mbps in Download in RRUL test with classified traffic in wired setups. The latency of BBR-n+ was found to be lower than BBR v3 in the Upload test in the wireless setup. In the RRUL test with best effort (without classification), BBR-n+ throughput was higher in the Upload and almost comparable with BBR v3 in the downlink. Although we see that pure loss-based congestion control algorithms,

Cubic, Reno, and even Yeah, gave better throughputs, it was only due to their aggressive strategy of filling the buffers that resulted in larger ping delays. On the other hand, both BBR v3 and BBR-n+ gave a better performance in terms of latency in most of the tests performed, with BBR-n+ performing better in terms of latency.

From the detailed tests performed, it is evident that BBR (both BBR v3 and BBR-n+) are indeed the game changers in the realm of congestion control algorithms. BBR-n+, with its better performance in both ubiquitous networking environments, is indeed the new leader in the congestion control block.

Data Availability Statement:

Data is available at the GitHub repository https://github.com/mahsan76/BBR-n-plus

Funding:

No funding was received for this research

Conflict of Interests:

The author declares no conflict of interest.

References

1. Postel J. Rfc0793: Transmission control protocol. RFC Editor; 1981.
2. Jacobson V. Congestion avoidance and control. ACM SIGCOMM computer communication review. 1995;25(1):157-87.
3. Tomar P, Panse P. A Comprehensive Analysis and Comparison of TCP Tahoe, TCP Reno and TCP Lite. International Journal of Computer Science Information Technologies. 2011;2(5):2467-71.
4. Xu L, Harfoush K, Rhee I, editors. Binary increase congestion control (BIC) for fast long-distance networks. IEEE INFOCOM 2004; 2004: IEEE.
5. Ahsan M, Awan MJ, Yasin A, Bahaj SA. 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.
6. Alizadeh M, Greenberg A, Maltz DA, Padhye J, Patel P, Prabhakar B, et al., editors. Data center tcp (dctcp). Proceedings of the ACM SIGCOMM 2010 Conference; 2010.
7. Brakmo LS, O'malley SW, Peterson LL, editors. TCP Vegas: New techniques for congestion detection and avoidance. Proceedings of the conference on Communications architectures, protocols and applications; 1994.
8. Fu CP, Liew SC. TCP Veno: TCP enhancement for transmission over wireless access networks. IEEE Journal on selected areas in communications. 2003;21(2):216-28.
9. Baiocchi A, Castellani AP, Vacirca F, editors. YeAH-TCP: yet another highspeed TCP. Proc PFLDnet; 2007.
10. Cardwell N, Cheng Y, Gunn CS, Yeganeh SH, Jacobson V. BBR: Congestion-based congestion control. Communications of the ACM. 2017;60(2):58-66.
11. 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.
12. Cardwell N, Cheng Y, Gunn CS, Yeganeh SH, Swett I, Iyengar J, et al. BBR Congestion Control: IETF 100 Update: BBR in shallow buffers. ACM SIGCOMM computer communication review. 2017.
13. Cardwell N, Cheng Y, Yeganeh SH, Jha P, Seung Y, Yang K, et al., editors. BBRv2: A model-based congestion control performance optimization. Proc IETF 106th Meeting; 2019.
14. Kleinrock L, editor Power and deterministic rules of thumb for probabilistic problems in computer communications. ICC 1979; International Conference on Communications, Volume 3; 1979.
15. Kleinrock L, editor On flow control in computer networks. Proceedings of the International Conference on Communications; 1978.
16. Kleinrock L. Internet congestion control using the power metric: Keep the pipe just full, but no fuller. Ad hoc networks. 2018;80:142-57.
17. Cardwell N, Cheng Y, Yang K, Morley D, Yeganeh SH, Jha P, et al., editors. BBRv3: algorithm bug fixes and public internet deployment. Proc IETF 117th Meeting; 2023.
18. Ahsan M, Muhammad SS. TCP BBR-n: Increased throughput for wireless-AC networks. PLoS One. 2023;18(12):e0295576.
19. 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.
20. Ahsan M. BBR-n+ test results and statistical analysis (ANOVA/MANOVA) [dataset] GitHub2025 [Available from: https://github.com/mahsan76/BBR-n-plus/.
21. 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.
22. Riley GF, Henderson TR. The ns-3 network simulator. Modeling and tools for network simulation: Springer; 2010. p. 15-34.

23. Gomez J, Kfoury EF, Crichigno J, Srivastava G. Evaluating TCP BBRv3 performance in wired broadband networks. Computer Communications. 2024;222:198-208.
24. Høiland-Jørgensen T, McKenney P, Taht D, Gettys J, Dumazet E. RFC 8290: The flow queue CoDel packet scheduler and active queue management algorithm. RFC Editor; 2018.
25. Claypool S, Chung J, Claypool M, editors. Comparison of TCP congestion control performance over a satellite network. International Conference on Passive and Active Network Measurement; 2021: Springer.
26. Wei W, Xue K, Han J, Xing Y, Wei DS, Hong P. BBR-based congestion control and packet scheduling for bottleneck fairness considered multipath TCP in heterogeneous wireless networks. IEEE Transactions on Vehicular Technology. 2020;70(1):914-27.
27. Zeynali D, Weyulu EN, Fathalli S, Chandrasekaran B, Feldmann A, editors. BBRv3 in the public Internet: a boon or a bane? Proceedings of the 2024 Applied Networking Research Workshop; 2024.
28. Jain RK, Chiu D-MW, Hawe WR. A quantitative measure of fairness and discrimination. Eastern Research Laboratory, Digital Equipment Corporation, Hudson, MA. 1984;21(1):2022-3.
29. 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.
30. Keselman HJ, Huberty CJ, Lix LM, Olejnik S, Cribbie RA, Donahue B, et al. Statistical practices of educational researchers: An analysis of their ANOVA, MANOVA, and ANCOVA analyses. Review of educational research. 1998;68(3):350-86.
31. Kim H-Y. Analysis of variance (ANOVA) comparing means of more than two groups. Restorative dentistry endodontics. 2014;39(1):74.