Looking at Stars: Analyzing Real-Time Video Conferencing over Starlink

Justus Fries
Technical University of Munich
Munich, Germany

Yannis Matezki
Technical University of Munich
Munich, Germany

Rohan Bose
Technical University of Munich
Munich, Germany

Nitinder Mohan
Delft University of Technology
Delft, The Netherlands

Abstract

SpaceX's Starlink has emerged as one of the largest network operators worldwide, serving over ten million subscribers across more than 150 countries. Real-time applications such as video conferencing, cloud gaming, and teleoperated control increasingly run over it. Yet how WebRTC's congestion controllers respond to Starlink's sub-second dynamics, namely its 15-second reconfigurations and satellite handovers, remains poorly understood. We present the first cross-layer measurement of WebRTC over Starlink, instrumenting Google Congestion Control (GCC) in a custom LibWebRTC testbed and pairing it with browser-side measurements of Microsoft Teams. Our campaign covers three vantage points in Europe and one in Antarctica at 2.5 and 10 Mbps targets. Even at Teams' conservative 2.5 Mbps uplink cap, the target is missed up to 34.9% of the time. Using browser metrics alone, we attribute up to 46% of these below-target periods to Starlink reconfigurations. GCC's delay-based estimator, which controls the bandwidth estimate 96–99% of the time, is the dominant pathway by which Starlink reaches the applications. Its overuse transitions cluster tightly around reconfigurations, while loss-based transitions remain uniformly distributed even at 10 Mbps. Satellite handovers, distinct from reconfigurations, cause longer-lasting overuse and the most aggressive sending-rate reductions in our dataset. Together, these findings reveal that GCC's conservatism, not intrinsic loss on the link, is the primary cost of Starlink's structural perturbations to real-time video.

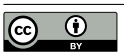
CCS Concepts

• **Networks** → **Network measurement**; **Mobile networks**; **Cross-layer protocols**.

Keywords

Starlink, Satellite Network, WebRTC, Video Conferencing

ACM Reference Format:
Justus Fries, Yannis Matezki, Rohan Bose, and Nitinder Mohan. 2026. Looking at Stars: Analyzing Real-Time Video Conferencing over Starlink. In *Proceedings of the 2026 ACM Internet Measurement Conference (IMC '26), October 12–16, 2026, Karlsruhe, Germany*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3777912.3839805



IMC '26, Karlsruhe, Germany

ACM ISBN 979-8-4007-2327-8/2026/10
https://doi.org/10.1145/3777912.3839805

1 Introduction

Real-time interactive video applications such as video conferencing, cloud gaming, and teleoperated control now run routinely over Low Earth Orbit (LEO) satellite networks (LSNs) [25, 37, 47, 50]. SpaceX's Starlink, the largest commercial LSN, serves over ten million subscribers in more than 150 countries [9, 37]. Recent infrastructure expansion that brings gateways and Points of Presence (PoPs) closer to users has narrowed the latency gap to terrestrial networks (TNs) in well-provisioned regions [9]. Underneath this surface comparability, however, Starlink behaves very differently from a terrestrial path at sub-second timescales. A globally synchronized scheduler triggers a network *reconfiguration* every 15 seconds that reassigns user-satellite-ground-station paths and resets the bandwidth allocation visible to a flow [20, 37, 42]. The local dish independently hands over between satellites that remain visible for only ~5–10 minutes [26, 34]. Both events momentarily disrupt connectivity, yet they are scheduled, structural, and orthogonal to the offered load [32, 45].

Real-time congestion control algorithms (CCAs) such as Google Congestion Control (GCC) [23] and SCReAM [29] are explicitly designed to be sensitive to short-timescale variations in one-way delay (OWD) and packet loss, keeping queuing delay bounded for interactive media [11, 14]. GCC, in particular, is the default in WebRTC and therefore in Chromium-based applications including Microsoft Teams and Google Meet [7, 11]. That same sensitivity makes them prone to misinterpreting Starlink's structural perturbations as congestion [21, 45]. Whether and how these signals propagate to the application is poorly understood. Which estimator dominates, how the response scales with target bitrate, and whether handovers compound or differ from reconfigurations together determine real-time video performance over Starlink. Yet prior measurement work on real-time applications over Starlink has stopped short of the CCA itself [13, 19, 25, 37, 47, 50], leaving these questions open (§2).

This paper closes that gap. We measure WebRTC video conferencing over Starlink from *four* geographically distributed vantage points: Antarctica (AQ), Germany (DE), the Netherlands (NL) and Norway (NO). We pair Microsoft Teams running headlessly in unmodified Chromium with a custom LibWebRTC testbed that logs GCC estimator state and bandwidth-estimate updates at every feedback packet. We measure both a 2.5 Mbps configuration matching Teams' uplink cap and a more demanding 10 Mbps one. Across over three days of cumulative video transmission, we compare GCC against SCReAMv1 and an unconstrained baseline. To the best of our knowledge, this is the first work to characterize the cross-layer interaction

between an LSN's sub-second dynamics and a real-time congestion controller's internal state. Specifically, we make three contributions:

(1) **Characterization of Teams over Starlink (§4.1).** We measure Microsoft Teams from four vantage points and find the uplink bitrate consistently capped at 2.5 Mbps. Even at this conservative target, AQ misses the cap 34.9% of the time and NO 24.8%, and using browser-side metrics alone we attribute up to 46% of below-target periods to Starlink reconfigurations.
(2) **Delay-based estimator drives GCC's response to Starlink (§4.2).** Through fine-grained instrumentation of LibWebRTC, we show that GCC's delay-based estimator, which controls the bandwidth estimate 96–99% of the time, is the dominant pathway by which Starlink dynamics propagate to the application. Its transitions to *overuse* cluster tightly around reconfiguration times. In contrast, loss-based estimator transitions remain uniformly distributed even at 10 Mbps where loss is substantially more frequent. The throughput shortfall at 10 Mbps is therefore driven by GCC's delay-based reaction rather than by intrinsic loss on the link, a conservatism we revisit alongside emerging IETF protocol designs in §5.
(3) **Reconfigurations and handovers leave distinct signatures (§4.2).** We separate *reconfigurations* (globally synchronized) from *satellite handovers* (dish-local), both of which trigger GCC overuse. Handovers cause longer-lasting overuse states than reconfigurations alone across locations. Irregular handovers (those not aligned with a reconfiguration) and unattributed overuse transitions are most common at AQ, which also exhibits the most aggressive sending-rate reductions in our dataset.

Our testbeds and LibWebRTC instrumentation [4], datasets [2], and analysis code [3] are publicly available.

