On the Suitability of BBR Congestion Control for QUIC over GEO SATCOM Networks

Aitor Martin
Department of Electrical Engineering & Computer Science
University of Stavanger, Norway
a.martin@stud.uis.no

Naeem Khademi
Department of Electrical Engineering & Computer Science
University of Stavanger, Norway
naeem.khademi@uis.no

ABSTRACT

Satellite broadband connectivity has received significant level of interest in recent years, partly due to the emergence of 5th and 6th generations of cellular networks and their novel use-cases. High-throughput GEO satellites are therefore expected to become an integral part of such networks both for access and backhauling. Almost simultaneously, major breakthroughs have occurred within the Internet transport layer – i.e., with the increasing deployment of user-space QUIC protocol and BBR congestion control. The impact of these innovations and their overall performance on the Internet paths traversing over satellite links is yet to be investigated. Although traditionally TCP-splitting methods with Performance-Enhancing-Proxies (PEPs) were used to boost the transport performance over the satellite links, such approaches become hard for QUIC due to its encrypted nature. This leads to QUIC's poor performance over satellite links, which is currently being investigated by the IETF's QUIC WG. In addition, the transport performance depends on the choice of congestion control and QUIC implementation. In this work we will explore these aspects and the suitability of BBR congestion control for QUIC over SATCOM networks through real-life experimentation in an emulated testbed environment.

CCS CONCEPTS

• **Networks** → **Transport protocols**; **Network experimentation**.

ANRW '22, July 25–29, 2022, PHILADELPHIA, PA, USA

ACM ISBN 978-1-4503-9444-4/22/07…$15.00
https://doi.org/10.1145/3547115.3547194

KEYWORDS

BBR, congestion control, satellite, QUIC

ACM Reference Format:
Aitor Martin and Naeem Khademi. 2022. On the Suitability of BBR Congestion Control for QUIC over GEO SATCOM Networks. In *Applied Networking Research Workshop (ANRW '22), July 25–29, 2022, PHILADELPHIA, PA, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3547115.3547194

1 INTRODUCTION

Geosynchronous (GEO) satellite networks have been the focus of significant recent research. This is both due to the emergence of high-throughput satellites (HTS) and the 5th/6th generation of cellular networks (5G/6G) and their novel use-cases. In such networks, broadband services can either be delivered by the ISPs to the end-user either directly or through a SATCOM-enabled cellular network in locations where terrestrial connectivity is limited or non-existent.

Simultaneously, recent years have witnessed major innovations within the Internet transport layer. This has mainly been focused on two areas: (1) de-ossifying transport protocol stack by deploying QUIC – a user-space transport protocol with a UDP substrate – to traverse the misbehaving middleboxes which made the transport evolution impossible [1]; (2) improving congestion control (CC) mechanisms to better utilize the available bandwidth while maintaining low latency through deployment of Bottleneck Bandwidth and Round-trip propagation time (BBR) congestion control [2]. Initially designed by Google and currently under standardization at the IETF, QUIC is a UDP-based, reliable, multiplexed and fully-encrypted transport protocol, aiming to solve many of TCP's shortcomings [3][4][5]. Although the IETF's QUIC standard specifies a CC similar to the TCP NewReno [5], different implementations of QUIC offer support for BBR CC. BBR's objective is to adapt the congestion window (cwnd) according to the actual level of congestion on the path by taking into consideration the increase in Round-Trip Time (RTT) and maximum achievable bandwidth – i.e., by mostly maintaining cwnd around an optimal value above which an increase in cwnd ceases to yield more bandwith yet incurs more latency [2]. Initial investigation of BBR's performance

over TCP and its related coexistence issues with loss-based CCs [6] has lead to the introduction of BBRv2 [7]. Unlike BBR, which was deemed to be *too aggressive* due to lack of reaction to packet loss and thus penalizing competing loss-based CCs, BBRv2 aims to score a better level of fairness by reacting to loss beyond a certain threshold.

Preliminary investigations of web performance with BBR-enabled QUIC over satellite links show that BBR can outperform the loss-based CUBIC [8] in presence of packet loss [9]. While the work in [9] focuses on small and medium-size web objects, a more thorough and general investigation of these CCs over satellite links particularly with the addition of BBRv2 is yet to be performed. Satellite links are shown to pose significant challenges to the transport protocols and congestion control algorithms [10] due to their high bandwidth-delay product (BDP) and likely asymmetric nature. While these issues were traditionally mitigated through the use of connection splitting methods and Performance Enhancing Proxies (PEPs) for TCP traffic [11], this has been proven to be very challenging for QUIC due to its end-to-end encrypted nature limiting the level of optimization methods that can be implemented on the middleboxes (e.g., SATCOM gateways). Recent discussions at the IETF aiming to address this issue involve several potential approaches: (a) improving end-to-end QUIC performance tailored for SATCOM environments, e.g., providing signaling of essential path characteristics (e.g., BDP) between the QUIC end-points [12]; and (b) voluntarily exposing certain header fields in QUIC to the middleboxes with proposals such as MASQUE [13].

