Simulating and Analyzing the Performance of TCP Under Extreme Conditions

Impact of SDN-induced routing changes on TCP BBR

Alexandru-Mihai Șologon

Supervisor(s): Fernando Kuipers, Adrian Zapletal

[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 14, 2025

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

An electronic version of this thesis is available at http://repository.tudelft.nl/.

Abstract

Frequent route changes in modern SDN-based networks are known to severely degrade the performance of TCP Cubic. This degradation is caused by two factors: sudden RTT changes, and packet reordering which Cubic misinterprets as congestion. This research investigates how a modern alternative, BBRv3, performs under these same conditions. Using ns-3 simulations with rerouting intervals of 3, 5, and 10 seconds, we show that BBRv3 is significantly more resilient. While Cubic throughput is reduced by nearly 50% at 3-second intervals, BBRv3 performance degrades by less than 10%, as its probing mechanism does not use packet reordering as a congestion signal. We also examined a second scenario where a flow leaves a saturated link. We concluded Cubic flows take much longer to fill the newly freed up bandwidth, as much as 8 seconds for a 25% increase in available bandwidth. BBR performs much better as its able to recognize that the link is not saturated during its next probing phase and mediately fill it. Therefore, we conclude that BBRv3 is better suited for the dynamic network conditions found in SDN environments.

1 Introduction

In recent years, the biggest cloud service providers (Google, Amazon, and Microsoft) have moved away from traditional networking and are using SDN-based networks [6], [8], [7]. This architectural shift is driven by significant gains in network efficiency. It replaces the distributed logic of older systems with a central controller that can globally determine the optimal network path for all traffic and directly manage the sending rates of individual services to prevent congestion. For instance, this SDN-based SWAN system, which leverages these capabilities, demonstrated the ability to carry 60% more traffic than a traditional MPLS TE deployment [7].

A direct consequence of this active, centralized management is that flows in SDN-managed networks commonly change routes at very short intervals, often 10 seconds or less [10]. Although network operators might make routing changes to efficiently use their resources, this will not always align with application requirements, such as those for latency-sensitive tasks. Research also shows that frequent routing changes can have negative effects on TCP flows and reduce throughput to as low as 35% compared to no changes when using the default congestion control algorithm in most networks, TCP Cubic [4]. This impact on Cubic arises primarily from two factors associated with route changes. First, switching to a path with a longer RTT directly decreases the achievable throughput (W/RTT) until the congestion window adapts. Second, switching to a shorter path can cause packet reordering, which loss-based algorithms like Cubic may interpret as congestion, leading to unnecessary window reductions.

Besides Cubic there are a few other CCAs deployed in real networks. One of them is BBR, first developed by Google in 2016. Unlike Cubic which uses packet loss as the sole indicator of congestion BBR uses multiple probing methods to model the bottleneck bandwidth and RTT of the route. This allows it to pace the flow such that it fully saturates the bandwidth while keeping the queue occupancy at a minimum [2]. Ware et al shows that in 2024 they found the majority of CDN websites to be using TCP BBR [12]. Mishra et al claims that more than 10% of the internet uses BBR [9].

We show that because BBR does not use weak signals to detect congestion, the throughput is going to be only slightly affected by routing changes even as frequent as every 3 seconds. The only cause of reduced throughput is when moving from lower delay to higher delay that the flow will have to increase its congestion window to match the new RTT.

A second scenario that highlights BBR's faster adaptation occurs when a competing flow leaves a saturated bottleneck. BBR performs much better in this situation, as its probing mechanism is able to recognize that the link is no longer saturated during its next probing phase and immediately fill the available bandwidth. In contrast, Cubic flows take much longer to fill the newly freed-up bandwidth, as they are limited to a slow, incremental growth of their congestion window.

2 Background and Context

2.1 SDN

Software-Defined Networking (SDN) is an approach to network management that enables dynamic, programmatically efficient network configuration to improve performance and monitoring. By decoupling the control plane from the data plane the network devices receive all their instructions on how packets are forwarded from a centralized source called SDN Controller. This allows links to be utilized more efficiently and support more traffic without requiring hardware improvements.