2 Background and Related Work

Starlink and the geography of LEO coverage. Starlink deploys ~10,000 satellites at 500–600 km altitude across multiple orbital shells with inclinations of 43°, 53°, 70°, and 97.6° [6, 34]. Each shell covers a distinct latitude band, so a user's location determines how many satellites are visible, which Point of Presence (PoP) the dish reaches, and what kind of ground-station path the flow takes (see fig. 9 in §B.2). At mid-latitudes, dense 43° and 53° coverage permits direct dish-to-ground-station bent-pipe relay. At higher latitudes, where no nearby ground station exists (e.g., Antarctica), traffic traverses inter-satellite links (ISLs) before reaching a distant PoP. Geographic diversity therefore matters for global-scale characterization: different vantage points see different satellite counts, different PoPs, and different routing topologies. Starlink allocates uplink bandwidth on demand and drops from the head of its queues [12]; both shape performance mainly for flows that saturate their allocation (~30 Mbps), well above the bitrates we target. Within each location, the path also remains dynamic at sub-second timescales. 15-second reconfigurations [20, 37, 42] and dish-local handovers between satellites visible for only 5–10 minutes [26, 34] both inject brief, scheduled delay-and-loss spikes that a delay-sensitive real-time CCA can read as congestion. Recent work proposes LSN-aware CCAs that infer these events explicitly [10, 21, 28, 31, 32, 45], but whether a real-time CCA reads them correctly remains an open question.

Live video conferencing. Web Real-time Communication (WebRTC) [7] is the open standard for browser-native real-time media in commercial conferencing platforms (Microsoft Teams, Google Meet), extensively studied via end-user measurement [33, 35, 38, 49]. WebRTC carries media as Secure Real-time Transport Protocol (SRTP) packets over UDP with Secure RTCP [8] feedback, and bundles its own real-time CCAs because UDP leaves congestion control to the application. Chromium's default is Google Congestion Control (GCC) [11, 23], which combines two parallel bandwidth estimators. The *delay-based estimator* groups RTP packets by sender timestamp, measures inter-group delay variation at the receiver, and fits a linear regression over the last N groups to obtain a *delay gradient* [11, 23]. A positive gradient triggers *overuse* (multiplicative decrease of the estimate); a negative gradient signals *underuse* (estimate grows). The *loss-based estimator* runs in parallel: it holds the estimate at 2–10% loss and decreases it multiplicatively above 10%. GCC takes the lower of the two estimates as the sending bitrate, updated by the receiver via REMB [23] or, in transport-wide variants, by the sender from per-packet RTCP feedback [40]. The other standardized RTP-media real-time CCA is Self-Clocked Rate Adaptation for Multimedia (SCReAM) [29], treated by prior comparisons on simulated [48] and 5G [27] paths as the natural alternative to GCC. SCReAM couples a TCP-style congestion window to a delay-based encoder rate target, driving the window toward a 100 ms queue-delay target and backing off multiplicatively on packet loss or ECN marks [44]. Comparing them on the same Starlink path isolates effects of GCC's estimator from those of delay sensitivity. Emerging RTP-media CCAs (NADA, Cross, draft SCReAM v2 with L4S) lack deployments and measurements and are out of scope.

Real-time applications on Starlink. Although GCC and SCReAM are well-studied in isolation, prior measurements of real-time applications on Starlink have stopped short of the CCA itself. Mohan et al. [37] report higher uplink loss and 15 s OWD shifts for a Zoom flow, but at a single low target bitrate and without instrumenting the controller. Fang et al. [19] measure Zoom's desktop client over Starlink, whose proprietary stack is opaque to CCA-level analysis. Zhao et al. [50] stress-test real-time multimedia services and observe video pauses during outages, treating each application as a black box. Malekinasab et al. [36] measure Teams, Zoom, and Meet over Starlink and show how its terrestrial and satellite topology shapes WebRTC relay selection, but do not study the effects Starlink's link characteristics have on the CCA and application performance. Streaming-oriented work targets TCP transports, not the UDP +RTP pipeline of real-time conferencing: Izhikevich et al. [25] modify NewReno [22] for VoD on LSNs, and Zhang et al. [47] characterize live-video surveillance over RTMP [5]. On terrestrial paths, Cherian et al. [13] demonstrate that microscopic instrumentation of WebRTC's congestion controller exposes effects that application-layer metrics miss; Yi et al. [46] apply the same approach on 5G to reveal how cellular dynamics influence GCC. We adopt this instrumentation philosophy on Starlink and pair it with browser-side measurements of a commercial platform, exposing how reconfigurations and handovers reach GCC. Our work therefore provides the first cross-layer view of how an LSN's sub-second dynamics propagate through a real-time congestion controller's internal state. It bridges the network-layer Starlink measurements above with the CCA-instrumented studies of terrestrial and cellular paths. By

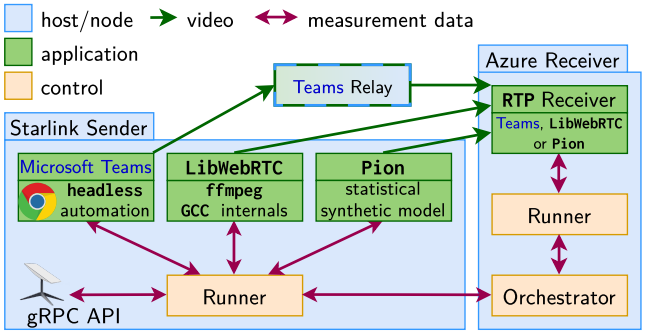


Figure 1: The different measurement testbeds. Microsoft Teams captures in-the-wild performance using an unmodified web client and a Teams relay in Europe; the LibWebRTC testbed exposes GCC internals under controlled bitrates against an Azure receiver. The Pion testbed generates synthetic traffic as a baseline. The Starlink-side peers run on LEOScope nodes and terminate inside Microsoft's backbone.

varying the target bitrate across two operating points and separating reconfigurations from satellite handovers, we further expose dependencies that single-bitrate, network-layer characterizations of Starlink leave invisible.

3 Measurement Methodology

Figure 1 shows our complementary measurement testbeds: a Microsoft Teams measurement that captures commercial behavior in the wild (§4.1), a controlled LibWebRTC testbed that exposes the parameters Teams hides (§4.2) and a Pion testbed for CCA comparison. Wherever possible, we reuse off-the-shelf components: a Teams web client in an unmodified headless Chromium, a LibWebRTC binary built from the production Chromium source tree, and an open-source SCReAMv1 implementation wrapped in Pion. We focus on the uplink, where Starlink's asymmetric capacity amplifies reconfigurations and handovers; preliminary downlink tests at 10 Mbps showed no similar effects.

Vantage points (VP). We deploy measurements on four LEOScope nodes spanning varied latitudes and different Starlink orbital shells. The DE node (Munich, ≈48° N) and NL node (Delft, ≈52° N) sit under the 43° & 53° shells with densest satellite coverage (see §B.2). The NO node (Oslo, ≈60° N) lies near the northern boundary of the 53° shell and is served primarily by the higher-inclination 70° and 97.6° shells. DE, NL, and NO all route via the Frankfurt PoP, with NO traversing a longer bent-pipe to its serving GS. The AQ node (McMurdo, ≈78° S) is covered almost exclusively by the 97.6° polar shell, sees the fewest visible satellites of our VPs, and is the only VP whose traffic crosses inter-satellite links to reach the Sydney PoP. Periodic traceroutes throughout the campaign confirm no PoP reassignments. The NL VP, a Raspberry Pi 5, could not sustain 10 Mbps due to hardware throttling and is excluded from those results.