This paper aims to thoroughly explore the suitability of BBR and BBRv2 for QUIC over satellite broadband links, which are yet to be fully investigated. The contributions of this paper are four-fold as follows:

(1) to the best of our knowledge, it is the first work to investigate the performance of QUIC with BBRv2 in satellite environments.
(2) it evaluates several important aspects of CC performance over long-haul satellite links such as inter- and intra-protocol fairness, latecomer fairness issues, and mice versus elephant flows.
(3) investigates the impact of QUIC implementation choice on the transport performance over satellite links.
(4) elaborates on the suitability of BBR for QUIC over satellite links, current issues and potential solutions.

The remainder of this paper is structured as follows: Section 2 outlines the related works on QUIC over satellite networks and BBR congestion control; Section 3 presents in detail the network testbed setup used for our real-life evaluations; Section 4 presents our evaluation scenarios and their experimental results; Section 5 provides a discussion based on our evaluation results on the suitability of BBR for QUIC over satellite links, its implications and potential solutions; and finally, Section 6 concludes the paper.

2 BACKGROUND

This section offers a context for our work by providing background on common transport layer issues over satellite links both for TCP and QUIC, as well as existing and proposed solutions to mitigate such issues. In addition, it presents prior works on the evaluation of BBR CC particularly over satellite links, and also investigates the works performed on the performance of different QUIC implementations.

2.1 Transport Layer over Satellite

The presence of a GEO satellite link along the network path introduces several challenges for transport layer mechanisms [14], often resulting in underutilized links. **Firstly**, as a consequence of large delays introduced by the signal propagation to GEO, the path's RTT increases to an order of 0.5-0.6 sec. This increases the transport's feedback loop, leading to the slow *cwnd* growth during both TCP slow-start and congestion avoidance phases. A high RTT thus implies a higher BDP, requiring larger buffers and windows in both the endpoints and the satellite transponder [14]. **Secondly**, satellite links can introduce significant bit error rates, which can be problematic for loss-based CCs (e.g., NewReno or CUBIC) which use packet loss as an indication of network congestion. **Thirdly**, satellite links can be asymmetrical, which can result in the upstream buffers becoming filled with ACKs and thus reducing downstream performance [15].

These challenges can be confronted by optimizing TCP mechanisms for satellite environments – for instance, satellite-optimized CCs (e.g., TCP Hybla [16]) or window scaling solutions (e.g., the TCP window scale option [17]). However, most satellite service providers have relied on PEPs to mitigate the issues of TCP over satellite [11]. These proxies usually rely on connection splitting, allowing optimizations such as local loss recovery, ACK segment spoofing [11] or the use of local CC in the satellite segment.

Nevertheless, more challenges arise QUIC traffic is introduced to the satellite networks [10]. Given the current trend with QUIC's deployment on the Internet (7.9% of all websites are already using QUIC [18]), the wide use of QUIC over satellite links is to be expected. Since QUIC advocates for complete end-to-end confidentiality with full header encryption, these PEP solutions become unfeasible, unless there is some cooperation mechanism between the endpoints and the network or some header information is exposed. The inability to use PEPs with QUIC has been shown to dramatically degrade performance in several studies [19, 20, 21].

Since then, efforts have been made at IETF's QUIC WG to study newly proposed experimental QUIC features that could boost its end-to-end performance over satellite links

[10]. These include: (1) the BDP Frame extension [12], which suggests a mechanism that allows endpoints to remember and exchange path characteristic parameters (i.e., the RTT and bottleneck bandwidth) and reuse them in following connections; (2) the ACK Frequency extension [22], which allows receivers to modify the ACK sending rate and avoid introducing congestion in the upstream link; and (3) the use of Forward Error Correction (FEC) to mitigate the effect of transmission errors in the satellite link [23].

2.2 BBRv1 and BBRv2

Traditional TCP CCs such as CUBIC or NewReno use packet loss as a sign of network congestion; on the contrary, BBR models congestion by measuring the bottleneck bandwidth and path RTT, aiming to find a *cwnd* value that optimizes the use of bandwidth resources while minimizing RTT increase. This is achieved through a series of bandwidth probing strategies that continuously search the maximum available bandwidth and re-measuring the base RTT in regular intervals.

Some early studies have shown that the first version of BBR [2] can significantly outperform CUBIC in many scenarios, especially when packet loss becomes significant [24]. However, later studies identify that BBR can be unfair towards loss-based CC, as a consequence of the aggressive STARTUP and PROBE-BW phases [6, 25, 26]. These works have also identified RTT unfairness issues.

These issues have led to an update to the algorithm, first proposed in 2019 - and referred to as BBRv2 hereafter - that aims to reduce the aggressiveness when sharing the link with CCs such as CUBIC, through the use of a more complex probing algorithm [7]. While BBRv1 did not react to packet loss at all, BBRv2 uses packet loss and Explicit Congestion Notification (ECN) signaling as inputs for the probing mechanism. BBRv2 also reduces the *cwnd* decrease rate in the PROBE-RTT phase, for a less aggressive throughput fluctuation. Several works have already evaluated BBRv2 for TCP over terrestrial networks [27, 28, 29, 30]. Authors in [27] show that BBRv2 significantly improves fairness when in competition with loss-based CC. Additionally, results in [28] and [29] provide in-depth fairness analysis, and point out convergence problems between BBRv2 flows when using large bottleneck buffers. Researchers in [30] also identified issues regarding BBRv2's capability of adapting to changing network conditions - e.g., bandwidth dynamics or random losses - and suggest new mechanisms to alleviate them.

