Low-Latency Live Video Streaming over Low-Earth-Orbit Satellite Networks

by

Jinwei Zhao
B.Eng., Zhejiang Gongshang University, China, 2017

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Computer Science

Low-Latency Live Video Streaming over Low-Earth-Orbit Satellite Networks

by

Jinwei Zhao
B.Eng., Zhejiang Gongshang University, China, 2017

Supervisory Committee

Dr. Jianping Pan, Supervisor
(Department of Computer Science)

Dr. George Tzanetakis, Departmental Member
(Department of Computer Science)

ABSTRACT

Video streaming currently dominates Internet traffic with an estimated share of 71% of all mobile data traffic, and it is forecast to increase to 80% in 2028. While 5G services are expanding beyond mobile devices to enterprises and Internet of Things (IoT) applications, the demand for service continuity and the surge in mobile data traffic is expected to further drive the evolution and expansion of network architectures into non-traditional areas. Space-Air-Ground Integrated Networks (SAGINs) are promising to transform future Internet connectivity by merging the capabilities of space, aerial, and terrestrial networks. Notably, Starlink's deployment of thousands of low-Earth-orbit (LEO) satellites provides global Internet coverage with significantly lower latency and higher bandwidth than conventional geostationary-equatorial-orbit (GEO)-based or medium-Earth-orbit (MEO)-based satellite networks, particularly enhancing connectivity in remote and rural areas. Yet, while the performance of video-on-demand (VoD) services over Starlink is on par with terrestrial networks, challenges remain in low-latency live video streaming, especially given the dynamic and fluctuating network latency in Starlink networks. In this thesis, we first conduct a comprehensive measurement study of Starlink's performance across different protocol layers, from access networks to live video streaming, at multiple geographical Starlink installations, including one where inter-satellite links (ISLs) are utilized in practice. Then, we present a novel adaptive video bitrate algorithm that utilizes the unique characteristics of Starlink networks to improve the quality of experience (QoE) of live video streaming. Finally, we discuss a joint decision-making solution with multipath QUIC protocol to address the multipath video streaming problem, which can further improve the video streaming experience.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

ACKNOWLEDGEMENTS

I would like to thank all the people who have helped, encouraged, and inspired me along the way. I am particularly grateful to my supervisor, **Dr. Jianping Pan**, for his guidance, support, and patience in all aspects of my graduate studies.

The valuable advice from other supervisory committee members, as well as the feedback from fellow graduate students, is much appreciated.

Lastly, I wish to thank my family and friends. Without them, I would never go this far.

DEDICATION

To my younger self, who decided to seek discomfort and go through this challenging life experience.

Chapter 1

Introduction

Video streaming currently dominates Internet traffic with an estimated share of 71% of all mobile data traffic, and it is forecast to increase to 80% in 2028 [16]. While 5G services were initially offered to consumers with mobile devices, there is also an increasing trend by network operators to offer 5G services to enterprises and massive Internet of Things (IoT) devices [54]. 5G subscriptions are forecast to reach 1.5 billion by the end of 2023 [16]. The demand for service continuity and the surge in mobile data traffic are expected to further drive the evolution and expansion of network architectures into non-traditional areas. 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 [36].

Starlink, a division of SpaceX [48], stands out as a pivotal player in offering global satellite 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, SpaceX'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 geostationary-equatorial-orbit (GEO) or medium-Earth-orbit (MEO) satellites, which are positioned at higher altitudes (35,786 km for GEO satellites and between 2,000 – 35,786 km for MEO satellites) with wider coverage but suffer from higher latency and limited

bandwidth performance. Typical round-trip-time (RTT) for GEO satellite Internet can exceed 800 milliseconds, which poses great challenges for a wide variety of Internet applications nowadays, while LEO satellite Internet can achieve RTT as low as 10–20 milliseconds. As of August 2023, SpaceX has deployed more than 5,000 LEO satellites and is currently offering Starlink Internet service to more than 2 million users in over 60 countries[1]. Nevertheless, SpaceX's LEO constellation ambition extends to launching up to 42,000 LEO satellites and eventually constructing multiple orbital shells [46].

The recent rapid growth and development of Starlink has attracted significant attention from both the industry and research community [29, 32, 47]. Different from the GEO satellites that remain at relatively fixed positions in the orbit, LEO satellites are in constant motion at approximately 7.8 km/s. Starlink UTs utilize phased array antennas to track the fast-moving satellites and perform frequent handovers between satellites in their line of sight to maintain network connectivity. Tanveer et al. [55] observed that Starlink employs a global controller for managing terminal-to-satellite scheduling. Notably, Starlink satellite handover events happen every 15 seconds, at the 12th, 27th, 42nd, and 57th (12-27-42-57) second past every minute, synchronized globally. Pan et al. [44] conducted an extensive measurement study on the Starlink satellite access network, gateway, point-of-presence (PoP) architectures and global backbone topology. It revealed that the RTT from UTs to GSs experiences significant fluctuations and is higher than that of conventional terrestrial Internet access via fiber optics, digital subscriber line (DSL), or cable modem.

For video streaming applications, users' QoE is significantly influenced by network latency, perceived video quality, and the number of bitrate fluctuations. Throughout the years, numerous adaptive video streaming methodologies and algorithms have been proposed to enhance the QoE. Dynamic Adaptive Streaming over HTTP (DASH) [52] has become the international standard for adaptive video streaming. The open-source reference player implementation dash.js [14] by the DASH Industry Forum is widely adopted by both the industry and research communities. Regarding the frequent satellite handover events and fluctuating network latency in Starlink networks, existing research indicate 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 bidirec-

[1] `https://en.wikipedia.org/wiki/Starlink`

tional video streaming such as interactive video conferencing, immersive AR/VR/XR applications, 360-degree and volumetric video streaming and low-latency live video streaming.

For VoD services over Starlink, the end-to-end performance is on par with conventional terrestrial networks [65]. 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, have distinct requirements for latency targets. Nowadays, typical terrestrial broadcasting (DVB-T) latencies range from 6–8 seconds. Given the widespread adoption of Starlink globally, analyzing the performance of low-latency live video streaming over LEO satellite networks and understanding the practical impacts are of great importance. O'Hanlon et al. [42] conducted a comprehensive analysis on 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. They employed trace-driven emulations with four different network profiles captured from real-world terrestrial networks. Nonetheless, it remains unknown how the fluctuating latency and frequent satellite handover events in Starlink networks affect the performance of low-latency live video streaming ABR algorithms in dash.js.

1.1 Main Contributions

We mainly focus on the following questions: How are the frequent satellite handover events and fluctuating network latency in Starlink networks affecting the performance of low-latency live video streaming systems? What are the possible solutions to improve the QoE of such systems?

In this thesis, we presented a study on the performance of the existing low-latency live video streaming techniques over Starlink networks. We first conducted comprehensive measurements of Starlink access networks across different protocol layers and multiple geographical Starlink installations, including one where ISLs are utilized in practice. We then proposed a novel low-latency live video streaming ABR algorithm

specifically designed for video streaming over Starlink satellite networks. It leverages the satellite handover patterns observed from the latency target-based measurements and analysis of low-latency live video streaming to dynamically adjust the video bitrate and playback speed, thereby ensuring a seamless viewing experience with low live latency, high average video bitrate, minimal rebuffering events and reduced visual quality fluctuation. Finally, we discussed a joint decision-making solution that can further improve the QoE of adaptive video streaming in multipath scenarios. This approach merges cross-layer metrics to create a context-aware online learning model, which can adaptively select the ideal network path and video bitrate simultaneously for multipath adaptive video streaming.

The main contributions of this thesis can be summarized as follows,

- Assessed the performance of Starlink access networks across different protocol layers and multiple geographical locations, taking into account scenarios both with and without ISLs.
- Conducted a latency target-based analysis of three state-of-the-art low-latency live video streaming ABR algorithms in dash.js over Starlink networks.
- Modelled the adaptive video streaming problem as an online learning process with contextual multi-armed bandit (CMAB) algorithms and proposed a novel ABR algorithm to improve the QoE of low-latency live video streaming over Starlink networks.
- Implemented an end-to-end prototype of the proposed algorithm with dash.js and evaluated its performance using both a purpose-built network emulation testbed and real Starlink networks.
- Discussed the potentials of multipath adaptive video streaming with multipath QUIC (MPQUIC) using a QoE-driven joint decision-making framework.

1.2 Thesis Overview

Chapter 1 contains a brief introduction to adaptive video streaming and Starlink networks, and contributions in this thesis followed by the thesis overview.
Chapter 2 presents a detailed background and related works for adaptive video streaming and LEO satellite networks.

Chapter 3 demonstrates the testbed setup and the measurement of Starlink access networks across different protocol layers.

Chapter 4 focuses on low-latency live video streaming over Starlink networks, and introduces the proposed algorithm with implementation details and performance evaluations.

Chapter 5 discusses the potentials of multipath adaptive video streaming with MPQUIC using a QoE-driven joint decision-making framework.

Chapter 6 concludes this thesis, presents open research challenges and discusses future work.

Appendix A includes the publication list of research work related to this thesis.

Chapter 2

Background and Related Works

In this chapter, we go through the history and evolution of online video streaming systems, with a particular focus on DASH, the international standard widely adopted by the industry and research communities. Within this context, we provide a brief overview of previous works on adaptive video streaming and cover ABR algorithms for both VoD and low-latency live video streaming scenarios. We also review the existing deployments and research works on LEO satellite constellations. While Starlink has attracted considerable research interest in different aspects, in this thesis, we mainly focus on the measurement of Starlink access networks and the practical performance evaluation of multimedia services over Starlink.

2.1 Adaptive Video Streaming

Online video streaming has become one of the most popular Internet applications in recent years. In the early 2000s, large-scale peer-to-peer video streaming platforms such as Coolstreaming were deployed on the Internet, attracting millions of users. In 2005, YouTube was created and provided an easy-to-use platform for users to upload and share videos. In 2010, Netflix, the previous DVD rental service provider, was transformed into an online video streaming platform and migrated its infrastructure to the Amazon Web Service (AWS) cloud computing platform.

Nowadays, the fundamental infrastructure of online video streaming is largely dependent on the Content Distribution Networks (CDNs). Service providers such as YouTube and Netflix deploy large-scale distributed systems consisting of hundreds of thousands of servers around the world at different data centers to deliver video content

to end users. Instead of passive proxy caching, CDN architectures are designed to proactively replicate content to edge servers in different data centers close to end users to reduce the last-mile content delivery latency to improve the video streaming QoE.

One of the main challenges in online video streaming is to provide high-quality videos to end users over heterogeneous network environments while maintaining high QoE. Different video streaming devices have distinct computing capabilities, such as different hardware support for video codecs. Each of them may have different and dynamic network conditions. While users are streaming videos, the network conditions may change dramatically due to the mobility of users or network congestion. Service providers of online video streaming systems develop bitrate adaptation algorithms to dynamically adjust the video resolution and bitrate to avoid video playback rebuffering events and maximize the perceptual video quality.

