Low Latency Live Video Streaming over a Low-Earth-Orbit Satellite Network with DASH

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

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

ABSTRACT

Despite Starlink's recent rapid growth in building a global low-earth-orbit satellite constellation and providing high-speed, low-latency Internet services, the implications of low-latency live video streaming over Starlink, especially given its fluctuating latency and regular satellite handovers, are yet to be thoroughly examined. In this paper, we conducted a thorough measurement study on the Starlink access network across different protocol layers and multiple geographical Starlink installations, including one where laser inter-satellite links are utilized in practice. We also performed a comprehensive latency target-based analysis of low-latency live video streaming with three state-of-the-art adaptive bitrate (ABR) algorithms in dash.js over Starlink. We presented a novel ABR algorithm designed for low-latency live video streaming over Starlink networks which leverages satellite handover patterns observed from measurements to dynamically adjust video bitrate and playback speed. Performance evaluations of the proposed algorithm were conducted using both a purpose-built network emulation testbed and actual Starlink networks. The results demonstrate that the proposed algorithm effectively utilizes the predictable Starlink satellite handover pattern and network characteristics. This effectively delivers a better quality of experience for low-latency live video streaming, characterized by high average video playback bitrates, minimal rebuffering events, and reduced visual quality fluctuation.

CCS CONCEPTS

• **Information systems** → **Multimedia streaming**; • **Networks** → **Network performance evaluation**.

KEYWORDS

DASH, Starlink, Low Latency Live Streaming

ACM Reference Format:
Jinwei Zhao and Jianping Pan. 2018. Low Latency Live Video Streaming over a Low-Earth-Orbit Satellite Network with DASH. In *Proceedings of (MMSys'24).* ACM, New York, NY, USA, 11 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

By integrating the distinct capabilities of space, aerial, and terrestrial networks, Space-Air-Ground Integrated Networks (SAGINs) are expected to revolutionize future Internet connectivity through enhanced flexibility and expansible network coverage. In addition to supporting the growing traffic of terrestrial networks, SAGINs will broaden Internet access to remote regions, including rural areas, oceans, and mountainous terrains [11]. Starlink, a division of SpaceX [17], stands out as a pivotal player in offering Internet service through a constellation of low-earth-orbit (LEO) satellites. These mass-produced small satellites bridge the communication between Starlink user terminals (UTs) and ground stations (GSs). While the concept of satellite Internet is not novel, Starlink's strategy in building a large LEO satellite constellation narrows the bandwidth and latency gap with conventional terrestrial networks. Specifically, SpaceX deploys Starlink satellites at an approximate altitude of 550 km, in contrast to traditional satellite communication networks that rely on either geosynchronous equatorial orbit (GEO) or medium-earth-orbit (MEO) satellites, which are positioned at higher altitudes and have wider coverage but suffers from higher latency and limited capacity. As of June 2023, SpaceX has deployed more than 4,600 LEO satellites and attained global coverage. Nevertheless, SpaceX's constellation ambition extends to launching up to 42,000 LEO satellites and eventually constructing multiple orbital shells [16].

The recent rapid growth and development of Starlink has attracted significant attention from both the industry and research community [10]. Due to the characteristics of LEO satellite networks, UTs utilize phased array antennas to track the moving satellites and perform frequent handovers between satellites to maintain network connectivity. Tanveer et al. [21] observed that Starlink employs a global controller for managing terminal-to-satellite scheduling. Notably, Starlink satellite handover happens every 15 seconds, specifically, at the 12th, 27th, 42nd, and 57th (12-27-42-57) second past every minute, synchronized globally. Pan et al. [14] conducted an extensive measurement study on the Starlink satellite access network, gateway, point-of-presence (PoP) architectures and the global backbone topology. It revealed that the round-trip-time (RTT) from the UT to the GS experiences significant fluctuations and is higher than that of conventional terrestrial Internet access via fiber optics, digital subscriber line (DSL), or cable modem.

Regarding the frequent satellite handover events and fluctuating latency, existing research indicates that Starlink can support a wide range of multimedia services with high-quality assurance, including video-on-demand (VoD) and live video streaming, given adequate playback buffers are configured properly. However, the performance remains insufficient for more demanding applications including bidirectional video streaming such as interactive video conferencing, immersive AR/VR/XR applications, 360-degree and volumetric video streaming and low-latency live (LLL) video streaming. For VoD services over Starlink, the end-to-end performance

MMSys'24, April 15–18, 2024, Bari, Italy

is on par with conventional terrestrial networks [23]. Despite frequent handover events and occasional outages of Starlink, the large playback buffer employed in VoD players is usually adequate to compensate for these interruptions, thereby ensuring a smooth viewing experience for end-users. However, for live video streaming, the necessity to meet latency targets introduces additional challenges for service providers endeavoring to ensure a low-latency experience, particularly within Starlink's dynamic network environment. Various application scenarios, such as live sports events, cloud gaming, and interactive live broadcasting, require distinct latency targets. Nowadays, typical terrestrial broadcasting (DVB-T) latencies range from 6 - 8 seconds. Given the widespread adoption of Starlink globally, the demand for LLL video delivery over satellite networks is undeniably significant. O'Hanlon et al. [13] conducted a comprehensive analysis of the low-latency performance of three adaptive bitrate (ABR) algorithms in dash.js, namely *Dynamic*, *L2A-LL*, and *LoL+*, considering a variety of latency targets (3, 5.5, 8, and 15 seconds) and configuration options. Nonetheless, it remains unknown how the fluctuating latency and frequent satellite handover events affect the performance of LLL video streaming ABR algorithms in dash.js.

In this paper, we conducted a thorough measurement of the Starlink access network across different protocol layers and geographical Starlink installations, including one where laser inter-satellite links (ISLs) are utilized in practice. We performed a latency target-based analysis of LLL video streaming over Starlink networks. We proposed a novel LLL video streaming ABR algorithm specifically designed for streaming over Starlink satellite networks. It leverages the satellite handover patterns observed from measurements to dynamically adjust the video bitrate and playback speed, thereby ensuring a seamless viewing experience with high average video playback quality, minimal rebuffering events and reduced visual quality fluctuation. The main contributions of this paper are therefore four-fold and can be summarized as follows:

- Assessed the latency and throughput of the Starlink access network across different protocol layers and geographical locations, taking into account scenarios both with and without ISLs.
- Conducted a latency target-based measurement and analysis of three state-of-the-art LLL video streaming ABR algorithms in dash.js over Starlink networks.
- Modelled the LLL video streaming with contextual multi-armed bandit (CMAB) algorithms and proposed a novel ABR algorithm, improving the QoE of LLL video streaming over Starlink networks.
- Implemented a prototype of the proposed algorithm with dash.js and evaluated its performance using both a purpose-built network emulation testbed and actual Starlink networks.

The remainder of the paper is organized as follows. Section 2 introduces some related works in LLL video streaming ABR algorithms and Starlink measurement studies. Section 3 details the testbed setup and outlines the results of our measurements. Section 4 articulates the problem of LLL video streaming using CMAB algorithms and presents the design of our proposed algorithm. Section 5 evaluates the performance of the proposed algorithm in both network emulation settings and real Starlink networks. Section 6 discusses the open research challenges and finally, Section 7 concludes the paper, highlighting potential improvement and future work.