Even though BBRv2 has been deeply evaluated experimentally over terrestrial networks and with TCP, none of these studies: (1) investigate its performance, inter- and intra-protocol fairness when traversing over high-BDP links (e.g., satellite links); and (2) investigate BBRv2 performance over QUIC. This paper investigates these two previously unexplored areas in detail.

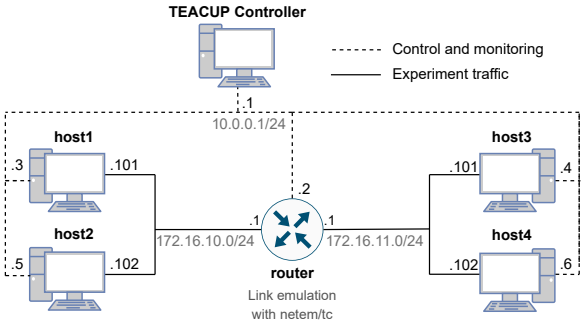


Figure 1: UiS TEACUP network testbed topology

2.3 QUIC implementations

Currently, there are several open-source IETF QUIC implementations available for experimentation. However, even though they are based on the same reference specifications, these implementations are highly heterogeneous [31]. Thus, QUIC performance evaluation results can be highly dependent on the particular implementation. As researchers in [32] show through the use of simulated and real satellite links, some implementations can perform weakly when used as a client, as a server, or even both.

3 EXPERIMENTAL TESTBED SETUP

Our approach in this work is based on real-life experimentation in an emulated network environment. We make use of a physical network testbed located at the University of Stavanger (UiS) with a dumbbell topology as shown in Figure 1. The testbed contains two pairs of hosts that act as endpoints: hosts 1-2 act as clients and hosts 3-4 act as servers. Another machine acts as a *router* between the two networks and performs satellite link emulation.

All hosts and the *router* in the networks are commodity off-the-shelf products and have identical hardware specifications: an Intel Quad Core i5-3470 @ 3.20GHz CPU and 16 GB Micron DDR3 1600 MHz RAM. They all run on the OpenSUSE Leap 15.1 Linux distribution, with Kernel 5.4.0. Each endpoint has two Ethernet interfaces, one for the control network traffic and another for the experimental traffic, and the router has an extra interface. The nodes are interconnected using a Gigabit Ethernet switch.

3.1 TEACUP for Experiment Orchestration

TCP Experiment Automation Controlled Using Python (i.e., TEACUP), developed by CAIA [33], is designed to orchestrate and run TCP experiments on emulated network testbeds. through a series of Python scripts that facilitate experiment design and automation. The scripts are run from a *controller* node running FreeBSD, which uses the *Fabric* Python library to control the experiment nodes remotely and extract statistics from them. The TEACUP architecture separates the

	SAT	TERR
One Way Delay (OWD)	300 ms	50 ms
Bandwidth (Down/Up)	20/(20|2) Mbps	20/20 Mbps
Bottleneck Buffer Size	0.25|0.5|1.0|2.0 x BDP	
Packet Loss Ratio (PLR)	0%, 0.1%, 1%	

Table 1: Network path emulation parameters

control network from the experiment network to isolate the experimental traffic from control traffic. For this work, we have extended the TEACUP to run experiments and extract statistics with multiple QUIC implementations.

3.2 Link emulation with *netem/tc*

To perform our evaluations at the transport layer level, we emulate a network path with typical characteristics of a satellite link using *netem/tc* in the *router*. Table 1 shows the parameter values used for two scenarios: the satellite scenario (SAT) and the terrestrial scenario (TERR). We consider a bandwidth asymmetry of ratio 1:10 and packet loss ratio values with an upper bound of 1% (see [34] section 5).

3.3 QUIC Traffic Generation and Logging

In this study, additional TEACUP traffic generators were developed for two major QUIC implementations: *ngtcp2* [35] and *picoquic* [36]. The former was chosen to be able to experiment with BBRv2 CC since it is one of the only QUIC stacks that implement it; the latter was selected because of the good performance reported over GEO satellite links [32]. Experiments in this work test these QUIC stacks against themselves - i.e., server and client use the same implementation.

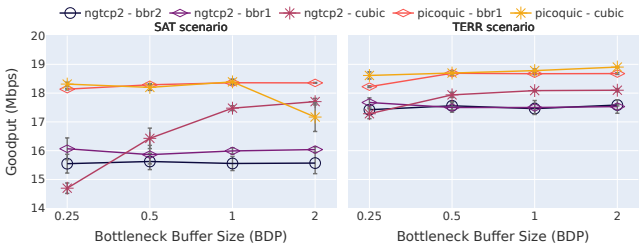
Since QUIC runs in user-space, no kernel tool was needed to extract transport layer statistics. We therefore used *qlog*, an easily structured, human-readable and standardized logging format for QUIC [37], supported by most QUIC stacks.

