TCP Congestion Control Performance over Starlink

Johan Garcia
johan.garcia@kau.se
Karlstad University
Karlstad, Sweden

Simon Sundberg
simon.sundberg@kau.se
Karlstad University
Karlstad, Sweden

Anna Brunstrom
anna.brunstrom@kau.se
Karlstad University
Karlstad, Sweden

Abstract

We examine the performance of 14 different Linux TCP congestion control (CC) variants over Starlink connectivity. We then focus on the two most commonly used CCs, BBR and Cubic, and the best performing other CC, Illinois. Multiple measurement campaigns were conducted to evaluate throughput performance for single and multiple flows, examine how the performance of the CCs relate to the length of the TCP flow, and investigate how the CCs share resources over Starlink. The results show that BBR (both v1 and v3) outperform Cubic in the current Starlink environment. Cubic, which is the Linux default CC, performs considerably worse than BBR, particularly for single flows. Cubic with HyStart also has additional slowstart performance issues, which can be mitigated by disabling HyStart. The Illinois CC is a promising alternative to BBR and although being less performant for single flows, it has better fairness properties. Leveraging earlier similar measurements for a longitudinal study, we further show that changes made to the Starlink infrastructure during 2023/2024 are likely to have resulted in significantly reduced Cubic throughput.

CCS Concepts

• **Networks → Transport protocols**; *Network measurement*; Wireless access networks.

Keywords

Starlink, Congestion Control, TCP, Network Measurements, Satellite Networks, Performance Evaluation

ACM Reference Format:
Johan Garcia, Simon Sundberg, and Anna Brunstrom. 2025. TCP Congestion Control Performance over Starlink. In *Applied Networking Research Workshop (ANRW '25), July 22, 2025, Madrid, Spain.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3744200.3744760

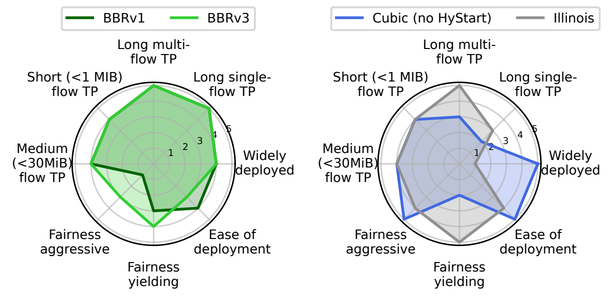

Figure 1: Congestion Controls in a Starlink perspective.

1 Introduction

Low-Earth-Orbit (LEO) satellite networks have emerged as a major opportunity for providing Internet connectivity to remote or under-served areas. While other LEO networks have also been examined [24, 37, 47], the majority of scientific study has been directed to Starlink [39]. A LEO communication system such as Starlink will introduce variations in throughput, latency, and losses due to the inherent characteristics of the communications channel, short-term scheduling, longer-term resource allocation, and the need for handoffs between satellites. The purpose of a congestion control (CC) is to provide fair and efficient usage of the available end-to-end communication resources, while protecting against network overload and avoiding unwarranted latency increases. Previous studies [12, 14, 16, 25, 33, 40, 46] have identified several Starlink-specific system properties that influence the working of CCs: 1) Time varying throughput; 2) 1.33 ms link layer slot timing, with silent periods of multiple slots thus leading to non-queueing delay variations; 3) spurious packet losses; and 4) periodic system reconfigurations at 15 s intervals.

In this work, we examine a range of TCP CC variants from several perspectives, beginning with a comparison of the long-flow throughput over Starlink of 14 CC variants. As discussed in the recent RFC on the specification of new CCs [9], the performance of short flows and the coexistence with other CC variants are important aspects, which we also examine here. The interplay between CCs and Starlink is examined and in most of our examinations we consider both single and multi-flow cases. A semi-subjective overview comparison of various CC characteristics within the Starlink context is provided in Figure 1.

2 Related Work

Previous work on LEO performance of multiple TCP CC variants include [7, 8, 25]. In [7], 10-flow measurements report that BBR performs better than Cubic, while [25] examines 9 CCs with a relative performance similar to ours for the joint subset of CCs. Other measurement studies include [33] that provides global throughput results based on 10s long single flow measurements from a large number of servers with unspecified CC configuration. In [16] TCP throughput is evaluated for single, four, and eight flows with an unspecified CC variant, likely Cubic. An early work [22] compared 5 CCs including BBR and Cubic using an unspecified number of flows, arriving at a similar relative performance as the one we show later in this paper. In [32] Iperf3 is used with Cubic, and an unspecified duration and number of flows. That work also measures with UDP, and reports TCP achieving 0.39 of the UDP throughput, which indicates considerable underutilization for the Cubic CC.

Other relevant studies focus on the underlying Starlink link layer characteristics [12, 13], which have an impact both on CC interactions, and the maximum capacity that Starlink can achieve. Some studies have used simulation or emulation to evaluate CC performance [2, 18, 23]. In [2] BBR, Cubic and NewReno are reported to have essentially the same throughput in a simulated LEO network. Our empirical findings do not match these simulation-derived throughput results. Other measurement studies explicitly consider effects such as weather [6, 27, 32], the impact of mobility [3, 15, 16, 26, 28, 43], the use of multi-connectivity [3, 16, 30, 31], CC for video streaming [10, 17, 45], and backbone/PoP aspects [34, 35]. There are also several studies focusing on LEO-specific CC improvements. LEO-adapted CC for QUIC is examined in [21, 44], while SaTCP is proposed and evaluated against Cubic in [4], and [29] proposes the in-orbit SatGuard.