Over the last two decades, the online video streaming industry has been evolving rapidly. Real-Time Messaging Protocol (RTMP) was developed by Macromedia to support online audio and video streaming with Flash Player. With the development of open web standards such as HTML 5, HTTP-based video streaming protocols such as HTTP Live Streaming (HLS) and Dynamic Adaptive Streaming over HTTP (DASH) have been widely deployed by major video streaming platforms to replace RTMP and Flash. DASH, developed by the MPEG group, became an international standard in November 2011, which was officially published as ISO/IEC 23009-1:2012 in April 2012 and was further revised in 2022 as ISO/IEC 23009-1:2022 [24]. The DASH reference player implementation dash.js [14], an open-source JavaScript library, is capable of both VoD and live video streaming. It is widely adopted by both the industry and research communities.

In this thesis, we mainly focus on DASH and its ABR algorithms in both VoD and live video streaming scenarios.

2.1.1 Dynamic Adaptive Streaming over HTTP (DASH)

The core idea of DASH is to divide the entire video into time-aligned small video segments at variable or fixed durations, typically ranging from 2 to 10 seconds. These segments are encoded into different video resolutions and bitrates with possibly different video and audio codecs. Metadata information describing the video segments, such as the segment duration, bitrate, frame rate, resolution, codec and the URLs

to the video segments are then generated in the corresponding Media Presentation Description (MPD) files. MPD files and video segments are stored on media servers and replicated to edge servers in CDN networks.

When a DASH client initiates a video streaming session, it first retrieves the MPD file and starts downloading video segments from the closest CDN edge server. The client requests video segments at specific video bitrate levels and resolutions based on the current network conditions, device capabilities, and user preferences. As the streaming session progresses, the client continuously monitors these factors and adjusts the bitrate level of the video segments to be downloaded. By dynamically switching between different bitrate levels, DASH ensures that the client can maintain the best possible video quality under dynamic network conditions while minimizing the risk of rebuffering events, ensuring a seamless playback experience.

DASH is codec agnostic, which means that it can support a wide range of video and audio codecs, such as H.264, H.265, VP9, and AV1. Service providers can take advantage of the most efficient codecs to reduce bandwidth consumption. New codec features including scalable video coding and content-aware encoding can further improve the efficiency of online video delivery systems.

2.1.2 Video on Demand Streaming

In this thesis, we classify existing adaptive bitrate (ABR) algorithms into two categories based on a widely accepted criterion [31]: (1) client-side techniques including throughput-based, buffer-based and hybrid algorithms; and (2) server-side or SDN-assisted techniques.

Traditional client-side ABR algorithms rely on network measurements including available bandwidth estimation or application contexts such as local playback buffer ratio. Spiteri et al. [50] proposed the buffer-based *BOLA* algorithm, which formulates bitrate adaptation as a utility-maximization problem. Lyapunov optimization is utilized to minimize rebuffering and maximize video quality. *BOLA* and its variants have been integrated into the DASH reference player dash.js and are widely used in production [49]. However, bitrate adaptation algorithms based on fixed sets of empirical rules often suffer from low performance in severely dynamic and unpredictable network environments. To address this issue, Cui et al. [12] used deep reinforcement learning to achieve the joint optimization of video bitrate, rebuffering time and latency. However, most deep reinforcement learning-based ABR algorithms are com-

putationally intensive and require powerful hardware such as GPUs. As a result, they are typically deployed on the server side. Yeo et al. [64] proposed *NEMO*, which uses server-assisted techniques to enable real-time video super-resolution on mobile devices to improve the QoE of adaptive video streaming. On the other hand, SDN-assisted ABR algorithms are promising in theory but usually lack real-world end-to-end support. Farahani et al. [18] proposed *ES-HAS* to improve adaptive video streaming performance over SDN-enabled networks. Virtualized edge components and an SDN controller are deployed to collect streaming clients' requests and retrieve underlying network information to determine the optimal cache server for clients. Nevertheless, provided that end-to-end SDN infrastructure cannot be guaranteed between the video streaming client and the server, the advantages of SDN-assisted algorithms vanish and the video streaming experience falls back to whatever the ABR algorithm can provide.

2.1.3 Low-Latency Live Video Streaming

Because of the necessity to meet specific latency targets, it is more challenging to ensure the QoE of live video streaming than VoD services. One of the key techniques enabling low-latency live video streaming over HTTP is the Common Media Application Format (CMAF) [23]. CMAF is a unified format for encoding core media segments and allows the use of different manifest formats and video streaming protocols, such as DASH and HLS. By using CMAF chunked encoding and chunked transfer with low-latency extensions of DASH and HLS, existing HTTP Adaptive Streaming (HAS) systems can achieve low-latency live video streaming towards one second of latency [6].

Over the past decade, numerous ABR algorithms have been proposed to improve the QoE of low-latency live video streaming in DASH. In this section, our concentration is on three low-latency live video streaming ABR algorithms available in the dash.js reference implementation, namely *Dynamic*, *L2A-LL*, and *LoL+*. The default *Dynamic* [49] algorithm in dash.js is a hybrid ABR algorithm consisting of throughput-based rule and buffer-based *BOLA* algorithm [50], similar to the techniques for VoD services in Section 2.1.2. Karagkioules et al. [28] 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. [7] introduced *LoL+*, a learning-based ABR algorithm that employs a self-organizing map (SOM) to adapt the bitrate level 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. [42] 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 low-latency live video streaming performance. The *Dynamic* algorithm performs the best in terms of low rebuffering duration, with the least number of rebuffering events 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 low-latency live video streaming.

Table 2.1: Satellite constellations providing Internet services (by October 2023) [59]

Index	Organization	Launched / Planned	First Launch
1	SpaceX (Starlink Gen 1)	4162 / 4408	2018
2	OneWeb	634 [43] / 648	2019
3	Iridium (NEXT)	75 / 75	2017
4	Globalstar (Gen 2)	25 / 48	2010
5	SES (O3b / mPOWER)	24 / 70	2013
6	Kepler (Gen 1)	19 / 140	2018

2.2 Low-Earth-Orbit Satellite Networks

This section offers a concise overview of the currently operational LEO satellite constellations that deliver Internet services. Subsequently, we delve into existing research on the measurement and analysis of Starlink access networks. Finally, we cover the practical performance evaluation of multimedia services over Starlink networks to conclude this section.

2.2.1 Low-Earth-Orbit Satellite Constellations

The concept of delivering global Internet coverage using LEO satellites has been around for decades. Compared to GEO and MEO alternatives, LEO satellite Internet can achieve lower latency comparable to conventional terrestrial Internet and higher bandwidth using smaller and cheaper satellites. The first LEO satellite Internet service provider, Teledesic, founded in the 1990s, aiming to build a commercial broadband satellite constellation using 288 active satellites at 1400 km[1]. The demonstration satellite for the Teledesic constellation, *Teledesic 1*, was launched on February 26, 1998. It was the first Ka-band satellite in orbit owned by a commercial enterprise. The satellite decayed from orbit on October 9, 2000. While Teledesic and other similar companies commercially failed in achieving the goal of delivering Internet service globally using LEO satellites, the idea has been revived in recent years, with the emergence of new technologies such as reusable rockets by SpaceX and mass-produced low-cost satellites. Wang et al. [59] provides an overview of satellite constellations providing Internet services, as shown in Table 2.1.

Among all the existing LEO satellite constellations, Starlink, a SpaceX division,

[1] `https://en.wikipedia.org/wiki/Teledesic`

stands out as the most successful one, with the largest number of LEO satellites in operation and more than 2 million subscribers worldwide as of October 2023. Since the launch of Starlink's beta testing in 2020, it has attracted considerable interest from both the industry and research communities. However, there are very few public details about the network architecture of Starlink, besides the limited information from SpaceX's FCC filings. Research topics span from access network measurements [44, 65, 55] to physical layer signal structure analysis [9] and routing protocol design [60], among other reverse engineering efforts of the Starlink networks.

Several other upcoming LEO satellite constellations are expected to be operational in the future. Amazon launched "Project Kuiper", which is an initiative to provide global broadband access through a constellation of 3,236 LEO satellites in orbit. As of October 2023, there are two prototype satellites operational in orbit, namely *KuiperSat-1* and *KuiperSat-2*. They have completed multiple successful demonstrations of ISLs, maintaining 100 Gbps links over a distance of nearly 1,000 km during the entire test window [4]. Amazon plans to begin satellite production before the end of 2023 and start beta testing in the second half of 2024 [3]. Telesat, a traditional GEO satellite Internet service provider, is also planning to launch a LEO satellite constellation with 198 LEO satellites in orbit. Telesat launched the first demonstration satellite *LEO Vantage 1* in 2018, which was later decommissioned and replaced by *LEO Vantage 3* in 2023, while the second prototype satellite *LEO Vantage 2* failed to reach orbit in 2017. Eutelsat OneWeb [17] aims to build a LEO satellite constellation featuring more than 630 satellites along 12 synchronized orbital planes in low Earth orbits at 1,200 km.

2.2.2 "Bent-Pipe" vs Inter-Satellite Links (ISLs)

Most of the existing LEO satellite constellations utilize the "bent-pipe" architecture, as shown in Figure 2.1. In this architecture, UTs utilize phased array antennas to track the moving satellites in line of sight and establish connections with one of the satellites. Network packets from users are transmitted to the connected satellite, which is responsible for forwarding packets to the nearest GS via satellite-to-ground links. The GSs are connected to Starlink's regional PoP via terrestrial links, which are then interconnected with the Internet backbone at different Internet exchange points (IXPs).

As of October 2023, there are several generations of Starlink satellites in operation.

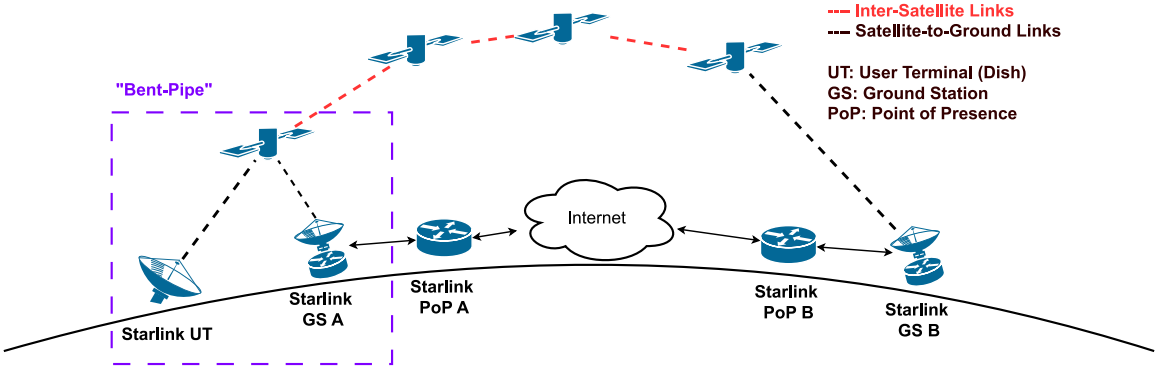


Figure 2.1: "Bent-Pipe" vs Inter-Satellite Links (ISLs)

The Starlink satellites are only ISLs-capable after Generation v1.5. When customers place orders on Starlink's website, they are required to provide their shipping address and Starlink will check whether the address is serviceable via ISLs. Having more than 5,000 satellites in operation and many GSs, the current usage of Starlink ISLs is limited. Most of the service regions still rely on the "bent-pipe" architecture. However, there are still some known locations where ISLs are being utilized and confirmed by SpaceX. The first one is Unalaska, located in the Aleutian Islands, Alaska, USA. It is also the first Starlink community gateway. ISLs are also utilized in certain African countries, including Kenya, Rwanda and Mozambique, where a mature terrestrial Internet backbone does not exist with limited GSs and PoPs across the continent. They are also being utilized for some island countries in the middle of the Pacific Ocean, and Starlink Maritime services, where boats or cruise ships are equipped with Starlink UTs but have no access to nearby GSs in the ocean.