4 EXPERIMENTS AND RESULTS

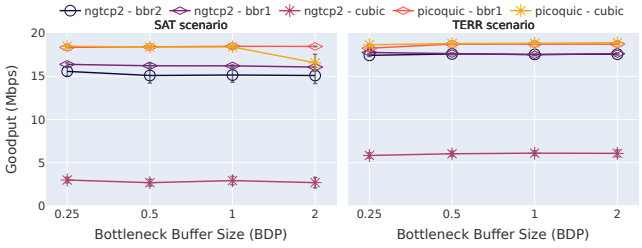
This section presents our experimental evaluation results by demonstrating how CC mechanisms perform for a variety of scenarios and parameters – e.g., bottleneck buffer size, PLRs, number of flows, object sizes, and number of objects. All experiments are run with the available CCs in QUIC implementations and over the SAT link scenario unless specified otherwise. Experiments are repeated for ten runs, which provides insight without heavily increasing experiment time. The error bars in graphs represent the standard deviation of results across experiment runs. Due to the absence of a BBRv2 implementation in *picoquic*, CC behavior in sections 4.2-4.4 is only studied with *ngtcp2* over symmetric SAT.

4.1 Bulk Download Performance

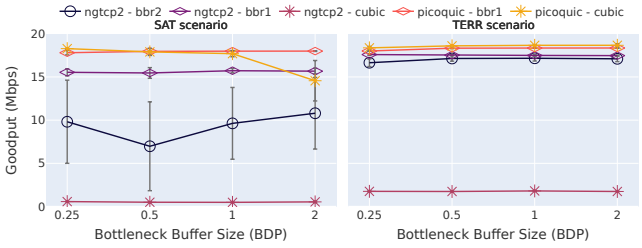
To measure the performance of bulk download traffic, the client downloads a very large file stored in the server, and the average goodput is measured after 120 seconds. Figure 2 shows the results using different QUIC implementations, CC algorithms, and bottleneck buffer sizes, for both symmetrical SAT and TERR scenarios. As expected, overall goodput is worse in the SAT scenario. It can however be seen that *picoquic* performs better, especially in the SAT scenario, and can sustain its goodput even in the presence of loss unlike ngtcp2. Irrespective of the buffer size, BBRv1 provides a slightly higher goodput than BBRv2. In the presence of packet loss, it is also noted that (1) ngtcp2 CUBIC's goodput significantly drops, while piqoquic's CUBIC generally sustains its goodput, and (2) BBRv2 reduces performance.



(a) $PLR = 0\%$



(b) $PLR = 0.1\%$



(c) $PLR = 1\%$

Figure 2: Bulk download goodput results

To evaluate the impact of uplink traffic in SAT scenarios on the downlink performance, we present the downlink goodput under different CC combinations in Table 2, for both a symmetric and an asymmetric SAT links. Results show that while *ngtcp2* goodput drops for an asymmetric bandwidth setup with 1:10 ratio, *picoquic* maintains performance.

4.2 Intra- and Inter-Protocol Fairness

When studying CC, it is important to evaluate the level of fairness between parallel flows of the same CC type (i.e.,

ngtcp2	**Symmetric 20/20**			**Asymmetric 20/2**		
Down | Up	BBRv2	BBRv1	CUBIC	BBRv2	BBRv1	CUBIC
BBRv2	14.73	14.94	13.56	5.31	8.74	6.36
BBRv1	15.10	14.97	14.66	8.03	8.25	5.25
CUBIC	14.90	13.64	11.70	7.38	8.64	7.74
picoquic	**Symmetric 20/20**			**Asymmetric 20/2**		
Down | Up	BBRv2	BBRv1	CUBIC	BBRv2	BBRv1	CUBIC
BBRv2	-	-	-	-	-	-
BBRv1	-	18.36	18.34	-	18.35	18.35
CUBIC	-	18.38	18.25	-	18.34	18.35

Table 2: Forward goodput for different SAT links

intra-protocol fairness) and different coexisting CC types (i.e., inter-protocol fairness). To measure fairness between parallel flows, we use Jain's Fairness Index (JFI) [38]. In this scenario a number of simultaneous bulk downloads are launched, and terminated after five minutes.

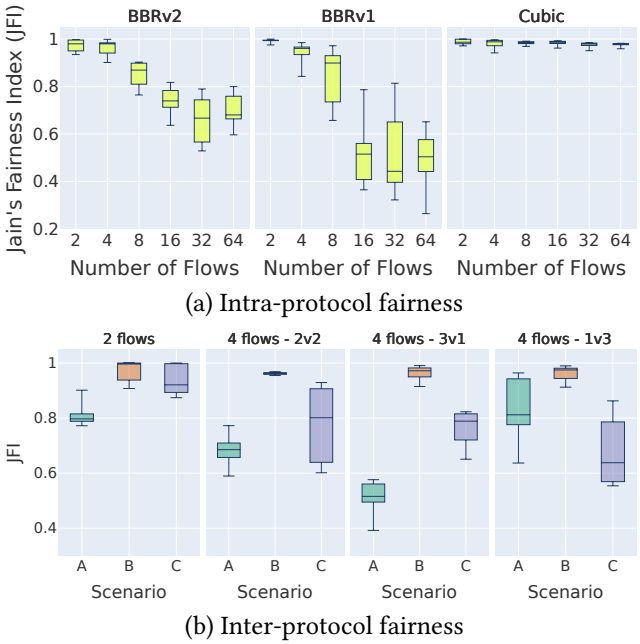
Figure 3a shows the intra-protocol fairness by presenting the JFI for different numbers of parallel flows, and for a buffer size of 1 BDP. Results show that BBRv1 achieves the best fairness for the 2-flow scenario, never descending below a JFI value of 0.970 and staying very close to 1. However, BBRv1 and BBRv2's fairness level decreases significantly as the number of flows increase reaching JFI values below 0.4 in the case of BBRv1. On the other hand, CUBIC maintains a very good level of fairness even with 64 flows.

