StarQUIC: Tuning Congestion Control Algorithms for QUIC over LEO Satellite Networks

Victor Kamel
University of Toronto
Toronto, ON, Canada
vkamel@cs.toronto.edu

Jinwei Zhao
University of Victoria
Victoria, BC, Canada
clarkzjw@uvic.ca

Daoping Li
University of Victoria
Victoria, BC, Canada
daopingli@uvic.ca

Jianping Pan
University of Victoria
Victoria, BC, Canada
pan@uvic.ca

Abstract

With the deployment of mega constellations of Low-Earth-Orbit (LEO) satellites, low latency and high throughput Internet coverage is extended globally. Latency-sensitive applications can benefit from the inherent lower transmission delay of LEO satellite networks compared to traditional Geostationary-Earth-Orbit (GEO) satellite networks. Starlink employs a globally time-synchronized controller to manage the association of satellite-to-ground communication links with an interval of 15 seconds, at fixed 12-27-42-57 seconds of every minute. Latency spikes and packet losses can occur during the handover period which can degrade the performance of transport layer protocols including TCP and QUIC, which rely on similar congestion control algorithms for fair data transmission. In this paper, we investigate the impact of the frequent Starlink handover events on QUIC performance. By leveraging the predictable handover patterns to avoid unnecessary congestion window reduction, we improved the performance of QUIC by up to 35% in terms of completion time in both network emulation and real-world experiments over Starlink networks. Our approach is independent of specific loss-sensitive congestion control algorithms and can be easily generalized.

CCS Concepts

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

Keywords

QUIC, Congestion Control, Network Measurement, LEO

ACM ISBN 979-8-4007-0489-5/24/11
https://doi.org/10.1145/3636534.3696227

ACM Reference Format:
Victor Kamel, Jinwei Zhao, Daoping Li, and Jianping Pan. 2024. StarQUIC: Tuning Congestion Control Algorithms for QUIC over LEO Satellite Networks. In *The 2nd ACM Workshop on LEO Networking and Communication 2024 (ACM MobiCom '24), November 18–22, 2024, Washington D.C., DC, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3636534.3696227

1 Introduction

Low-Earth-Orbit (LEO) satellite networks have seen a surge in interest in recent years, driven by the widespread deployment of mass-produced small satellites and reduced launch costs. Starlink, the pilot project of SpaceX, is the most successful LEO satellite constellation, with over 6,000 satellites in operation as of May 2024, covering 3 million subscribers in nearly 100 countries and regions [10]. LEO satellite networks have the unique advantage of low latency connectivity, compared with traditional satellite networks which operate in geostationary Earth orbits at an altitude of 35,786 km. However, the high mobility of LEO satellites makes the Starlink user terminal (UT) experience frequent handovers between satellites. Starlink employs a globally time-synchronized controller to manage the handover process [11]. Specifically, the handovers between UT and satellites happen every 15 seconds, at the 12-27-42-57 seconds of each minute, synchronized globally. This unique characteristic enables the UT to track satellites and maintain connectivity with its built-in phased array antenna throughout the handover process. However, handovers result in latency spikes and introduces packet loss during the handover period, which could potentially impact the performance of upper layer protocols and applications [14].

The highly dynamic network conditions in LEO satellite networks can potentially degrade the throughput performance of transport layer protocols, including TCP and QUIC. Specifically, the latency spikes and packet loss introduced by the frequent handover process in Starlink can be treated as a false congestion signal by different congestion control (CC) algorithms. Zhao et al. [14] have shown that the TCP downlink throughput in Starlink is significantly affected by the handover process, and CC algorithms have to initiate a slow start phase post-handover to recover, which has a significant negative impact on applications such as low-latency live video streaming.

Different CC algorithms follow their individual heuristics to adjust the cwnd and update the CC state. Loss-based CC algorithms, such as Reno and CUBIC, are overly sensitive to packet losses. More recently, latency-based CC algorithms, such as BBR, have been adopted widely to improve the transport layer performance. While having multiple variants, BBRv1 mainly uses bandwidth and round-trip time (RTT) estimation as the primary signals to adjust the cwnd. BBRv2 and BBRv3 incorporate ECN and loss signals into the CC state to compete more fairly with other CC algorithms. BBRv3 has already been adopted as the default TCP CC algorithm by Google for all internal WAN traffic and for all Google.com and YouTube public Internet traffic [2].