However, due to the limited number of ISLs-capable satellites in operation, the limited regions where ISLs are utilized in practice and the lack of public information disclosure about the routing algorithms used in the ISLs deployment, there are very few research works on the practical performance evaluation related with ISLs. Most of the existing research works on Starlink access networks are based on the "bent-pipe" architecture, and there are few works on the multipath ISL routing protocol design that use simulation-based evaluation [51].

2.2.3 Starlink Access Networks

A simplified topology of Starlink access networks is shown in Figure 2.2. Clients connect to the Starlink user router via either Ethernet or wireless connections. Starlink's

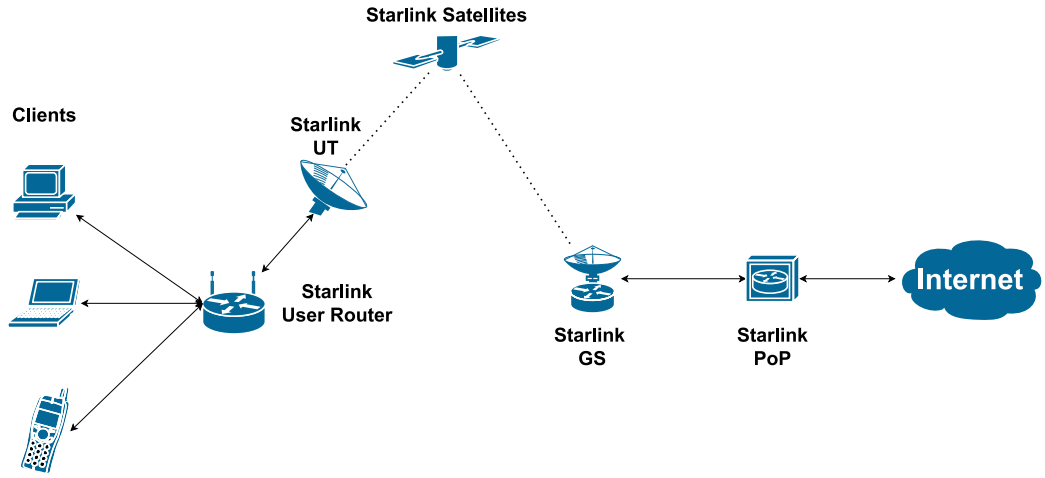


Figure 2.2: Starlink access networks

UT has a built-in gRPC interface, which is used by the Starlink mobile application and web portal[2] to configure the UT and retrieve various performance metrics. Several open-source projects such as `starlink-grpc-tools`[3] utilize the gRPC interface to retrieve performance metrics, including the satellite obstruction map, network latency, packet loss and outage events, etc.

Among the existing measurement studies, Pan et al. [44] provided comprehensive insights into the Starlink access network, gateway, PoP, and global backbone network, illustrating the findings with detailed network topology diagrams, derived from collaborative measurements across the world. Their findings indicate that, from a user's perspective, the GS is always accessible at a carrier-grade 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. Each Starlink subscriber is associated with a home PoP, where the user AAA (Authentication, Authorization, and Accounting) and IPv4 network address translation (NAT) happen. Only subscribers with a static IPv4 address can be reached by external public Internet. Starlink also supports IPv6 natively with end-to-end IPv6 connectivity. Each dish is delegated a `/56` IPv6 prefix for network clients.

Tanveer et al. [55] 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

[2] `https://dishy.starlink.com`
[3] `https://github.com/sparky8512/starlink-grpc-tools`

suggests that SpaceX utilizes a globally time-synchronized scheduler to manage the access network between the UTs and satellites. By analyzing the satellite obstruction maps retrieved from the Starlink UT's gRPC interface, utilizing the Two-Line Element (TLE) set dataset and the SGP4 algorithm, they developed a model to predict the characteristics of the satellite allocated to serve a UT in a given location and time.

2.2.4 Multimedia Services over Starlink

In an early study by Michel et al. [39], besides latency and throughput measurements, they quantified the web browsing performance over Starlink, and the results show Starlink outperforms traditional satellite Internet service providers using GEO satellites and has similar performance to regular terrestrial Internet access. Based on their measurements, Starlink is 75–80% faster than traditional satellite Internet in terms of QoE-related metrics including web page *onLoad* time and *SpeedIndex*. Ma et al. [38] conducted end-to-end latency and throughput measurements using four UTs deployed across British Columbia, Canada, including urban and remote areas. They conducted end-to-end latency tests utilizing 9 destination servers deployed worldwide on the AWS platform, including São Paulo, Singapore, Sydney, North California, Bahrain, Tokyo, London, Cape Town, and Mumbai. They mainly investigated Starlink's "bent-pipe" architecture and the routing strategy used by Starlink. Zhao et al. [65] conducted a systematic measurement on real-time multimedia services over Starlink, including VoD (YouTube), live video 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 low-latency live video streaming, face more pronounced performance challenges. Other research efforts such as Garcia et al. [21] focused on network layer measurements utilizing precise timestamps to analyze throughput variations at multiple timescales and infer system timing details of the Starlink networks.

Chapter 3

Measuring a Low-Earth-Orbit Satellite Network

In this chapter, we present our measurements of Starlink networks across different protocol layers and multiple geographical locations. We first present the testbed setup for our measurements, followed by the measurement results and analysis of Starlink access networks. We then outline the approach behind the latency target-based analysis of low-latency live video streaming over Starlink networks. The subsequent measurement results, in conjunction with the performance evaluation of a novel low-latency ABR algorithm, are presented in Section 4.3.

3.1 Testbed Setup

The architecture of our Starlink measurement testbed is shown in Figure 3.1. We utilized two Starlink UTs, one located in the University of Victoria, Canada (referred to as Pacific Northwest, without ISLs being utilized in this region), and the other one located in a remote island country in the western Indian Ocean (with ISLs being utilized in this region). For each Starlink installation, we deployed a virtual machine in the nearest Google Cloud Platform (GCP) availability zone. The virtual machine is used as the target server in the end-to-end latency and throughput measurements and as the media server that hosts video assets for the low-latency live video streaming measurements.

For the Starlink installation in the Pacific Northwest, the nearest Starlink GSs are located in the State of Washington, USA and the associated home PoP is the Starlink

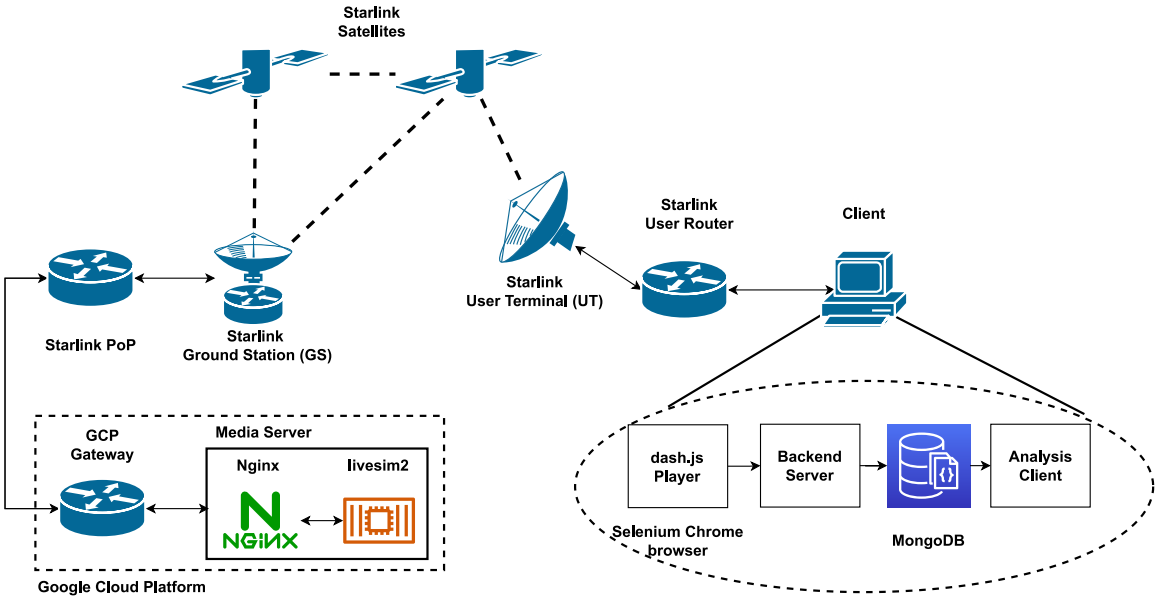


Figure 3.1: Starlink DASH testbed architecture

Seattle PoP. To minimize additional terrestrial network latency in the end-to-end latency and throughput measurements, we deployed the virtual machine 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, which is negligible compared to the latency of Starlink access networks. The Starlink installation in the western Indian Ocean is approximately located near the eastern coast of Africa. As of July 2023, the only Starlink PoP in Africa is in Lagos, Nigeria [40], located on the western coast of the continent. Considering the absence of Starlink GSs within a 5,000 km radius of this Starlink installation, our inference is that the network packets from this UT to the Internet are relayed and transmitted through multiple inter-satellite links, as illustrated in Figure 2.1. To support this inference, we utilized `mtr` and `traceroute` to trace the packet from this Starlink UT to the Internet. We also used `nslookup` to perform reverse DNS resolution on the public IP address associated with this Starlink installation. The result (`customer.lgosnga1.pop.starlinkisp.net`) revealed that packets are routed via the Starlink Lagos PoP in Nigeria for this Starlink installation. When ISLs are utilized, packets travel through multiple satellites before being transmitted to the GS. Because of the absence of GCP data centers in Africa, we deployed the virtual machine for this region in GCP's *europe-southwest1-a* availability zone, physically located in Madrid, Spain, which is closely interconnected with

Table 3.1: Starlink access network latency (milliseconds) to GS gateway

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

Starlink Madrid PoP. Starlink Madrid PoP is interconnected with Starlink Lagos PoP through Starlink's terrestrial backbone infrastructure [44].