2.2 Congestion Control Algorithms

The two Congestion Control Algorithms that we looked at are Cubic and BBR.

CUBIC is a TCP congestion control algorithm designed to work best in networks characterized by a large bandwidth-delay product (BDP). Its central mechanism adjusts the congestion window ($W(t)$) using a cubic function based primarily on the elapsed time (t) since the last congestion event (packet loss). This relationship is defined by the formula: $W(t) = C(t - K)^3 + W_{\max}$. In this equation, $W_{\max}$ represents the window size recorded just before the loss occurred, C is a protocol constant, and K is a parameter derived from $W_{\max}$. A key property arising from this time-based approach is that CUBIC's window growth rate is independent of the network's Round Trip Time (RTT). The cubic function also dictates distinct window growth phases: the window increases rapidly when significantly below $W_{\max}$ but slows its growth considerably as it nears $W_{\max}$, promoting stability around the point of previously detected network saturation [5].

BBR, developed by Google in 2016, uses a different method to detect congestion in the network. Unlike Cubic,

which uses packet loss as its primary signal for congestion, BBR uses minimum RTT and bandwidth bottleneck estimations to model the characteristics of the network path [2]. However, in the first version, BBR would sometimes overestimate the available bandwidth and would never respond to congestion. This made it greatly unfair to Cubic. The unfairness is addressed in the latest versions (BBRv2 and BBRv3) but it's still not completely resolved. The new versions also support ECN. This allows end-to-end notification of network congestion without dropping packets [3].

2.3 Related Work

There has been extensive work on the matter of frequent routing changes using Cubic. Past research showed that on routes with shorter RTTs (45/55 ms and 90/105 ms) if rerouting happens at 4 seconds or higher the negative effect on throughput is present but not greatly significant, less than 10%. However when the frequency is less than 4 seconds the throughput is significantly affected. As the rerouting period moves towards 0.1 s the throughput exponentially decreases until it reaches 35% of the no rerouting value [4]. While the behavior of Cubic under these conditions is well-understood, the performance of BBR in such dynamic environments remains an open question, motivating the work presented in this paper.

2.4 Predictions

Our expectations when placing TCP BBR through conditions similar to those in [4] are that BBR will be significantly less affected by packet reordering. BBR does not use packet loss as its primary congestion signal and will not react as aggressively when packet reordering is confused with packet loss. We still predict that changing to a shorter path will affect the model created by BBR and impact its performance.

3 Experimental setup

In this chapter, we introduce and describe the tools and setup used for the experiments. The code can be found here [11].

3.1 Tools

To simulate TCP behavior, we use the NS-3 network simulation software. The CUBIC congestion control algorithm is natively supported in NS-3. While the official NS-3 release includes only BBRv1, we use an implementation of BBRv3 developed by Agnieszka Brachman, which is based on the Linux kernel version [1]. The testbed was developed collaboratively within the research project team.

3.2 Topology and experiment design

Two network topologies were designed for the following experiments.

Effect of frequent routing changes on BBR throughput

Our goal with this experiment is that by switching between the two parallel links during a transmission, we simulate routing changes and observe their effects on TCP flows using either BBR or Cubic as the congestion control algorithm, across varying RTTs.

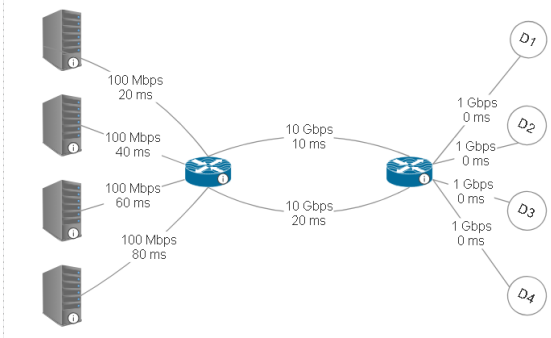

Figure 1: Topology of experiment 1

The first topology includes four sender nodes connected to a router via four 100 Mbps links, each with respective delays of 20, 40, 60, and 80 ms. This router connects to a second router through two parallel links with effectively unlimited bandwidth (10 Gbps) and delays of 10 ms and 20 ms. The second router then connects to four corresponding sink nodes via links with unlimited bandwidth and zero delay. During the total duration of the simulation the bottleneck remains the same constituted by the 100 Mbps link that connects senders to the first router. This design ensures that the only variable between the two available paths is the RTT.