2 RELATED WORKS

In this section, we provide a brief overview of the related works involved with LLL video streaming in DASH and the measurement of Starlink networks.

2.1 LLL Video Streaming in DASH

Over the past decade, numerous ABR algorithms have been proposed to improve the QoE of video streaming in DASH. In this section, our concentration is on three LLL video streaming ABR algorithms available in the dash.js [5] reference player, namely *Dynamic*, *L2A-LL*, and *LoL+*.

The default *Dynamic* [18] algorithm in dash.js is a hybrid ABR algorithm consisting of throughput-based rule and buffer-based BOLA algorithm [19]. Karagkioules et al. [9] proposed Learn2Adapt-LowLatency (*L2A-LL*), which uses online convex optimization to provide robust video bitrate adaptation strategies without relying on specific parameter tuning, channel model assumptions, throughput estimation or application-specific adjustments. Bentaleb et al. [2] introduced *LoL+*, a learning-based ABR algorithm that employs a self-organizing map (SOM) to adapt the bitrate at every segment download boundary. *LoL+* contains four different modules, namely a playback speed control module that combines the current latency and buffer level to control the playback speed; a throughput measurement module that accurately provides throughput estimation based on CMAF chunks; a QoE evaluation module that computes the QoE considering five metrics: selected bitrate, number of bitrate switches, rebuffering duration, latency and playback speed; and a weight selection module that implements a dynamic weight assignment algorithm for the SOM model features.

O'Hanlon et al. [13] conducted a latency target-based analysis of these three ABR algorithms concerning a range of latency targets (3, 5.5, 8, and 15 seconds) and configuration options for the LLL video streaming performance. The *Dynamic* algorithm performs the best in terms of low rebuffering duration, with the least number of stalls and the shortest overall rebuffering time. In terms of live latency, the *Dynamic* algorithm also provides the smallest deviation from the latency target in all the scenarios evaluated and provides the most stable but lower video bitrate quality, whilst *L2A-LL* and *LoL+* can reach a higher video quality level. They further demonstrated the impact of the `FastSwitching` option in dash.js. When enabled, this option replaces low-bitrate video segments in the playback buffer with high-bitrate segments during a quality increase, rather than appending them directly to the end of the current playback buffer. However, the `FastSwitching` option brings a significant number of re-requests that consume more bandwidth but do not generally increase the QoE. Thus, the option is recommended to be disabled in LLL video streaming.

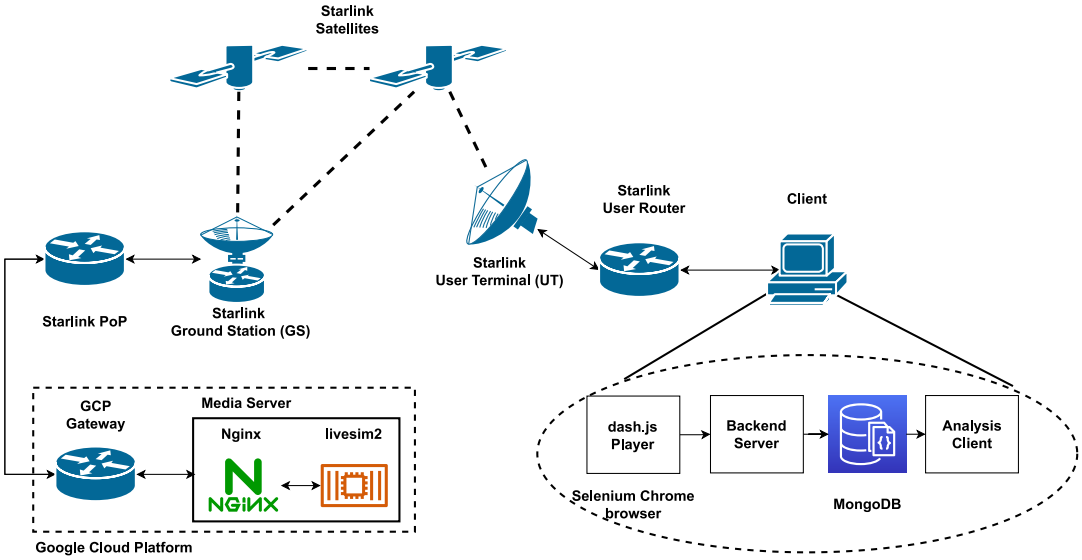


Figure 1: Starlink DASH testbed architecture

2.2 Starlink

Since the launch of Starlink's beta testing in 2020, it has attracted considerable research interest from both the industrial and academic sectors. Research topics span from access network measurements [14, 21, 23] to physical layer signal structure analysis [3] and routing protocol design [22], among others. In this section, we focused on the evaluation of Starlink access networks.

Zhao et al. [23] conducted a systematic measurement on real-time multimedia services over Starlink, including VoD (YouTube), live-streaming (Twitch) and video conferencing service (Zoom). The Starlink network typically delivers satisfactory performance for multimedia services. However, factors like extreme weather, satellite handover events, and changes in packet routing paths can impact its performance. VoD services remain largely unaffected due to their substantial playback buffers, which can compensate for Starlink's occasional short-time outages and frequent satellite handover events. In contrast, interactive applications such as video conferencing and live video streaming, face more pronounced performance challenges. Tanveer et al. [21] noted a consistent pattern in Starlink's satellite handover events occurring every 15 seconds. Specifically, these events manifest in the latency characteristics at the 12th, 27th, 42nd, and 57th seconds of each minute. This pattern suggests that SpaceX utilized a globally time-synchronized scheduler to manage the access network between the UTs and satellites. Pan et al. [14] provided comprehensive insights into the access, gateway, PoP, and backbone network architectures of Starlink, illustrating the findings with detailed network topology diagrams, derived from collaborative measurements across the world.

3 MEASUREMENT

In this section, we first present the testbed setup and measurement results of Starlink access networks. Our measurements were conducted across different protocol layers and multiple geographical Starlink installations. We then outline the approach behind our latency target-based analysis of LLL video streaming over Starlink networks. The subsequent measurement results, in conjunction with the performance evaluation of the proposed algorithm, are presented in Section 5.

3.1 Setup

The architecture of our Starlink measurement testbed is shown in Figure 1. For each Starlink installation, we deployed a virtual machine as the media server in the nearest Google Cloud Platform (GCP) availability zone. For example, for the Starlink installation in the Pacific Northwest, the nearest Starlink GS and PoP are located in Seattle, USA. To minimize additional terrestrial network latency, we deployed the media server in GCP's *us-west1-a* availability zone, physically located in Oregon, USA. Our measurements indicate that a single hop exists between the Starlink Seattle PoP and the GCP gateway in this availability zone, with a latency below 10 milliseconds. We also have access to a Starlink installation on an island in the western Indian Ocean, proximate to the eastern coast of Africa. As of July 2023, the only Starlink GS and PoP in Africa are physically located in Lagos, Nigeria [12], located on the western coast of the continent. Considering the absence of Starlink GSs within a 5,000 km radius of this Starlink installation, we utilized `mtr` and `traceroute` to trace the packet from this Starlink UT to the Internet. The results derived from the reverse DNS resolution using `nslookup`, (`customer.lgosnga1.pop.starlinkisp.net`), revealed that packets are routed via the Starlink Lagos PoP in Nigeria. Our inference is that the packets are relayed and transmitted through multiple laser inter-satellite links, commonly referred to as ISLs. When ISL is in use, packets travel through multiple satellites before being transmitted to the GS as shown in Figure 1. We deployed the media server for this region in GCP's *europe-southwest1-a* availability zone, physically located in Madrid, Spain, where the Starlink Madrid PoP is interconnected with Starlink Lagos PoP through Starlink's terrestrial backbone infrastructure [14].

