Testing the impact of in-transmission bandwidth and delay variation on selected TCP variants

Konrad Gniaz[1]
Supervisor(s): Fernando Kuipers[1]**, Adrian Zapletal**[1]
[1]EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 13, 2025

Name of the student: Konrad Gniaz
Final project course: CSE3000 Research Project
Thesis committee: Fernando Kuipers, Adrian Zapletal, Asterios Katsifodimos

Abstract

The Transmission Control Protocol (TCP) remains the cornerstone of modern network communication, enabling reliable and ordered data delivery across a wide range of network environments. Despite its ubiquity, TCP's variants' performance under extreme and highly variable network conditions remains insufficiently explored. This paper investigates the behavior of two common TCP variants - CUBIC and BBRv1 - when subjected to dynamic bandwidth and delay fluctuation. Such conditions are increasingly common in real-world wireless networks and can have a significant impact on TCP flows. We conduct a series of tests using the ns-3 framework, employing a dumbbell topology to simulate wireless connections. The results of 6 different testing scenarios are presented. They showcase that both algorithms experience significant packet loss in the event of bandwidth variance in-transmission, with BBRv1 adapting to these changes better, but not dominating over CUBIC in a multi-flow connection. In addition, both TCP variants experience harsh throughput drops and lose very few packets in the event of delay spikes. When faced with both delay and bandwidth variance, BBRv1 experiences high packet loss while CUBIC's connection remains stable.

1 Introduction

The Transmission Control Protocol (TCP) is a network protocol that is a fundamental part of transmitting data over the World Wide Web by providing reliable end-to-end byte streams over networks. It provides aid in error detection, tools for packet retransmission and flow and congestion control. Connections over TCP are common in all types of networks and the majority of internet traffic uses TCP. Therefore, it is vital for us to know its boundaries and limits. Testing TCP under extreme conditions helps with maximizing efficiency of TCP connections and prevent potentially catastrophic errors.

Currently, not enough research has been done to consider TCP's testing exhaustive. While time-varying bandwidth and delay in wireless connections have been investigated, they've been a topic of research dating back to the early-to-mid 2000s, and were highly limited to specific environments and TCP variants that were of common use back then. Recent work on this matter, on the other hand, either covers a performance comparison of TCP protocols [14] or suggests improvements to the existing TCP infrastructure [12], but rarely does it test the currently available TCP variations under extreme circumstances. This research paper aims to aid in resolving that issue by contributing to the testing of two commonly used TCP variants under one of such extremes.

"How do TCP CUBIC and BBRv1 perform when bandwidth or delay varies significantly during transmission?" is the question that this paper aims to answer. It showcases how the BBRv1 and CUBIC Congestion Control Algorithms, which are both very popular across connections over TCP, behave when utilized in wireless connections with varying bandwidth or varying delay on the bottleneck link.

A dumbbell topology for wireless connections is utilized in order to make the connections. The ns-3 framework is used in order to recreate such a topology and properly simulate the experiments. Six different testing scenarios are utilized in order to test the TCP variants' performance under a variance in delay, bandwidth, or both. Both single and multi-flow connections are tested. The variants' throughput and congestion window size changes are measured, as well as their packet loss on each connection. In the case of multi-flow connections, the possibility of domination of one variant over the other is checked by measuring the Jain's Fairness Index (JFI).

The tests show that BBRv1 offers better adaptability to varying bandwidth than CUBIC, as it reaches the new bottleneck bandwidth fast in case of a sudden bandwidth increase, and adapts equally as fast in case of a sudden bandwidth decrease. However, it does not dominate over it in short, low-bandwidth connections with bandwidth variance. Sudden bandwidth changes mid-transmission also cause the connection to become unstable in terms of packet loss for both CCAs. In the event of delay spikes, both CCAs experience little to no packet loss, but decrease their throughput significantly during the spike. When both the bandwidth and the delay change at the same time in a connection, BBRv1 struggles to adapt and,in turn, experiences significantly larger packet loss than CUBIC.

2 Background and Related Work

This section aims to provide background on the domain of our research, as well as explain what work has already been done on the topic.

2.1 Background

The Transmission Control Protocol, more commonly known as TCP, is an internet protocol designed to provide reliable, ordered, and error-checked delivery of data between applications communicating over an IP network [4]. It operates at the transport layer of the OSI model and is widely applied in cases where reliable data transfer is essential, such as web browsing, email or file transfers. Since TCP focuses on reliability and correctness, it introduces additional latency and overhead to data transfers, which is an inevitable trade-off that is acceptable in scenarios where the former 2 characteristics are vital. However, over time, many TCP variants were developed in order to push the limits of TCP's efficiency while maintaining its reliability. The most notable and commonly used TCP variants as of now are CUBIC and BBR, both of which are congestion control algorithms (CCAs).