In this paper, we evaluate the performance of different CC algorithms with QUIC over Starlink, and improve QUIC performance by leveraging the predictable handover patterns to fine-tune the CC algorithm to avoid unnecessary cwnd reduction. We conduct experiments over a real Starlink network and a repeatable Mininet-based emulation testbed. We target QUIC, as its modular architecture and user space implementation allow more resilience towards potential changes in handover behaviour both in Starlink and across different constellations. Our preliminary results show that by incorporating the handover awareness into CC algorithms, the QUIC throughput performance can be markedly improved over Starlink, with the potential to be adapted to other LEO satellite networks.

The remainder of this paper is organized as follows. In Section 2, we introduce some existing work on measuring the performance of LEO satellite networks, and efforts to improve the performance of transport layer protocols over Starlink. In Section 3, we present our methodology and demonstrate the characteristics of Starlink that motivate the need for additional tuning of CC algorithms. Then, we present our novel handover-aware CC algorithm improvements. In Section 4, we present our extensive evaluation results over real-world Starlink networks and a Mininet-based emulation testbed. In Section 5, we discuss future work. Finally, we conclude our paper in Section 6.

2 Related Work

Cao et al. [1] proposed SaTCP, a CUBIC variant that forecasts the timestamps of satellite handover events or routing updates by utilizing the predictability of satellite locations to inform TCP to adapt its CC decisions accordingly. However, their evaluation was solely limited to their LeoEM emulator, which assumes the ground station (GS) can report the disruption event in advance in order to counteract satellite location prediction errors. Moreover, the LeoEM emulator assumes that the Starlink UT can notify end devices (UE) about upcoming handover events, and the nearby GS would be responsible for estimating the upcoming intermediate handover timestamps for the users associated with the GS and notify them in advance. In reality, such information is not available and this assumption is not physically plausible.

Pan et al. [8][9] measured the Starlink global backbone topology and the performance of Starlink access networks. It is worth noting that every Starlink UT is associated with one home Point-of-Presence (PoP). Network traffic from the UE is first sent through UT, then relayed through one or potentially multiple satellites, when inter-satellite links (ISLs) are being utilized, before reaching a landing GS. The traffic is then tunnelled back to the home PoP before exiting to the Internet. Thus, the cross-layer information sharing scheme for improving CC algorithm performance in [1] is not only infeasible but also impractical due to the proprietary "black-box" nature of the Starlink infrastructure.

Tiwari et al. [12] investigated TCP throughput performance with different CC algorithms over Starlink, namely CUBIC, NewReno, BBRv1 and BBRv2. They found that BBRv1 and BBRv2 could only achieve the median throughput of 32% and 8% of the maximum capacity (maximum observed throughput) over Starlink. They also noted that the results were contrary to previous simulation-based studies [7], due to the high loss rates alongside latency variations over real-world Starlink networks. They found that BBRv2 performs significantly worse than BBRv1, due to the introduction of a greater sensitivity to loss factors in BBRv2. Thus, they conducted a hyperparameter tuning to find the best (α, β) values for BBRv2 over Starlink, to find a balance between the more frequent RTT probing and the more robust loss resistance.

Li et al. [6] proposed StarTCP, which proactively stalls transmission during handovers to avoid bursty losses and erroneous congestion signals. They designed a *Handover Manager* with kernel density estimation to identify the handover timestamp every 15 seconds. In fact, existing research has revealed that Starlink employed a time synchronized controller globally to schedule handover events at the 12-27-42-57 seconds every minute [9] [11]. On the other hand, StarTCP employed an actor-critic model to learn the optimal transmission stalling duration under different network conditions with an offline training process. However, StarTCP only evaluated their performance on a set of static network emulation environments, lacking performance evaluation over real-world Starlink networks. In addition, stalling transmission may delay some packets that would otherwise have made it across, increasing latency.