Table 1: Starlink access network latency (ms) to the GS

Starlink Installations	Min	Median	Average	σ
Without ISL	16.7	42.8	47.5	20.9
With ISL	59.1	109.0	119.8	43.7

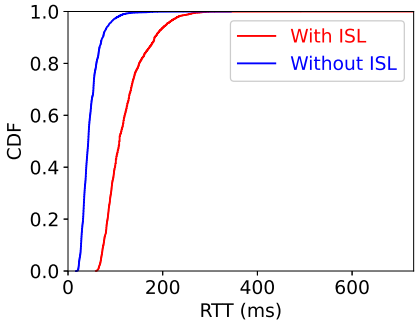
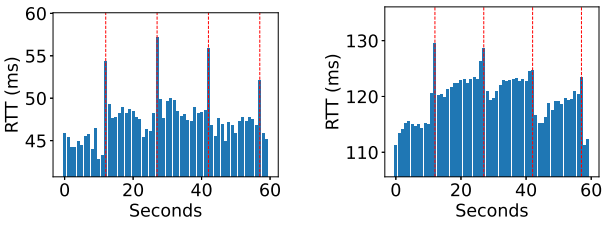


Figure 2: CDF of the RTT to the GS



(a) Avg. RTT to GS without ISL **(b) Avg. RTT to GS with ISL**

Figure 3: Average RTT to GS at every second

For each Starlink installation, a mini PC is directly connected to the Starlink user router via an Ethernet cable. Network measurement scripts and the LLL video streaming stack are deployed on the mini PC. The LLL video streaming stack consists of a modified dash.js player, a backend server, a MongoDB database and an analysis client. The modified dash.js player sends the playback metrics to the backend server through REST APIs, which are then stored in the MongoDB database. The analysis client queries the MongoDB database after playback sessions and generates corresponding figures. On each media server, we deployed `livesim2` [6], a DASH live source simulator implemented in Go by the DASH Industry Forum, which takes segmented DASH source videos as the input assets, and produces a wall-clock (UTC) synchronized linear stream of video segments. By looping the input VoD DASH assets and dynamically adjusting the timestamps, a perpetual "live" video stream is made available to the clients. Nginx is installed as the frontend web server to serve the MPD files and the corresponding "live" video streams.

3.2 Measuring Starlink Access Network

Pan et al. [14] measured the structure of Starlink access networks. Their findings indicate that, from a user's perspective, the GS is always accessible at a carrier-grad NAT (CGNAT) address `100.64.0.1`. This applies to regular Starlink subscribers, except those on the business or priority subscription plan with the public address option. Subscribers of the latter are allocated a static public IPv4 address bounded to their Starlink user routers. The IPv4 network address translation (NAT) happens at the nearest GS for regular Starlink subscribers.

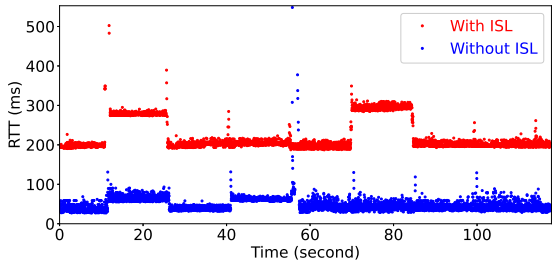
Our first measurement is to evaluate the latency performance of Starlink access networks. We started with measuring the RTT between clients and the nearest GS using `ping`. The interval for the `ping` command is set at 10 milliseconds, with a total continuous duration of 60 minutes. A summary of the Starlink access network latency to the GS is shown in Table 1.

For the Starlink installation without ISL, the overall access latency can easily be maintained at around 50 milliseconds, with the minimal RTT being 16.7 milliseconds, the median RTT being 42.8 milliseconds and the average RTT being 47.5 milliseconds. For the Starlink installation with ISL, the overall access latency fluctuated more than the one without ISL, with the minimal RTT being 59.1 milliseconds, the median RTT being 109.0 milliseconds, the average RTT being 119.8 milliseconds and a higher standard deviation σ=43.7 milliseconds than σ=20.9 milliseconds without ISL. The CDF of both access latency distributions is shown in Figure 2.

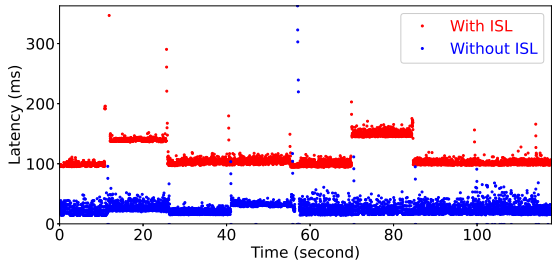
We calculated the average RTT to the nearest GS at every second for both Starlink installations, as shown in Figure 3. It shows the comparison of averaged RTT to the nearest GS for two geographical Starlink installations (western Indian Ocean and Pacific Northwest) with and without ISL utilized. It revealed an obvious pattern that at 12-27-42-57 seconds of every minute, the average RTT between Starlink UTs and their corresponding GSs spikes, which indicates that satellite handover events happen at synchronized seconds globally at different geographical locations. This observation is consistent with the findings in [21]. However, our observations in Figure 3(a) and Figure 3(b) indicate that the potential impact of satellite handover events is more pronounced on the Starlink installation that did not utilize ISL during our measurement.

To gain deeper insights into how satellite handover events influence the end-to-end (E2E) performance of different applications, we carried out time-synchronized measurements of latency and throughput, as indicated in Figure 4. E2E latency and throughput are measured with `IRTT` and `iPerf3` respectively. We deployed `IRTT` and `iPerf3` daemon programs on the media servers as illustrated in Section 3.1 and Figure 1. All the media servers and clients are configured with NTP time synchronization using `chrony` and Google Public NTP service[1], such that `IRTT` can provide accurate one-way delay (OWD) measurements. `IRTT` can also provide latency measurement with higher time resolution than `ping`, by keep sending UDP packets on a fixed time interval regardless of whether replies are received. By utilizing UDP instead of ICMP, it can also avoid the potential ICMP deprioritization on some network devices and provide more realistic measurement results close to real-world applications. In our measurements, the `IRTT` request interval is set