To evaluate inter-protocol fairness three CC combinations are investigated. Figure 3b shows the JFI for these combinations with 2 and 4 flows (scenario *A* denotes BBRv2 vs BBRv1; *B* denotes BBRv2 vs CUBIC; and *C* denotes BBRv1 vs CUBIC). It can be observed that BBRv2 and CUBIC coexist fairly in all configurations, unlike BBRv1 and CUBIC. Some level of unfairness can also be observed between BBR versions - i.e. BBRv1 flows get a bigger bandwidth share due to their more aggressive nature -, but this may not be very critical since BBRv2 is intended to replace BBRv1.

4.3 Latecomer Issue

In the context of CC fairness, the *latecomer issue* indicates that unfair distribution of bandwidth and perhaps flow starvation may exist when a latecomer flow sharing the bottleneck with an already established flow which is utilizing a significant portion of available bandwidth. On SAT scenarios, the large RTT may potentially exacerbate this issue. To investigate this, a scenario was designed consisting of four flows over a SAT link scenario each starting at 0, 40, 80 and 120 seconds respectively, and all of which lasting for 180 seconds. Figure 4 shows the goodput and smoothed RTT for each flow over time in a single run, for each CC.

In overall, we observe that all new flows take a relatively long time to converge due to the long RTT feedback loop in SAT scenarios. We however observe that while CUBIC suffers more significantly from latecomer unfairness at the



(a) Intra-protocol fairness

(b) Inter-protocol fairness

Figure 3: JFI for fairness tests (ngtcp2)

beginning (i.e., with less number of flows), this issue becomes less profound as the number of flows increases — i.e., in the episode 160 sec–200 sec where CUBIC flows share the bandwidth more fairly compared to BBRv1 and BBRv2. BBRv1 shows the most aggressive behavior of latecomer flows: they gain their fair share of available bandwidth faster, and also overtake the pre-existing flows. Meanwhile, BBRv2 projects a less aggressive behavior in this regard. BBRv2 results also show that the last flow (Flow 4) fails to reclaim the available bandwidth in the last tens of seconds, which might be caused by *ngtcp2* BBRv2 implementation issues.

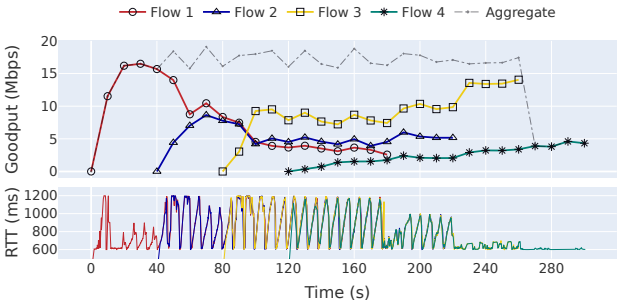
4.4 Mice versus Elephant Flows

Since satellites are used for Internet broadband access, it is relevant to study the impact of web object size. Small web objects lead to the transmission of *mice flows* in contrast to the bulk traffic (i.e., *elephant flows*) studied in §4.1. Figure 5 presents the download time for different object sizes and numbers of objects, with *ngtcp2*. The experiments are run in the presence of a background BBRv1 flow.

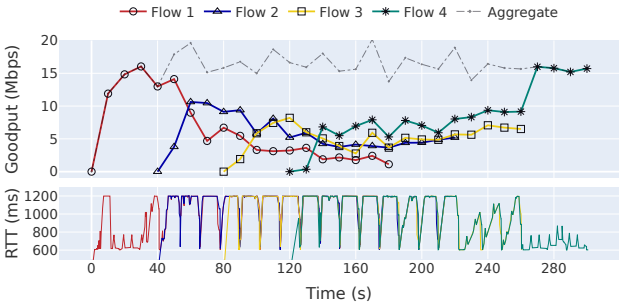
Results clearly show that download times are higher with CUBIC, especially for larger object sizes and number of objects - e.g., CUBIC takes twice as long as BBRv1 to download the 100x100KB object set. Even when using CUBIC as background traffic, and thus eliminating the potential bias of BBRv1's aggressiveness towards CUBIC, we have observed that BBRv1 still yields the lowest download times.