QUIC, as standardized by RFC 9000 [5], builds on top of UDP and is implemented in user space. This provides a more flexible and modular transport layer protocol compared to

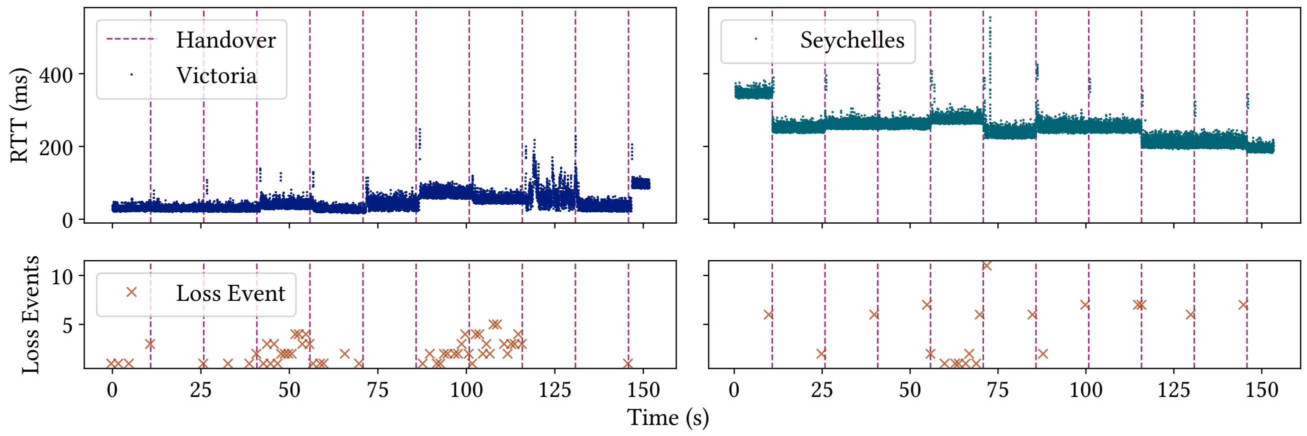

Figure 1: RTT and packet loss over Starlink [13], plotted against handovers. Right trace uses ISLs; left does not.

TCP. QUIC begins every connection in slow start with the congestion window (cwnd) set to an initial value. During the transmission process, the CC state changes over time as the receivers send ACK frames to the sender with ACK ranges, acknowledging that one or more packets are received. When packet loss is detected, QUIC enters the recovery state, where the cwnd is reduced and the sender retransmits lost packets. QUIC also supports the Explicit Congestion Notification (ECN) mechanism once a path is validated as ECN-capable.

In this paper, we proposed our generic handover-aware improvements to CC algorithms, and evaluated its practical performance on real-world Starlink networks. We also built a Mininet-based emulator to provide a repeatable and controllable environment, and utilized a publicly available Starlink latency dataset [13] for our evaluation and reproducibility.

3 Methodology

In this work, we adopt picoquic[3] as our reference QUIC implementation, as it implements most of the relevant features of QUIC, is easily-extensible, and implements several different CC algorithms including Reno, CUBIC, BBRv1 and BBRv3. We note that some of these algorithms, created primarily for TCP, have been optimized for QUIC.

3.1 CC Algorithm Performance Over Starlink

Current CC algorithms can broadly be classified into *loss-based* and *delay-based* categories.

3.1.1 Loss-based. Algorithms like Tahoe, Reno and CUBIC rely on packet loss events as the primary signal for congestion. When a link is sufficiently reliable, packet loss is caused by intermediate routers dropping packets due to their buffer capacity being exceeded. However, as observed by various network measurements over Starlink [9], since the handover events also produce packet loss, this is no longer a reliable indicator of congestion, at least for periods during which handover is in effect. In Figure 1, we plot two representative latency traces along with number of packet loss events reported by IRTT aggregated every second from [13], along with the handover events. Packet loss appears to be correlated with the handover, especially when ISLs are used (Seychelles UT). However, the packet loss events reported by CC algorithms and applications over Starlink networks could be categorized as: genuine path congestion, obstructions in the UT's field-of-view, packet reordering due to latency fluctuations, or other factors. Thus, the ability to differentiate the different sources of packet loss is critical.