While several of the aspects we examine here have been studied in some previous work, none of these works jointly consider a large variety of CCs, single/multi-flow aspects, varying flow lengths, fairness, and longitudinal aspects, as we do here.

3 Measurement setup

To be able to perform fine-grained analysis of CC behavior we use Iperf3 to generate traffic and capture pcaps using tcpdump. A Gen-2 Starlink Standard Actuated receiver located at the roof of the Computer Science building at Karlstad University is used [36]. The client-side measurements are collected from a physical machine with a 4-core Intel i7-6700 CPU and 16 GB RAM, running Ubuntu 22.04 LTS with a 6.3.2 Linux kernel. We use an Intel 10G X550-T2 Ethernet card. On the server the Ubuntu 24.04 LTS stock 6.8.0 kernel is used with default configuration, except for tests with (the not yet upstreamed) BBRv3-250318, which use Google's patched 6.13.7 kernel [5] instead. We disable client-side Large Receive Offload (LRO) and Generic Receive Offload (GRO) to avoid capturing merged superpackets.

Each measurement run is 180 seconds long and measure one CC with either a single TCP connection, or with 10 parallel TCP connections. Runs are scheduled within replication blocks, with a block encompassing all combination of CC and flow counts. The combinations are allocated randomly within each block. We collect data for 15 blocks, and thus we have 15 replications for each combination of CC and flow count, with some exceptions as a few runs failed to execute properly. The data collection took place in April 2025[1].

With this setup we expect the different CCs measured within a campaign to, on average, observe similar Starlink communications conditions, and thus potentially being able to achieve similar throughput. We performed additional measurements to examine specific aspects, and leveraged some previous measurements to allow the study of longitudinal aspects. In total, more than 1.4 billion packets were captured and processed to obtain the presented results.

4 Congestion Control Variants

Much of the data carried in todays networks is transferred using the TCP transport protocol in order to ensure reliable communication between the endpoints. TCP uses a congestion control (CC) mechanism to control the sending rate. The main purpose of a CC is to ensure that the sender does not overload the end-to-end path with a too high sending rate. Timewise, the CC is divided into two phases, with the initial so-called slow-start phase aiming to quickly arrive at a sending rate appropriate for the end-to-end path. The subsequent congestion avoidance phase aims to track changes in the underlying available capacity and adjust the sending rate accordingly, and can be seen as a steady-state mechanism after the end of the transitory slow-start phase. There are several approaches to CC as used in different TCP variants. These approaches can be grouped based on if the congestion signals are tied either to 1) the detection of packet losses, or 2) increases in end-to-end delay periods, or 3) some hybrid approach [42].

We initially focus on examining the long-flow performance of a range of CC variants available in the Linux kernel, thus being easily deployable on a server that runs Linux. As the experiments are performed using a well-provisioned academic network we assume that Starlink will be the bottleneck link. Thus, the achieved TCP throughput of long lived flows

[1]Data for the CCs Highspeed, Veno and CDG is from a previous measurement campaign from February 2024, utilizing the same setup but with 25 replications instead. Runs for UDP and Cubic without HyStart (cubic-nh) are for 15 replications measured in June 2025.

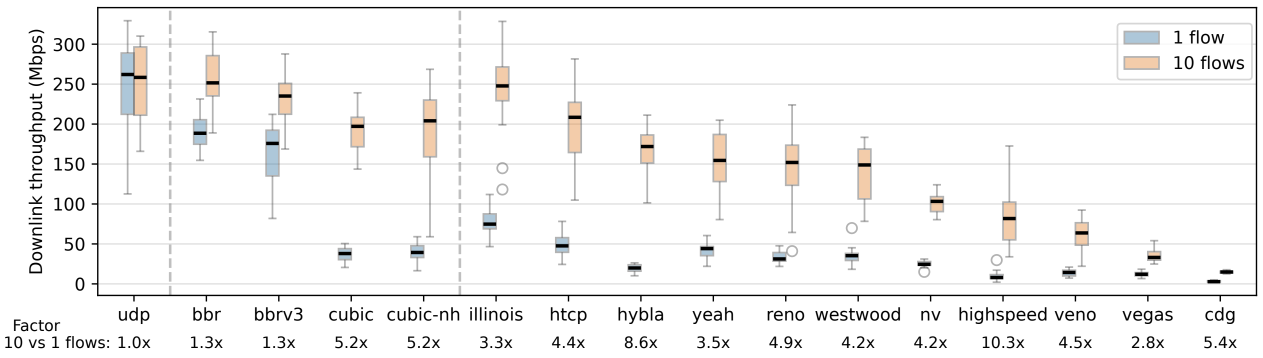

Figure 2: Average per-run throughput of 180 second long replicated Iperf3 measurement runs. The box shows the throughput for the 25th, 50th and 75 percentile runs, while the whiskers show the runs with min and max average throughput. Circles denotes runs outside of 1.5 times the inter-quartile range (IQR), and thus deemed outliers.

will be limited by either a) the underlying throughput provided by Starlink as it allocates the shared satellite capacity among users, b) CC response to non-congestion variations in packet loss and/or delays, or c) inability of the CC to effectively track large throughput variations in the underlying Starlink connectivity.