Microsoft Teams. On the Starlink side, our test peer runs an unmodified Microsoft Teams web client in headless Chromium 143.0.7499.169, driven via the Chrome DevTools Protocol. We enable fully unattended runs using FIDO passkey emulation. The receiving peer runs in Microsoft's Azure backbone, keeping the measurement path inside the same network Teams uses in production. Teams selects its relay PoP closest to the first meeting joiner, so we have the receiver join first. This pins the relay to Europe across all four vantage points; we geolocate the relays to Sweden, Ireland, and France during our measurement window. The source video is a 1080p 60 FPS webcam-style talking-head clip, looped, with no artifact at the loop seam. We collect bit & frame rates, packet loss, and RTT through the standard WebRTC browser API every 50 ms.

LibWebRTC. The controlled platform lets us configure the parameters that Teams hides: target bitrate, congestion controller, encoder, and source content. We link a custom C++ binary against an instrumented LibWebRTC shared library. The instrumentation is built on top of LibWebRTC from the Chromium 142.0.7401.2 source tree. Our instrumentation follows Yi et al. [46]: we patch LibWebRTC to log in GCC's interval. This captures every delay-gradient update, estimator-state transition, bandwidth-estimate update, and RTCP feedback interval at the granularity at which GCC actually decides. The receiving peer runs the same LibWebRTC binary in an Azure VM, so sender and receiver speak the same WebRTC stack with no application-layer intermediary. The source video is a 1080p 60 FPS dashcam recording encoded with H.264 via an external ffmpeg process; we choose this content to emulate teleoperated driving and live game streaming, where sustained high uplink bitrates are realistic.

Pion baseline. For configurations that are not easily expressed in LibWebRTC, we run a Pion-based test peer with two settings: an unconstrained sender (*No-CC*) and SCReAMv1 via an open-source wrapper [1]. Since our Pion baseline is mainly concerned with whether bitrate targets are met, we drive it with RFC 8593's statistical reference model. This generates synthetic traffic that approximates the bitrate dynamics of an encoder using a statistical model. We use these baselines for the DE-only CCA comparison and flag the synthetic-content caveat where relevant.

Telemetry and ground truth. Each test run collects four parallel data streams. *(i)* WebRTC stats at the application layer (bitrate, FPS, loss, RTT). *(ii)* Per-feedback-packet GCC logs from the modified LibWebRTC (delay gradient, estimator state, bandwidth estimate, RTCP feedback interval). *(iii)* Independent latency probes via UDP (IRTT) and ICMP, run as sidecar processes alongside the WebRTC traffic for cross-checking. *(iv)* The Starlink dish's gRPC obstruction map polled at 1 Hz, from which we identify satellite handovers via the angular shift between consecutive maps. Network reconfigurations are anchored at the globally synchronized :12, :27, :42, and :57 second boundaries [42]. We classify a satellite handover as *regular* if within ±1 s of a reconfiguration, *irregular* otherwise; handovers principally drive long lasting overuse states (§4.2).

Our final dataset comprises three sender configurations (LibWebRTC GCC, Pion *No-CC*, Pion SCReAMv1) at two target bitrates (2.5 and 10 Mbps), alongside Microsoft Teams running GCC at its native 2.5 Mbps cap. A round consists of different configurations that are repeated multiple times and are scheduled in round-robin fashion per VP. A receiver node is always exclusively used for a single sender at a time to avoid cross-VP interference. Each run lasts 15 minutes and is repeated multiple times per location; we discard the first and last 30 seconds to avoid connection-establishment and tear-down artifacts. The campaign ran in multiple rounds from

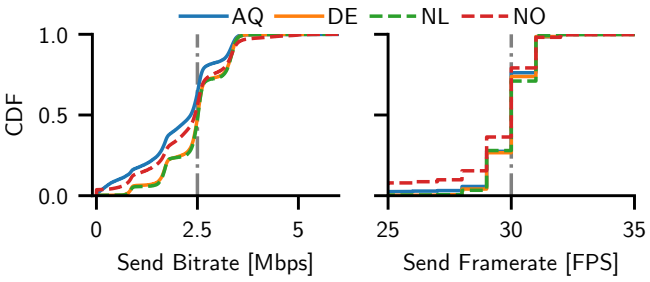


Figure 2: Teams sender-side `WebRTC` metrics across locations. Both metrics exhibit baseline variability due to the encoder, but AQ and NO show more frequent drops below the cap.

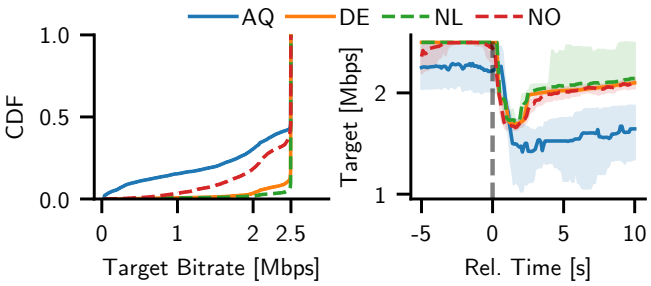


Figure 3: Distribution and median (with 95% CI) time series of the `WebRTC` target bitrate. The time series is relative to reconfigurations that cause bitrate drops.

November 2025 to April 2026, accumulating over 10 hours of cumulative video transmission for AQ, DE and NL and over 7 hours for NO at 2.5 Mbps. For DE, we additionally collect 13 hours of 10 Mbps data and 19 hours of `Pion` baselines, see Table 1 (§B.2).

4 Results

4.1 Microsoft Teams

We begin our analysis with a case study on Microsoft Teams. Teams downscales the source video to 720p at 30 FPS and caps the uplink bitrate at 2.5 Mbps. Figure 2 shows that DE and NL reach these targets more consistently than AQ and NO: DE and NL have fewer above and below-cap samples, while AQ and NO fall short more often. The achieved bitrates hover around the 2.5 Mbps cap, slightly exceeding or falling short of the target across all VPs. This variability is an artifact of the encoder not converging exactly to the target. AQ further achieves the framerate target more frequently than NO (73.5% and 63.6%, respectively) despite operating at lower bitrates. We hypothesize that sustained low bitrates allow the encoder to compress more aggressively, but our measurements do not capture encoder-internal state to confirm this. Additional measurements from terrestrial reference networks (TNs) in DE and NL (see §B.3) exhibit similar bit and frame rates (fig. 10). We find that bitrates achieved by Starlink are comparable to terrestrial, with a median of 2.51 Mbps and 2.52 Mbps respectively. For both networks Teams achieves the target framerate of 30 FPS for around 75% of the measurements and does not exhibit long drops in FPS (freezes). This suggests that some variability in performance is inherent, further supporting a root cause in the encoder dynamics.