3.1.2 Delay-based. Algorithms such as BBRv1 are delay-based (referred to as model-based by its authors). BBR estimates the bandwidth-delay product (BDP) and adjusts the sending rate in order to prevent buffers in intermediate routers from overflowing. This is accomplished by periodically measuring the RTT of the connection. Bandwidth is adjusted by periodically sending at a higher rate (in case the available bandwidth increases), followed by a drain period whereby buffers can be "drained" if the elevated sending rate was excessive. In practice, this causes BBRv1 to be completely loss-insensitive. This is not always desirable. Therefore, the BBRv1 implementation in picoquic is extended to support ECNs and react to massive packet loss. BBRv2 and v3 incorporate loss-sensitivity in order to increase fairness with other flows.

In addition, this family of algorithms is sensitive to RTT and delay fluctuations. However, these characteristics are subject to change after handover, as seen in Figure 1. We may see the delay increase, decrease, or remain the same depending on factors such as the distance to the connected satellite and the topology of the new connection determined by SpaceX's routing algorithm. Thus, this high delay variation will also hamper the algorithm's ability to accurately determine the bandwidth and delay of the connection if it is not aware of the handover logic.

3.2 Handover-Aware Congestion Control

In order to counteract the impact of high packet loss rates and delay fluctuations during handover periods, we propose a *congestion window freeze* mechanism. The `cwnd` freeze mitigates the impact of packet loss by preventing the CC algorithm from interpreting loss that is likely to be a result of handover as indicators of network congestion. This is effective for both loss-based algorithms such as CUBIC, and loss-sensitive delay-based algorithms such as BBRv2 and BBRv3.

As Starlink utilizes a global synchronized controller for scheduling satellite handovers on the 12-27-42-57 second of each minute, we can correlate packet loss events to a particular handover event. However, since the sender must either wait for an ACK frame or a timeout event in order to recognize that packet loss has occurred, the sender's view of the packet loss lags behind. In addition, in QUIC, the acknowledgements may be delayed [4]. Thus, when a loss event is detected, we check if the time that the packet was sent coincides with a handover. If so, we prevent the loss from reducing the `cwnd`, while still allowing it to increase.

We have observed that handover events typically last about 100 ms. In our measurements, we choose the sender location to be physically close to the PoP, thus minimizing the impact of the transmission delay between the satellite and the sender. If this were not the case, the sender would have to incorporate an estimate of the delay to the PoP in order to accurately determine which packets were affected by handover.

Finally, this assumes that the sender's clock is in sync in order to determine the handover times. Otherwise, it would be possible to estimate based on the latency pattern and knowledge of the 15-second gap between handovers.[1]

4 Evaluation & Analysis

In order to comprehensively benchmark our modified CC algorithms[2], we perform experiments both in an emulated environment based on real Starlink traces and with two geographically-diverse active Starlink UTs. This allows us to produce consistent and repeatable results, as well as to subsequently validate those results in real-world experiments. We

[1]Starlink has exposed the GPS receiver inside the dish through an NTP server in Sept. 2024 through a firmware upgrade, so applications and transport protocols are more likely to have a synchronized clock.
[2]Available: https://github.com/HeroHFM/picoquic_leo

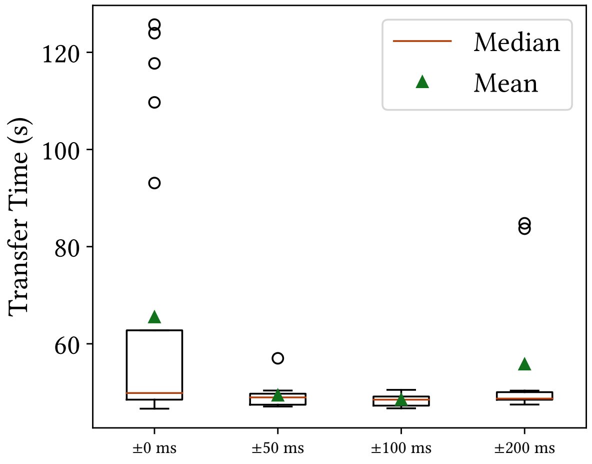

Figure 2: Finding optimal `cwnd` freeze time in BBRv3.

evaluate the `cwnd` freeze in BBRv3, as it is a delay-based algorithm that is still sensitive to some types of loss in addition to CUBIC, a pure loss-based algorithm. Due to `picoquic`'s heavy optimization of CUBIC, we expect that our modifications will have a reduced improvement when compared to the original versions of those algorithms.