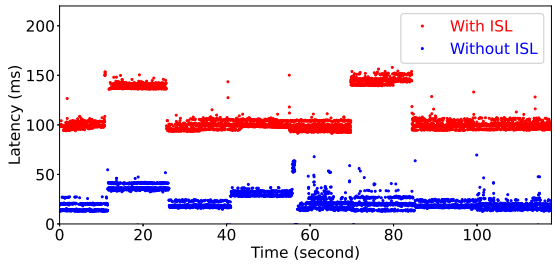
[1] https://developers.google.com/time



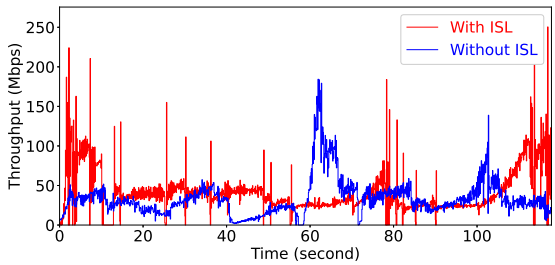
(a) E2E Latency



(b) Uplink OWD



(c) Downlink OWD



(d) Downlink Throughput

Figure 4: Time synchronized latency and throughput measurements

to 10 milliseconds, consistent with the ICMP `ping` experiments. To measure the throughput, we utilized `iPerf3` and set the report interval to 100 milliseconds, which is the minimal `iPerf3` report interval. In this paper, our primary focus was on evaluating the downlink throughput performance from the media server to the video streaming clients, aligning with typical video streaming scenarios. However, a brief discussion on the uplink performance for live broadcasting scenarios is provided in Section 6.1. TCP was used for all the `iPerf3` throughput measurements and the TCP congestion control algorithm used is Cubic, which is the default congestion control algorithm in the Ubuntu 22.04 operating system we used.

Table 2: Bitrate ladder of the LLL video dataset

Resolution	Frame rate (fps)	Bitrate (Kbps)
1920x1080	50	6000
1920x1080	25	5100
1920x1080	50	4900
1024x576	25	1500
1024x576	25	1200
768x432	25	900
512x288	25	450
480x270	12.5	300

Figure 4 shows a 2-minute window of the time-synchronized latency and throughput measurements. There is a notable variance in latency patterns across each 15-second interval. At the boundaries of each timeslot, both the uplink and downlink OWD exhibit surges, aligning with the satellite handover events. Regarding OWD, the downlink OWD exhibits a more pronounced "strip band" pattern compared to the uplink OWD, especially for Starlink installations without ISL in Figure 4(c). This distinction likely arises because downlink access is allocation-based, designating specific timeslots in a media access frame for a particular UT. Conversely, the uplink operates on either a contention-based system or a poll-randomize-grant mechanism [8]. Figure 4(d) shows that while the RTT might stay relatively stable between different 15-second timeslots as the satellite handover events happen, the client and server have to go through the slow start pattern in each timeslot because of RTT timeout or packet loss, which can significantly impact the throughput performance. It is important to note that the utilization of ISL does not directly impact downlink throughput. Instead, it is influenced by Starlink's capacity limitations in specific regions.

3.3 Measuring LLL Video Streaming

Our measurement on LLL video streaming over Starlink networks is based on dash.js v4.7.1, which was released in June 2023. Three ABR algorithms and four different latency targets are evaluated, namely *Dynamic*, *LoL+* and *L2A-LL* and the latency targets range from 3 seconds to 6 seconds. The video dataset used in our measurements is obtained from the CTA WAVE Test Project [4]. The bitrate ladder is shown in Table 2, which contains 8 representations with different resolutions and frame rates, and the target H.264 encoding bitrate ranges from 300 Kbps to 6000 Kbps. The video is segmented into 2-second aligned video segments. We played back the "live" video stream produced by `livesim2` for 5 minutes in each round of measurement and repeated the same measurement

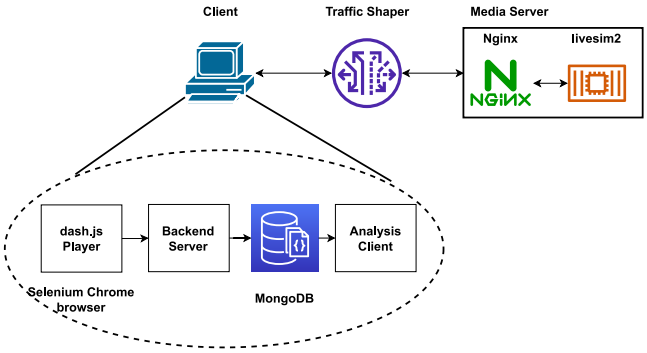


Figure 5: Starlink DASH emulation testbed architecture

10 times for each latency target and ABR algorithm. The following metrics are collected every 100 milliseconds on our modified dash.js player during playback,

- Average playback bitrate
- Average live latency
- Rebuffering time (%)
- No. of bitrate switches

and they are periodically sent to the backend server and then stored in MongoDB databases for the analysis client to evaluate the performance of LLL video streaming ABR algorithms.

In addition to evaluating the performance of LLL video streaming ABR algorithms in real Starlink networks, we also conducted measurements in emulated network environments. The architecture of the purpose-built emulation testbed is similar to the real Starlink testbed with minor modifications, which is shown in Figure 5. All the components in Figure 5 are deployed on a single machine using Docker containers and orchestrated by Docker Compose. A "Traffic Shaper" container is added to the emulation testbed between the dash.js player and the frontend Nginx web server to add artificial network latency following the Starlink satellite handover pattern and the measured latency performance. The motivation behind network emulation is to provide a repeatable environment for performance evaluation under extreme network conditions. While Starlink indeed performs satellite handovers every 15 seconds as shown in Section 3.2, we also observed that in some cases, the latency variation is not significant enough to affect the performance of LLL video streaming. With network emulation, we can add periodic and artificial but still realistic packet losses and latency variations according to the Starlink satellite handover pattern at 12-27-42-57 seconds every minute. In our emulation, we employed the Linux network utility `tc` and `netem` to introduce artificial delay and packet loss. Specifically, this was done one second before and after 12-27-42-57 seconds every minute. That is, during the intervals of (11, 12, 13), (26, 27, 28), (41, 42, 43), and (56, 57, 58) seconds every minute, artificial network delays and packet loss were added in the links between dash.js player and the frontend Nginx web server. The presumption of adding extra seconds is even though the actual Starlink satellite handover events at the physical layer take roughly 100 milliseconds, they might introduce extended disturbances to upper-layer applications. For comparison, we also incorporated a control group utilizing terrestrial fiber optics Internet access. The measurement results are shown in Figure 6 to Figure 8, along with a detailed performance evaluation and comparison with the proposed algorithm in this paper presented in Section 5.

Table 3: Summary of key notations

Notation	Definition
K	Number of arms
T	Total number of rounds
$a(t)$	Arm selected by agent at round t
$b(t)$	Context vector revealed to agent at round t
μ_k	Distribution parameter for arm k
q	1st round of the current 15-second timeslot
$\mathcal{H}_q^{t-1}$	History beginning from q up to round $t-1$
$a^*(t)$	Optimal arm at round t
$r(i)$	Reward for video segment i
C	Total number of rebuffering events
C_k	Total number of rebuffering events at bitrate k
t_j	Rebuffering time for event j
t_j^k	Rebuffering time for event j at bitrate k
$\mathrm{B}(i)$	Video bitrate of segment i
$\mathrm{B}_{\max}$	Highest video bitrate available in the MPD file
$\mathrm{R}(t)$	Playback speed at t
$\mathrm{X}(t)$	Estimated network throughput at t
$\mathrm{L_N}(t)$	Measured network latency at t
$\mathrm{LL_{target}}$	Latency target for the playback session
$\mathrm{LL_{current}}(i)$	Playback latency when downloading segment i
$\mathrm{QoE_{P1203}}(i)$	ITU-T P.1203 Mode 0 QoE score for segment i
$\mathrm{QoE}(i)$	Final QoE for segment i

4 PROBLEM FORMULATION

In this section, we first present the problem formulation of LLL video streaming using CMAB algorithms. Then, a novel QoE-driven reward function and catch-up policy are proposed to improve the QoE of LLL video streaming under Starlink networks. The key notations used in this paper are summarized in Table 3.