CUBIC is a loss-based variant of TCP - meaning it uses packet loss as the primary signal to detect network congestion - that has been the default in Linux for nearly two decades (since 2006), and has more recently become the default in most major operating systems [7]. Its name comes from its main characteristic, which is that the congestion window grows as a cubic function over time since the last congestion event, which means that window growth is independent of RTT. It is designed to work well in high bandwidth-delay product (BDP) networks. Given how long it has been in use as a standard in the Linux Operating System, it is definitely a worthwhile variant of TCP to test under previously untested conditions.

BBR (which stands for Bottleneck Bandwidth and RTT) is a TCP variant developed by Google that presents a fundamentally different approach to congestion control [2]. Unlike traditional loss-based algorithms, BBR utilizes estimation of bandwidth and delay in order to send packets at the exact rate the network can support. BBR has three versions: BBRv1 developed in 2016, BBRv2 developed in 2020 and BBRv3, which is currently still in development. While BBRv3 is utilized in the infrastructure of most large-scale providers, BBRv1 is still very common across all TCP connections [11]. Because of its characteristics, BBR is a very interesting algorithm to test under time-varying bandwidth and delay in a data transmission in order to see how well it can handle such variance and whether it can maintain the efficiency it promises to maintain.

2.2 Related Work

Contributions regarding variable bandwidth include [3, 15, 13]. The form of delay variation this paper explores is delay "spikes" - which have also been researched in the past [6, 9, 16]. Xin's and Jamalipour's paper [16] is especially interesting - it concludes that delay spikes heavily impact TCP's throughput, especially at the start of the spike(which, in their research, lasted 5-10 seconds - way longer than this paper's experiments).

One could assume that would mean that the current knowledge base about this matter is sufficient - however, there are two major caveats to the currently published research on this topic. First of all, it mostly tests low-bandwidth connections [3, 6, 9, 13, 16]. Second of all, due to the fact that the research is relatively old, TCP variants that are currently not as common were tested, such as TCP Reno, Eifel or Tahoe.

When it comes to more recent work on testing TCP, a lot of it has been done in terms of testing TCP in specific connection types [1, 5, 8, 14]. Whenever the given domain is generalized to contain and emulate the most commonplace types of connections, the conditions provided are also quite generic, as these kinds of contributions aim to evaluate performances of variants rather than testing their limits and boundaries [10, 17].

Thus, though some TCP variants have already been tested under in-transmission bandwidth and delay variance, we're still missing evaluation of the CCAs that are currently the most common under such conditions, especially in mid-to-high (>5Mbps) bandwidth wireless connections.

3 Methodology

In this section, we elaborate on the way our experiments and evaluation are set up from the theoretical standpoint and describe how our setup is implemented in order to run our simulations.

3.1 Setup

The connections over TCP can be divided into two types - wired connections and wireless connections. Wireless connections are far more common in the current day and age - and far more susceptible to variance in bandwidth and delay - which is why this paper's test scenarios are run exclusively on those. As for the setup of access and bottleneck links, a dumbbell topology (as shown in Figure 1) is utilized.

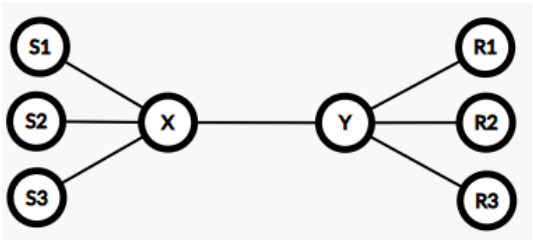

Figure 1: An example of a dumbbell topology. S and R stand for Sender and Receiver nodes respectively, connected to routers X and Y by access links. Routers themselves are connected by a bottleneck link.

In the topology itself, each access link has a bandwidth of 110% of the highest bandwidth value across the simulation. Otherwise, these properties of the links could interfere with the

test results. The bottleneck link between routers is the link at which the bandwidth and delay highly vary in each test. The connection in each experiment is time-based - the sender sends packets to the receiver for a predefined amount of time without stopping. For each test case, a buffer size of twice the Bandwidth-Delay Product (BDP) was used to reflect standard network conditions.

There are many metrics that can be chosen for measuring performance of TCP variants. Baseed on previous research made on this topic, we decided that the following properties of the flows will be measured in each experiment:

- The number of lost packets in the connection
- Congestion window size changes (in Bytes) across the connection
- Throughput (in Megabits per second) across the connection
- Fairness (in terms of Jain's Fairness Index) for multi-flow tests

3.2 Implementation

To evaluate the Transmission Control Protocol under realistic conditions, the ns-3 framework was employed as the core tool for creation of the testing environment. The ns-3 emulation framework is a versatile tool designed to bridge the gap between simulated and real-world network testing. Ns-3 was selected due to its robust support for experimentation, as many of the conditions and metrics mentioned hereinbefore and hereinafter are built-in in ns-3, making it very accessible for such research.