4.1 Emulation

Our emulation[3] is created in Mininet based on measurements collected in [13]. The traces consist of IRTT latency measurements collected at 10 ms intervals and iPerf3 throughput measurements collected at 100 ms intervals. The emulation synchronizes with the wall clock such that handovers occur at exactly the correct fixed seconds in each minute. The bandwidth and latency of the link is updated every 100 ms using `tc-netem` and `tc-tbf` software available on Linux based on the traces. This configuration results in packet loss probed to be about 2–4% up/down.[4] Although we do not vary the packet loss during handover, the spike in RTT captured by the traces will cause `picoquic` to perceive packet loss due to retransmission timeout (RTO). Our emulation uses traces from [13], where the UT is associated with the Starlink Seattle PoP and does not utilize ISLs.

For our tests, we transfer a 256 MB file from a server running the modified CC algorithms to a client, and record the transfer completion time. We perform 2 runs per trace over 5 traces. In Figure 2, we implement a symmetric `cwnd` freeze interval before and after the scheduled handover timestamps (12-27-42-57 seconds of every minute) in BBRv3. We see that

[3]Available: https://github.com/HeroHFM/starquic
[4]Packet loss values observed in traces from [13] indicate that this is only slightly more elevated than expected.

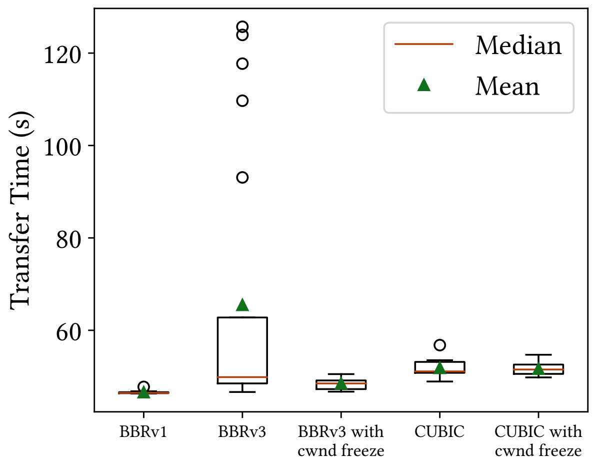

Figure 3: Transfer time for BBRv3 and CUBIC in Mininet emulation. The cwnd freeze is ±100 ms.

an interval of ±100 ms (total of 200 ms) produces the best results among those tested. This is longer than the handover itself, and accounts for the latency between the GS and the server. In Figure 3, we observe that our modifications to BBRv3 stabilizes the inter-run variance, thus decreasing tail latency. In addition, the mean transfer time reduces 26% from 65.5 s to 48.5 s. With the cwnd freeze, we see that BBRv3 is almost competitive with BBRv1—an algorithm that is not loss-sensitive.

We see that CUBIC is unaffected by the modification. This is because we do not simulate loss during handover, and thus the spurious loss recovery mechanism in picoquic's implementation of CUBIC prevents the loss due to the RTO for packets that arrive eventually from lowering the cwnd. This is effectively our method, achieved by different means. However, this will not work as well with real packet loss. The Figure 4 demonstrates that the modified algorithms indeed increase the throughput achieved by the application during the transfer for BBRv3.

4.2 Real-World Tests

In order to test our modified CC algorithms over real-world Starlink networks, we run both the original and the modified CC algorithms through a Starlink UT located in Asia, associated with the Starlink PoP in Tokyo and a UT located in North America, with its PoP in Seattle. The servers are located in datacenters close to their respective PoPs.

In Figure 5, we repeatedly transfer a 1 GB file from the server to the client at different times in the day. For BBRv3, the time required to complete the transfer decreases 14.5%

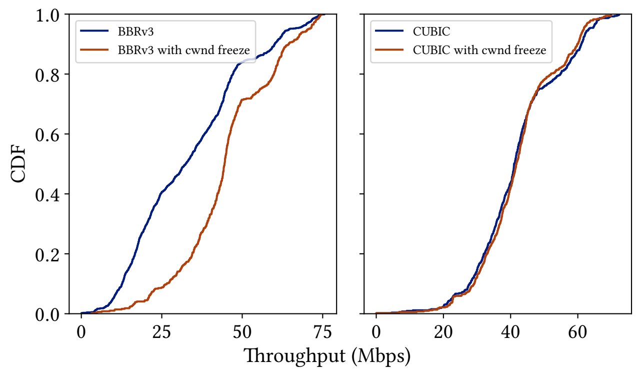

Figure 4: CDF of throughput for BBRv3 and CUBIC in Mininet emulation.