Effect of rerouting a flow through a saturated link

The goal of this experiment is to observe how periodically introducing a new flow into an already saturated link affects the throughput of both the rerouted flow and the existing flows, depending on the congestion control algorithm in use.

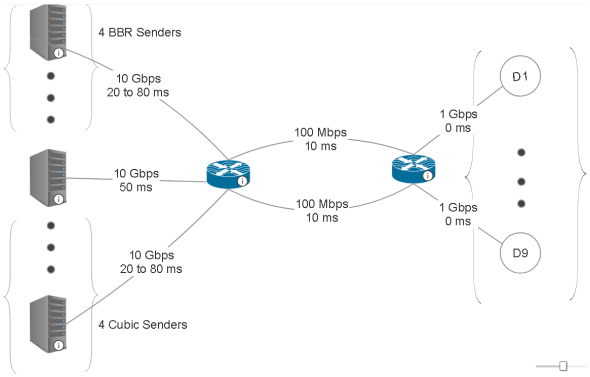

Figure 2: Topology of experiment 2

This experiment involves nine senders: four static TCP Cubic senders, four static BBR senders, and one dynamic sender that will use either BBR or Cubic and be periodically rerouted during the simulation. All senders are connected to the first router through 10 Gbps links.

Each group of static senders (Cubic and BBR) is connected to the first router through links with delays of 20 ms, 40 ms, 60 ms and 80 ms. The dynamic sender is connected through a link with a delay of 50 ms.

The first router is connected to the second router through two parallel links, each with 100 Mbps bandwidth and 10 ms

delay. The BBR flows are routed through the first link, and the Cubic flows through the other. The dynamic flow is initially on the first link. During the simulation, this flow is periodically switched between the two links to simulate rerouting events.

3.3 Metrics

Performance metrics are congestion window size (captured upon change) and average throughput sampled every 100 ms. These metrics are used to evaluate the performance and behavior of the protocol in the proposed scenarios.

4 Results and evaluation

4.1 TCP performance under frequent routing changes

The first setup was run using six different parameter variations. For each of the 2 CCAs we run 3 simulations where the rerouting happens every 3, 5 and 10 seconds.