The achieved target bitrate frequently falls below the 2.5 Mbps cap (fig. 3, left). Allowing a 0.375 Mbps margin around the cap, the share of samples below target is 34.9% at AQ and 24.8% at NO, against 5.7% at DE and 2.7% at NL. This margin is derived from `GCC`'s 0.85 multiplicative decrease factor, multiplied by the 2.5 Mbps cap to capture the effect of `GCC`'s reaction to Starlink reconfigurations. To attribute bitrate drops to Starlink reconfigurations, we apply a simple heuristic. For each reconfiguration, we compute the maximum target bitrate in the 1.5 s before and the minimum in the 1.5 s after; if the relative difference exceeds 15% (i.e., a multiplicative decrease), we attribute the drop to that reconfiguration. We choose 1.5 s based on `GCC`'s reaction time to Starlink reconfigurations, see §B.4. Figure 3 (right) shows the median and 95% confidence interval of the target bitrate around all attributed reconfigurations, computed in 100 ms bins.

All four VPs show the same pattern: the target bitrate sits at the 2.5 Mbps cap (2.25 Mbps for AQ) before each attributed reconfiguration, drops by ≈0.8 Mbps within 1–2 s, and only partially recovers over the next 10 s. The reaction is sharpest at AQ: by t=10 s, the other three VPs have recovered to 2.10–2.14 Mbps, while AQ's median only climbs back to 1.65 Mbps. Integrating the recovery time to above 2.125 Mbps (0.85 × cap), we attribute up to 46% of the time spent below target to reconfigurations: 46.3% at AQ, 32.0% at NO, 5.5% at NL and 4.9% at DE. Several LEO-specific factors plausibly amplify Starlink's dynamics at AQ. Its polar location is served almost exclusively by the high-inclination 97.6° shell with sparse satellite visibility, and we observe more frequent satellite handovers per measurement. Its traffic also traverses inter-satellite links to reach the Sydney PoP, which may attenuate or compound the underlying signal. Antarctic weather may add a further confounder [43]. Our data does not separate these contributions.

Takeaway — Microsoft Teams performance differs between geographic locations over Starlink, even when the same PoP is used. Up to 46% of the time spent below the 2.5 Mbps cap is attributable to Starlink reconfigurations using browser-side metrics alone. Comparing Starlink to a TN, we find overall similar performance.

4.2 LibWebRTC

Based on our findings for Microsoft Teams, we use a custom `LibWebRTC` testbed to dive deeper into congestion control behavior. Similar to our Teams measurements, we characterize the performance in different locations using baseline measurements at 2.5 Mbps. However, 2.5 Mbps is the target and not an upper cap. In addition, for one location (DE) we measure with a maximum target bitrate of 10 Mbps. Since our testbed does not require a relay, we use receiver-side data to quantify overall performance, while relying on sender-side data to analyze Starlink's dynamics and their effects.

The baseline performance shown in fig. 4 (left) varies, similar to our Teams measurements, by location. We find that while NL and DE achieve the highest bitrates, NO and especially AQ are affected by reconfigurations. The distribution of send rates relative to reconfigurations in fig. 4 (right) highlights this effect, with more

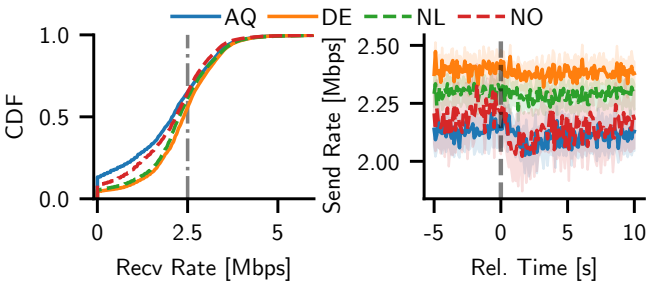


Figure 4: WebRTC received bitrate distribution and median send bitrate (95% CI) relative to all reconfigurations.

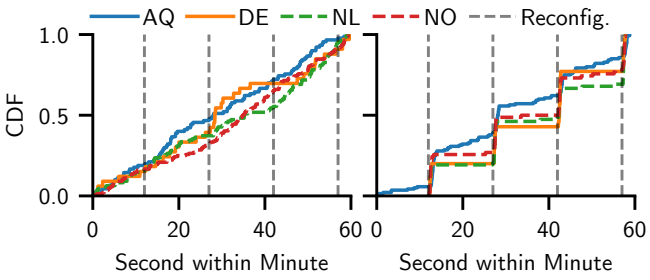


Figure 5: GCC's loss-based (left) and delay-based (right) estimator state transition times and reconfigurations.

aggressive send rate reductions in NO and AQ. Specifically, the delta between highest and lowest median send rates is 0.31 Mbps for NO and 0.18 Mbps for AQ, but only 0.15 Mbps for DE and 0.12 Mbps for NL. Although the absolute delta for AQ matches that of DE, its impact is more pronounced due to AQ's lower overall send rates, resulting in a relatively larger proportional reduction.

To further understand how reconfigurations interact with GCC we look at the state of GCC's two bandwidth estimators. For the loss-based estimator, we look at state transitions to *decrease* and for the delay-based one we look for transitions to *overuse*. Both of these transitions will directly cause a bitrate reduction. Figure 5 shows the times that state transitions and Starlink reconfigurations occur at in terms of seconds. While the loss-based estimator's transitions occur relatively uniformly within a minute, the delay-based estimator transitions show a clear pattern of correlation to reconfiguration times. Note that we find that the loss is generally low because a 2.5 Mbps target is below Starlink's uplink allocation [12]. Thus, GCC uses the delay-based estimator (99.0% for AQ, 99.6% for DE, 99.1% for NL and 96.4% for NO) most of the time. The delay-based estimator spends little of the total measurement time in overuse: 0.39% for AQ, 0.07% for DE and NL and 0.40% for NO.

To assess the application-level impact of bandwidth estimation reductions, we analyze freezes, which WebRTC defines as periods where the frame duration exceeds a threshold based on a linear average of previous values. The start of a freeze correlates with state transitions of the delay-based but not the loss-based estimator. The time from an estimator transition to the next freeze start is skewed toward lower values. Figure 6 (left) shows the distribution of the time from an overuse transition to the next freeze start. We find that the median time from overuse to a freeze within 15 s is 3.8 s for AQ, 3.6 s for DE, 3.5 s for NL and 2.9 s for NO.

While Starlink exhibits reconfigurations every 15 seconds, the satellite handover frequency is lower. We attribute a regular handover to a reconfiguration if the handover is within ±1 s of the reconfiguration. This window covers ≥89% of the handovers for DE, NL and NO. For AQ, we find the highest number of irregular handovers: 208 out of 678 (30.7%). We use the same window-based method to find whether estimator transitions coincide with reconfigurations and/or handovers. We find that 21 of those 208 irregular handovers cause the delay-based estimator to detect overuse. However, the total number of overuse transitions at AQ is high, with 140 transitions in total, while we find 78 for NL and NO, and 35 for DE.

The send rate reduction caused by handovers is higher compared to reconfigurations across all locations. Figure 6 (right) shows the duration spent in overuse for handovers and non-handover reconfigurations across VPs. In the underlying data, we find that the difference between handovers and reconfigurations depends on the location: the difference is higher for AQ and DE (>50 ms), while it is smaller for NL and NO. Overall, we find that the number of overuse transitions we can attribute to Starlink dynamics varies by location. For AQ we can only attribute 60.7% of transitions to reconfigurations or handovers, but 100% for DE, 97.4% for NL and 91.0% for NO. This indicates that delay is less stable for AQ in general, which may be caused by the polar shell, weather effects or the inter-satellite links that traffic from AQ has to traverse.