For each Starlink installation, a mini PC is directly connected to the Starlink user router via an Ethernet cable. Network measurement scripts and a low-latency live video streaming client are deployed on the mini PC. The low-latency live video streaming client 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. The playback metrics are then stored in the MongoDB database. The analysis client queries the MongoDB database after each playback session ends and generates corresponding figures. On each media server, we deployed `livesim2` [15], a DASH live source simulator implemented in Go by the DASH Industry Forum, which takes segmented DASH VoD source videos as input assets, and produces a wall-clock (UTC) synchronized linear stream of video segments. By looping through the input DASH VoD 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 Networks

Our first measurement is to evaluate the latency performance of Starlink access networks. We started with measuring the RTT between clients and the GS gateway (`100.64.0.1`) in the associated home PoP using `ping`. The interval for the `ping` command is set to 10 milliseconds, with a total continuous duration of 60 minutes. A summary of the Starlink access network latency to the GS gateway is shown in Table 3.1.

For the Starlink installation without ISLs, the overall access latency can easily be maintained at around 50 milliseconds, with the minimal RTT being 16.7 milliseconds,

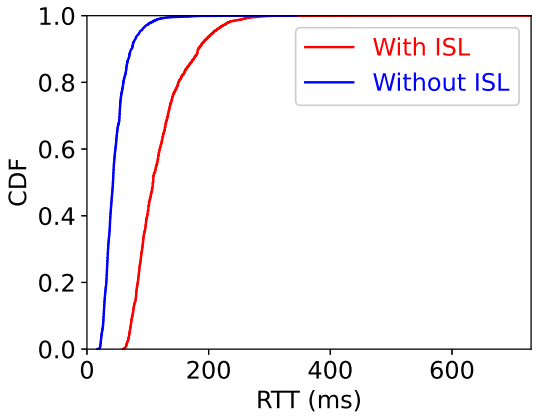
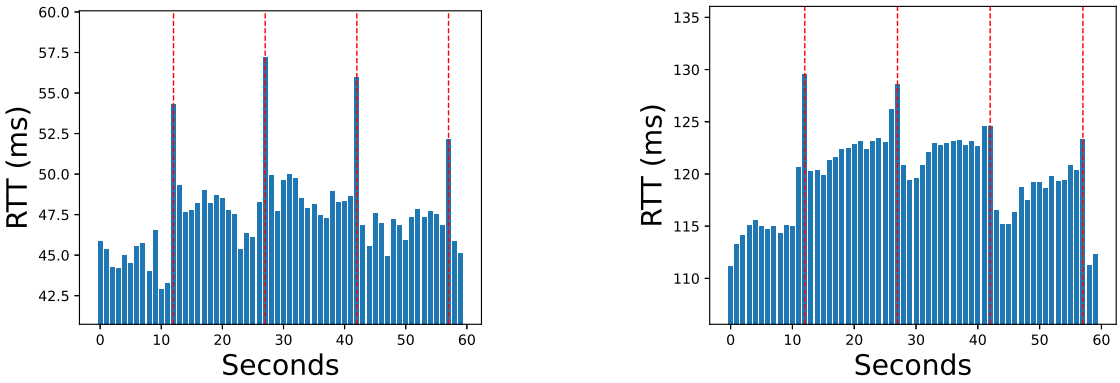


Figure 3.2: CDF of the RTT to GS gateway



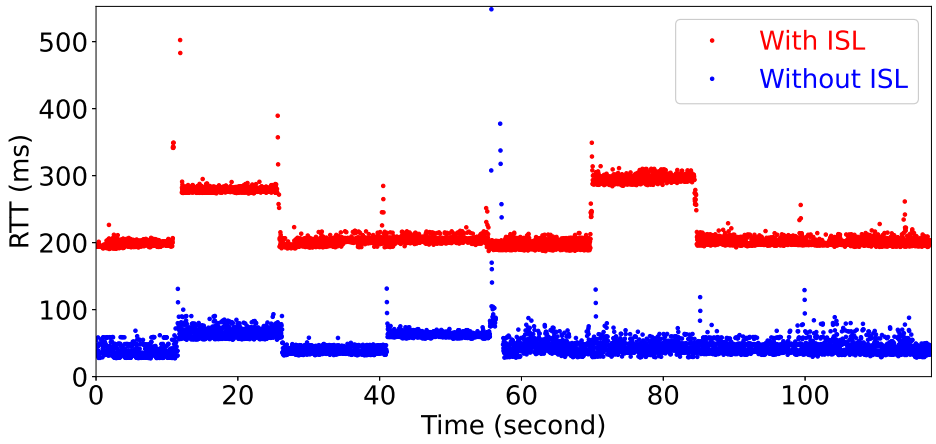
(a) Average RTT to GS gateway without ISLs

(b) Average RTT to GS gateway with ISLs

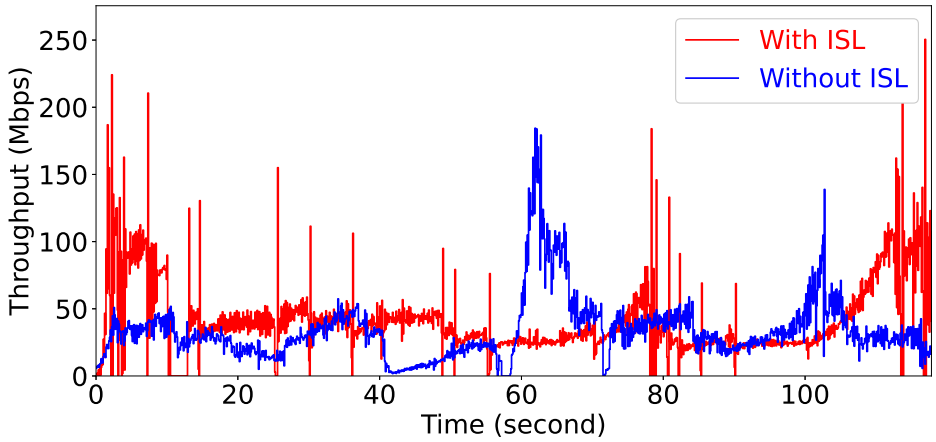
Figure 3.3: Average RTT to GS gateway at every second

the median RTT being 42.8 milliseconds and the average RTT being 47.5 milliseconds. For the Starlink installation with ISLs, the overall access latency fluctuated more than the one without ISLs, 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 ISLs. The cumulative distribution function (CDF) of both access latency distributions is shown in Figure 3.2.

We calculated the average RTT to the GS gateway at every second for both Starlink installations, as shown in Figure 3.3. It revealed an obvious pattern that at (12-27-42-57) seconds of every minute, the average RTT between Starlink UTs



(a) E2E Latency

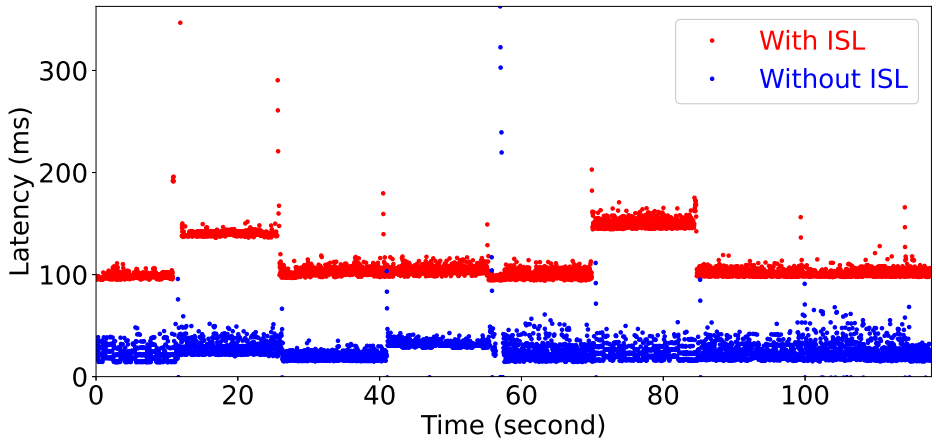


(b) Downlink Throughput

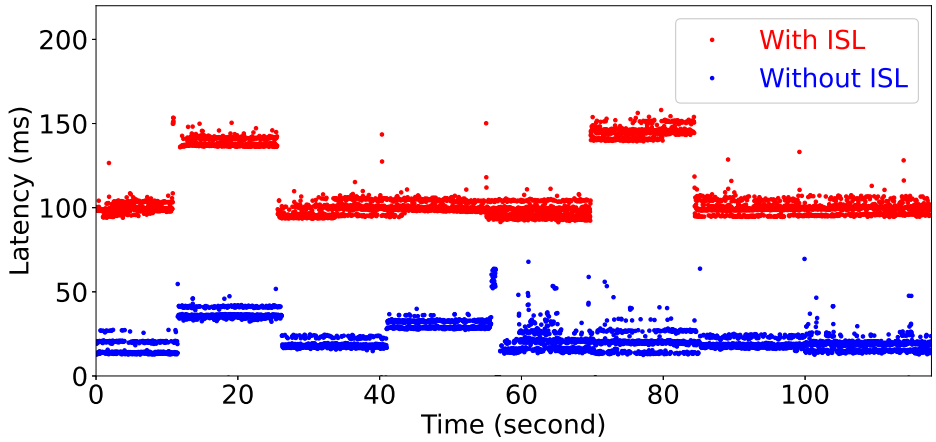
Figure 3.4: Time synchronized RTT and downlink throughput

and their corresponding GS gateways spikes, which indicates that satellite handover events happen at synchronized seconds globally at different geographical locations. This observation is consistent with the findings in [55]. However, our observations in Figure 3.3 indicate that the potential impact of satellite handover events is more pronounced on the Starlink installation that did not utilize ISLs during our measurement, while the technical details of ISLs remain unknown and require further investigation.

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 3.4 and Figure 3.5.



(a) Uplink OWD



(b) Downlink OWD

Figure 3.5: Time synchronized One-Way Delay (OWD)

E2E latency and throughput are measured with `IRTT`[1] and `iPerf3`[2] respectively. `IRTT` (Isochronous Round-Trip Tester) is a tool to measure round-trip-time (RTT), one-way delay (OWD) and other metrics using UDP packets sent on a fixed period. `iPerf3` is a tool for active measurements of the maximum achievable bandwidth on IP networks using TCP, UDP or SCTP protocols. Both `IRTT` and `iPerf3` use the client-server model, requiring a server daemon program running on the target server and a client program running on the client machine. We deployed `IRTT` and `iPerf3` daemon programs on the media servers as illustrated in Figure 3.1. All

[1] `https://github.com/heistp/irtt`
[2] `https://iperf.fr/`

the media servers and clients are configured with NTP time synchronization using `chrony`[3] and Google Public NTP service[4], such that `IRTT` can provide accurate OWD measurements. `IRTT` provides latency measurements by sending UDP packets on a fixed time interval regardless of whether replies are received and is not affected by the Linux kernel scheduling or other factors that can affect the packet interval of ICMP `ping` measurements. 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 to 10 milliseconds, consistent with the ICMP `ping` experiments. With the short interval at 10 milliseconds, we can obtain a more accurate RTT distribution to reveal the latency pattern and traffic engineering at a finer granularity. To measure the throughput of Starlink access networks, we utilized `iPerf3` and set the report interval to 100 milliseconds, which is the minimal `iPerf3` report interval. In this thesis, 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.