5 DISCUSSION

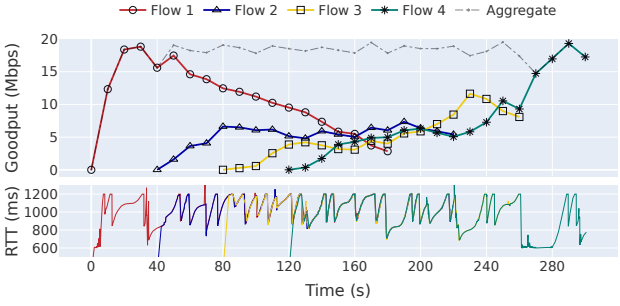
The choice of CC algorithm has proven to play a big role in QUIC's performance over SATCOM. Unlike CUBIC, BBR can



(a) BBRv2



(b) BBRv1



(c) CUBIC

Figure 4: Latecomer fairness with 4 flows (ngtcp2)

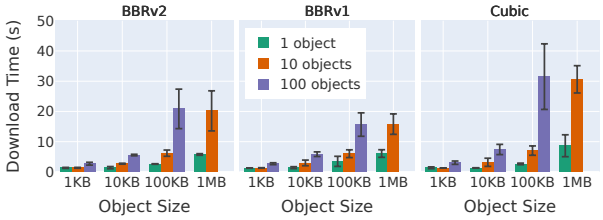


Figure 5: Download time for various sets of objects (ngtcp2)

maintain high performance when packet loss appears on the link, and it reduces download times for mice flows. Fairness towards loss-based CC has also improved significantly with BBRv2, even with the long feedback loop caused by the SAT link, and BBRv2 latecomers appear to converge faster than CUBIC while behaving less aggressively than BBRv1. Nevertheless, BBRv2 flows appear to perform significantly worse on lossy SAT links, and BBRv2's intra-protocol fairness is far from acceptable. Therefore, we encourage upcoming BBR iterations to try to tackle the issues above that arise on large-BDP paths.

As stated before in this paper, packet loss and bandwidth asymmetry are highly present on most GEO SATCOM links, and they play a big role in transport performance. While the presence of packet loss can be mitigated using layer 1 and 2 mechanisms such as Adaptive Coding and Modulation (ACM) and Forward Error Correction (FEC), bandwidth asymmetry is difficult to avoid in satellite broadband services in the market. Additionally, the problem of ACK congestion becomes especially critical nowadays due to the high presence of uplink traffic in modern Internet use. While TCP can alleviate return link congestion through ACK aggregation in PEPs, current QUIC deployments need to implement ACK policies either (1) in the endpoints - e.g. with the ACK Frequency extension - or (2) using MASQUE. Our study has shown that an ACK policy such as *picoquic*'s, which has been found to send around 10 times fewer ACK frames than *ngtcp2* on average, can maintain high performance over highly asymmetric links. This motivates the further study of ACK policy negotiations and their implementation for QUIC traffic.

Our study has also reported very significant performance differences between QUIC implementations: *picoquic* is providing better link utilization across CC algorithms, even with high packet loss and bandwidth asymmetry. Given our observations and the discussion on [39], we believe that these might be related to flow control window adaptation and acknowledgement strategies.

6 CONCLUSION

This work has studied the suitability of BBR congestion control over emulated satellite networks and using different QUIC implementations, as well as the impact of packet loss and bandwidth asymmetry of these networks.

The main findings of these paper are as follows: (1) BBR beats CUBIC in both short and long downloads, especially under packet loss conditions; (2) BBRv2 alleviates BBRv1's fairness issues, but it suffers from long protocol feedback loops in satellite links; and (3) satellite-optimized ACK policies using QUIC's ACK Frequency extension can help minimize ACK congestion on asymmetric bandwidth setups.

Future work should consider a wider set of QUIC implementations and a more complex satellite link model, including satellite PHY-MAC mechanisms, more realistic packet loss models (e.g. considering the likely bursty nature of errors), and a broader set of asymmetric bandwidth setups. This would help identify potential strategies to further improve BBRv2 over lossy and long latency paths, as well as design clever ACK policies that mitigate the effects of asymmetry while maintaining reliable delivery and loss detection and recovery.

REFERENCES

[1] Giorgos Papastergiou et al. 2017. De-ossifying the internet transport layer: a survey and future perspectives. *IEEE Communications Surveys Tutorials*, 19, 1, 619–639. DOI: 10.1109/COMST.2016.2626780.

[2] Neal Cardwell, Yuchung Cheng, C. Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. 2016. Bbr: congestion-based congestion control. *ACM Queue*, Volume 14, no. 5, 20–53. http://queue.acm.org/detail.cfm?id=3022184.

[3] Jana Iyengar and Martin Thomson. 2021. QUIC: A UDP-Based Multiplexed and Secure Transport. RFC 9000. (May 2021). DOI: 10.17487/RFC9000.

[4] Martin Thomson and Sean Turner. 2021. Using TLS to Secure QUIC. RFC 9001. (May 2021). DOI: 10.17487/RFC9001.

[5] Jana Iyengar and Ian Swett. 2021. QUIC Loss Detection and Congestion Control. RFC 9002. (May 2021). DOI: 10.17487/RFC9002.

[6] Mario Hock, Roland Bless, and Martina Zitterbart. 2017. Experimental evaluation of BBR congestion control. In *2017 IEEE 25th International Conference on Network Protocols (ICNP)*. IEEE, Toronto, ON, (Oct. 2017), 1–10. ISBN: 978-1-5090-6501-1. DOI: 10.1109/ICNP.2017.8117540.