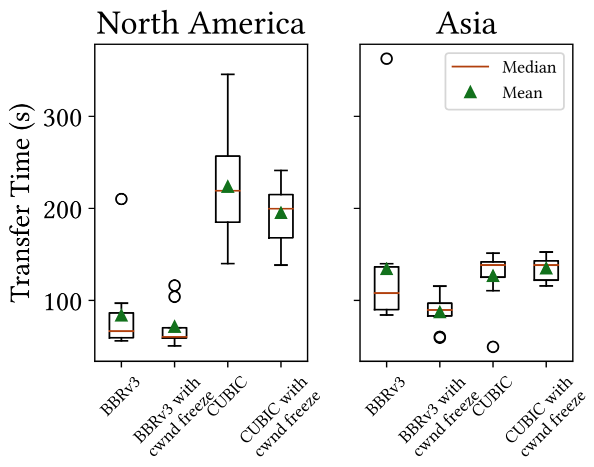

Figure 5: Transfer time for BBRv3 and CUBIC over Starlink.

and 35% in North America and Asia, respectively, when compared to the baseline. We note that our modified CUBIC performs faster in one location, but not the other. In the highly-dynamic Starlink environment, there are many factors outside our control, such as the satellite beam allocation for users in the same cell and the updated routing topology after handover. However, we are able to control these in emulation in order to provide a fair and repeatable comparison.

5 Future Work

We have identified that packet loss and delay spikes during handover, and changes to the connection characteristics post-handover, as reasons that current CC algorithms struggle to utilize the full available bandwidth over Starlink networks. In this work, we have addressed the packet loss and latency spike that occurs during handover, but an aspect that remains unattended to is the change in latency and throughput after each handover. While the current research community is still working methods to model this behaviour, we propose that for delay-based CC algorithms such as BBR, we need to make modifications for them to achieve accurate bandwidth and delay estimation over LEO networks. This could be achieved by preventing BBR from measuring the RTT during handover, when such measurements are likely to produce an incorrect result. In addition, after handover, we may force the algorithm to re-measure the bandwidth. This ensures that the changes to the link condition are accurately captured by the algorithm when they occur. These changes would allow us to rapidly adapt to the new link condition. Finally, supporting additional constellations such as Eutelsat OneWeb should be possible, as they experience handover as well. Adapting the algorithm to automatically detect the constellation and distance from the PoP would thus be useful.

6 Conclusion

In this paper, we investigated the impact of Starlink handover events on the performance of different CC algorithms in QUIC. By utilizing the predictable handover events every 15 seconds at fixed 12-27-42-57 seconds of every minute, we propose a CC-agnostic mechanism to avoid unnecessary `cwnd` reduction. We adopted our mechanism with BBRv3 and CUBIC and conducted comprehensive performance evaluation with both Mininet-based network emulation and real-world Starlink networks. The results demonstrate the effectiveness of our mechanism and by incorporating the handover awareness into CC algorithms, QUIC performance can be significantly improved up to 35% with straightforward modifications to current CC algorithms. Future work may involve more fine-grained tuning of the state changes in the algorithms.

Acknowledgments

This research was supported in part by NSERC, CFI, BCKDF and an NSERC USRA research award while Victor was with the University of Victoria. This work also is not possible without other Starlink users such as Shun and Dominique who generously allowed us to access their dishes remotely.