When varying bandwidth, the Token Bucket Filter Queue Discipline (Tbfqueuedisc) is utilized on the bottleneck links and its "Rate" value is changed throughout the connections using the SetAttribute method. When varying delay, we use SetAttribute on the bottleneck link's "Delay" value.

4 Evaluation

This section elaborates on the tests that we perform and analyses their results.

4.1 Tests Performed

Name	Duration	Bandwidth	Delay	Flow characteristics
Large bandwidth step	45s	0-15s: 5 Mbps, 15-30s: 100 Mbps, 30-45s: 5 Mbps	50ms/100ms	Single
Small bandwidth step	45s	0-15s: 30 Mbps, 15-30s: 5 Mbps, 30-45s: 30 Mbps	50ms/100ms	Single
Multi-flow bandwidth step	9s	0-3s: 1 Mbps, 3-6s: 5 Mbps, 6-9s: 1Mbps	50ms/100ms	Multi(1 for each variant)
Small delay spikes	25s	50Mbps	every 5s for 1s: 50ms, otherwise: 5ms	Single
Big delay spike	20s	10Mbps	10-12s: 300ms, otherwise: 5ms	Single
Bandwidth and delay step	20s	0-10s: 50Mbps, 10-20s: 5Mbps	0-10s: 20ms, 10-20s: 100ms	Single

Table 1: Description of each test scenario.

The detailed description of the test cases can be found in Table 1. The purpose of each test case is described further in the corresponding section.

4.2 Test Results And Analysis

Before examining the test results, it is important to highlight two key considerations. First of all, each of the figures below contains dashed lines. In the scenarios involving varying bandwidth, each dashed line represents the point at which the bandwidth increases or decreases. In scenarios involving delay spikes, dashed lines represent the start and end of each spike. Secondly, each simulation starts on the 1st second, so the time intervals at which the changes in bandwidth or delay occur are 1 second later than the table above shows.

The first 3 test scenarios were run in order to test CUBIC's and BBRv1's capabilities of handling abrupt changes in bandwidth mid-transmission. For each of the scenarios, the connection was split into three intervals, the first and the third one having the same bandwidth, and the second one having a much higher or much lower bandwidth, in order to simulate real-world situations where, for a certain period of time, the bandwidth can decrease or increase.

Scenarios 4 and 5 test the reaction of CUBIC and BBRv1 to delay "spikes" - sudden and short increases in the delay during a connection. Scenario 4 simulates a connection with frequent, smaller spikes, whereas scenario 5 simulates a connection with one big delay spike.

Scenario 6 simulates a connection where both a bandwidth drop and a delay increase are suddenly experienced mid-connection, to see how the TCP variants adapt to both at the same time.

Scenario 1 - Large Bandwidth Step

The first scenario represents a situation where a single flow in a connection experiences a sudden and high bandwidth increase and then a sudden and high bandwidth decrease to the original value. It was created to check how fast both TCP variants react to the change and how fast they adapt when bandwidth goes back to it's usual low value.

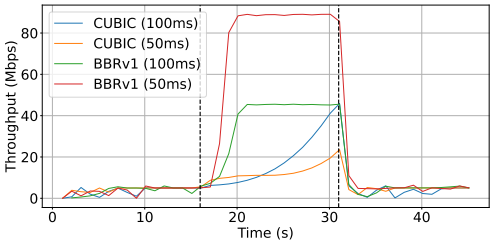

Figure 2: Throughput over time for each single-flow run in the large bandwidth step scenario.

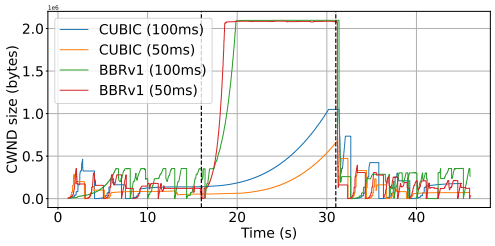

Figure 3: Congestion window size over time for each single-flow run in the large bandwidth step scenario.

Test case	Packet loss
CUBIC (100ms)	4427(8.2%)
CUBIC (50ms)	3587(8%)
BBRv1 (100ms)	5167(5.6%)
BBRv1 (50ms)	7263(4.3%)

Table 2: Packet loss for each single-flow run in the large bandwidth step scenario.