An overview of long flow throughput performance is provided in Figure 2, with the naming used in the Linux kernel (where the NewReno CC is labeled Reno). BBR refers to BBRv1, the CC available in the upstream kernel, while BBRv3 is used to distinguish the newer BBR version. The figure is subdivided into a UDP, left CC, and right CC part. The UDP part shows the throughput obtained for a single UDP flow sending at 500 Mbps, and for 10 x 50 Mbps UDP flows. Both cases show essentially the same underlying throughput, as expected. The left-hand CC part shows results for BBR, BBRv3, Cubic, and Cubic with HyStart turned off (cubic-nh) with the rationale that these are the CCs that currently see or are expected to see large-scale deployment. On the right-hand CC part, 11 additional CCs available in the Linux kernel are displayed, sorted in order of decreasing median throughput for 10 concurrent flows. As a conservative rule of thumb for visual interpretation, non-overlapping interquartile ranges (boxes) indicate a significant difference in the median, although a statistical test is required to make a formal statement. From the figure, BBR provides best overall performance with both the highest 10 flow and single flow median throughput. However, BBRv3 is fairly close to BBR. Both variations on Cubic show somewhat worse performance for the 10 flow case, but much worse performance for the single flow case. In terms of the other CCs, Illinois is the best performer for both 10 flows and single flow.

Delay-based CCs such as Vegas and CDG do not perform well for either single or ten flow cases. In terms of utilizing the underlying transmission capacity, none of the other CCs come close to full utilization since they do not achieve throughput similar to UDP, which can be seen as an indication of the capacity that can be provided by the network. One reason for the poor performance of the delay-based CCs Vegas and CDG may be short term delay variations that occur due to Starlink scheduling mechanisms. Starlink transmissions are performed in one or more 1.33 ms time-slots, with multiple silent slots between transmissions to a single user [12]. This scheduling is somewhat predictable [11] and give rise to a scheduling waiting delay that varies between zero and upwards of 15 milliseconds. As discussed in Section 7.1 of [1], the AIMD mechanism of loss-based CCs can also make them sensitive to large variations in packet delay and contribute to link underutilization. We can also note that BBR and BBRv3 use a design that makes them more robust to spurious losses, and this is also one likely reason for their higher throughput.

An interesting aspect of the CCs is also the relationship between the obtained throughput for 10 flows versus a single flow, which is shown as a factor in the bottom row of Figure 2. For both BBR versions the 10 flow throughput is a factor 1.3 of the single flow, and for Cubic it is 5.2. In an ideal end-to-end connection without non-congestion losses, no underlying link throughput variations, and no buffer-unrelated delay variations, most CC variants allow a single long-running TCP connection to achieve the same throughput as multiple parallel TCP connections would achieve, which would be the capacity of the underlying link. It is clear that none of the examined CCs achieve similar throughput for a single as for 10 flows. However, if the the sole underlying reason for the underutilization was randomly placed spurious non-congestion related losses, then the throughput for a loss-sensitive CC should be proportional to the number of flows, as long as the flow aggregate does not reach the link capacity, as is typically the case here. Thus, the factor would in such case be 10.

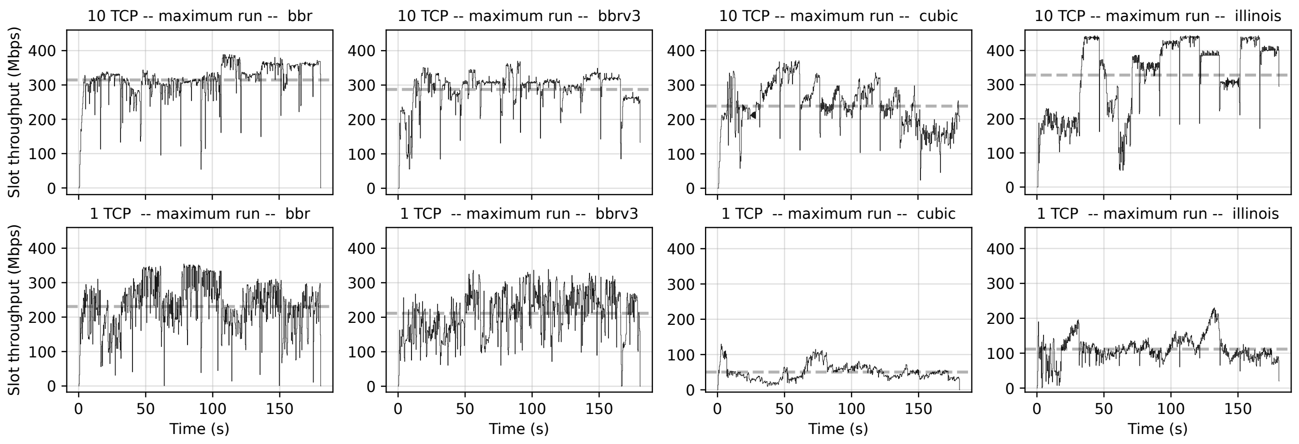

Figure 3: Throughput evolution within the measurement run for the run with highest average throughput. Total throughput per time slot is displayed with a resolution of 266 ms. Dashed line shows per-run average throughput.