4.1 System Model

The problem setting of a general CMAB algorithm can be defined as in [1]. In this paper, we model the LLL video streaming scenario with CMAB algorithms as follows.

The video bitrate adaptation problem in live video streaming can be formulated as an online decision-making process. The video player, referred to as an agent, is presented with K different bitrate levels to choose from in each of T rounds. The total video playback time T could be infinite. The K video bitrate levels are referred to as K arms. The decision on which bitrate should be selected for playback is analogous to choosing an arm to pull by the agent. Before the agent pulls an arm in each round, a context vector $b(t) \in \mathbb{R}^d$ is presented, which contains the context information when the current round happens. The context information can include metrics such as the current network latency, current video playback speed, estimated network throughput, etc. In this paper, we define the

context vector $b(t)$ as follows,

$$b(t) = [\mathrm{L_N}(t), \mathrm{R}(t), \mathrm{X}(t)] \tag{1}$$

where at time t, $\mathrm{L_N}(t)$ is the measured network latency to the media server, $\mathrm{R}(t)$ is the current playback speed, and $\mathrm{X}(t)$ is the estimated network throughput, respectively.

The agent chooses an arm that is anticipated to yield the highest expected reward in the current round. That is, the video bitrate selection should yield the highest expected QoE in the current round, without causing playback interruptions or rebuffering events. In each round, only the reward of the chosen arm is revealed to the agent, leaving the rewards of the unselected arms undisclosed.

A history $\mathcal{H}^{t-1}$ containing all the previous rewards of the selected arms and their respective contexts up to round $t-1$ can be compiled by the agent before round t. In this paper, we only consider the history $\mathcal{H}_q^{t-1}$ during the current 15-second timeslot beginning from round q,

$$\mathcal{H}_q^{t-1} = \left\{a(s), r_{a(s)}(s), b(s), s = q, \ldots, t-1\right\} \tag{2}$$

where $a(s)$ denotes the arm played at round s and $r_{a(s)}(s)$ is the reward for arm $a(s)$ at round s, and q is the first round during the current 15-second timeslot.

Given $b(t)$, the reward for arm k at round t is derived from an unknown distribution with mean $b(t)^T \mu_k$, where $\mu_k \in \mathbb{R}^d$ is a constant parameter unknown to the agent. $b(t)^T$ denotes the matrix transpose of $b(t)$. The expected reward of $r_k(t)$ for each arm k given $b(t)$ and $\mathcal{H}_q^{t-1}$ can be defined as,

$$\mathbb{E}\left[r_k(t) \mid b(t), \mathcal{H}_q^{t-1}\right] = b(t)^T \mu_k. \tag{3}$$

In a CMAB scenario, an agent employing an online learning algorithm must decide which arm $a(t)$ to pull at each round t, considering both the history $\mathcal{H}_q^{t-1}$ and the context vector $b(t)$ of the current round made available to the agent.

Define $a^*(t)$ as the optimal arm at time t that provides the maximum expected reward, formulated as $a^*(t) = \arg\max_k b(t)^T \mu_k$. Let $\Delta_k(t)$ represent the difference in reward between the optimal arm $a^*(t)$ and arm k at time t, i.e.,

$$\Delta_k(t) = b(t)^T \mu_{a^*(t)} - b(t)^T \mu_k \tag{4}$$

Then, the regret at time t is defined as

$$\mathrm{regret}(t) = \Delta_{a(t)}(t) \tag{5}$$

It is worth noting that the CMAB problem setting presented here deviates from the one outlined in [1]. In our scenario, the assumption is that each arm k is revealed with the identical context vector $b(t)$. Moreover, each arm follows an unknown yet unique distribution defined by its respective μ_k. We also only consider the history $\mathcal{H}_q^{t-1}$ during the current 15-second timeslot beginning from round q, because of the unique and fluctuating latency pattern in each timeslot as shown in Figure 4.

4.2 Reward Function Design

We employed ITU-T P.1203 [15] for the QoE-driven reward function. Specifically, we used the ITU-T P.1203 Mode 0, which derives from video stream metadata and yields an overall QoE score represented as a Mean Opinion Score (MOS) for video segments. We took the O.46 score from ITU-T P.1203 model outputs, which is a single media session quality score, on a 1–5 quality scale in real numbers. We develop a QoE-driven reward function with the primary objective of reducing the live latency, minimizing rebuffering events and increasing playback bitrate,

$$\mathrm{QoE}(i) = \mathrm{QoE}_{\mathrm{P1203}}(i) * \frac{\mathrm{LL_{target}}}{\mathrm{LL_{current}}(i)} * \frac{\mathrm{B}(i)}{\mathrm{B_{max}}} - \frac{\sum_{j=1}^{C_k} t_j^k}{\sum_{j=1}^{C} t_j} \tag{6}$$

where $\mathrm{QoE}_{\mathrm{P1203}}(i)$ represents the MOS score for video segment i calculated by ITU-T P.1203 Mode 0, $\mathrm{LL_{target}}$ represents the latency target in this playback session, $\mathrm{LL_{current}}(i)$ represents the current live latency when downloading video segment i, C represents the total number of rebuffering events, C_k represents the number of rebuffering events happens at bitrate k, t_j represents the duration of rebuffering event j, t_j^k represents the duration of rebuffering event j happens at bitrate k, $\mathrm{B}(i)$ is the video bitrate for segment i, and $\mathrm{B_{max}}$ is the highest video bitrate available in the MPD file.

The agent pulls an arm before downloading each video segment i, and the corresponding reward $r(i)$ for video segment i obtained is defined as,

$$r(i) = \mathrm{QoE}(i) \tag{7}$$

The objective of the agent is to pull the best arm which yields the highest expected reward in each round for video segment i.