Figure 3.4 and Figure 3.5 show 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 the Starlink installation without ISLs in Figure 3.5(b). 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 [27]. Figure 3.4(b) 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 downlink

[3] `https://chrony-project.org/`

[4] `https://developers.google.com/time`

Table 3.2: Bitrate ladder of the low-latency live 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

TCP throughput performance. It is important to note that the utilization of ISLs does not directly impact downlink throughput. Instead, it is influenced by Starlink's capacity limitations and QoS policies in specific regions. There are more subscribers in the Pacific Northwest region than in the western Indian Ocean region, which possibly results in a higher contention ratio and lower downlink throughput in the throughput measurements we conducted.

3.3 Latency Target-based Analysis of Live Video Streaming

Besides the measurement and analysis of Starlink access networks in Section 3.2, we also conducted a latency target-based analysis of low-latency live video streaming over Starlink networks. The motivation behind this analysis is to provide a better understanding of the real-world impact of Starlink's frequent satellite handover events and fluctuating network latency on low-latency live video streaming applications. By examining these impacts in conjunction with the state-of-the-art low-latency ABR algorithms, this analysis seeks to offer insights and guidance for future algorithm design to improve the QoE of low-latency live video streaming over satellite networks.

Our measurement of low-latency live 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 [5]. The bitrate ladder

is shown in Table 3.2, which contains 8 video 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 time-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 10 times for each latency target and ABR algorithm. The following metrics are collected every 100 milliseconds on our modified dash.js player during each playback session,

- Average live latency
- Average playback bitrate
- Rebuffering time ratio (%)
- Number of bitrate switches
- Bitrate standard deviation

and they are periodically (every 5 seconds) sent to the backend server and then stored in the MongoDB database for the analysis client to evaluate the performance of each ABR algorithm with different latency targets.

The measurement results for this section, along with a detailed performance evaluation and comparison with the proposed algorithm in this thesis are shown in Figures 4.4 to 4.8, presented in Section 4.3.

3.4 Open Dataset

To benefit the research community, we have publicly released our measurement dataset on Starlink access networks on Zenodo[5].

This dataset contains the end-to-end measurements of Starlink access network latency and downlink throughput as described in Section 3.2. In this dataset, we included the results obtained from the Starlink installation in the Pacific Northwest without ISLs being utilized. The Starlink UT is physically located in the University of Victoria, BC, Canada, whereas the associated Starlink PoP is located in Seattle, WA, USA.

The duration of data collection spans from 2023-09-13 to 2023-09-17. The data was collected using the `irtt-iperf3.py` script. The script is scheduled to run every

[5] `https://doi.org/10.5281/zenodo.10020034`

10 minutes throughout the day. Each session lasts 2 minutes. The script starts concurrent Python threads for `IRTT` and `iPerf3` clients to measure time-synchronized latency and downlink throughput. To reduce potential network fluctuation introduced by wireless connections, the script was run on a mini PC directly connected to the Starlink user router via an Ethernet connection. Both the client and the virtual machine are configured with the Google Public NTP service. TCP and Cubic congestion control algorithm was used for the `iPerf3` throughput measurements. The number of TCP streams in `iPerf3` throughput measurement is 1.

The time interval between `IRTT` UDP packets is set to 10 milliseconds. The `iPerf3` report interval is set to 100 milliseconds, which is the smallest value that can be set. Both `IRTT` and `iPerf3` clients write measurement metrics to JSON files after each session ends. A demonstration script `plot.py` is provided to parse the JSON files and generate example figures.

Occasionally, `IRTT` and `iPerf3` clients exit abnormally due to connection outages, thus leaving corrupted or incomplete JSON files. Those files are removed from the dataset. The IP address and domain names of the virtual machine and the client are labeled as redacted in post-processing.

Chapter 4

Low-Latency Live Video Streaming over Starlink

In this chapter, we propose a novel ABR algorithm for low-latency live video streaming over Starlink networks. The motivation for this approach is based on the unique network characteristics of Starlink access networks, measured across different protocol layers and multiple geographical Starlink installations in Chapter 3. Contextual multi-armed bandit (CMAB) algorithms are employed to model the low-latency live video streaming problem as an online learning process. An end-to-end prototype is implemented in dash.js, and performance evaluation is conducted with both network emulation and real Starlink networks.

4.1 System Model and Problem Formulation

In this section, we first present the problem formulation of low-latency live video streaming using CMAB algorithms. Then, a novel QoE-driven reward function and catch-up policy are proposed to improve the QoE of low-latency live video streaming over Starlink networks. The key notations used in this chapter are summarized in Table 4.1.

4.1.1 Contextual Multi-Armed Bandit

Contextual multi-armed bandit (CMAB) algorithms are a family of lightweight and efficient online learning approaches for solving the exploration-exploitation dilemma in online decision-making problems. CMAB requires less computational resources

Table 4.1: 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	First 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
$\mathrm{B}_{\max}$	Highest video bitrate available in the MPD file
$\mathrm{R}(t)$	Playback speed at round t
$\mathrm{X}(t)$	Estimated network throughput at round t
$\mathrm{L}_{\mathrm{N}}(t)$	Measured network latency at round t
$\mathrm{Buffer}_{\min}$	Minimal playback buffer threshold
$\mathrm{Buffer}_{\mathrm{current}}$	Current playback buffer level
$\mathrm{LL}_{\mathrm{target}}$	Latency target for the playback session
$\mathrm{LL}_{\mathrm{current}}(i)$	Playback latency when downloading segment i
$\mathrm{QoE}_{\mathrm{P1203}}(i)$	ITU-T P.1203 QoE score for segment i
$\mathrm{QoE}(i)$	Final QoE for segment i

than reinforcement learning-based algorithms but also provides a competitive performance guarantee in terms of low regret bound. The problem setting of general CMAB algorithms can be outlined below as described in [1]. In an online sequential decision-making problem, an agent is presented with K actions, referred to as K arms. In each of the T rounds, the agent must pull an arm k and observe the corresponding reward r_k. The rewards of the unselected arms remain unknown to the agent. Before pulling an arm k, the agent observes a d-dimensional feature vector b_k, associated with each arm k. In CMAB algorithms, the feature vector b_k is referred to as the *context*. The agent employs the context vectors b_k in conjunction with the rewards from previous rounds to determine which arm should be selected in the current round. Over time, the agent aims to gather enough information about

how the context vectors and rewards relate to each other so that it can predict, with some certainty, which arm is expected to yield the highest reward by observing the context vector in the current round. The agent competes with a class of predictors, in which each predictor takes in the context vectors and predicts which arm will give the best reward. If the agent can guarantee to do nearly as well as the predictions of the best predictor in hindsight (i.e., with the lowest regret bound), then the agent is considered to successfully compete with that class.

In this chapter, we model the low-latency live video streaming scenario with CMAB algorithms as follows. The video bitrate adaptation problem in low-latency live video streaming can be formulated as an online learning process. The video player, referred to as an agent, is presented with a bitrate ladder of K different bitrate levels to choose from in each of T rounds. We assume T is finite but unknown to the agent. 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. In this section, we define the context vector $b(t)$ as follows,

$$b(t) = [\mathrm{L_N}(t), \mathrm{R}(t), \mathrm{X}(t)] \tag{4.1}$$

where at round t, $\mathrm{L_N}(t)$ is the latest 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. From Figures 3.4 and 3.5, we can observe the latency and throughput pattern varies significantly in each 15-second timeslot over Starlink networks. In this section, we only consider the history $\mathcal{H}_q^{t-1}$ of the current 15-second timeslot beginning from round q,

$$\mathcal{H}_q^{t-1} = \{k, r_k(s), b(s), s = q, \ldots, t-1\} \tag{4.2}$$

where k denotes the arm played at round s and $r_k(s)$ is the reward for arm k at round s, $b(s)$ is the context vector for round s, and q is the first round of 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,

$$\mathbb{E}\left[r_k(t) \mid b(t), \mathcal{H}_q^{t-1}\right] = b(t)^T\mu_k. \tag{4.3}$$

In a CMAB scenario, an agent employing an online learning algorithm must decide which arm k 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 round 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 round t, i.e.,

$$\Delta_k(t) = b(t)^T\mu_{a^*(t)} - b(t)^T\mu_k \tag{4.4}$$

Then, the regret at round t is defined as

$$\text{regret}(t) = \Delta_k(t) \tag{4.5}$$

It is worth noting that the CMAB problem setting presented in this thesis 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. This means that before the agent selects the next video bitrate, each arm k (representing a video bitrate level) is revealed with the same network measurements and playback metrics. We also only consider the history $\mathcal{H}_q^{t-1}$ of the current 15-second timeslot beginning from round q instead of the entire history $\mathcal{H}^{t-1}$ of the current playback session, because of the unique and fluctuating latency and throughput pattern in each timeslot as shown in Figure 3.4 and Figure 3.5.

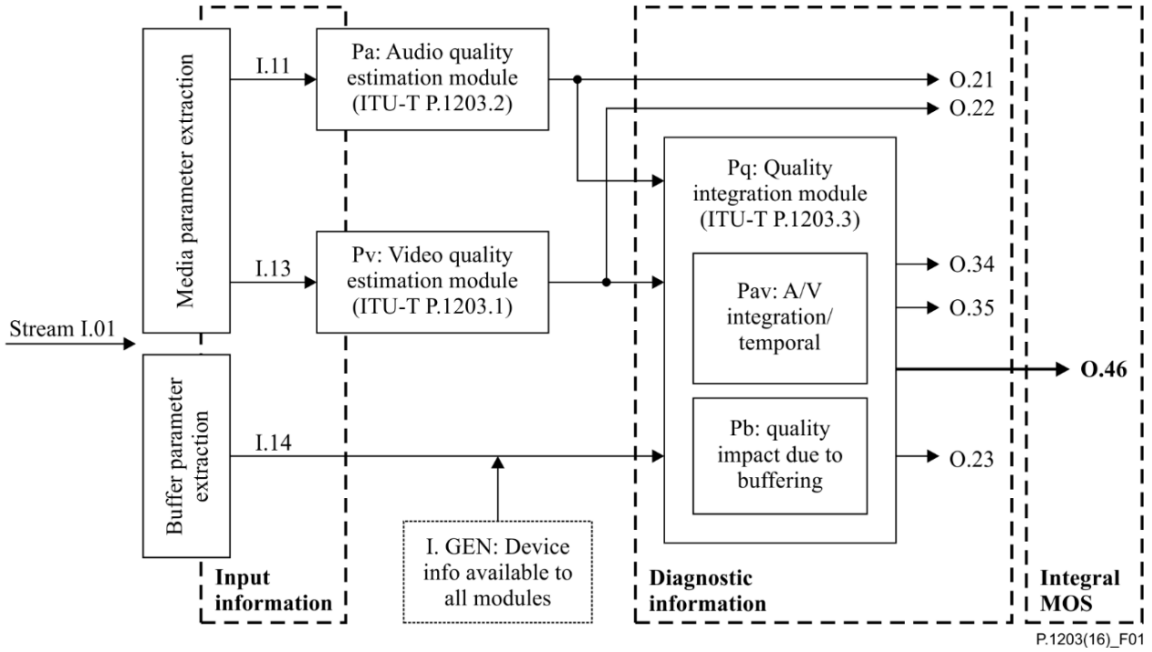


Figure 4.1: ITU-T P.1203 model [25]