The results of this scenario's runs are shown on Figures 2 and 3 and Table 2. Both

CCAs begin adjusting their congestion window sizes correctly at moments of bandwidth variance - however, BBRv1 reaches the optimal throughput and congestion window size almost instantly, whereas CUBIC slowly ramps up in order to reach it and does not get there in the time window of the simulation. Such a result is expected - BBR is known to be really good at reaching the bottleneck bandwidth fast due to its congestion window adjustment and throughput being based on estimation of the bottleneck bandwidth, compared to CUBIC's exponential growth since the last loss event. Interestingly enough, despite CUBIC being loss-based and BBRv1 being delay-based(and thus ignoring packet loss as a sign to decrease the congestion window size) both seem to react to the bandwidth drop at the same time.

In terms of differences between higher and lower delay runs, two interesting properties can be noticed. Firstly, in its 100-millisecond run, it appears that BBRv1 miscalculated the optimal bottleneck bandwidth in its internal algorithm, most probably due to the in-transmission variance confusing it. Consequently, while it did converge on a high congestion window fast, it's throughput was capped at around half the available bandwidth, under-utilizing it. Secondly, CUBIC adjusted to the higher bandwidth faster while under higher delay - even though it reached the optimal congestion window slower at the start, which is quite unexpected. It might have been caused by the small loss causing a slight decrease in the congestion window size in the 50-millisecond CUBIC run around the 13th second, that caused CUBIC's growth function to yield smaller values due to more recent loss compared to the 100-millisecond CUBIC run.

In all the runs, the connection is quite unstable, with a 5-8% packet loss - which is expected, similarly to the fact that BBRv1 lost more packets than CUBIC (though percentage-wise, the packet loss was higher for CUBIC, as BBRv1 managed to transfer a lot more data than CUBIC in the scenario's runtime, especially in the 100Mbps bandwidth timeframe).

Scenario 2 - Small Bandwidth Step

The second scenario presents a similar situation, with two key differences - the bandwidth drops first and increases later, and the values aren't as extreme (5-30Mbps compared to the previous 5-100Mbps).

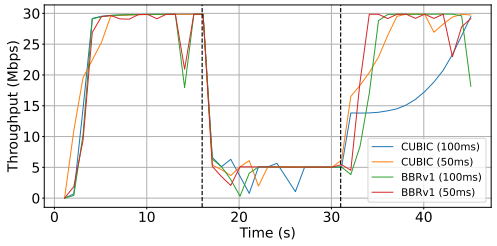

Figure 4: Throughput over time for each single-flow run in the small bandwidth step scenario.

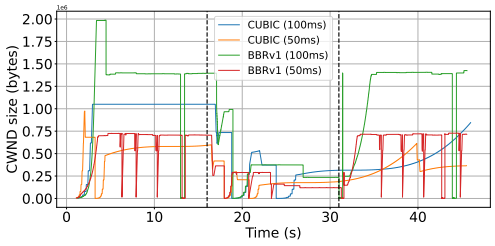

Figure 5: Congestion window size over time for each single-flow run in the small bandwidth step scenario.

Test case	Packet loss
CUBIC (100ms)	1609(1.7%)
CUBIC (50ms)	1802(1.7%)
BBRv1 (100ms)	1506(1.4%)
BBRv1 (50ms)	1754(1.6%)

Table 3: Packet loss for each single-flow run in the small bandwidth step scenario.

The results of this scenario's runs are shown on Figures 4 and 5 and Table 3. Similarly to how it was in the first scenario, BBRv1 reacts to the sudden bandwidth increase faster (as expected) and reacts to the sudden bandwidth decrease as fast as CUBIC (contrary to expectations). In addition, 50 millisecond delay runs on both CCAs result in better performance in terms of throughput and congestion window size adjustments than their 100-millisecond counterparts. What is interesting, however, is that both TCP variants struggle with maintaining the optimal 5Mbps throughput at the start of the bandwidth switch.

In terms of packet loss, all runs experienced around 1-2% - a large improvement comparatively to the previous scenario, most probably stemming from smaller bandwidth changes. This time, BBR experienced a largely similar amount of packets lost as CUBIC, which is still quite surprising.

Scenario 3 - Multi-flow Bandwidth Step

The third scenario tests something different - namely how both TCP variants will behave when competing over bandwidth on a bottleneck link with varying bandwidth in a short(9-second) connection.

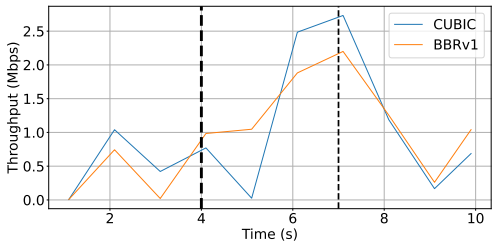

Figure 6: Throughput over time for the 50ms run in the multi-flow bandwidth step scenario.