4.3 Catch-up Policy

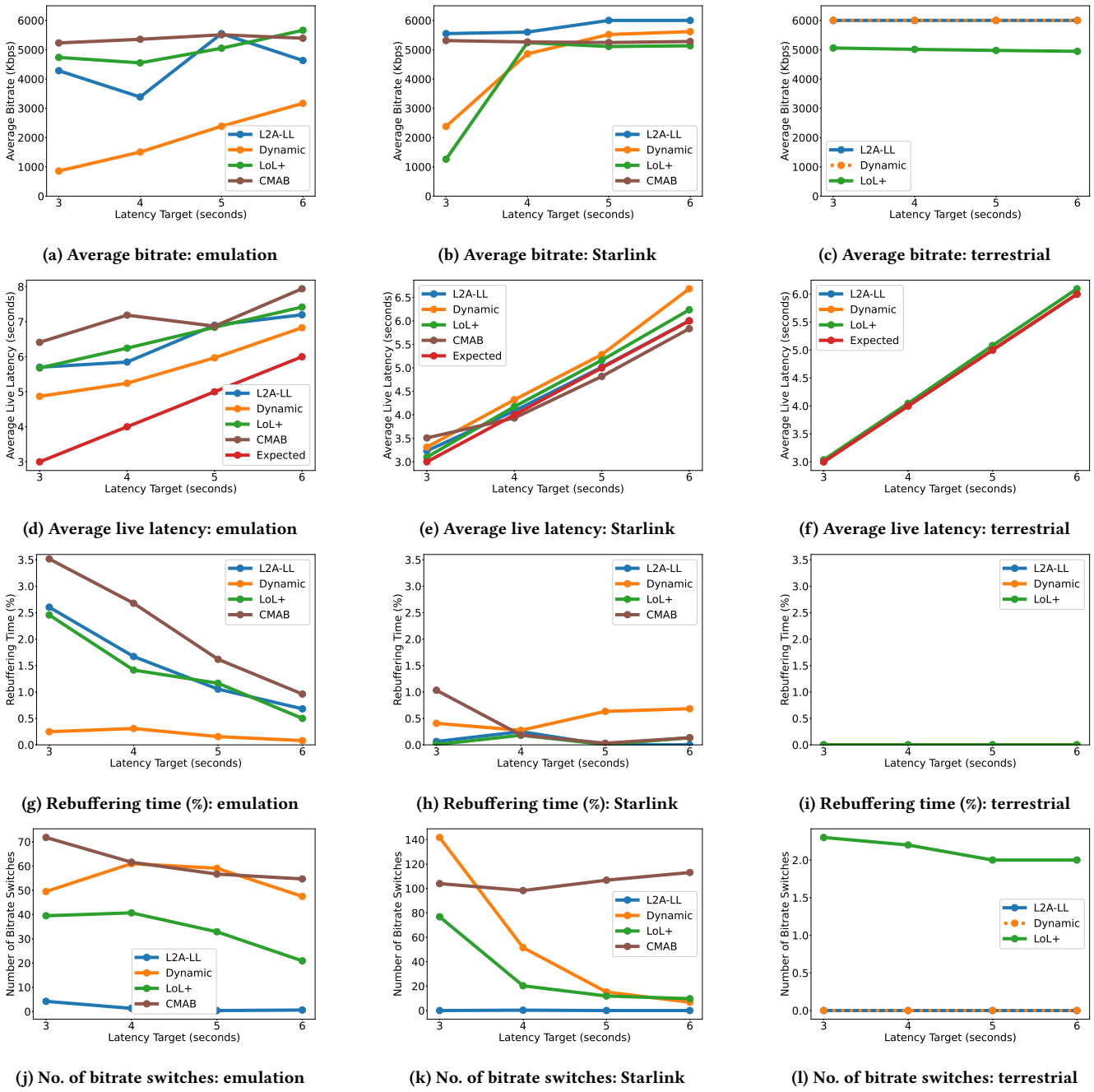
A similar playback speed control module as in *LoL+* [2] is employed in the proposed algorithm. The main goal of our catch-up policy and playback speed control module is to avoid potential playback interruptions during satellite handover periods. Although the actual satellite handover at the physical layer only takes around 100 milliseconds, it could bring a prolonged impact on upper-layer applications. Thus, we define our playback speed control module as follows,

- The current buffer level is below the safe threshold, or it is currently within the satellite handover period: slow down the playback speed below 1.0.
- The current buffer level is sufficient, and it is not within the satellite handover period:
  - The live latency is close to the latency target ($\epsilon = \pm 2\%$): maintain playback speed at 1.0.
  - The current live latency is lower than the latency target: slow down playback speed.
  - The current live latency is higher than the latency target: speed up playback speed.

The satellite handover period is defined using the same motivation as in Section 3.3. The intervals of (11, 12, 13), (26, 27, 28), (41, 42, 43), and (56, 57, 58) seconds every minute, are considered as the satellite handover periods.

5 EVALUATION

We evaluated the performance of the proposed algorithm and compared it with the other three LLL video streaming ABR algorithms both in network emulation settings and real Starlink networks. The performance evaluation on Starlink networks was only conducted



(a) Average bitrate: emulation

(b) Average bitrate: Starlink

(c) Average bitrate: terrestrial

(d) Average live latency: emulation

(e) Average live latency: Starlink

(f) Average live latency: terrestrial

(g) Rebuffering time (%): emulation

(h) Rebuffering time (%): Starlink

(i) Rebuffering time (%): terrestrial

(j) No. of bitrate switches: emulation

(k) No. of bitrate switches: Starlink

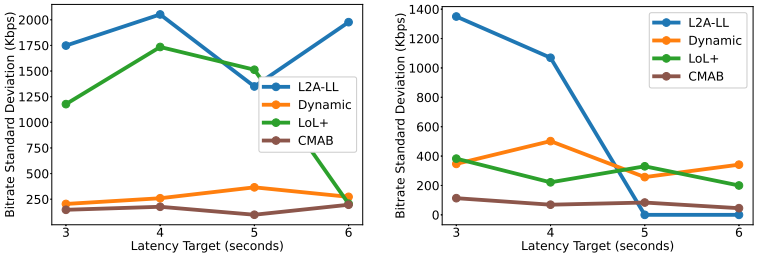
(l) No. of bitrate switches: terrestrial

Figure 6: The measurement and performance evaluation

on the Starlink installation without ISL being utilized. The implementation of this paper is available on GitHub[2]

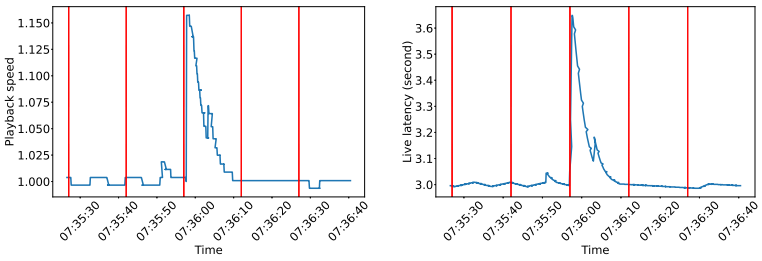
We implemented our CMAB-based ABR algorithm on dash.js v4.7.1 and built an end-to-end evaluation prototype. To solve the online learning problem with CMAB algorithms, `MABWiser` [20] which provides fast prototyping with various CMAB algorithms is chosen in our implementation. While other CMAB algorithms are available, in this paper, our implementation chose Linear Thompson Sampling (LinTS) [1] as our solution. `MABWiser` is originally implemented in Python. To integrate it with dash.js reference player implemented in JavaScript and based on Media Source Extensions (MSE), we utilized `Pyodide` [7], which is a WebAssembly-based Python runtime for browsers. We implemented the core CMAB algorithm for bitrate adaptation and QoE calculation in Python

[2]The URL is anonymized for double-blind review.