The fact that the empirically observed factors are well below 10 for all but Highspeed suggests that while spurious losses play a role, there are other characteristics of Starlink that also contribute to the observed capacity underutilization.

5 Starlink and TCP Interaction

Our measurement setup allows us to observe high-resolution per-packet throughput data, which we can aggregate to suitable time slot sizes. We use this to gain additional insights into the relationship between the different CCs and the underlying Starlink behavior. In the remainder of this paper we mainly focus on BBR, BBRv3 and Cubic due to their deployment interest, and Illinois as being the best performing outsider. An overview of the throughput evolution for the runs which achieved the best average throughput is provided in Figure 3, using 266 ms resolution for the throughput calculation interval (i.e. 200 Starlink frame times of 1.33ms).

In the top row subgraphs, which show 10 flow results, there are regular downward spikes in slot throughput. These characteristic drop spikes result from the periodic reconfiguration that Starlink performs every 15 seconds, as has been previously identified and discussed in [14, 33]. Between the 15s drop spikes, there are many instances of intervals with a fairly horizontal evolution within them. Such horizontal intervals we consider to be a indication that throughput is at the underlying Starlink communications capacity[2]. The link capacity is in turn decided by the used modulation and frame scheduling as elaborated on in [11].

In the top row we can also see that the underlying Starlink throughput occasionally is subject to large fluctuation between two 15s intervals, here most visible for the Illinois run. Such large variations in underlying capacity may be challenging for some CCs to adapt to, especially when there is only a single flow that should adapt to these fluctuations. The Illinois single flow subgraph gives an example where the throughput is increasing in some intervals but the sending rate seem unable to reach a possible horizontal Starlink capacity bound within each 15 second interval.

The BBR 10 flow subgraphs also exhibit a number of intervals with clear horizontal evolution, but with more spurious variation within them compared to Illinois. For the single flow case, only BBR provides visual clues towards horizontal intervals that would suggest reaching a capacity bound. Within such intervals BBR still has considerable downwards variation, which is one reason why single flow BBR has somewhat worse performance than 10 flow BBR as seen in Figure 2. For Cubic, there are severe problems of reaching the Starlink capacity even with 10 flows, as seen in many intervals where the aggregate increase for the 10 flows is not sufficient to timely arrive at the likely Starlink link capacity. This suggests that 10-flow Cubic is less frequently fully utilizing the underlying capacity, which is consistent with its lower average throughput as noted in Section 4. For the single flow case, Cubic struggles even more, and is very far from the link capacity.

6 Short TCP Flows

It is well known that the majority of flows traversing the Internet are short. One empirical examination of Internet-bound flows showed that more than 99 % of TCP flows were less than 512 KiB in size [19]. To study short-flow performance, we exploit our per-packet receiver-side downlink data. For the single flow runs we measure the amount of time passing from the reception of first SYNACK packet

[2]Previous experiments indicate that there can also be intervals of full utilization which do not have stable horizontal evolution.

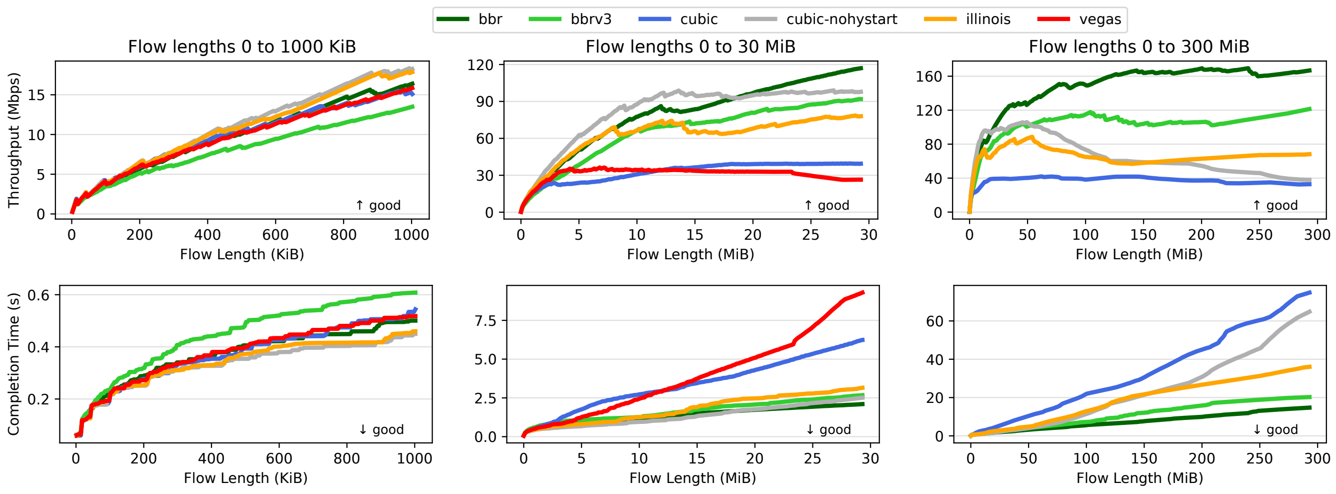

Figure 4: Computed average effective throughput and flow completion time for one short flow of varying length.