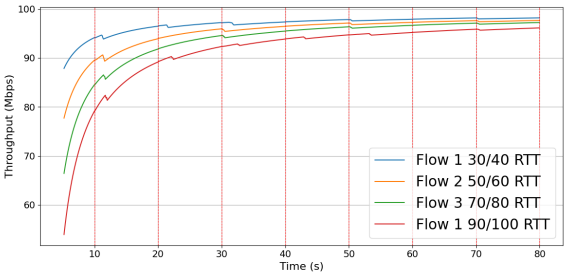

Figure 3: Throughput when rerouting 4 BBR flows every 10s

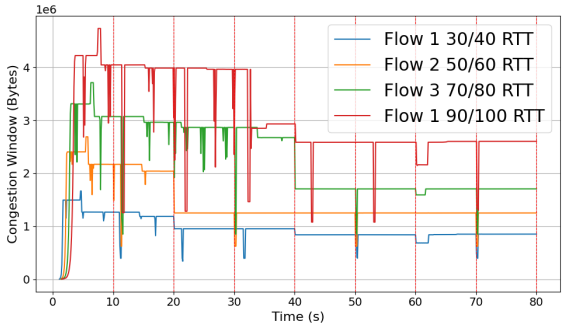

Figure 4: Congestion window when rerouting 4 BBR flows every 10s

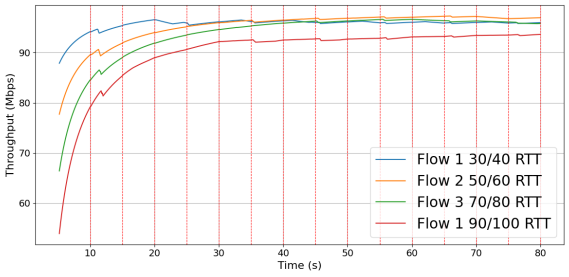

Figure 5: Throughput when rerouting 4 BBR flows every 5s

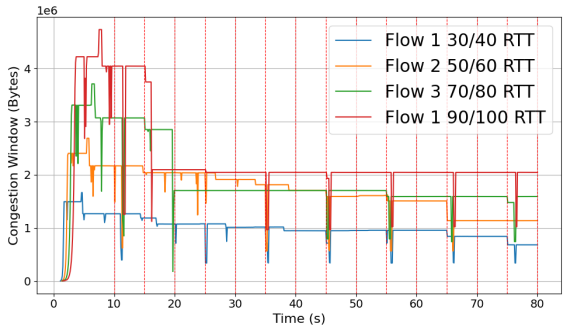

Figure 6: Congestion window when rerouting 4 BBR flows every 5s

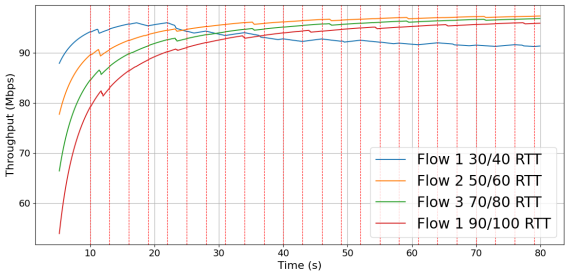

Figure 7: Throughput when rerouting 4 BBR flows every 3s

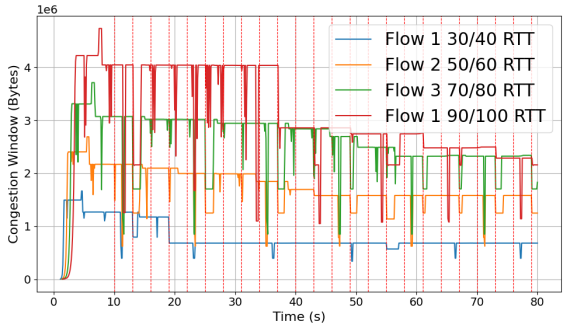

Figure 8: Congestion window when rerouting 4 BBR flows every 3s

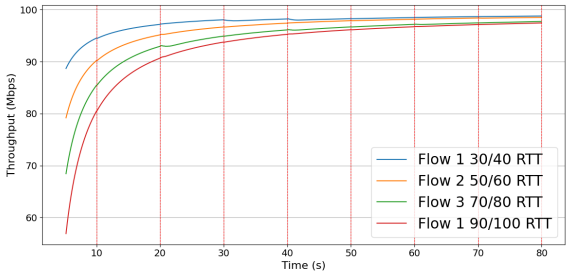

Figure 9: Throughput when rerouting 4 CUBIC flows every 10s

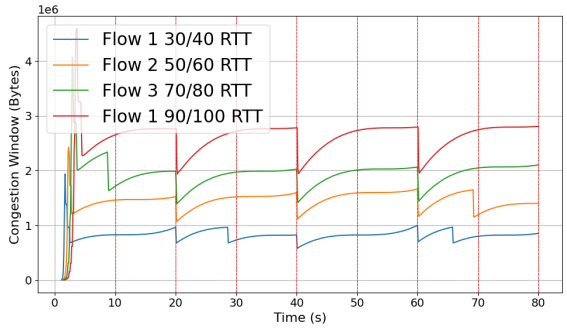

Figure 10: Congestion window when rerouting 4 CUBIC flows every 10s

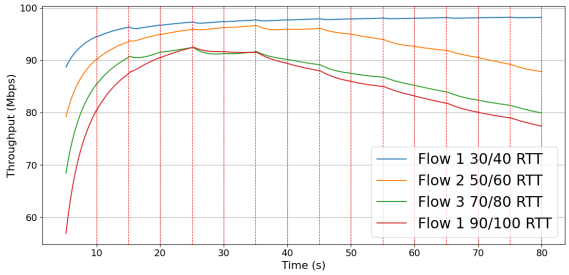

Figure 11: Throughput when rerouting 4 CUBIC flows every 5s

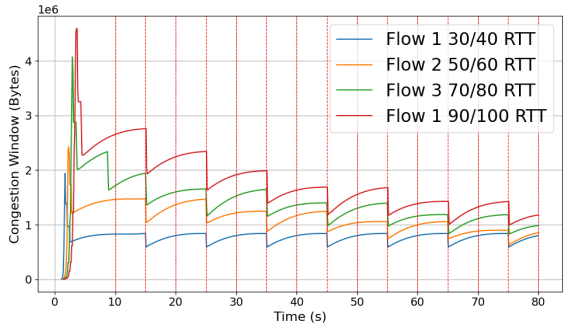

Figure 12: Congestion window when rerouting 4 CUBIC flows every 5s

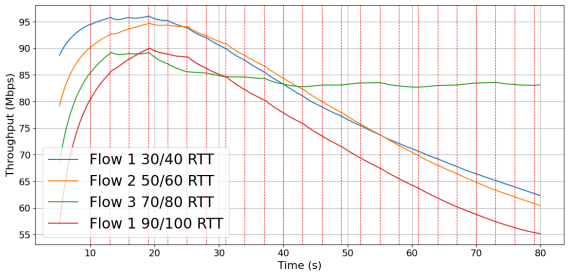

Figure 13: Throughput when rerouting 4 CUBIC flows every 3s

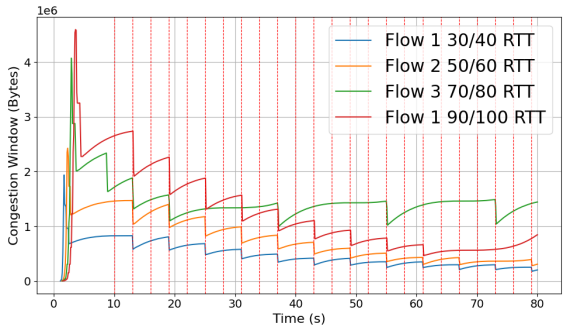

Figure 14: Congestion window when rerouting 4 CUBIC flows every 3s

From the throughput of the BBR flows when rerouting takes place at 10, 5, and 3 seconds intervals in figures 3, 5, and 7, we observe that rerouting at 10-second intervals has almost no impact on the average throughput of the BBR flows. However, in setups with more frequent rerouting, the low-delay flow (30/40 RTT) is noticeably affected. This is to be expected, because on shorter routes, that 10 ms variation causes a disproportionately larger mismatch between the current congestion window and the new optimal value. Its average throughput drops to 94 Mbps and 91 Mbps at 5-second and 3-second intervals, respectively, compared to 98 Mbps with less frequent rerouting. In contrast, flows with higher RTTs remain largely unaffected, maintaining average throughput of around 95–96 Mbps regardless of the rerouting frequency. Looking at the Congestion Window figures for the same flows in figures 4, 8, 6 we can get a more in-depth picture why the throughput is lower. We can see that BBR will lower the congestion window in response to rerouting from high to low RTT but its able to almost immediately recover. Sometimes BBR is not able to recover fast enough, and this results in a small negative effect on throughput such as figure 4 at the 60-second mark. This occurs because the rerouting frequency is very close to BBRv3 probing frequency. Here we can see that the CWND is not able to recover and keeps decreasing. This results in a greater than usual performance degradation for the flow.

From the throughput of cubic flows that get rerouted every 10, 5 and 3 seconds, in figures 9, 11, 13, we observe that rerouting at 10-second intervals has almost no impact on the average throughput of the CUBIC flows either. However,