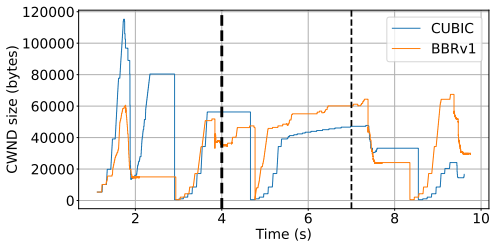

Figure 7: Congestion window size over time for the 50ms run in the multi-flow bandwidth step scenario.

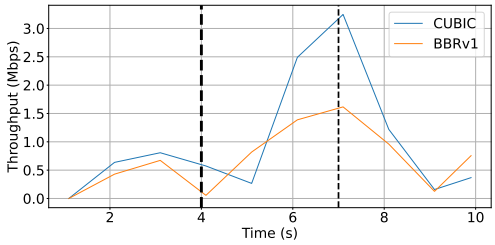

Figure 8: Throughput over time for the 100ms run in the multi-flow bandwidth step scenario.

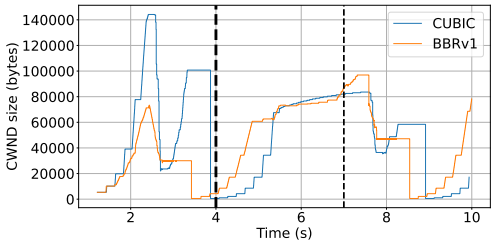

Figure 9: Congestion window size over time for the 100ms run in the multi-flow bandwidth step scenario.

Delay	Packet loss (CUBIC)	Packet loss (BBRv1)	JFI
100ms	455(27%)	218(20.6%)	0.972347
50ms	564(33.3%)	355(23.2%)	0.999628

Table 4: Packet loss for each flow and JFI for each run in the multi-flow bandwidth step scenario.

The results of this scenario's runs are shown on Figures 6, 7, 8 and 9 and Table 4. BBRv1 is the more aggressive algorithm and BBR is known to dominate over CUBIC under normal circumstances[17]. In addition, BBRv1 seems to react to bandwidth increases faster, as seen in the previous scenarios. However, it does not mean that it would dominate in such a scenario - it actually seems that both algorithms achieve similar throughput in this case, as for both 100 and 50 millisecond RTT flows, the JFI index is very high (>0.97). In the 100-millisecond scenario, CUBIC actually achieves visibly higher throughput during the 5Mbps bandwidth timeframe despite adapting to the bandwidth changes slower (in terms of adjusting congestion window size). It seems to be the case due to it grabbing more bandwidth early (in the first 2 seconds of the connection) and maintaining that advantage due to both algorithms not catching up to the available bandwidth that fast in the small timeframe of the second phase. However, it appears that CUBIC's slight prevalence would only be temporary - the trend seems to suggest that if the connection persisted, BBRv1 would begin dominating over CUBIC.

Another interesting observation is that around the first part of the connection (first 2 seconds), the average throughput exceeds the available throughput of the bottleneck link. We believe this is caused by the rapid data sending of CUBIC at the start of the connection. This causes the buffer to absorb packet bursts, which queue on the bottleneck link and are then received (seemingly) very quickly, confusing the throughput estimation algorithm.

In terms of packet loss, both variants experienced huge packet loss (20-35%), rendering the connection extremely unstable. It might seem counterinituitive that the packet loss was higher in the low-delay runs for both variants, but it is actually quite reasonable - low RTT can cause higher packet burstiness, which can lead to higher immediate loss of packets on a sudden bandwidth drop (which occurs between the second and third time interval).

Scenario 4 - Small Delay Spikes

The first scenario testing the two CCAs against delay spikes contains four of them, each lasting 1 second in 5-second intervals in a 25-second connection. Each spike isn't huge, however - the increase is only from 5 to 50 milliseconds.

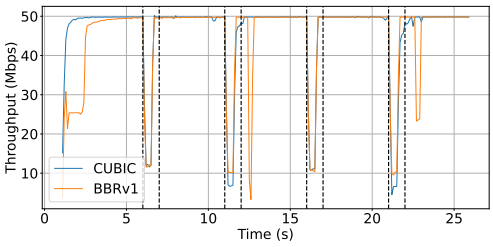

Figure 10: Throughput over time for each single-flow run in the small delay spikes scenario.

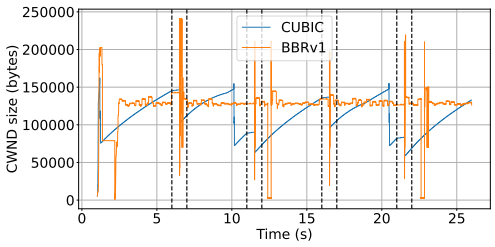

Figure 11: Congestion window size over time for each single-flow run in the small delay spikes scenario.