(a) Emulation **(b) Starlink**

Figure 7: Bitrate standard deviation



(a) Playback speed **(b) Live latency**

Figure 8: Playback speed and live latency during one playback session

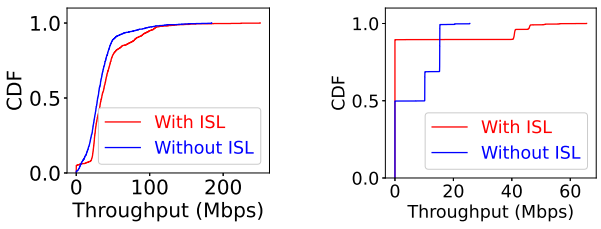
and interacted with dash.js through `Pyodide` APIs. We acknowledge that implementing our proposed algorithm in dash.js with WebAssembly-based Python is slower than a native JavaScript implementation, with our measurement, however, the performance overhead and the time cost to solve the CMAB problem in each round is below 100 milliseconds, which is negligible in our scenario. Similar to the latency target-based measurements in [13], we disabled the `FastSwitching` option in dash.js and used the "moof" parsing method for throughput calculation. The network latency to the media server $L_N(t)$ is measured in the backend server as shown in Figure 1 and Figure 5 and queried by dash.js clients through REST APIs. We set the *maxDrift* to 5s and *playbackRate* to 0.17. For *Dynamic* and *L2A-LL* algorithms, we set the catch-up mechanism to the *Default*, while *LoL+* and our proposed algorithm have their distinct catch-up mechanisms. As our proposed ABR algorithm takes advantage of predictable satellite handover patterns, we only evaluated its performance in network emulation settings and real Starlink networks, while the other three ABR algorithms are also evaluated in the terrestrial network setting as the control group. The exploration rate of LinTS is set to 1.0.

From Figure 6(a) to Figure 6(c), we can see all ABR algorithms show the same trend in increasing the average bitrate as the latency target increases, with the exception that *L2A-LL* shows a slight fluctuation in average bitrate in the emulation setting in Figure 6(a). In Figure 6(c), both *L2A-LL* and *Dynamic* maintain the highest average bitrate while the average bitrate of *LoL+* maintains at around 5100 Kbps for all the latency targets. In Figure 6(a) and Figure 6(b), both *Dynamic* and *LoL+* algorithms show significant fluctuations in average bitrate at different latency targets, especially when the latency target is below 4 seconds. On the other hand, the proposed CMAB-based ABR algorithm can always maintain a high average bitrate at different latency targets in both network emulation and real Starlink networks.

Figure 6(d) to Figure 6(f) shows the average live latency in all three network settings. In Figure 6(d), all ABR algorithms cannot reach the latency target with at least 1 second deviation. This might caused by the extreme network profile with a 3-second handover period in the network emulation setting. The *Dynamic* algorithm has the lowest average live latency, while our proposed CMAB-based ABR algorithm has the highest average live latency. However, in Figure 6(e) and Figure 6(f), all ABR algorithms have similar performance close to the latency target. Specifically, the proposed algorithm can achieve lower live latency than the expected latency target in Figure 6(e), when the latency target is larger than 4 seconds.

Figure 6(g) to Figure 6(i) show the ratio of rebuffering events in all three network settings. In Figure 6(g), all ABR algorithms have a decreasing rebuffering time ratio as the latency target increases. In the real Starlink network settings, the proposed algorithm can achieve a low rebuffering time ratio close to *L2A-LL* as shown in Figure 6(h).

Figure 6(j) to Figure 6(l) shows the number of bitrate switches in all three network settings. *L2A-LL* has the least number of bitrate switches across different latency targets. Both *Dynamic* and *LoL+* show a similar decreasing trend in the number of bitrate switches as the latency target increases. Our proposed algorithm



(a) Downlink throughput CDF **(b) Uplink throughput CDF**

Figure 9: CDF of throughputs

has a generally higher number of bitrate switches than other ABR algorithms, especially in Figure 6(j). However, Figure 7 shows that our proposed algorithm has the lowest bitrate standard deviation. This suggests that shifts in playback quality under the proposed algorithm bring less visual impact to users compared with other ABR algorithms. However, this implication should be further verified by other visual quality assessment methods such as VMAF in the future.

Figure 8 demonstrates the significance of considering satellite handover awareness when designing an ABR algorithm for LLL video streaming over satellite networks. It shows the actual playback speed and live latency during one playback session using the *Dynamic* ABR algorithm with a 3-second latency target. The figure shows that at `07:35:57`, the satellite handover event happens. However, the algorithm made a decision to increase playback speed, which resulted in a significant increase in live latency.

6 DISCUSSION

In this paper, we mainly focused on measuring the performance of Starlink access network and low latency ABR algorithms in one-way live video streaming scenarios from media servers to end users. However, there are still many open research challenges left to be explored in the future.

6.1 Video Ingestion Performance on the Uplink

As the popularity of live broadcasting and short video-sharing platforms such as Twitch and TikTok grows, the video ingest performance on the uplink is also becoming more important. However, the uplink performance of Starlink networks is significantly subpar compared to the downlink performance. As shown in Figure 9, the uplink throughput is significantly lower than the downlink throughput with very high fluctuations. Similar to Section 3.2, it is important to note that the utilization of ISL does not directly impact throughput performance. Instead, it is influenced by Starlink's capacity limitations in different geographical regions. In contrast to the "pull"-based model of live streaming on the downlink path, live broadcasting workflow involves streaming software such as OBS and uses RTMP/HLS/DASH to ingest video streams to the service provider's servers using a "push"-based model. In this case, the video ingestion clients can better utilize the predictable satellite handover patterns to dynamically adjust the video bitrate and sending rate to avoid potential bufferbloat and improve the video ingestion performance.

6.2 Other Research Challenges

With the widespread deployment of Starlink and other LEO satellite constellations, the dynamic nature of LEO satellite networks poses challenges to the traditional CDN architecture for media delivery. Given the inherent mobility of LEO satellites, the conventional paradigms of "local" or "edge" computing are being redefined, leaving issues such as resource allocation, storage, and caching as open areas for exploration.

As discussed in Section 3.2, the TCP performance over Starlink is significantly affected by the slow start pattern as satellite handover events happen every 15 seconds. Other congestion control algorithms, such as BBR, BBRv2, and HyStart++, may yield better TCP throughput performance when compared to Cubic on Starlink networks. On the other hand, QUIC, as the next-generation transport-layer protocol, is designed with 1-RTT connection establishment and 0-RTT connection resumption, which can potentially significantly reduce the impact of satellite handover events on TCP performance and therefore improve upper-layer application performance such as live video streaming.

The performance of demanding applications such as cloud gaming over LEO satellite networks remains an area requiring comprehensive investigation. Essentially, cloud gaming is interactive ultra-low latency live video streaming. Latency-sensitive scenarios, especially in games like first-person shooters, are particularly vulnerable to fluctuating latency and frequent satellite handovers, given they also encompass human interactions via Starlink's uplink channels.