(which is part of the TCP handshake), and until the client has received data corresponding to a particular simulated flow length. When measured this way, the time will include one round-trip time overhead for connection establishment, but no round-trip times for connection termination. In Figure 4 we provide the effective complete-flow throughput, and the completion time. The values are computed for three different ranges, with 250 uniformly distributed flow sizes per range. The displayed throughput and completion times are the median of the replicated runs, for each specific flow size. To broaden the view in terms of short flow performance, we in this figure also partly include results for Vegas, which had very poor long-flow performance as seen in Figure 2.

As seen in Figure 4, there are noticeable throughput differences between the different CCs. For short flows the performance is dependent on the slow start behavior, rather than the congestion avoidance part of the CC. This results in different relative performance between the CCs in comparison to Figure 2. In the leftmost column, for short flows with flow sizes < 1000 KiB, all CCs are quite similar. For the more medium-sized flows seen in the middle column, Cubic with HyStart and Vegas perform poorly, while Cubic without HyStart performs similar to the other CCs. As evident by the results for Cubic without HyStart, the poor short flow performance of Cubic is related to the HyStart mechanism prematurely exiting from slow-start. This issue is discussed in [41] where a mitigation is also specified. An alternate mitigation, SEARCH, is proposed and evaluated for a LEO system in [20]. From the rightmost column it is clear that the CCs differ in their flow-size dependence. Cubic with HyStart plateaus early while Cubic-nohystart has high initial throughput and then a steady decline towards the same throughput as with HyStart. The BBR variations trend slightly upwards towards their eventual long-flow throughput of Figure 2.

In summary, we note that in a Starlink context, the flow length is an important factor. Slow-start dynamics can result in significantly different relative CC performance compared to the long-flow case where steady-state behavior decides the performance.

7 Congestion control fairness

The evaluation so far have considered each TCP CC in isolation. We now consider how CC variants coexist when they are concurrently sharing the Starlink capacity. We set up an experiment with a primary and secondary CC, both of which have 10 Iperf3 flows. The primary CC starts and runs in isolation for 30s, and then the secondary CC starts and the two CCs run in parallell for 30s. We compute the relative factor of all-flows throughput the primary CC gets in relation to the secondary CC during the 30s they compete. A factor of 1 signifies equal throughput sharing between the primary and

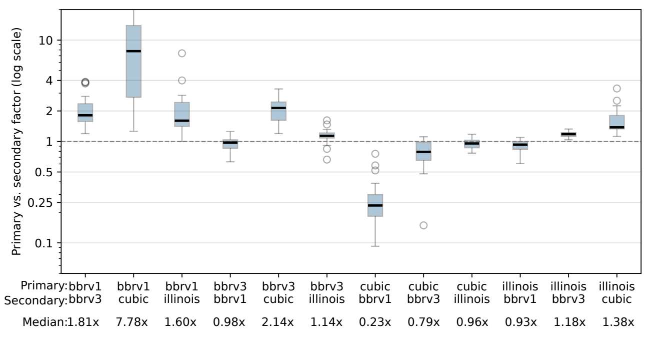

Figure 5: Concurrent CCs, 30s runs with 2 x 10 flows.

secondary CCs. The distribution of this factor for the 25 replications are shown in Figure 5, note the logarithmic y axis. It is clear that BBR have severe problems coexisting fairly with Cubic in the Starlink environment. Regardless of if Cubic is the primary or secondary CC, BBR will take a disproportionate amount of the throughput. The overall best performer from an fairness perspective is Illinois, which does not yield overly much to BBR, and is not very aggressive towards Cubic.

8 Longitudinal evaluation

In addition to the April 2025 measurement campaign mainly discussed so far, we now also consider previous measurement campaigns. These campaigns involve a single CC, or a small number of CCs, and have the same structure with 180 second measurement runs replicated over longer time periods. As Cubic is the most widely deployed CC, and was measured in all our previous campaigns, we will use Cubic for our longitudinal evaluation. We discuss the previous campaigns in terms of their date in the form YYMM where YY is year and MM month. The campaigns 2307, 2402, 2406, and 2410 all used similar measurement setups, with the sender using kernel version 5.15.0 (stock kernel for Ubuntu 22.04) for all runs, while campaign 2504 uses version 6.8.0 (stock kernel for Ubuntu 24.04). The results from our different measurement campaigns are shown in Figure 6. As can be seen, the Cubic performance sees a drastic performance decrease between 2307 and 2402 with half the throughput for the 10 flow case, and only a quarter of the performance for the single flow case.