[7] Neal Cardwell, Yuchung Cheng, Soheil Hassas Yeganeh, Ian Swett, and Van Jacobson. 2022. BBR Congestion Control. Internet-Draft draft-cardwell-iccrg-bbr-congestion-control-02. Work in Progress. Internet Engineering Task Force, (Mar. 2022). 66 pp. https://datatracker.ietf.org/doc/html/draft-cardwell-iccrg-bbr-congestion-control-02.

[8] Injong Rhee, Lisong Xu, Sangtae Ha, Alexander Zimmermann, Lars Eggert, and Richard Scheffenegger. 2018. CUBIC for Fast Long-Distance Networks. RFC 8312. (Feb. 2018). DOI: 10.17487/RFC8312.

[9] Yue Wang, Kanglian Zhao, Wenfeng Li, Juan Fraire, Zhili Sun, and Yuan Fang. 2018. Performance Evaluation of QUIC with BBR in Satellite Internet. In *2018 6th IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE)*. ISSN: 2380-7636. (Dec. 2018), 195–199. DOI: 10.1109/WiSEE.2018.8637347.

[10] Tom Jones, Gorry Fairhurst, Nicolas Kuhn, John Border, and Stephan Emile. 2021. Enhancing Transport Protocols over Satellite Networks. Internet-Draft draft-jones-tsvwg-transport-for-satellite-02. Work in Progress. Internet Engineering Task Force, (Oct. 2021). 29 pp. https://datatracker.ietf.org/doc/html/draft-jones-tsvwg-transport-for-satellite-02.

[11] Jim Griner, John Border, Markku Kojo, Zach D. Shelby, and Gabriel Montenegro. 2001. Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations. RFC 3135. (June 2001). DOI: 10.17487/RFC3135.

[12] Nicolas Kuhn, Francklin Simo, David Pradas, and Emile Stephan. 2021. Evaluating BDP FRAME extension for QUIC. en. *arXiv:2112.05450 [cs]*, (Dec. 2021). arXiv: 2112.05450. Retrieved Jan. 18, 2022 from http://arxiv.org/abs/2112.05450.

[13] Tommy Pauly and David Schinazi. 2022. QUIC-Aware Proxying Using HTTP. Internet-Draft draft-pauly-masque-quic-proxy-03. Work in Progress. Internet Engineering Task Force, (Mar. 2022). 19 pp. https://datatracker.ietf.org/doc/html/draft-pauly-masque-quic-proxy-03.

[14] Luis A. Sanchez, Mark Allman, and Dr. Dan Glover. 1999. Enhancing TCP Over Satellite Channels using Standard Mechanisms. RFC 2488. (Jan. 1999). DOI: 10.17487/RFC2488.

[15] S Oueslati-Boulahia, A Serhrouchni, and S Tohm. 2000. TCP Over Satellite Links : Problems and Solutions. en, 12.

[16] Carlo Caini and Rosario Firrincieli. 2004. TCP Hybla: a TCP enhancement for heterogeneous networks. en. *International Journal of Satellite Communications and Networking*, 22, 5, (Sept. 2004), 547–566. DOI: 10.1002/sat.799.

[17] David Borman, Robert T. Braden, Van Jacobson, and Richard Scheffenegger. 2014. TCP Extensions for High Performance. RFC 7323. (Sept. 2014). DOI: 10.17487/RFC7323.

[18] 2022. Usage Statistics of QUIC for Websites, May 2022. (2022). Retrieved May 3, 2022 from https://w3techs.com/technologies/details/ce-quic.

[19] Nicolas Kuhn, François Michel, Ludovic Thomas, Emmanuel Dubois, and Emmanuel Lochin. 2020. QUIC: Opportunities and threats in SATCOM. In *2020 10th Advanced Satellite Multimedia Systems Conference and the 16th Signal Processing for Space Communications Workshop (ASMS/SPSC)*. ISSN: 2326-5949. (Oct. 2020), 1–7. DOI: 10.1109/ASMS/SPSC48805.2020.9268814.

[20] Jorg Deutschmann, Kai-Steffen Hielscher, and Reinhard German. 2019. Satellite Internet Performance Measurements. en. In *2019 International Conference on Networked Systems (NetSys)*. IEEE, Munich, Germany, (Mar. 2019), 1–4. ISBN: 978-1-72810-568-0. DOI: 10.1109/NetSys.2019.8854494.

[21] John Border, Bhavit Shah, Chi-Jiun Su, and Rob Torres. 2020. Evaluating QUIC's Performance Against Performance Enhancing Proxy over Satellite Link. In *2020 IFIP Networking Conference (Networking)*. (June 2020), 755–760.

[22] Jana Iyengar and Ian Swett. 2021. QUIC Acknowledgement Frequency. Internet-Draft draft-ietf-quic-ack-frequency-01. Work in Progress. Internet Engineering Task Force, (Oct. 2021). 12 pp. https://datatracker.ietf.org/doc/html/draft-ietf-quic-ack-frequency-01.

[23] Ian Swett, Marie-Jose Montpetit, Vincent Roca, and François Michel. 2020. Coding for QUIC. Internet-Draft draft-swett-nwcrg-coding-for-quic-04. Work in Progress. Internet Engineering Task Force, (Mar. 2020). 17 pp. https://datatracker.ietf.org/doc/html/draft-swett-nwcrg-coding-for-quic-04.