4.1.2 QoE-driven Reward Function

To evaluate the video streaming experience, we employ the ITU-T P.1203 [45] Recommendation model to derive the QoE score for each video segment. The ITU-T P.1203-series of Recommendations provide a standardized methodology for evaluating the QoE of multimedia streaming services. The models comprise a set of modules for short-term video quality (ITU-T P.1203.1) and audio quality (ITU-T P.1203.2) estimation, usually referred to as Pv and Pa modules, respectively. A block diagram of the ITU-T P.1203 model is shown in Figure 4.1. Four different modes, referred to as modes 0 to 3, can be used for the Pv module. Specifically, we used the ITU-T P.1203.1 mode 0 for video quality evaluation, which derives from video segment metadata (I.13) and yields an overall QoE score represented as a Mean Opinion Score (MOS) for video segments. The available metadata for I.13 in mode 0 includes the video bitrate, frame rate, resolution, video codec, encoding profile, and the duration of the video segment. In this thesis, we mainly focused on the video bitrate adaptation problem. The video dataset we used in Section 3.3 only contains one audio bitrate track. Therefore, the Pa module calculation remains the same regardless of the arm selected in each round. For I.14, we assume the streaming device is a PC and the viewing distance is 1.5 meters. 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.

To integrate the ITU-T P.1203 model with low-latency live video streaming, we develop a QoE-driven reward function with the primary objective of reducing the live latency, increasing playback bitrate and minimizing rebuffering events. The reward function is defined as follows,

$$\mathrm{QoE}(i) = \mathrm{QoE}_{\mathrm{P1203}}(i) * \frac{\mathrm{LL}_{\mathrm{target}}}{\mathrm{LL}_{\mathrm{current}}(i)} * \frac{k}{\mathrm{B}_{\mathrm{max}}} - \frac{\sum_{j=1}^{C_k} t_j^k}{\sum_{j=1}^{C} t_j} \tag{4.6}$$

where $\mathrm{QoE}_{\mathrm{P1203}}(i)$ represents the O.46 MOS score for video segment i calculated by ITU-T P.1203 model, $\mathrm{LL}_{\mathrm{target}}$ represents the required latency target in the current playback session, $\mathrm{LL}_{\mathrm{current}}(i)$ 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 video bitrate level k, t_j represents the duration of rebuffering event j, t_j^k represents the duration of rebuffering event j happens at bitrate k, arm k is the video bitrate for segment i, and $\mathrm{B}_{\mathrm{max}}$ is the highest video bitrate available in the MPD file.

The agent makes a decision on which arm to pull before downloading each video segment i, and the corresponding reward $r(i)$ for video segment i is obtained right after the download is completed, which is defined as,

$$r(i) = \mathrm{QoE}(i) \tag{4.7}$$

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

4.1.3 Catch-up Policy

A similar empirical catch-up policy as in *LoL+* [7] is employed in the proposed algorithm. The pseudocode of the catch-up policy is shown in Algorithm 1. The main goal of our catch-up policy is to avoid potential playback interruptions by slowing down playback speed during satellite handover periods, in addition to the case when the playback buffer is below the safe threshold. 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, the satellite handover period is defined using

Algorithm 1 Catch-up Policy (Pseudocode)

```
procedure CalculateSpeed(Δ)
    catchupRate ← 0.17
    d ← Δ * 5
    s ← (catchupRate * 2)/(1 + e^{-d})
    R(t) ← (1 − catchupRate) + s
    return R(t)
end procedure
procedure isHandoverPeriod
    return Now.Second in {12, 27, 42, 57}
end procedure
procedure PlaybackSpeed(LL_current, LL_target, Buffer_current, Buffer_min)
    if (isHandoverPeriod or Buffer_current < Buffer_min) then
        Δ ← |Buffer_current − Buffer_min|
        R(t) ← CalculateSpeed(Δ)
    else
        ϵ ← 0.02
        if (|LL_current − LL_target| ≤ (ϵ * LL_target)) then
            R(t) ← 1.0
        else
            Δ ← LL_current − LL_target
            R(t) ← CalculateSpeed(Δ)
        end if
    end if
    return R(t)
end procedure
```

the same motivation as in Section 3.3. The (12-27-42-57) seconds of every minute, are considered as the satellite handover periods. We define our catch-up policy as follows,

- The current buffer level is below the safe threshold ($\text{Buffer}_{\text{min}}$), 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.

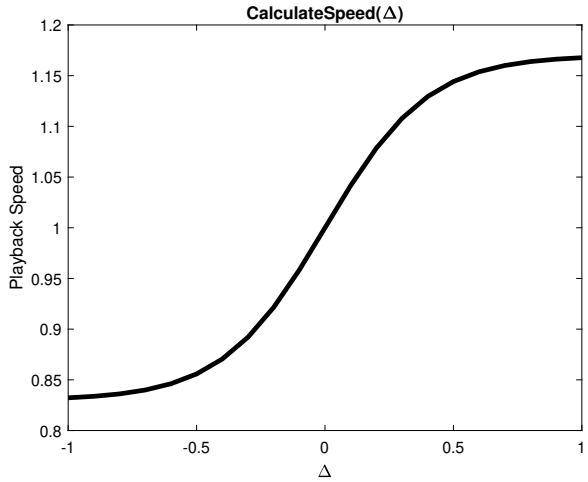


Figure 4.2: Sigmoid function for catch-up policy

– The current live latency is higher than the latency target: speed up playback speed.

Note that the playback speed is calculated with an empirical sigmoid function as shown in Figure 4.2 based on how large the deviation is between the current live latency ($\mathrm{LL}_{\mathrm{current}}$) and latency target ($\mathrm{LL}_{\mathrm{target}}$), and the current buffer level and safe threshold ($\mathrm{Buffer}_{\mathrm{min}}$), as shown in Algorithm 1. The larger the deviation, the faster/slower the playback speed. *catchupRate* defines the minimum and maximum catch-up rates, which represent the range of the playback speed.

4.2 Implementation

In this section, we describe the implementation details of our proposed ABR algorithm for low-latency live video streaming over Starlink networks in dash.js. We evaluated the performance of the proposed algorithm and compared it with the other three low-latency live video streaming ABR algorithms in network emulation settings and real Starlink networks. The performance evaluation with real Starlink networks was only conducted on the Starlink installation without ISLs being utilized. The implementation of this thesis is available online[1].

[1] `https://doi.org/10.5281/zenodo.10028731`

4.2.1 An End-to-End Prototype in dash.js

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` [53] which provides fast prototyping with various CMAB algorithms is chosen in our implementation. Our implementation chose Linear Thompson Sampling (LinTS) [1] as the solution. LinTS models the uncertainty by building a probability distribution from historical contexts and rewards and then samples from the distribution when choosing actions.

`MABWiser` is originally implemented in Python. The DASH reference player dash.js is based on the Media Source Extensions (MSE) of browsers and implemented in JavaScript. To integrate both, we utilized a WebAssembly-based Python runtime for browsers, `Pyodide` [56]. We implemented the core logic of the CMAB-based video bitrate adaptation algorithm and QoE calculation in Python and interacted with dash.js through `Pyodide` APIs. We acknowledge that implementing the 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 less than 100 milliseconds, which is negligible in our scenario.

Similar to the latency target-based measurements in [42], the `FastSwitching` option is disabled in dash.js, and we 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 3.1 and queried by the dash.js player through REST APIs. We set the *maxDrift* to 5s and *catchupRate* to 0.17, which means the maximum latency deviation allowed before dash.js triggers a seek-to-live event is 5s, and the player would catch up approximately 5s within a 30s window. We set the safe threshold of minimal playback buffer $\text{Buffer}_{\text{min}}$ to 0.5s. 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. The exploration rate of LinTS is set to 1.0.

4.2.2 Network Emulation

In addition to evaluating the performance of low-latency live video streaming ABR algorithms in real Starlink networks, we also built a network emulation testbed to provide repeatable environments for performance evaluation. The architecture of the purpose-built emulation testbed is similar to the real Starlink testbed with minor

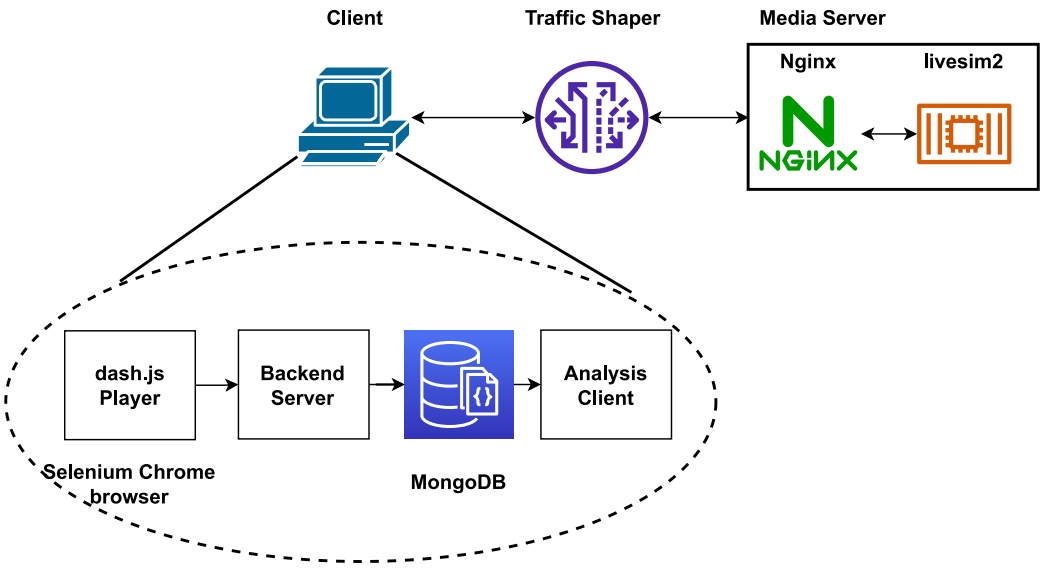


Figure 4.3: Starlink DASH emulation testbed architecture

modifications, which is shown in Figure 4.3. All the components in Figure 4.3 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 and packet loss following the Starlink satellite handover pattern and the measured latency performance. Specifically, the "Traffic Shaper" container is implemented using the Linux network utility `tc` and `netem` to introduce artificial delay and packet loss during handover periods. In our emulation, we set the latency to 100 ms and the packet loss rate to 1% during handover periods, which is defined as (12-27-42-57) seconds every minute. The network latency and packet loss rate outside the handover periods are set to 40 ms and 0.1% respectively.