With the higher bitrate (10Mbps, 60 FPS, DE-only) scenario, the target bitrate is only achieved 9.5% of the time, while the target framerate is achieved more frequently, 41.5% of the time (see fig. 12 in §B.5). We find that for 10 Mbps, the delay-based estimator is in overuse for 0.46% of the total measurement time, which is slightly higher than 0.07% when the target is 2.5 Mbps. However, we find that the loss-based estimator is in the state decrease for 4.62% of the time (0.25%). Similar to 2.5 Mbps, the transitions of the loss-based estimator do not occur during reconfigurations or handovers. This means that the base level of loss is higher and affecting congestion control more.

Next to the high base loss, we also find that the effects of reconfigurations and handovers are more pronounced. Figure 7 shows the delay gradient and loss around estimator state changes. Note

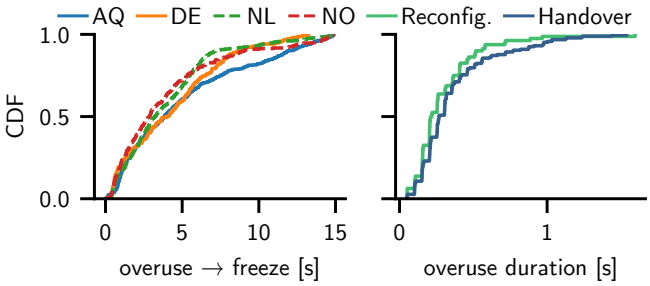


Figure 6: Delay from GCC's overuse transitions to freeze starts per VP and duration spent in overuse across VPs split by handovers or non-handover reconfigurations.

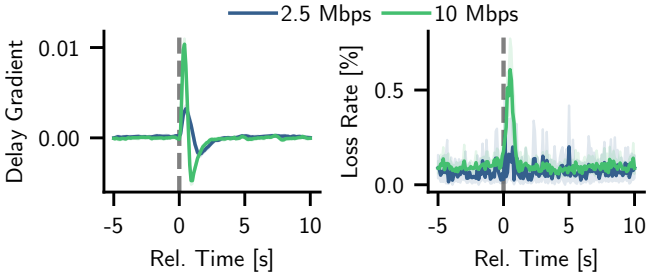


Figure 7: GCC's median delay gradient and mean loss (95% CI) relative to reconfigurations for DE.

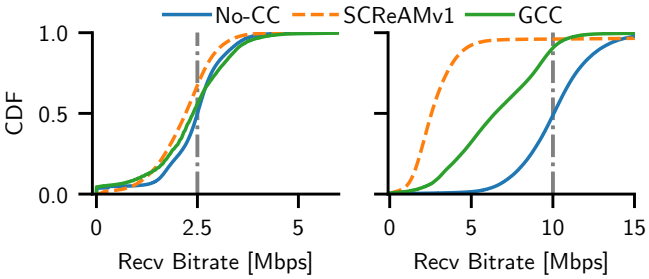


Figure 8: WebRTC receiver-side bitrate for CCAs with 2.5 Mbps (left) and 10 Mbps (right) targets for the DE vantage point.

that we show the mean loss because the median loss around reconfigurations is 0% (see §B.6). We observe that both the delay gradient and the loss rate are more pronounced around reconfigurations and handovers when targeting 10 Mbps compared to 2.5 Mbps.

Finally, we compare different real-time CCAs against each other for the DE location in fig. 8. LibWebRTC is used for GCC, while Pion is used for *No-CC* and SCReAMv1. Since the Pion testbed relies on synthetic video, the SCReAMv1 results may not be indicative of real-world performance. For *No-CC*, the bitrate is purely determined by an encoder model without feedback. Comparing GCC against *No-CC* shows that GCC could achieve higher bitrate by choosing to ignore congestion signals. We find that reconfigurations do not have a large negative impact as loss is low without congestion control.

Takeaway — The 2.5 Mbps LibWebRTC measurements closely match our Teams results. We are able to correlate reconfigurations to bitrate reductions, bandwidth estimator state transitions and freezes. Even when targeting 10 Mbps, reconfigurations and handovers do not cause a loss-based reaction by GCC. While an unconstrained Pion-based sender is able to deliver 10 Mbps to the receiver, GCC, despite no loss, is limited by Starlink effects to achieve the target.

5 Discussion and Conclusion

Prior Starlink measurements found that reconfigurations and handovers cause delay-and-loss spikes. We quantify how these events shape bit and frame rate reductions at the application. Two results depart from that expectation: even at 10 Mbps, these events do not generate sufficient loss for GCC's loss-based estimator to react, leaving delay as their sole route to the application; and AQ's delay varies well beyond what reconfigurations and handovers explain.

We report below an initial experiment toward improving real-time CC for Starlink based on these findings and discuss implications for ongoing IETF protocol work. Real-time CCAs rely on delay as their primary congestion signal, so Starlink's reconfiguration- and handover-driven delay spikes break the assumptions on which they are built. Explicit Congestion Notification (ECN) offers explicit signaling but is rarely deployed at endpoints [39].

Any Starlink-aware CCA modification must distinguish reconfigurations and handovers from actual congestion using only implicit signals. While the delay variations themselves are attributable to Starlink events, in our preliminary experiments, they are not periodic enough to be predictable over longer timescales. LeoCC [32] detects and reacts to reconfigurations using a high-frequency ICMP ping; we instead use a structural asymmetry: reconfigurations raise delay in both directions, while congestion typically affects only the sender's uplink. We track per-packet RTCP feedback for uplink and the RTCP feedback interval for downlink delay. We freeze the delay-based bandwidth estimator when both deviate significantly, similar to [10, 31]. LeoCC instead re-estimates bandwidth on every detection. At a 10 Mbps target, this design recovers a substantial fraction of the bitrate otherwise lost to reconfigurations in our LibWebRTC tests. However, the approach depends on receiver behavior: Teams emits RTCP feedback too infrequently to detect reconfigurations.

Two lines of IETF protocol design work are directly relevant. RTP-over-QUIC [16–18] enables media and non-media streams to share a congestion controller instead of competing, which is helpful even for WebRTC. Receiver-side feedback for QUIC real-time CCAs is being specified [41]. QUIC's existing controllers, Cubic and BBR, underperform GCC even on terrestrial paths [15, 24]. SCReAM version 2 [30] introduces L4S support that uses per-class queues and ECN to improve real-time QoS. Its delay-based fallback remains vulnerable to Starlink reconfigurations whenever ECN is unavailable.

Future directions. Our findings point to several natural extensions. The same instrumentation could probe the downlink (currently outside our scope), multi-party calls and other stacks like Zoom or Google Meet. Application-layer adaptation, such as encoder-aware bitrate scheduling, complements the CCA-side modification sketched above. Evaluating SCReAM v2's L4S path once Starlink supports classic or L4S ECN would close the protocol comparison.