when rerouting occurs more frequently, the impact becomes significant across all flows. At a 5-second interval, throughput drops below 90 Mbps for the three higher RTT flows, with two of them falling below 80 Mbps. At a 3-second interval, the second highest RTT flow remains just below 85 Mbps, while the other three experience a sharp decline, reaching values around 60 Mbps, representing a drop of nearly 50%. From the figures of the congestion window 10, 11, 14 we can observe that rerouting from a higher RTT to a lower RTT is almost universally interpreted as congestion and as a consequence the size of the CWND is reduced. This is a direct result of packet reordering; packets sent on the new, faster route arrive ahead of older packets still in transit, causing a burst of duplicate acknowledgments that Cubic incorrectly attributes to network congestion. This does not cause a significant impact at 10-second intervals, as a switch from a higher to a lower RTT path only occurs once every 20 s. This is because Cubic has time to restore the CWND to a degree high enough to saturate the link. When the rerouting happens more frequently the CWND cannot recover and keeps decreasing. This is especially impactful on the higher RTTs flows. This confirms our prediction: while CUBIC is highly sensitive to frequent rerouting, BBR is affected to a much lesser extent, showing performance degradation of less than 10%.

4.2 Effect of rerouting a flow through a saturated link

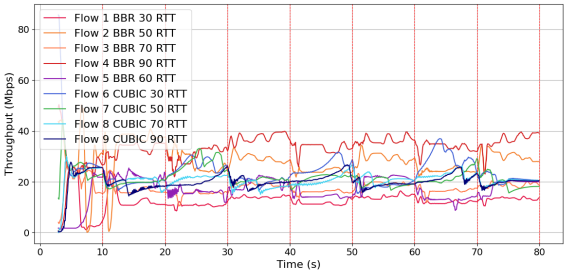

Figure 15: Throughput when rerouting a BBR flow through a saturated link

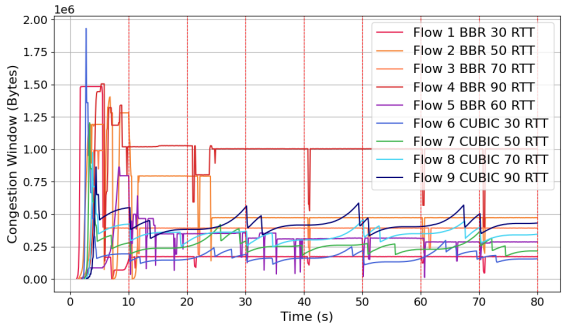

Figure 16: Congestion window when rerouting a BBR flow through a saturated link

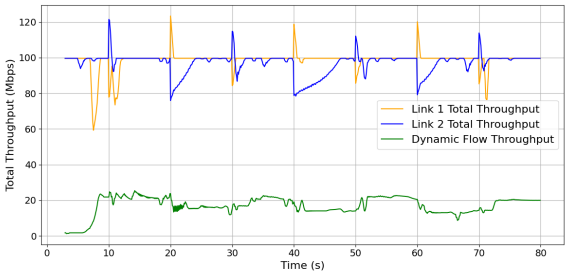

Figure 17: Cumulated throughput inside the 2 links when rerouting a BBR flow through a saturated link

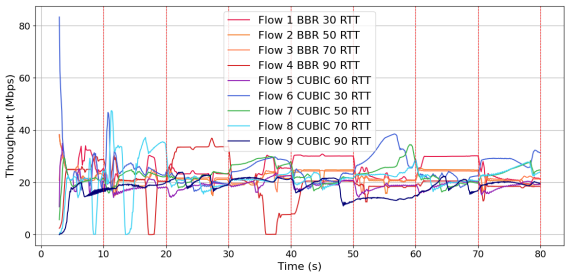

Figure 18: Throughput when rerouting a Cubic flow through a saturated link

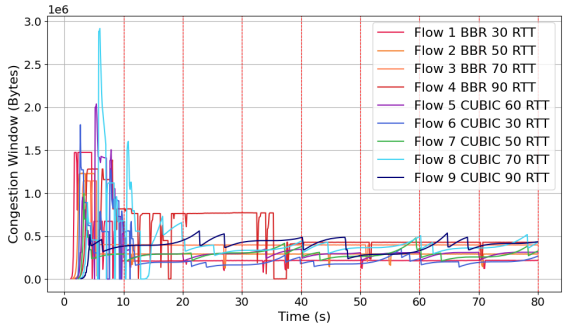

Figure 19: Congestion window when rerouting a Cubic flow through a saturated link