Test case	Packet loss
CUBIC	60(<0.1%)
BBRv1	333(0.2%)

Table 5: Packet loss for each single-flow run in the small delay spikes scenario.

The results of this scenario's runs are shown on Figures 10 and 11 and Table 5. The CCAs' behavior mimicks one by older TCP variants from [16] - their throughput experiences a harsh drop when the spikes occur. BBRv1 seems to recover from each of them perfectly fine, whereas CUBIC appears to recover a bit slower on the second and fourth spike. On the other hand, CUBIC experiences about 5 times less packets lost than BBRv1 - which again, makes sense given that CUBIC is loss-based and BBRv1 is delay-based. This being the reason for such a packet loss difference is further supported by the fact that BBRv1 maintains it's congestion window size in a fairly stable manner throughout the connection, whereas CUBIC drops it each time the spike occurs (reacting to loss).

Scenario 5 - Big Delay Spike

This scenario presents the two TCP variants with a singular, but bigger spike - one that increases the delay from 5 to 300 milliseconds for 2 seconds in the middle of a 20-second connection.

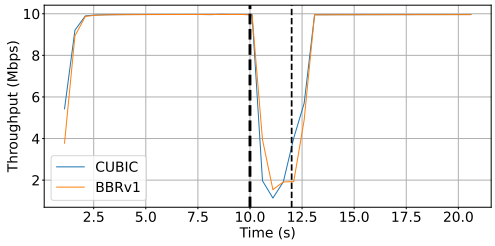

Figure 12: Throughput over time for each single-flow run in the big delay spike scenario.

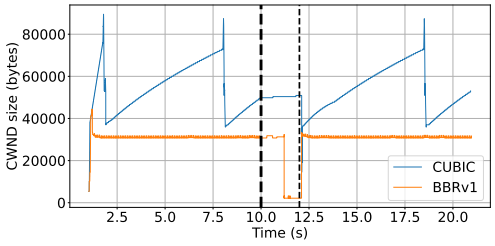

Figure 13: Congestion window size over time for each single-flow run in the big delay spike scenario.

Test case	Packet loss
CUBIC	94(0.4%)
BBRv1	0(0%)

Table 6: Packet loss for each single-flow run in the big delay spike scenario.

The results of this scenario's runs are shown on Figures 12 and 13 and Table 6. Both CCAs apppear to have recovered from the longer spike similarly well in terms of throughput. The congestion window size evolution over time appears to have produced quite odd results. Firstly, at the time of the spike (starting around mid-way through it) BBRv1 dropped its congestion window completely - which might be a malfunction stemming from the spike's occurence interfering with its probing phase. What makes this explanation more sensible is the fact that BBRv1 does not probe throughout the experiment at all unless the probe did actually happen at the time of the spike. In addition, CUBIC's congestion window size is significantly larger than BBRv1's at all stages of the experiment, even though they both

reach and maintain the maximum available bandwidth the entire time, which is extremely odd. One possible explanation (excluding graphing and ns3-specific errors) would be the low queue size (since its twice the BDP, it would be very small in a 10Mbps, 5 millisecond delay connection) impacting CUBIC's capabilities to transmit packets at the maximum possible rate.

Scenario 6 - Bandwidth And Delay Step

The last scenario presents CUBIC and BBRv1 with a situation in which both the bandwidth and the delay on the link change suddenly in the middle of the connection.

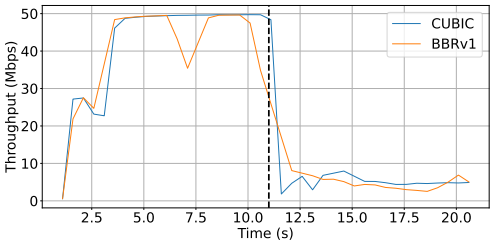

Figure 14: Throughput over time for each single-flow run in the small bandwidth and delay step scenario.

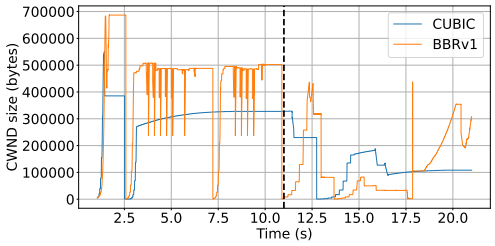

Figure 15: Congestion window size over time for each single-flow run in the bandwidth and delay step scenario.

Test case	Packet loss
CUBIC	650(1.1%)
BBRv1	2619(5.8%)

Table 7: Packet loss for each single-flow run in the bandwidth and delay step scenario.