Our findings highlight a fundamental tension: real-time CCAs rely on delay as their congestion indicator, yet Starlink's periodic reconfigurations make that signal ambiguous. L4S explicit signaling reduces this dependency but does not eliminate it as long as a delay-based fallback remains necessary. Resolving the tension will likely require richer receiver-side feedback to disambiguate reconfiguration artifacts from congestion, or cooperation from the satellite network to signal reconfigurations out of band.

Acknowledgements

This research was supported by the National Growth Fund through the Dutch 6G flagship project "Future Network Services".

References

[1] 2026. *SCReAM cgo wrapper.* Retrieved September 1, 2026 from https://github.com/mengelbart/scream-go

[2] 2026. *WebRTC over Starlink: Dataset.* doi:10.14459/2026mp1860734

[3] 2026. *WebRTC over Starlink: Scripts.* Retrieved September 1, 2026 from https://github.com/justus237/starlink-webrtc

[4] 2026. *WebRTC over Starlink: Testbed Code.* Retrieved September 1, 2026 from https://github.com/yannismate/webrtc-bench

[5] Adobe Systems Incorporated. 2012. *Real-Time Messaging Protocol (RTMP) Specification.* Technical Report. Adobe Systems Incorporated. https://veovera.org/docs/legacy/rtmp-v1-0-spec.pdf

[6] Muaz Ali, Utkarsh Upadhyay, Sean McCormick, Joseph Hill, and Beichuan Zhang. 2026. Starlink Constellation: Deployment, Configuration, and Dynamics. *arXiv preprint arXiv:2603.25835* (2026).

[7] Harald Alvestrand. 2021. Overview: Real-Time Communication with WebRTC. RFC 8825. doi:10.17487/RFC8825

[8] Mark Baugher, David McGrew, Mike Naslund, Elisabetta Carrara, and Karl Norrman. 2004. The Secure Real-time Transport Protocol (SRTP). RFC 3711. doi:10.17487/RFC3711

[9] Rohan Bose, Jinwei Zhao, Tanya Shreedhar, Jianping Pan, and Nitinder Mohan. 2026. Investigating Web Content Delivery Performance over Starlink. In *Proceedings of the ACM Web Conference 2026 (WWW '26)*. Association for Computing Machinery, USA, 5165–5176. doi:10.1145/3774904.3792227

[10] 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. doi:10.1109/INFOCOM53939.2023.10228914

[11] Gaetano Carlucci, Luca De Cicco, Stefan Holmer, and Saverio Mascolo. 2016. Analysis and design of the google congestion control for web real-time communication (WebRTC). In *Proceedings of the 7th International Conference on Multimedia Systems (MMSys '16)*. Association for Computing Machinery, USA, Article 13, 12 pages. doi:10.1145/2910017.2910605

[12] Hendrik Cech, Nitinder Mohan, and Jörg Ott. 2026. Dissecting the StarLink: Characterizing Queuing and Flow Dynamics in the Starlink Network. In *Proceedings of the ACM SIGCOMM 2026 Conference (SIGCOMM '26)*. Association for Computing Machinery, USA, 1475–1495. https://doi.org/10.1145/3789240.3829162

[13] Nathaniel Cherian, Akhil Prasad, and Sonia Fahmy. 2026. A Microscopic View of Congestion Control Behavior in Video Conferencing Applications. In *International Conference on Passive and Active Network Measurement*. Springer, 152–167. doi:10.1007/978-3-032-18268-5_7

[14] Luca De Cicco, Gaetano Carlucci, and Saverio Mascolo. 2013. Experimental investigation of the google congestion control for real-time flows. In *Proceedings of the 2013 ACM SIGCOMM Workshop on Future Human-Centric Multimedia Networking (FhMN '13)*. 21–26. doi:10.1145/2491172.2491182

[15] Rebecca Drucker, Aruna Balasubramanian, and Anshul Gandhi. 2025. Investigating WebRTC BBR as an alternative to GCC for live video streaming. In *2025 17th International Conference on COMmunication Systems and NETworks (COMSNETS)*. 225–232. doi:10.1109/COMSNETS63942.2025.10885565

[16] Mathis Engelbart, Fabian Willi Fromwald, and Jörg Ott. 2026. QUIC as Multiplexing Layer in WebRTC. In *Proceedings of the 2026 Applied Networking Research Workshop (ANRW '26)*. Association for Computing Machinery, USA, 119–125. doi:10.1145/3822163.3827919

[17] Mathis Engelbart and Jörg Ott. 2021. Congestion control for real-time media over QUIC. In *Proceedings of the 2021 Workshop on Evolution, Performance and Interoperability of QUIC (EPIQ '21)*. Association for Computing Machinery, USA, 1–7. doi:10.1145/3488660.3493801

[18] Mathis Engelbart, Joerg Ott, and Spencer Dawkins. 2025. *RTP over QUIC (RoQ).* Internet-Draft. Work in Progress. https://datatracker.ietf.org/doc/draft-ietf-avtcore-rtp-over-quic/14/

[19] Hao Fang, Haoyuan Zhao, Feng Wang, Yi Ching Chou, Long Chen, Jianxin Shi, and Jiangchuan Liu. 2025. Streaming Media over LEO Satellite Networking: A Measurement-Based Analysis and Optimization. *ACM Transactions on Multimedia Computing, Communications and Applications* 21, 9 (2025), 1–24. doi:10.1145/3694976

[20] 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 '23)*. Association for Computing Machinery, USA, 31–36. doi:10.1145/3614204.3616108

[21] Aashish Gottipati and Lili Qiu. 2025. Dynamic Pacing for Real-time Satellite Traffic. *arXiv preprint arXiv:2507.09798* (2025).

[22] Tom Henderson, Sally Floyd, Andrei Gurtov, and Yoshifumi Nishida. 2012. The NewReno Modification to TCP's Fast Recovery Algorithm. RFC 6582. doi:10.17487/RFC6582

[23] Stefan Holmer, Henrik Lundin, Gaetano Carlucci, Luca De Cicco, and Saverio Mascolo. 2016. *A Google Congestion Control Algorithm for Real-Time Communication.* Internet-Draft. Work in Progress. https://datatracker.ietf.org/doc/draft-ietf-rmcat-gcc/02/

[24] Christian Huitema, Suhas Nandakumar, and Cullen Fluffy Jennings. 2024. *BBR Improvements for Real-Time connections.* Internet-Draft. Work in Progress. https://datatracker.ietf.org/doc/draft-huitema-ccwg-bbr-realtime/00/

[25] 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. *Proc. ACM Meas. Anal. Comput. Syst.* 8, 3, Article 30 (Dec. 2024), 22 pages. doi:10.1145/3700412

[26] Liz Izhikevich, Manda Tran, Katherine Izhikevich, Gautam Akiwate, and Zakir Durumeric. 2024. Democratizing LEO Satellite Network Measurement. *Proc. ACM Meas. Anal. Comput. Syst.* 8, 1, Article 13 (Feb. 2024), 26 pages. doi:10.1145/3639039

[27] Ahmed Samir Jagmagji, Haider Dhia Zubaydi, and Sandor Molnar. 2022. Exploration and Evaluation of Self-Clocked Rate Adaptation for Multimedia (SCReAM) Congestion Control Algorithm in 5G Networks. In *2022 45th International Conference on Telecommunications and Signal Processing (TSP)*. 230–237. doi:10.1109/TSP55681.2022.9851382