7 CONCLUSION

In this paper, we conducted a thorough measurement study on the Starlink access network at multiple geographical Starlink installations, across different protocol layers from access latency and raw TCP throughput to application-layer LLL video streaming performances. Then, we proposed a novel ABR algorithm for LLL video streaming, tailored for Starlink satellite networks. The proposed algorithm utilized CMAB algorithms, with a novel reward function and catch-up policy considering satellite handover patterns. We implemented an end-to-end prototype of the proposed algorithm in dash.js and the performance evaluation was conducted on both a purpose-built network emulation testbed and real Starlink networks. The results illustrated that the proposed algorithm can achieve LLL video streaming with high video bitrate, low playback latency, low rebuffering ratio and less visual quality fluctuation. For future works, VMAF can also be used to evaluate the visual quality fluctuation of different ABR algorithms in LLL video streaming. The network emulation testbed can use trace-driven methodology to conduct more realistic emulations. Utilizing CMAF chunked encoding and chunked transfer over Starlink networks could further enhance the live streaming latency and QoE. More work can be done to improve the catch-up policy and reward function with more detailed analyses of satellite handover patterns and present theoretical regret-bound analysis for the CMAB-based algorithm. It is also worth investigating the performance of existing LLL video streaming ABR algorithms with satellite handover awareness and the performance of uplink video streaming over ISL-enabled Starlink networks.

REFERENCES

[1] Shipra Agrawal and Navin Goyal. 2013. Thompson Sampling for Contextual Bandits with Linear Payoffs. In *Proceedings of the 30th International Conference on Machine Learning*. PMLR, 127–135. https://proceedings.mlr.press/v28/agrawal13.html

[2] Abdelhak Bentaleb, Mehmet N. Akcay, May Lim, Ali C. Begen, and Roger Zimmermann. 2022. Catching the Moment With LoL+ in Twitch-Like Low-Latency Live Streaming Platforms. *IEEE Transactions on Multimedia* 24 (2022), 2300–2314. https://doi.org/10.1109/TMM.2021.3079288

[3] Rodrigo Blázquez-García, Diego Cristallini, Martin Ummenhofer, Viktor Seidel, Jörg Heckenbach, and Daniel O'Hagan. 2023. Experimental Comparison of Starlink and OneWeb Signals for Passive Radar. In *2023 IEEE Radar Conference (RadarConf23)*. 1–6. https://doi.org/10.1109/RadarConf2351548.2023.10149580

[4] CTA. 2023. *CTA The Wave Project: Web Application Video Ecosystem Interoperability Project.* https://github.com/cta-wave/Test-Content

[5] DASH-Industry-Forum. 2023. *DASH-Industry-Forum/dash.js.* https://github.com/Dash-Industry-Forum/dash.js

[6] DASH-Industry-Forum. 2023. *DASH-Industry-Forum/livesim2.* https://github.com/Dash-Industry-Forum/livesim2

[7] The Pyodide development team. 2023. *Pyodide.* https://doi.org/10.5281/zenodo.8123342

[8] Jayasuryan V. Iyer, Khasim Shaheed Shaik Mahammad, Yashodhan Dandekar, Ramakrishna Akella, Chen Chen, Phillip E. Barber, and Peter J. Worters. 2022. *System and Method of Providing a Medium Access Control Scheduler.* https://patents.google.com/patent/US11540301B1/en

[9] Theo Karagkioules, Rufael Mekuria, Dirk Griffioen, and Arjen Wagenaar. 2020. Online learning for low-latency adaptive streaming. In *Proceedings of the 11th ACM Multimedia Systems Conference (MMSys '20)*. Association for Computing Machinery, New York, NY, USA, 315–320. https://doi.org/10.1145/3339825.3397042

[10] Debopam Bhattacherjee Kassem, Mohamed. 2023. LEOCONN Webinar Series. https://leoconnws.github.io/

[11] Jiajia Liu, Yongpeng Shi, Zubair Md. Fadlullah, and Nei Kato. 2018. Space-Air-Ground Integrated Network: A Survey. *IEEE Communications Surveys & Tutorials* 20, 4 (2018), 2714–2741. https://doi.org/10.1109/COMST.2018.2841996

[12] Owens Nathan. 2023. Unoffical Starlink Global Gateways & PoPs. Retrieved September, 2023 from https://www.google.com/maps/d/viewer?mid=1805q6rlePY4WZd8QMOaNe2BqAgFkYBY

[13] Piers O'Hanlon and Adil Aslam. 2023. Latency Target based Analysis of the DASH.js Player. In *Proceedings of the 14th Conference on ACM Multimedia Systems (MMSys '23)*. Association for Computing Machinery, New York, NY, USA, 153–160. https://doi.org/10.1145/3587819.3590971

[14] Jianping Pan, Jinwei Zhao, and Lin Cai. 2023. Measuring a Low-Earth-Orbit Satellite Network. arXiv:2307.06863 [cs.NI]

[15] Werner Robitza, Steve Göring, Alexander Raake, David Lindegren, Gunnar Heikkilä, Jörgen Gustafsson, Peter List, Bernhard Feiten, Ulf Wüstenhagen, Marie-Neige Garcia, Kazuhisa Yamagishi, and Simon Broom. 2018. HTTP Adaptive Streaming QoE Estimation with ITU-T Rec. P. 1203 - Open Databases and Software. In *Proceedings of the 9th ACM Multimedia Systems Conference (MMSys '18)*. Association for Computing Machinery, New York, NY, USA, 466–471. https://doi.org/10.1145/3204949.3208124

[16] Crist Ry and Paul Trey. 2023. *Starlink Internet Explained.* Retrieved June, 2023 from https://www.cnet.com/home/internet/starlink-satellite-internet-explained/

[17] SpaceX. 2023. Starlink. https://www.starlink.com

[18] Kevin Spiteri, Ramesh Sitaraman, and Daniel Sparacio. 2019. From Theory to Practice: Improving Bitrate Adaptation in the DASH Reference Player. *ACM Transactions on Multimedia Computing, Communications, and Applications* 15, 2s (July 2019), 67:1–67:29. https://doi.org/10.1145/3336497

[19] Kevin Spiteri, Rahul Urgaonkar, and Ramesh K. Sitaraman. 2020. BOLA: Near-Optimal Bitrate Adaptation for Online Videos. *IEEE/ACM Transactions on Networking* 28, 4 (Aug. 2020), 1698–1711. https://doi.org/10.1109/TNET.2020.2996964

[20] Emily Strong, Bernard Kleynhans, and Serdar Kadioglu. 2021. MABWISER: Parallelizable Contextual Multi-armed Bandits. *International Journal on Artificial Intelligence Tools* 30, 04 (June 2021), 2150021. https://doi.org/10.1142/S0218213021500214

[21] Hammas Bin Tanveer, Mike Puchol, Rachee Singh, Antonio Bianchi, and Rishab Nithyanand. 2023. Making Sense of Constellations: Methodologies for Understanding Starlink's Scheduling Algorithms. arXiv:2307.00402 [cs.NI]

[22] Yufei Wang, Lin Cai, and Jun Liu. 2023. High-Reliability, Low-Latency, and Load-Balancing Multipath Routing for LEO Satellite Networks. In *2023 Biennial Symposium on Communications (BSC)*. 107–111. https://doi.org/10.1109/BSC57238.2023.10201829

[23] 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, New York, NY, USA, 43–49. https://doi.org/10.1145/3592473.3592562