The results of this scenario's runs are shown on Figures 14 and 15 and Table 7. Surprisingly, CUBIC appears to have adapted to the sudden changes faster than BBRv1 - in fact, BBRv1 seems still be struggling to find the new optimal congestion window size when the experiment ends. It might look like BBRv1's throughput transition is smooth, but that is most probably due to the fact that it experienced throughput loss around the time of the spike (having it drop bandwidth pre-emptively before the spike occurs is impossible as it does not know about it happening beforehand). What solidifies BBRv1's struggles with a sudden change of both the bandwidth and the delay in a connection is the fact that it experienced 5 times the packet loss of CUBIC, making it relatively unstable. It is most probably caused by a combination of BBRv1's estimation of bottleneck bandwidth being "outdated" during the second part of the connection and the fact that loss does not make BBRv1 drop its congestion window size - the impact of both being quite significant in this situation.

5 Responsible Research

When conducting research, it is important that ethical considerations are made. What it means in the context of this paper can be broken down into two main points. First of all, we

have to ensure that the data generated for and used by this paper's analyses is reproducible. Second of all, we have to ensure that the simulation of real networks via ns-3 code is accurate and reliable.

For the first point, in order to showcase how the data was generated and ensure its reproducibility, the code used in order to simulate the connections has been uploaded to a public GitHub repository that can be found under github.com/maselko13/tcp-testing. The code is annotated with comments that showcase how to replicate similar testing environments (as it does not include each test scenario specifically - test scenarios were created by changing a few lines of code each time) in order to get the same results as shown in this paper.

As for the second point, extra care has been put towards ensuring the reliability and accuracy of data coming from the simulations by solely utilizing the functions and classes that are available in the public release of ns-3 itself. As mentioned prior, ns-3 (or ns in general) is a highly respectable framework for research and has been utilized in research papers before and every contribution that is available in its standard libraries has undergone extensive testing.

Use of AI assistance

Large-language model tool usage for this paper was limited solely to gaining more knowledge about the ns-3 framework while coding the experiments. The paper was fully written by the authors with no AI assistance involved.

6 Discussion And Future Work

This section provides additional context on our findings and provides recommendations for potential work that can be done on this topic in the future.

6.1 Discussion

As the in-depth evaluation and analysis of the results have been made in the Evaluation section, this section aims to provide some additional clarification on the paper's contents.

There are two aspects of this paper that have not been elaborated upon, but could be questioned by the reader.

Firstly, the choice of testing BBRv1 instead of BBRv3 might seem confusing at first. However, as mentioned in the previous section, we decided on solely utilizing the implementations available in the ns-3 framework itself, which does not contain an official implementation of the BBRv3 algorithm. While community contributions have been considered, they proved to be quite unreliable after running experiments on them. Thus, the older version of BBR - BBRv1 is the one being tested. It is also important to highlight that BBRv1 is still quite common across computer networks [11], so testing it is by all means still relevant.

Secondly, the analyses of each scenario might appear to focus on the comparative performance of each variant rather than the general properties of both. That is because this paper aims to focus on variant-specific behavior, as the behavior of the TCP protocol itself under these circumstances has been tested in the past, whereas the evaluation of CUBIC or BBRv1 specifically has not.

6.2 Future Work

This paper aims to begin filling in the gaps in terms of this area of testing the modern TCP variants. With that said, it is not nearly enough in order to do so. Many more experiments can be run in order to extend upon this knowledge base. The suggestions we propose for further research include:

- Utilizing ns-3's base library or community contributions (as long as it is ensured that they are reliable representations of such algorithms) in order to test other modern, highly used variations of TCP against connections with varying bandwidth and delay. For example, TCP variants such as BBRv2, BBRv3 or New Reno.
- Introducing new scenarios and building upon the ones included in this paper, making the testing of TCP under such conditions more thorough.
- Running these experiments on TCP CUBIC and BBRv1 on actual, physical connections instead of using simulation frameworks such as ns-3.

7 Conclusion

The goal of this paper was to increase our understanding of the behavior of TCP CUBIC and BBRv1 in wireless connections with time-varying bandwidth, delay, or both, as the current knowledge base on this topic is quite old and limited to older TCP variants and specific types of connections.

In order to reach that goal, multiple testing scenarios were run to check their impact on fairness, throughput, congestion window size and packet loss. These scenarios were run on a wireless dumbbell topology set up in the ns-3 network simulation environment with bandwidth and delay variance on the bottleneck link. Overall, six scenarios were run - three of them featuring bandwidth variance, two of them featuring delay "spikes" and one of them combining delay and bandwidth variance in a connection.