In addition to network emulation and real Starlink networks, we also included a terrestrial network setting as the control group, which was conducted on the campus network of the University of Victoria. The campus network and the virtual machine in GCP's *us-west1-a* availability zone are interconnected through research and educational networks including BCNET (AS271), CANARIE (AS6509) and Pacific Northwest Gigapop (PNWGP) (AS101), with very low latency (average latency at 10 ms) and abundant bandwidth. As our proposed ABR algorithm takes advantage of predictable satellite handover patterns, when evaluating the performance of the proposed algorithm in the terrestrial network setting, we disabled the handover awareness and replaced the catch-up mechanism with the *Default* catch-up policy in

dash.js.

4.2.3 Complexity Analysis

The complexity of the proposed algorithm mainly comes from the calculation of the expected reward of each arm and is largely dependent on the complexity of the specific CMAB algorithm utilized.

The LinTS implementation in `MABWiser` assumes that the reward of each arm follows a Gaussian distribution and maintains a ridge regression model to estimate the mean and variance of the reward distribution for each arm. In each round, the Gaussian distribution is randomly sampled to obtain the expected reward of each arm and hence the arm with the highest expected reward is selected. We assume T is finite but unknown to the agent and the agent only compiles the history $\mathcal{H}_q^{t-1}$ of the current 15-second timeslot beginning from round q and up to $t-1$. In DASH, the video is segmented into small segments. The duration of each DASH video segment normally ranges between 2–10 seconds. Thus, the maximum number of rounds that can be played in a 15-second timeslot is upper bounded.

In our evaluation, the low-latency live video streaming dataset we used has a fixed segment duration of 2 seconds. The player can allocate a fixed amount of space for the calculation. The space complexity of the algorithm is $\mathcal{O}(1)$. The time complexity of the algorithm is $\mathcal{O}(d^3 + K)$, where K is the number of arms (video bitrate levels). $\mathcal{O}(d^3)$ comes from the complexity of matrix inversion operations in linear regression, d is the number of features, i.e., $d = |b(t)|$. In practice, the number of features and the number of video bitrate levels is finite and usually small. Thus, the matrix inversion operations in linear regression can be done in a relatively short time, and the time complexity could be close to $\mathcal{O}(K)$.

4.3 Performance Evaluation

In this section, we provide a detailed performance evaluation of the proposed ABR algorithm for low-latency live video streaming over satellite networks, and compare it with the other three state-of-the-art low-latency live ABR algorithms in dash.js. We conducted the performance evaluation in three different network settings: network emulation with the purpose-built emulation testbed, real Starlink network without ISLs, and terrestrial networks. Five video playback metrics are collected to evaluate

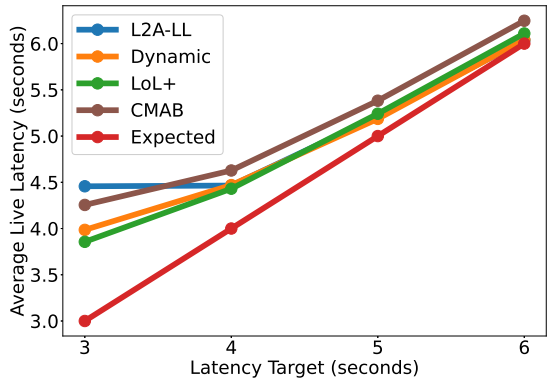
the performance of low-latency live video streaming ABR algorithms: average live latency, average bitrate, rebuffering time ratio, the number of bitrate switches, and bitrate standard deviation.

4.3.1 Average Live Latency

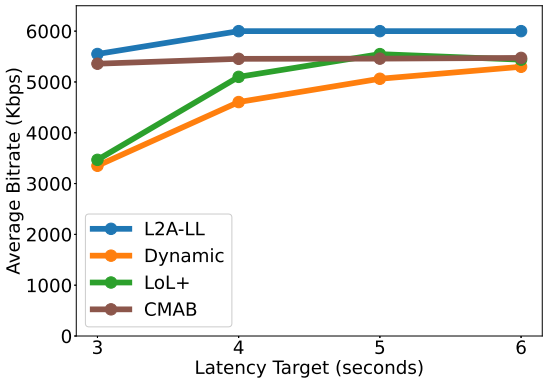
The first performance metric is average live latency, which plays the most critical role and significantly affects user experience in low-latency live video streaming. Figure 4.4 shows the average live latency in all three network settings. In network emulation as shown in Figure 4.4(a), when the latency target is 3 seconds, all the ABR algorithms cannot reach the expected latency requirement with at least around 1-second deviation. As the latency target increases, the gap between average live latency and expected latency requirement narrows. This is partially because of the 2-second segment duration in our performance evaluation, which leaves the catch-up policy with less space to adjust the playback speed to reach the latency target. In real Starlink networks, as shown in Figure 4.4(b), only the proposed algorithm can achieve lower live latency than the expected latency target, when the latency target is larger than 4 seconds. In the terrestrial network setting as shown in Figure 4.4(c), all ABR algorithms have very similar performance in achieving the expected latency target requirements.

4.3.2 Average Bitrate

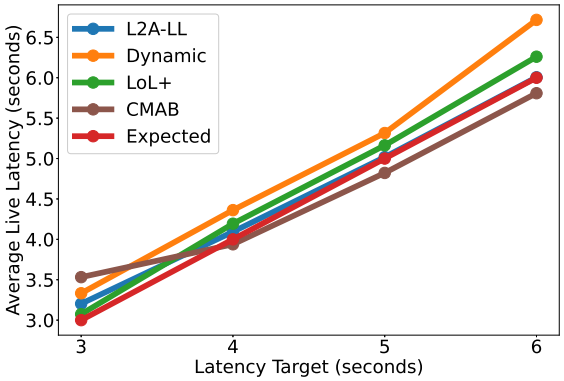
In Figure 4.5, we can see all ABR algorithms show the same trend in increasing the average bitrate as the latency target increases. In Figure 4.5(a) and Figure 4.5(b), both *Dynamic* and *LoL+* algorithms show significant fluctuations in average bitrate at different latency targets, especially when the latency target is below 4 seconds. In Figure 4.5(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. On the other hand, the proposed CMAB-based ABR algorithm can always maintain a high average bitrate across different latency targets in both network emulation and real Starlink networks. Although the proposed algorithm has a lower average bitrate than *Dynamic* and *L2A-LL* in the terrestrial network setting, it is a beneficial trade-off when considering the performance in dynamic network environments.