[28] 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, USA, 169–170. doi:10.1145/3663408.3665803

[29] Ingemar Johansson and Zaheduzzaman Sarker. 2017. Self-Clocked Rate Adaptation for Multimedia (SCReAM). RFC 8298. doi:10.17487/RFC8298

[30] Ingemar Johansson, Magnus Westerlund, and Mirja Kühlewind. 2026. *Self-Clocked Rate Adaptation for Multimedia.* Internet-Draft. Work in Progress. https://datatracker.ietf.org/doc/draft-johansson-ccwg-rfc8298bis-screamv2/07/

[31] 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 '24)*. Association for Computing Machinery, USA, 43–48. doi:10.1145/3697253.3697271

[32] Zeqi Lai, Zonglun Li, Qian Wu, Hewu Li, Jihao Li, Xin Xie, Yuanjie Li, Jun Liu, and Jianping Wu. 2025. LeoCC: Making Internet Congestion Control Robust to LEO Satellite Dynamics. In *Proceedings of the ACM SIGCOMM 2025 Conference (SIGCOMM '25)*. Association for Computing Machinery, USA, 129–146. doi:10.1145/3718958.3750491

[33] Insoo Lee, Jinsung Lee, Kyunghan Lee, Dirk Grunwald, and Sangtae Ha. 2021. Demystifying Commercial Video Conferencing Applications. In *Proceedings of the 29th ACM International Conference on Multimedia (MM '21)*. Association for Computing Machinery, USA, 3583–3591. doi:10.1145/3474085.3475523

[34] 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*. 1–10. doi:10.1109/INFOCOM53939.2023.10228912

[35] Kyle MacMillan, Tarun Mangla, James Saxon, and Nick Feamster. 2021. Measuring the performance and network utilization of popular video conferencing applications. In *Proceedings of the 2021 ACM Internet Measurement Conference (IMC '21)*. Association for Computing Machinery, USA, 229–244. doi:10.1145/3487552.3487842

[36] Kousar Malekinasab, Mostafa Abdollahi, Jinwei Zhao, and Jianping Pan. 2026. Where Does My Call Go? Measuring Google Meet, Zoom, and Microsoft Teams on Starlink. In *Proceedings of the ACM SIGCOMM 2026 Conference (SIGCOMM '26)*. Association for Computing Machinery, USA, 1937–1945. doi:10.1145/3789240.3827595

[37] 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 Web Conference 2024 (WWW '24)*. Association for Computing Machinery, USA, 2723–2734. doi:10.1145/3589334.3645328

[38] Nanditha Rao, A Maleki, F Chen, Wenjun Chen, C Zhang, Navneet Kaur, and Anwar Haque. 2019. Analysis of the effect of QoS on video conferencing QoE. In *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*. IEEE, 1267–1272. doi:10.1109/IWCMC.2019.8766591

[39] Constantin Sander, Ike Kunze, Leo Blöcher, Mike Kosek, and Klaus Wehrle. 2023. ECN with QUIC: Challenges in the Wild. In *Proceedings of the 2023 ACM Internet Measurement Conference (IMC '23)*. Association for Computing Machinery, USA, 540–553. doi:10.1145/3618257.3624821

[40] Zaheduzzaman Sarker, Colin Perkins, Varun Singh, and Michael Ramalho. 2021. RTP Control Protocol (RTCP) Feedback for Congestion Control. RFC 8888. doi:10.17487/RFC8888

[41] Ian Swett and Joseph Beshay. 2026. *QUIC Extended Acknowledgement for Reporting Packet Receive Timestamps.* Internet-Draft. Work in Progress. https://datatracker.ietf.org/doc/draft-ietf-quic-receive-ts/03/

[42] 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 2023)*. Association for Computing Machinery, USA, 37–43. doi:10.1145/3624354.3630586

[43] Muhammad Asad Ullah, Antti Heikkinen, Mikko Uitto, Antti Anttonen, and Konstantin Mikhaylov. 2025. Impact of Weather on Satellite Communication:

Evaluating Starlink Resilience. In *2025 IEEE 101st Vehicular Technology Conference (VTC2025-Spring)*. 1–7. doi:10.1109/VTC2025-Spring65109.2025.11174614

[44] Magnus Westerlund, Ingemar Johansson, Colin Perkins, Piers O'Hanlon, and Ken Carlberg. 2012. Explicit Congestion Notification (ECN) for RTP over UDP. RFC 6679. doi:10.17487/RFC6679

[45] 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 (HotNets '24)*. Association for Computing Machinery, USA. doi:10.1145/3696348.3696867

[46] Fan Yi, Haoran Wan, Kyle Jamieson, and Oliver Michel. 2025. Automated, Cross-Layer Root Cause Analysis of 5G Video-Conferencing Quality Degradation. In *Proceedings of the 2025 ACM Internet Measurement Conference (IMC '25)*. Association for Computing Machinery, USA, 835–850. doi:10.1145/3730567.3764434

[47] 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 (MM '24)*. Association for Computing Machinery, USA, 7909–7917. doi:10.1145/3664647.3680785

[48] Songyang Zhang, Weimin Lei, Wei Zhang, and Yunchong Guan. 2019. Congestion control for RTP media: A comparison on simulated environment. In *International Conference on Simulation Tools and Techniques*. Springer, 43–52. doi:10.1007/978-3-030-32216-8_4

[49] Xinggong Zhang, Yang Xu, Hao Hu, Yong Liu, Zongming Guo, and Yao Wang. 2012. Profiling Skype video calls: Rate control and video quality. In *IEEE INFOCOM 2012 - IEEE Conference on Computer Communications*. 621–629. doi:10.1109/INFCOM.2012.6195805

[50] Haoyuan Zhao, Hao Fang, Feng Wang, and Jiangchuan Liu. 2023. Realtime Multimedia Services over Starlink: A Reality Check. In *Proceedings of the 33rd Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV '23)*. Association for Computing Machinery, USA, 109–110. doi:10.1145/3592473.3592562

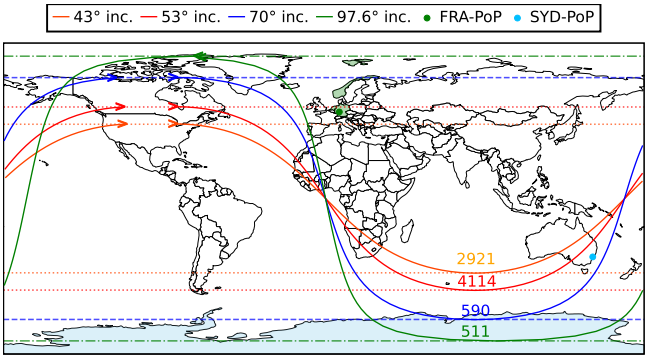


Figure 9: Starlink orbital inclinations (43°, 53°, 70° and 97.6°) with number of satellites in each orbit for a given inclination. The countries for the vantage points in our WebRTC measurement campaign are shaded in green (NO,NL,DE) and blue (AQ), with their respective assigned PoPs in Frankfurt (DE) and Sydney (AU).