We found that a change of bandwidth in-transmission in a wireless connection has a big impact on the stability of BBRv1's and CUBIC's connections, rendering them incredibly unreliable in case of frequent or high variance. In case of sudden bandwidth increases, BBRv1 reaches the new bottleneck bandwidth near-instantly while CUBIC requires ramp-up in order to do so. In case of sudden bandwidth decreases, despite not being a loss-based algorithm, BBRv1 adjusts its congestion window similarly as fast as CUBIC and does not experience significantly higher loss. In terms of fairness, no CCA dominates the other in short bandwidth-varying connections, but this could change if the connections persist.

When delay spikes occur, both TCP variants experience a temporary, huge drop in throughput, but manage to recover quite fast. In addition, neither of them experiences substantial packet loss (all packet losses are lower than 0.5%).

In a connection with a variance in both the bandwidth and the delay, BBRv1 struggles to find the optimal bottleneck bandwidth fast, while CUBIC manages to do so in a reasonably fast time. Consequently, it experiences way higher packet loss than CUBIC (5.8% to 1.1%).

Overall, it appears that both algorithms experience similar issues to ones tested in the research done in the past. BBRv1 appears to be better suitable for connections with in-transmission varying bandwidth or delay, whereas CUBIC performs better in the event of both changing at the same time. However, further research is needed on this topic in order to understand the limitations and behavior of these modern TCP variants better, as well as to examine the behavior of other modern CCAs such as BBRv3 under these circumstances.

References

[1] Philipp Bruhn, Mirja Kuehlewind, and Maciej Muehleisen. Performance and improvements of tcp cubic in low-delay cellular networks. *Computer Networks*, 224:109609, 2023.

[2] Neal Cardwell, Yuchung Cheng, C. Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. Bbr: congestion-based congestion control. *Commun. ACM*, 60(2):58â66, January 2017.

[3] Mun Choon Chan and Ramachandran Ramjee. Tcp/ip performance over 3g wireless links with rate and delay variation. pages 71–82, 2002.

[4] Wesley Eddy. Rfc 9293: Transmission control protocol (tcp), 2022.

[5] Junkai Fei, Xiaojun Zhu, and Ruyan Zhang. Evaluation of tcp variants on dynamic uav networks. In *2023 26th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pages 11–16. IEEE, 2023.

[6] Shaojian Fu, Mohammed Atiquzzaman, and William Ivancic. Effect of delay spike on sctp, tcp reno, and eifel in a wireless mobile environment. In *Proceedings. Eleventh International Conference on Computer Communications and Networks*, pages 575–578. IEEE, 2002.

[7] Sangtae Ha, Injong Rhee, and Lisong Xu. Cubic: a new tcp-friendly high-speed tcp variant. *ACM SIGOPS operating systems review*, 42(5):64–74, 2008.

[8] Hanlin Huang, Ke Xu, Xinle Du, Yiyang Shao, Jie Li, Xiangyu Gao, and Tong Li. Performant tcp over wi-fi direct. In *2024 IEEE/ACM 32nd International Symposium on Quality of Service (IWQoS)*, pages 1–10. IEEE, 2024.

[9] Pasi Lassila and Pirkko Kuusela. Performance of tcp on low-bandwidth wireless links with delay spikes. *European transactions on telecommunications*, 19(6):653–667, 2008.

[10] Douglas J Leith, Robert N Shorten, and Gavin McCullagh. Experimental evaluation of cubic-tcp. 2008.

[11] Ayush Mishra, Lakshay Rastogi, Raj Joshi, and Ben Leong. Keeping an eye on congestion control in the wild with nebby. In *Proceedings of the ACM SIGCOMM 2024 Conference*, pages 136–150, 2024.

[12] Gonzalo Olmedo, Román Lara-Cueva, Diego Martínez, and Celso de Almeida. Performance analysis of a novel tcp protocol algorithm adapted to wireless networks. *Future Internet*, 12(6):101, 2020.

[13] Fengyuan Ren and Chuang Lin. Modeling and improving tcp performance over cellular link with variable bandwidth. *IEEE Transactions on Mobile Computing*, 10(8):1057–1070, 2010.

[14] Sharada U Shenoy, M Sharmila Kumari, Udaya Kumar K Shenoy, and N Anusha. Performance analysis of different tcp variants in wireless ad hoc networks. In *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)*, pages 891–894. IEEE, 2017.

[15] Kai-Yeung Siu and Hong-Yi Tzeng. Performance of tcp over atm with time-varying available bandwidth. *Computer Communications*, 19(11):927–936, 1996.

[16] Fei Xin and Abbas Jamalipour. Tcp throughput and fairness performance in presence of delay spikes in wireless networks. *International Journal of Communication Systems*, 18(4):395–407, 2005.

[17] Danesh Zeynali, Emilia N Weyulu, Seifeddine Fathalli, Balakrishnan Chandrasekaran, and Anja Feldmann. Promises and potential of bbrv3. In *International Conference on Passive and Active Network Measurement*, pages 249–272. Springer, 2024.