References

[1] Xuyang Cao and Xinyu Zhang. 2023. SaTCP: Link-Layer Informed TCP Adaptation for Highly Dynamic LEO Satellite Networks. In *IEEE INFOCOM 2023 - IEEE Conference on Computer Communications*. 1–10. https://doi.org/10.1109/INFOCOM53939.2023.10228914

[2] Neal Cardwell, Yuchung Cheng, Kevin Yang, David Morley, Soheil Hassas Yeganeh, Priyaranjan Jha, Yousuk Seung, Van Jacobson, Ian Swett, Bin Wu, and Victor Vasiliev. 2024. BBRv3: Algorithm Overview and Google's Public Internet Deployment. https://datatracker.ietf.org/meeting/119/materials/slides-119-ccwg-bbrv3-overview-and-google-deployment-00.

[3] Christian Huitema. 2024. Picoquic. Private Octopus. https://github.com/private-octopus/picoquic

[4] Jana Iyengar, Ian Swett, and Mirja Kühlewind. 2024. *QUIC Acknowledgment Frequency*. Internet-Draft draft-ietf-quic-ack-frequency-09. Internet Engineering Task Force. https://datatracker.ietf.org/doc/draft-ietf-quic-ack-frequency/09/

[5] Jana Iyengar and Martin Thomson. 2021. *QUIC: A UDP-Based Multiplexed and Secure Transport*. Request for Comments RFC 9000. Internet Engineering Task Force. https://doi.org/10.17487/RFC9000

[6] Li Jiang, Yihang Zhang, Yannan Hu, Yong Cui, and Xinggong Zhang. 2024. StarTCP: Handover-aware Transport Protocol for Starlink. In *Proceedings of the 8th Asia-Pacific Workshop on Networking (APNet '24)*. Association for Computing Machinery, New York, NY, USA, 169–170. https://doi.org/10.1145/3663408.3665803

[7] Simon Kassing, Debopam Bhattacherjee, André Baptista Águas, Jens Eirik Saethre, and Ankit Singla. 2020. Exploring the "Internet from Space" with Hypatia. In *Proceedings of the ACM Internet Measurement Conference (IMC '20)*. Association for Computing Machinery, New York, NY, USA, 214–229. https://doi.org/10.1145/3419394.3423635

[8] Jianping Pan, Jinwei Zhao, and Lin Cai. 2023. Measuring a Low-Earth-Orbit Satellite Network. In *2023 IEEE 34th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC '23)*. 1–6. https://doi.org/10.1109/PIMRC56721.2023.10294034

[9] Jianping Pan, Jinwei Zhao, and Lin Cai. 2024. Measuring the Satellite Links of a LEO Network. In *2024 IEEE 59th International Conference on Communications (ICC '24)*. 4439–4444. https://doi.org/10.1109/ICC51166.2024.10623111

[10] Starlink. 2024. Starlink is connecting more than 3M people with high-speed internet across nearly 100 countries, territories and many other markets. https://twitter.com/Starlink/status/1792678386353213567.

[11] Hammas Bin Tanveer, Mike Puchol, Rachee Singh, Antonio Bianchi, and Rishab Nithyanand. 2023. Making Sense of Constellations: Methodologies for Understanding Starlink's Scheduling Algorithms. In *Companion of the 19th International Conference on Emerging Networking EXperiments and Technologies (CoNEXT '23)*. Association for Computing Machinery, New York, NY, USA, 37–43. https://doi.org/10.1145/3624354.3630586

[12] Shubham Tiwari, Saksham Bhushan, Aryan Taneja, Mohamed Kassem, Cheng Luo, Cong Zhou, Zhiyuan He, Aravindh Raman, Nishanth Sastry, Lili Qiu, and Debopam Bhattacherjee. 2023. T3P: Demystifying Low-Earth Orbit Satellite Broadband. https://doi.org/10.48550/arXiv.2310.11835 arXiv:2310.11835 [cs]

[13] Jinwei Zhao and Jianping Pan. 2024. LENS: A LEO Satellite Network Measurement Dataset. In *Proceedings of the 15th ACM Multimedia Systems Conference (MMSys '24)*. Association for Computing Machinery, New York, NY, USA, 278–284. https://doi.org/10.1145/3625468.3652170

[14] Jinwei Zhao and Jianping Pan. 2024. Low-Latency Live Video Streaming over a Low-Earth-Orbit Satellite Network with DASH. In *Proceedings of the 15th ACM Multimedia Systems Conference (MMSys '24)*. Association for Computing Machinery, New York, NY, USA, 109–120. https://doi.org/10.1145/3625468.3647616