Included in the graph are also the BBR measurements that were interleaved with the Cubic measurements at 2402. The BBR throughput shows that the poor Cubic performance was not caused by lower Starlink capacity, but by Cubic now being unable to utilize the available bandwidth to the same extent. Further, the throughput evolution graphs in Figure 7 show that with 10 flows, Cubic was at 2307 able to saturate the Starlink capacity giving rise to the characteristic horizontal upper-bounded intervals. At 2402, the same Cubic is unable to produce any sustained intervals of capacity saturation (and which, although not shown, 2402 BBR does produce). Since the same Cubic version and measurement setup is used for the 2307 and 2402 measurements, the most likely explanation for the throughput drop is some change to Starlink that impacts the Starlink-Cubic CC interaction. We hypothesize that a potential reason for this drastic reduction in Cubic performance can be found in the Starlink network configuration in terms of buffer sizes, queuing behavior, etc. Here we note that Starlink has reported the introduction of a large number of modifications of various aspects of their network in the period between the 2307 and

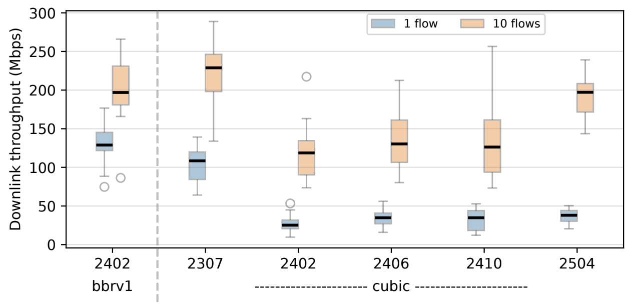

Figure 6: Measured throughput at yymm campaigns.

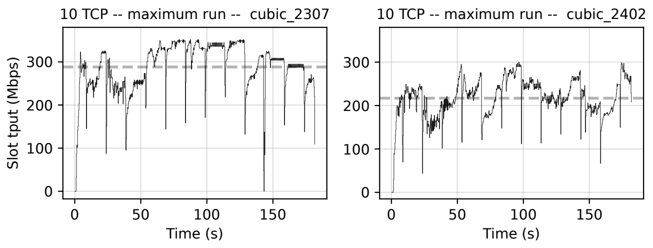

Figure 7: Cubic throughput evolution at 2 campaigns.

2402 measurements [38]. This implies that queuing, latency, and loss characteristics may be different over time even as the underlying provided communications capacity may be similar. These Starlink network setup variations may impact the CC interactions and the resulting throughput as demonstrated here. Consequently, the issue of potential internal Starlink configuration changes should be considered when comparing CC related performance results spread over time.

9 Conclusions

Our results show considerable differences in achieved throughput between the examined CCs. BBRv1 has the best throughput, regardless of the number of parallel flows and the flow length. Cubic has difficulties to fully utilize the capacity even with 10 long flows and for short flows it has issues with slow-start. Delay-based CCs, such as Vegas, generally perform the worst, except for short flow sizes. When also considering fairness aspects, Illinois becomes a strong overall performer.

From a service provider perspective, these results might be useful when considering how to configure the CC in servers likely to be serving material to Starlink-connected users. Our longitudinal study shows the importance for service providers to regularly measure and assess the performance of CCs as changes to the internal Starlink infrastructure may severely impact performance and invalidate previous configuration choices.

Acknowledgments

This research was partly funded by the Swedish Knowledge Foundation and Karlstad Internet Privacy Lab (KIPL).

References

[1] Venkat Arun, Mina Tahmasbi Arashloo, Ahmed Saeed, Mohammad Alizadeh, and Hari Balakrishnan. 2021. Toward formally verifying congestion control behavior. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*. 1–16.

[2] George Barbosa, Sirapop Theeranantachai, Beichuan Zhang, and Lixia Zhang. 2023. A comparative evaluation of TCP congestion control schemes over low-Earth-orbit (LEO) satellite networks. In *Proceedings of the 18th Asian Internet Engineering Conference*. 105–112.

[3] Claes Beckman, Johan Garcia, Herman Mikkelsen, and Patrik Persson. 2024. Starlink and Cellular Connectivity under Mobility: Drive Testing Across the Arctic Circle. In *2024 Wireless Telecommunications Symposium (WTS)*. IEEE, 1–9.

[4] Xuyang Cao and Xinyu Zhang. 2023. SaTCP: Link-Layer Informed TCP Adaptation for Highly Dynamic LEO Satellite Networks. In *IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 1–10.

[5] Neil Cardwell. 2025. *Release bbrv3-2025-03-18*. https://github.com/google/bbr/releases/tag/bbrv3-2025-03-18

[6] Céline Careau, Emil Fredriksson, Robert Olsson, Peter Sjödin, and Claes Beckman. 2025. Throughput Analysis of Starlink Satellite Internet: Study on the Effects of Precipitation and Hourly Variability with TCP and UDP. In *SPACOMM 2025: The Seventeenth International Conference on Advances in Satellite and Space Communications*. International Academy, Research and Industry Association (IARIA).

[7] Jörg Deutschmann, Saeid Jahandar, Kai-Steffen Hielscher, and Reinhard German. 2023. Internet via Satellite: GEO vs. LEO, OpenVPN vs. Wireguard, and CUBIC vs. BBR. In *Proceedings of the 1st ACM MobiCom Workshop on Satellite Networking and Computing*. 19–24.

[8] Zhuoxuan Du, Jiaqi Zheng, Hebin Yu, Hongquan Zhang, and Guihai Chen. 2024. Libra: A Congestion Control Framework for Diverse Application Preferences and Network Conditions. *IEEE/ACM Transactions on Networking* (2024).