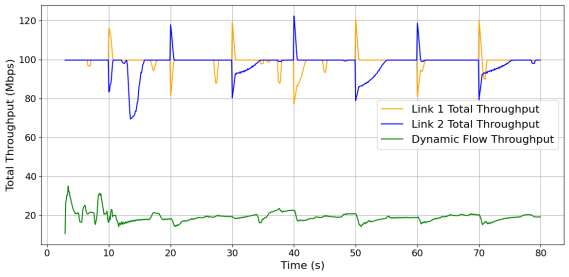

Figure 20: Cumulated throughput inside the 2 links when rerouting a Cubic flow through a saturated link

In this experiment there are 9 flows split between two identical bottleneck links. One of the flows gets rerouted at specific intervals between the 2 links. On each of the links the flows which do not get rerouted will share the same congestion control algorithm either Cubic or BBR. Two experiments were run one with the dynamic flow using Cubic and one using BBR.

The figures 15, 16, 17 show the per flow throughput, congestion window evolution and per link and dynamic flow throughput for the first simulation. Figure 15 shows us that the higher RTT BBR flows are unfair to the others. The 90 ms and 70 ms ones maintain for the whole duration around 30 Mbps each which means that they constitute more than 60% of the total flow on their link despite the fact that they share with 2 or 3 other flows. From figure 17 we can see that BBR reacts very well to the dynamic flow leaving the link. Due to BBR's reliance on its probing mechanism, the link almost immediately becomes saturated as BBR recognizes the newly available bandwidth.

The Cubic flows manage to have an evenly split bandwidth even when the 5th flow enters the route. In figure 17 we see that Cubic is slow to saturate the new capacity after a flow leaves. This happens because Cubic slowly increases its congestion window based on time until it encounters congestion, as it cannot directly "detect" if more bandwidth has become available.

The dynamic flow maintains an almost constant throughput. Slightly lower than 20 Mbps in the BBR link and slightly higher in the Cubic link.

The figures 18, 19, 20 show the per flow throughput, congestion window evolution and per link and dynamic flow throughput for the second simulation. The trend of higher RTT BBR flows taking the most bandwidth does not hold here. The flows achieve a mostly even split with the lowest RTT flow having a bigger throughput. The trend of BBR being fast to react to bandwidth changes does however hold up. In figure 20 we can see that the throughput recovers almost as fast as in the last simulation, the difference being the presence of a cubic flow on the link.

The Cubic flows remain even when all 5 of them are present. When the dynamic one leaves we can see that the lower RTT benefit the most from the newly available bandwidth to the great detriment of the 90 ms Cubic flow.

The dynamic flow behaves as expected. It's treated fairly by both BBR and Cubic flows. Switching the route causes congestion and the flow has to lower its congestion window as a consequence which then grows until the next switch. It maintains a constant 20 Mbps throughout the whole duration of the experiment.

5 Responsible Research

No confidential or sensitive information was handled during this research. There are no ethical implications of this study.

6 Usage of Large Language Models

We have not used any LLM created content when writing this thesis. During this project we used it exclusively for implementation help and proofreading for grammatical errors.

7 Limitations and Future Work

The tests in this study were run in the NS-3 simulator. A real-world testbed would improve confidence in these results because it includes network hardware and operating system behaviors not captured by the simulation. The BBRv3 implementation, while following the Linux Kernel code, is not an official release and may differ from the mainline version. Therefore, it is necessary to redo these experiments on a physical setup. This would confirm that the observed behavior is correct and not caused by inaccuracies in the simulation or the implementation.

This paper focused only on long-lived TCP flows. This is a limitation because the majority of internet traffic consists of short, latency-sensitive flows. For these flows, metrics like completion time are more important than the steady-state throughput measured in our tests. Future work should therefore extend this analysis to short flows and evaluate how modern CCAs maintain low latency under frequent route changes.

8 Conclusion

This research examined the impact of frequent routing changes on TCP flows using either BBR or Cubic congestion control. It looked at 2 situations, one where flows are rerouted between 2 non-bottleneck links with different delays. We showed that when moving from a higher RTT route to a lower RTT route with frequencies of 5 s or lower Cubic throughput is greatly reduced, as low as 50%. This is caused by packet reorder which is interpreted as congestion. Unlike Cubic packet reorder has no effect on BBR. As a result BBR flows are affected by less than 10% even when rerouting happens as fast as every 3 seconds. The cause is how BBR tracks available bandwidth using probing and doesn't react to weak signals for congestion.