[24] Neal Cardwell, Yuchung Cheng, C. Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. 2016. BBR: Congestion-Based Congestion Control: Measuring bottleneck bandwidth and round-trip propagation time. en. *Queue*, 14, 5, (Oct. 2016), 20–53. DOI: 10.1145/3012426.3022184.

[25] Benedikt Jaeger, Dominik Scholz, Daniel Raumer, Fabien Geyer, and Georg Carle. 2019. Reproducible Measurements of TCP BBR congestion control. *Computer Communications*, 144, 31–43. DOI: https://doi.org/10.1016/j.comcom.2019.05.011.

[26] Dominik Scholz, Benedikt Jaeger, Lukas Schwaighofer, Daniel Raumer, Fabien Geyer, and Georg Carle. 2018. Towards a Deeper Understanding of TCP BBR Congestion Control. In

2018 IFIP Networking Conference (IFIP Networking) and Workshops, 1–9. DOI: 10.23919/IFIPNetworking.2018.8696830.

[27] Jose Gomez, Elie Kfoury, Jorge Crichigno, Elias Bou-Harb, and Gautam Srivastava. 2020. A Performance Evaluation of TCP BBRv2 Alpha. In *2020 43rd International Conference on Telecommunications and Signal Processing (TSP)*. (July 2020), 309–312. DOI: 10.1109/TSP49548.2020.9163512.

[28] Yeong-Jun Song, Won-Ju Eom, Jeong-Keun Kim, Chang-Hoon Park, Geon-Hwan Kim, and You-Ze Cho. 2020. Intra-protocol Convergence Problem in BBRv2's Bandwidth Probing. In *2020 International Conference on Information and Communication Technology Convergence (ICTC)*. ISSN: 2162-1233. (Oct. 2020), 1016–1018. DOI: 10.1109/ICTC49870.2020.9289384.

[29] Yeong-Jun Song, Geon-Hwan Kim, Imtiaz Mahmud, Won-Kyeong Seo, and You-Ze Cho. 2021. Understanding of BBRv2: Evaluation and Comparison With BBRv1 Congestion Control Algorithm. *IEEE Access*, 9, 37131–37145. Conference Name: IEEE Access. DOI: 10.1109/ACCESS.2021.3061696.

[30] Furong Yang, Qinghua Wu, Zhenyu Li, Yanmei Liu, Giovanni Pau, and Gaogang Xie. 2022. BBRv2+: Towards balancing aggressiveness and fairness with delay-based bandwidth probing. en. *Computer Networks*, 206, (Apr. 2022), 108789. DOI: 10.1016/j.comnet.2022.108789.

[31] Robin Marx, Joris Herbots, Wim Lamotte, and Peter Quax. 2020. Same Standards, Different Decisions: A Study of QUIC and HTTP/3 Implementation Diversity. en. In *Proceedings of the Workshop on the Evolution, Performance, and Interoperability of QUIC*. ACM, Virtual Event USA, (Aug. 2020), 14–20. ISBN: 978-1-4503-8047-8. DOI: 10.1145/3405796.3405828.

[32] Sebastian Endres, Jörg Deutschmann, Kai-Steffen Hielscher, and Reinhard German. 2022. Performance of QUIC Implementations Over Geostationary Satellite Links. *arXiv:2202.08228 [cs]*, (Feb. 2022). arXiv: 2202.08228. Retrieved Apr. 4, 2022 from http://arxiv.org/abs/2202.08228.

[33] 2016. TCP Experiment Automation Controlled Using Python (TEACUP) – A Tool for Automated TCP Testbed Experiments. (2016). Retrieved Apr. 26, 2022 from http://caia.swin.edu.au/tools/teacup/.

[34] ETSI. 2013. Satellite Earth Stations and Systems (SES); Air Interface for S-band Mobile Interactive Multimedia (S-MIM); Part 2: Forward Link Subsystem Requirements. Technical Specification ETSI TS 102 721-2.

[35] 2022. Ngtcp2. original-date: 2017-06-25T08:28:58Z. (Apr. 2022). Retrieved Apr. 26, 2022 from https://github.com/ngtcp2/ngtcp2.

[36] 2022. Picoquic. original-date: 2017-06-26T19:08:37Z. (Apr. 2022). Retrieved Apr. 26, 2022 from https://github.com/private-octopus/picoquic.

[37] Robin Marx, Luca Niccolini, and Marten Seemann. 2022. Main logging schema for qlog. Internet-Draft draft-ietf-quic-qlog-main-schema-02. Work in Progress. Internet Engineering Task Force, (Mar. 2022). 49 pp. https://datatracker.ietf.org/doc/html/draft-ietf-quic-qlog-main-schema-02.

[38] R. Jain, D. Chiu, and W. Hawe. 1998. A Quantitative Measure Of Fairness And Discrimination For Resource Allocation In Shared Computer Systems. *arXiv:cs/9809099*, (Sept. 1998). arXiv: cs/9809099. Retrieved Apr. 10, 2022 from http://arxiv.org/abs/cs/9809099.

[39] 2022. EToSat discussion: Interop runner with satellite links. (2022). Retrieved July 3, 2022 from https://mailarchive.ietf.org/arch/msg/etosat/luOR6oL7ujdcpg7GwazR4FATQdQ/.