A Ethics

This work does not raise any ethical issues.

B Appendix

B.1 Measurement Campaign: Run Details

Table 1: Detailed measurement rounds per vantage point (VP) and configuration.

Configuration	VP	Rounds	Video
Teams	AQ	2025-12-04–2025-12-06	5 h
	DE	2025-11-16–2025-11-19	8.5 h
	NL	2025-11-25–2025-11-27	4.5 h
		2026-01-06–2026-01-08	3.7 h
	NO	2025-11-30–2025-12-03	5.7 h
`LibWebRTC GCC` 2.5 Mbps	AQ	2025-12-04–2025-12-06	5.3 h
	DE	2026-01-23–2026-01-24	3.3 h
		2026-04-29	1.3 h
	NL	2025-12-23–2025-12-27	8.7 h
	NO	2025-12-01–2025-12-02	2 h
`LibWebRTC GCC` 10 Mbps	DE	2025-11-16–2025-11-19	8.5 h
		2026-01-23–2026-01-24	3.3 h
		2026-04-29	1.3 h
`Pion` *No-CC* 2.5 Mbps	DE	2026-04-24–2026-04-26	5.5 h
`Pion SCReAMv1` 2.5 Mbps	DE	2026-04-24–2026-04-26	4.3 h
`Pion` *No-CC* 10 Mbps	DE	2026-04-24–2026-04-26	4 h
`Pion SCReAMv1` 10 Mbps	DE	2026-04-24–2026-04-26	5.5 h

B.2 Starlink Constellation

The Starlink constellation comprises of LEO satellites deployed in either of the 43°, 53°, 70° and 97.6° inclination angles as illustrated in Figure 9. The 43° and 53° orbits host the most number of satellites [6], given that majority of the world's population lies in this belt. Our vantage points in Germany (DE) and Netherlands (NL) are primarily serviced by these orbits, while the Norway (NO) vantage points may connect to satellites in the 53°, in addition to the 70° or the 97.6° shell. The vantage point for Antarctica (AQ) was located in McMurdo station (77° 50' 47" S, 166° 40' 06" E) and could only be serviced by the 97.6° polar orbit. Starlink users from all 3 EU countries use the Frankfurt (Germany) PoP, while the AQ location used the Sydney (Australia) PoP. Since no ground stations can be reached by a LEO satellite serving the AQ vantage point via a single bent-pipe relay, it is likely that inter-satellite links (ISLs) are utilised.

B.3 Microsoft Teams: Starlink vs Terrestrial

We compare the performance of Starlink to a terrestrial network in fig. 10. Our terrestrial connection is a symmetric 1 Gbps link at the same large university where the Starlink connection we compare to is located. For both bitrate and framerate, Starlink exhibits a higher frequency of values below the respective medians.

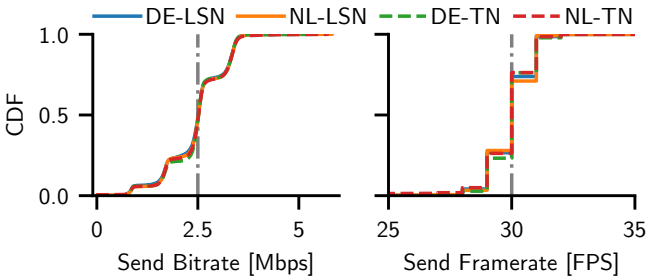


Figure 10: Teams performance in Central Europe from Starlink and a terrestrial university network.

B.4 Teams/LibWebRTC: Starlink Reconfigurations

For Microsoft Teams, we use a simple attribution heuristic with a 1.5 s window before and after a reconfiguration to determine whether a drop in bitrate is attributable to a Starlink reconfiguration. This window is chosen based on empirical observations from the `LibWebRTC` analysis, where we can inspect the time between a reconfiguration and `GCC`'s state change to *overuse* (see §4.2). For our European vantage points, we argue that a 1 s window would suffice. However, for AQ we observe that the reaction time is increased, likely due to the additional ISL hops and the longer RTTs involved. Figure 11 shows the distribution of overuse transitions within 5 s of a reconfiguration for all four vantage points. We argue that a sufficient value for AQ is 1–2 s, and we therefore choose 1.5 s as a compromise for all vantage points.

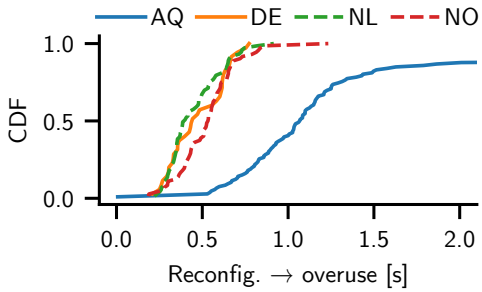


Figure 11: Distribution of overuse transitions within 5 s of a reconfiguration for all four vantage points.

B.5 LibWebRTC: Performance for 10 Mbps Target

Figure 12 shows the sender and receiver-side bitrate and framerate for 10 Mbps targeting 60 FPS at the Germany (DE) location. Only for 9.5% of the time, the target bitrate is achieved, while the target framerate is achieved more consistently for 41.5% of the time.

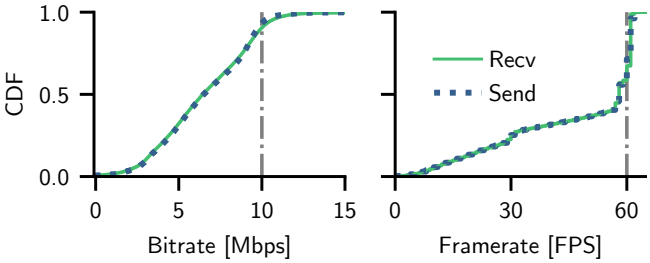


Figure 12: Received and send bitrate and framerate for 10 Mbps from the DE vantage point.

B.6 LibWebRTC: 2.5 Mbps vs 10 Mbps Tail Loss

Since the median loss is 0%, we characterize the tail loss for both 2.5 Mbps and 10 Mbps bitrate targets. Figure 13 shows the CCDF (log scale) of per-interval loss rate over 100 ms intervals. We observe that the 2.5 Mbps target experiences loss less frequently (loss occurs in 1.0% of intervals vs 3.5% for 10 Mbps). Both targets have similar loss rates in the long tail from ~2.5–100%. Thus, tail losses appear independent of target bitrate, despite increased loss rates around reconfigurations for 10 Mbps (see fig. 7 in §4.2).

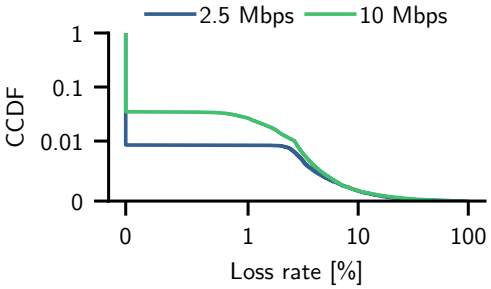


Figure 13: CCDF (logarithmic y-axis) of tail loss for 2.5Mbps and 10Mbps for DE.