The second scenario examined is how flows react when one is introduced or taken out of a saturated bottleneck. We concluded Cubic flows take much longer to fill up the newly freed up bandwidth, as much as 8 seconds for a 25% increase in available bandwidth. BBR performs much better as its able to recognize that the link is not saturated during its next probing phase and immediately fill it.

In conclusion BBR adapts better to network changes and is more suited for SDN-based networks. It upholds its promise of performing better in unstable network conditions

References

[1] Agnieszka Brachman. ns-3-dev - dev-bbrv3-fixed branch. https://gitlab.com/agnieszka.brachman/ns-3-dev/-/tree/dev-bbrv3-fixed?ref_type=heads, 2025. Accessed: 2025-06-02.

[2] Neal Cardwell, Yuchung Cheng, C. Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. BBR: Congestion-Based Congestion Control. *ACM Queue*, 14(5):20–53, September 2016.

[3] Neal Cardwell, Ian Swett, and Joseph Beshay. BBR Congestion Control. Internet-Draft draft-ietf-ccwg-bbr-

02, Internet Engineering Task Force, February 2025. Work in Progress.

[4] Radu Cârpa, Marcos Dias de Assunção, Olivier Glück, Laurent Lefèvre, and Jean-Christophe Mignot. Evaluating the Impact of SDN-Induced Frequent Route Changes on TCP Flows. In *2017 13th International Conference on Network and Service Management (CNSM)*, pages 1–7. IEEE, November 2017.

[5] Sangtae Ha, Injong Rhee, and Lisong Xu. CUBIC: A New TCP-Friendly High-Speed TCP Variant. *ACM SIGCOMM Computer Communication Review*, 38(3):67–74, July 2008.

[6] James Hamilton. AWS re:Invent 2016: Tuesday Night Live. YouTube Video, 2016.

[7] Chi-Yao Hong, Srikanth Kandula, Ratul Mahajan, Ming Zhang, Vijay Gill, Mohan Nanduri, and Roger Wattenhofer. Achieving High Utilization with Software-Driven WAN. In *Proceedings of the ACM SIGCOMM 2013 Conference*, SIGCOMM '13, pages 15–26. ACM, 2013.

[8] Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, Junlan Zhou, Min Zhu, Jonathan Zolla, Urs Hölzle, Stephen Stuart, and Amin Vahdat. B4: Experience with a globally-deployed software defined WAN. In *Proceedings of the ACM SIGCOMM 2013 Conference*, SIGCOMM '13, pages 3–14, Hong Kong, China, 2013. ACM.

[9] Ayush Mishra, Lakshay Rastogi, Raj Joshi, and Ben Leong. Keeping an eye on congestion control in the wild with nebby. In *Proceedings of the 2024 ACM Special Interest Group on Data Communication (SIGCOMM '24)*, pages 136–150. ACM, 2024.

[10] Waleed Reda, Kirill Bogdanov, Alexandros Milolidakis, Hamid Ghasemirahni, Marco Chiesa, Gerald Q. Maguire Jr., and Dejan Kostić. Path Persistence in the Cloud: A Study of the Effects of Inter-Region Traffic Engineering in a Large Cloud Provider's Network. *ACM SIGCOMM Computer Communication Review*, 50(2):1–13, April 2020.

[11] Alexandru Şologon. TCP-Thesis: Code for Evaluating the Impact of SDN-Induced Route Changes on TCP Flows. https://github.com/AlexandruSologon/TCP-Thesis, 2025.

[12] Ranysha Ware, Adithya Abraham Philip, Nicholas Hungria, Yash Kothari, Justine Sherry, and Srinivasan Seshan. Ccanalyzer: An efficient and nearly-passive congestion control classifier. In *Proceedings of the 2024 ACM Special Interest Group on Data Communication (SIGCOMM '24)*, pages 181–196. ACM, 2024.