[9] Martin Duke and Gorry Fairhurst. 2025. Specifying New Congestion Control Algorithms. Internet Requests for Comments. https://www.rfc-editor.org/info/rfc9743

[10] Hao Fang, Haoyuan Zhao, Jianxin Shi, Miao Zhang, Guanzhen Wu, Yi Ching Chou, Feng Wang, and Jiangchuan Liu. 2024. Robust Live Streaming over LEO Satellite Constellations: Measurement, Analysis, and Handover-Aware Adaptation. In *Proceedings of the 32nd ACM International Conference on Multimedia*. 5958–5966.

[11] Johan Garcia, Matthias Beckerle, Simon Sundberg, and Anna Brunstrom. 2025. Modeling and predicting starlink throughput with fine-grained burst characterization. *Computer Communications* (2025), 108090.

[12] Johan Garcia, Simon Sundberg, and Anna Brunstrom. 2024. Fine-grained starlink throughput variation examined with state-transition modeling. In *2024 19th Wireless On-Demand Network Systems and Services Conference (WONS)*. IEEE, 69–76.

[13] Johan Garcia, Simon Sundberg, and Anna Brunstrom. 2024. Inferring starlink physical layer transmission rates through receiver packet timestamps. In *in Proceedings of the 25th IEEE Wireless Communications and Networking Conference (WCNC)*.

[14] Johan Garcia, Simon Sundberg, Giuseppe Caso, and Anna Brunstrom. 2023. Multi-Timescale Evaluation of Starlink Throughput. In *Proceedings of the 1st ACM Workshop on LEO Networking and Communication (LEO-NET)*. 31–36.

[15] Amirreza Ghafoori, Alireza Famili, and Angelos Stavrou. 2024. Stars and towers on the wheels: Global perspective on satellite networks vs. terrestrial 5G. In *IEEE Global Communications Conference (GLOBECOM)*. IEEE, 301–306.

[16] Bin Hu, Xumiao Zhang, Qixin Zhang, Nitin Varyani, Z. Morley Mao, Feng Qian, and Zhi-Li Zhang. 2023. LEO Satellite vs. Cellular Networks: Exploring the Potential for Synergistic Integration. In *Companion of the 19th International Conference on Emerging Networking EXperiments and Technologies (CoNEXT 2023)*. 45–51.

[17] Liz Izhikevich, Reese Enghardt, Te-Yuan Huang, and Renata Teixeira. 2024. A Global Perspective on the Past, Present, and Future of Video Streaming over Starlink. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 8, 3 (2024), 1–22.

[18] Li Jiang, Yihang Zhang, Jinyu Yin, Xinggong Zhang, and Bin Liu. 2023. Leotp: An information-centric transport layer protocol for LEO satellite networks. In *2023 IEEE 43rd International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 579–590.

[19] Piotr Jurkiewicz, Grzegorz Rzym, and Piotr Boryło. 2021. Flow length and size distributions in campus Internet traffic. *Computer Communications* 167 (2021), 15–30.

[20] Maryam Ataei Kachooei, Jae Chung, Feng Li, Benjamin Peters, Joshua Chung, and Mark Claypool. 2024. Improving TCP Slow Start Performance in Wireless Networks with SEARCH. In *2024 IEEE 25th International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. IEEE, 279–288.

[21] Victor Kamel, Jinwei Zhao, Daoping Li, and Jianping Pan. 2024. StarQUIC: Tuning Congestion Control Algorithms for QUIC over LEO Satellite Networks. In *Proceedings of the 2nd International Workshop on LEO Networking and Communication (LEO-NET)*. 43–48.

[22] Mohamed M Kassem, Aravindh Raman, Diego Perino, and Nishanth Sastry. 2022. A browser-side view of Starlink connectivity. In *22nd ACM Internet Measurement Conference (IMC '22)*. 151–158.

[23] Fátima Khan, Cristina Hervella, Luis Diez, Fátima Fernández, Néstor J Hernández Marcano, Rune Hylsberg Jacobsen, and Ramón Agüero. 2023. Realistic assessment of transport protocols performance over LEO-based communications. *Computer Networks* 236 (2023), 110008.

[24] Heli Kokkoniemi-Tarkkanen, Kimmo Ahola, Jani Suomalainen, Marko Hoyhtya, Markus Saynevirta, and Atte Kivisto. 2024. Mission-critical connectivity over LEO satellites: Performance measurements using OneWeb system. *IEEE Aerospace and Electronic Systems Magazine* (2024).

[25] Zeqi Lai, Zonglun Li, Qian Wu, Hewu Li, Weisen Liu, Yijie Liu, Xin Xie, Yuanjie Li, and Jun Liu. 2024. Mind the Misleading Effects of LEO Mobility on End-to-End Congestion Control. In *Proceedings of the 23rd ACM Workshop on Hot Topics in Networks*. 34–42.

[26] Dominic Laniewski, Eric Lanfer, and Nils Aschenbruck. 2025. Measuring Mobile Starlink Performance: A Comprehensive Look. *IEEE Open Journal of the Communications Society* (2025).

[27] Dominic Laniewski, Eric Lanfer, Bernd Meijerink, Roland van Rijswijk-Deij, and Nils Aschenbruck. 2024. Wetlinks: a large-scale longitudinal starlink dataset with contiguous weather data. In *2024 8th Network Traffic Measurement and Analysis Conference (TMA)*. IEEE, 1–9.

[28] Dominic Laniewski, Eric Lanter, Simon Beainn, Jan Dunker, Michael Dückers, and Nils Aschenbruck. 2024. Starlink on the Road: A First Look at Mobile Starlink Performance in Central Europe. In *2024 8th Network Traffic Measurement and Analysis Conference (TMA)*. IEEE, 1–8.

[29] Jihao Li, Hewu Li, Zeqi Lai, Qian Wu, Yijie Liu, Qi Zhang, Yuanjie Li, and Jun Liu. 2024. SatGuard: Concealing Endless and Bursty Packet Losses in LEO Satellite Networks for Delay-Sensitive Web Applications. In *Proceedings of the ACM on Web Conference 2024*. 3053–3063.

[30] Melisa López, Sebastian Bro Damsgaard, Ignacio Rodríguez, and Preben Mogensen. 2022. An Empirical Analysis of Multi-Connectivity between

5G Terrestrial and LEO Satellite Networks. In *2022 IEEE Globecom Workshops*. IEEE, 1115–1120.

[31] Melisa López, Sebastian Bro Damsgaard, Ignacio Rodríguez, and Preben Mogensen. 2023. Connecting Rural Areas: An Empirical Assessment of 5G Terrestrial-LEO Satellite Multi-Connectivity. In *2023 IEEE 97th Vehicular Technology Conference (VTC2023-Spring)*. IEEE, 1–5.

[32] Sami Ma, Yi Ching Chou, Haoyuan Zhao, Long Chen, Xiaoqiang Ma, and Jiangchuan Liu. 2023. Network Characteristics of LEO Satellite Constellations: A Starlink-Based Measurement from End Users. In *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*. IEEE, 1–10.

[33] Nitinder Mohan, Andrew E. Ferguson, Hendrik Cech, Rohan Bose, Prakita Rayyan Renatin, Mahesh K. Marina, and Jörg Ott. 2024. A Multifaceted Look at Starlink Performance. In *Proceedings of the ACM on Web Conference 2024*. 2723–2734.

[34] 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)*. IEEE, 1–6.

[35] Jianping Pan, Jinwei Zhao, and Lin Cai. 2024. Measuring the Satellite Links of a LEO Network. In *IEEE International Conference on Communications*.

[36] Mohammad Rajiullah, Giuseppe Caso, Anna Brunstrom, Jonas Karlsson, Stefan Alfredsson, and Ozgu Alay. 2023. CARL-W: a Testbed for Empirical Analyses of 5G and Starlink Performance. In *In 3rd ACM Workshop on 5G and Beyond Network Measurements, Modeling, and Use Cases (5G-MeMU '23)*. doi:10.1145/3609382.3610510

[37] Aravindh Raman, Matteo Varvello, Hyunseok Chang, Nishanth Sastry, and Yasir Zaki. 2023. Dissecting the performance of satellite network operators. *Proceedings of the ACM on Networking* 1, CoNEXT3 (2023), 1–25.

[38] SpaceX. 2024. *Improving Starlink's latency*. Retrieved June 25, 2024 from https://api.starlink.com/public-files/StarlinkLatency.pdf

[39] Starlink. 2024. *HIGH-SPEED INTERNET AROUND THE WORLD*. Retrieved June 25, 2024 from https://www.starlink.com/

[40] 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*. 37–43. doi:10.1145/3624354.3630586

[41] Joe Touch, Yoshifumi Nishida, and Gorry Fairhurst. 2023. HyStart++: Modified Slow Start for TCP. Internet Requests for Comments. doi:10.17487/RFC9406

[42] Belma Turkovic, Fernando A Kuipers, and Steve Uhlig. 2019. Fifty shades of congestion control: A performance and interactions evaluation. *arXiv preprint arXiv:1903.03852* (2019).

[43] Muhammad Asad Ullah, Antti Heikkinen, Mikko Uitto, Marko Höyhtyä, Antti Anttonen, Konstantin Mikhaylov, and Timo Lind. 2025. Starlink in Northern Europe: A New Look at Stationary and In-motion Performance. *arXiv preprint arXiv:2502.15552* (2025).

[44] Wenjun Yang, Lin Cai, Shengjie Shu, and Jianping Pan. 2024. Mobility-aware congestion control for multipath QUIC in integrated terrestrial satellite networks. *IEEE Transactions on Mobile Computing* (2024).

[45] Miao Zhang, Jiaxing Li, Haoyuan Zhao, Linfeng Shen, and Jiangchuan Liu. 2024. StarStream: Live Video Analytics over Space Networking. In *Proceedings of the 32nd ACM International Conference on Multimedia*. 7909–7917.

[46] Haoyuan Zhao, Hao Fang, Feng Wang, and Jiangchuan Liu. 2023. Realtime Multimedia Services over Starlink: A Reality Check. In *33rd Workshop on Network and Operating System Support for Digital Audio and Video*. 43–49.

[47] Jinwei Zhao, Owen Perrin, Ali Ahangarpour, and Jianping Pan. 2025. Measuring the OneWeb Satellite Network. In *Proceedings of the IEEE/IFIP Network Traffic Measurement and Analysis Conference (TMA'25)*.