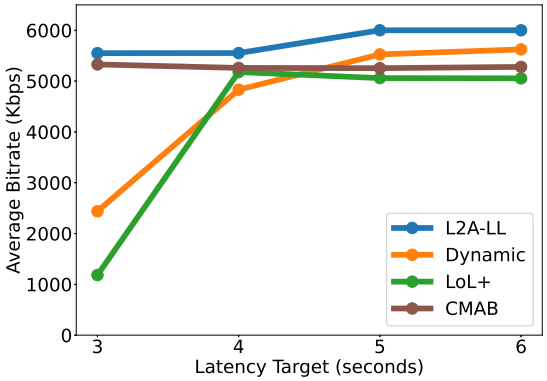
(a) Emulation



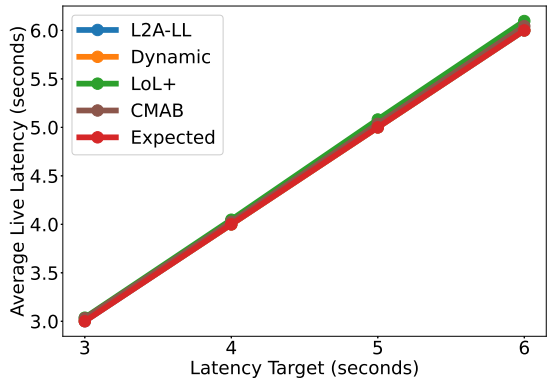
(a) Emulation



(b) Starlink

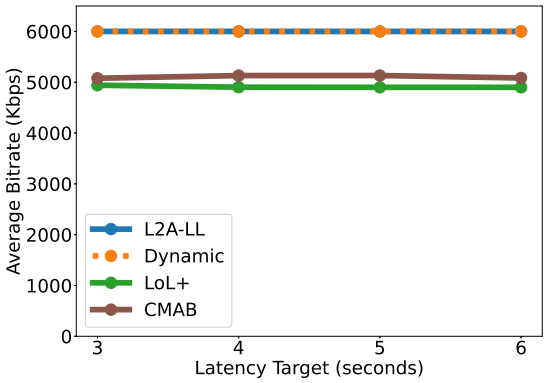


(b) Starlink



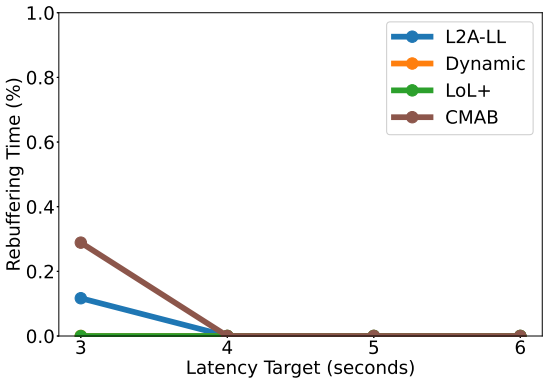
(c) Terrestrial

Figure 4.4: Average live latency

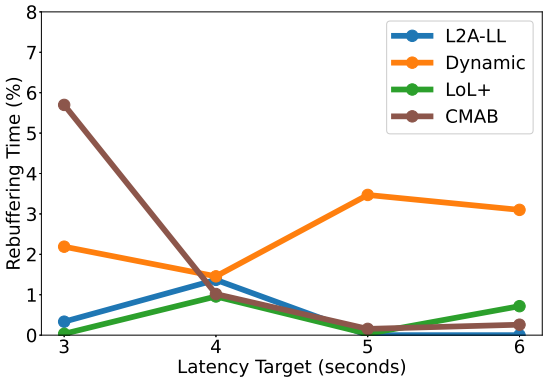


(c) Terrestrial

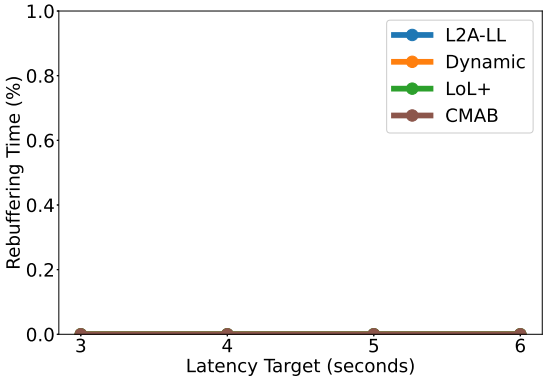
Figure 4.5: Average bitrate



(a) Emulation

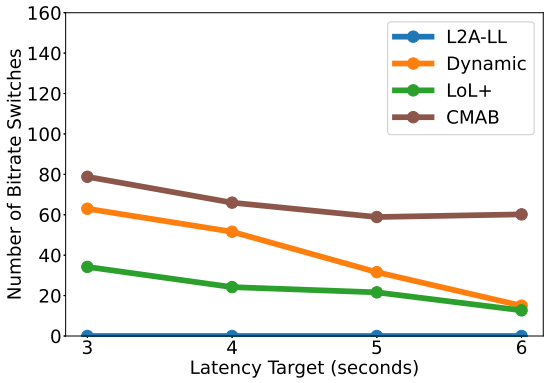


(b) Starlink

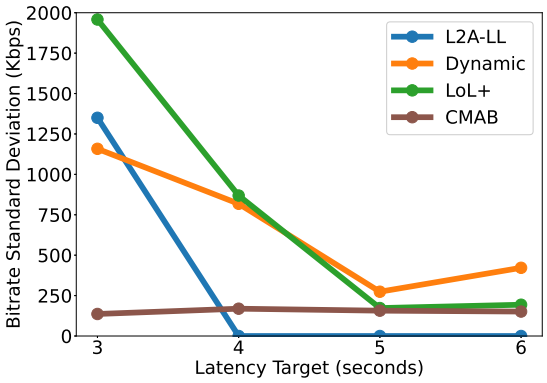


(c) Terrestrial

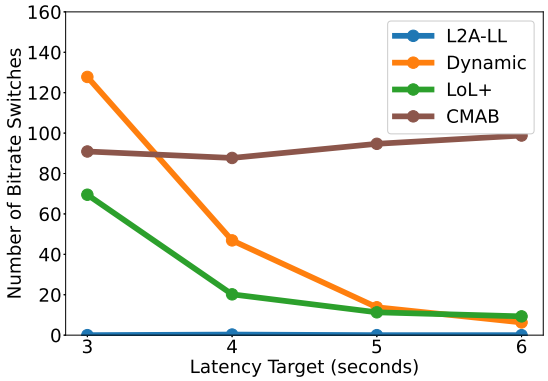
Figure 4.6: Rebuffering time ratio



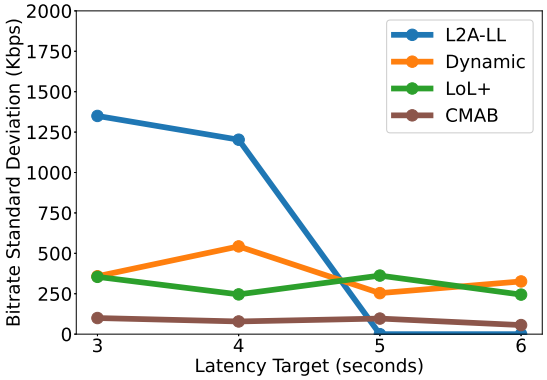
(a) Emulation



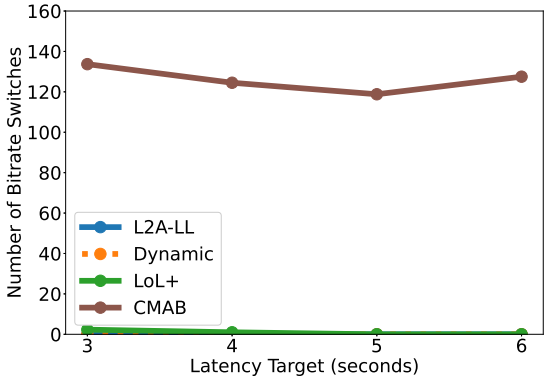
(a) Emulation



(b) Starlink

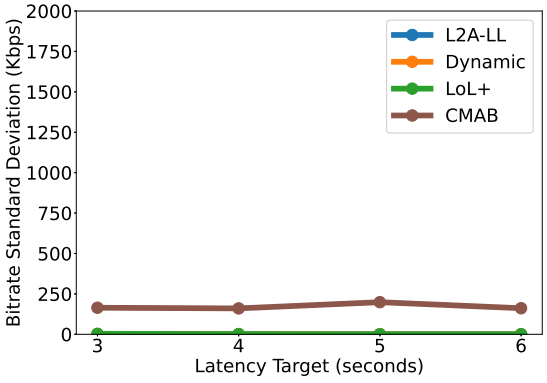


(b) Starlink



(c) Terrestrial

Figure 4.7: Number of bitrate switches



(c) Terrestrial

Figure 4.8: Bitrate standard deviation

4.3.3 Rebuffering Time Ratio

Rebuffering events are the worst experience for live video streaming as they interrupt the playback and cause a delay in live latency. Figure 4.6 shows the rebuffering time ratio in all three network settings. A larger latency target allows the player to build a more substantial local playback buffer, thereby better tolerating network fluctuations and reducing rebuffering events. In Figure 4.6(a), all ABR algorithms have a decreasing trend in rebuffering time ratio as the latency target increases. In the real Starlink network setting, the proposed algorithm can achieve a low rebuffering time ratio close to *L2A-LL* as shown in Figure 4.6(b), especially when the latency target is larger than 4 seconds. No rebuffering events are observed in the terrestrial network setting for all ABR algorithms as shown in Figure 4.6(c).

4.3.4 Number of Bitrate Switches

Figure 4.7 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 in network emulation and real Starlink networks as shown in Figure 4.7(a) and Figure 4.7(b). Our proposed algorithm has a generally higher number of bitrate switches than other ABR algorithms in all network scenarios.

4.3.5 Bitrate Standard Deviation

However, Figure 4.8 shows that our proposed algorithm has a relatively low bitrate standard deviation in both network emulation and real Starlink networks. A lower bitrate standard deviation indicates that the algorithm can maintain a more stable playback quality and the shifts in playback quality with 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 [41] in the future.

There is one observation from the results worth mentioning is that our proposed algorithm also has a high number of bitrate switches in the terrestrial network environment. The potential explanation is that our algorithm only considers the history within the current 15-second timeslot without a moving average of previous measurements, and it has to restart the exploration phase from scratch after every handover

event. In conjunction with the 2-second DASH video segment duration, and the limited playback buffer size in live video streaming, it is challenging for the agent to converge to the optimal bitrate in a short time. We also did not include any precautions or adjustments on the learning rate in different network environments.

Chapter 5

Further Discussion

Previously, HTTP was built with the underlying transport protocol TCP. TCP suffers from high latency connection establishment and head-of-line (HoL) blocking issues, making it unsuitable for latency-critical live video streaming scenarios. QUIC is a new UDP-based transport protocol to enhance the performance and security of Internet connections. It has recently been proposed as RFC 9000 [26] by IETF and forms the foundation of HTTP/3 [8]. QUIC's built-in features, such as stream multiplexing, resolve the HoL blocking issue in previous HTTP/1.1 and HTTP/2. Additionally, 1-RTT connection establishment and 0-RTT connection resumption considerably reduce network latency, making QUIC a promising option for scenarios that require fast and reliable communications.

On the other hand, in recent years, multipath transport protocols have attracted increased attention [35, 19]. Multipath TCP (MPTCP) [20] was proposed as an extension to TCP, enabling the utilization of several distinct network paths for concurrent data transmission to enhance both network reliability and performance. MPTCP faces challenges in efficiently developing new scheduling algorithms and achieving widespread adoption due to the high optimization of the existing TCP network stack. Multipath QUIC (MPQUIC), one of the ongoing QUIC extension drafts, has collected significant interest in the community [37]. Similar to multipath TCP (MPTCP), MPQUIC aims to utilize multiple network paths for simultaneous data transmission to increase reliability and performance. Additionally, it also leverages QUIC's advantages such as stream multiplexing, 1-RTT connection establishment, and 0-RTT connection resumption. MPQUIC presents a new opportunity to develop state-of-the-art multipath scheduling algorithms in the user space, integrating application-specific contexts with less effort. The majority of current multipath adaptive video streaming

solutions employ two separate and independent algorithms for multipath scheduling and video bitrate adaptation [63, 66]. At the transport layer, network metrics such as throughput and latency are typically employed to carry out the multipath packet scheduling process. However, at the application layer, video players lack the context regarding transport layer packet scheduling algorithms. Consequently, the inter-path differences and intra-path fluctuations of network throughput and latency can impact the QoE of various video streaming scenarios. However, reconciling both control loops with empirical rules can be challenging due to their distinct objectives.

5.1 Multipath Transport Protocols

With proper configuration, MPTCP is expected to provide better network performance, such as higher throughput [30, 57], increased network resilience and redundancy in the face of network failures [33]. Xing et al. [62] proposed *OLAPS* for MPTCP to generate adaptive scheduling policies and quickly respond to abrupt network changes. *OLAPS* uses UCB1 with a custom reward function based on instantaneous throughput at both the subflow and connection levels, rather than relying on network measurements of congestion windows, RTT, and loss rates. Zhao et al. [66] proposed *MPTCP+*, which incorporated the path use decision and the multipath congestion control, to address the problems of inter-path throughput difference and congestion window fluctuation. On the other hand, while the MPQUIC extension draft is still under IETF discussion [37], many efforts have already been made to explore different scheduling algorithms for it. *Peekaboo* is a learning-based MPQUIC scheduler presented by Wu et al. [61]. *Peekaboo* utilized LinUCB [34] to address the multipath scheduling problem in heterogeneous network environments with dynamically changing path characteristics.

5.2 Joint Decision-Making for Multipath Adaptive Video Streaming

In this section, we briefly introduce our work on QoE-driven joint decision-making for multipath adaptive video streaming [67]. Similar to the problem formulation in Section 4.1, we utilized contextual multi-armed bandit algorithms to model the multipath adaptive video streaming problem.

The main difference lies in the formulation of arms and context vectors. In multipath adaptive video streaming, we consider there are N video segments and P available network paths. Each video segment is encoded into M different bitrate levels. The CMAB action space is defined as the cartesian product $M \times P = \{(m, p)|m \in M \text{ and } p \in P\}$. An arm k is defined as any possible combination of available video bitrate and network path (m, p).

We define the context vector $b(t)$ as follows,

$$b(t) = \{\mathrm{R}(t), \mathrm{RTT}_p(t), \mathrm{BW}_p(t), p = 1, 2, \ldots, P\} \tag{5.1}$$

where at time t, $\mathrm{R}(t)$ is the local playback buffer ratio, $\mathrm{R}(t) \in [0, 1]$. $\mathrm{RTT}_p(t)$, $\mathrm{BW}_p(t)$ is the estimated average RTT and throughput for path p at time t respectively. Typically, a low playback buffer ratio $\mathrm{R}(t)$ indicates that the network cannot sustain the required video bitrate. Thus, a rebuffering event would occur unless the video streaming client adjusts to a lower video bitrate. However, this assumption does not always hold. A high playback buffer ratio does not necessarily mean the network is healthy because the player could be just downloading many low-bitrate video segments. Thus, the local playback buffer ratio alone cannot be referred to as a reliable context for the player to make video bitrate and multipath scheduling decisions. Therefore, $\mathrm{RTT}_p(t)$ and $\mathrm{BW}_p(t)$ are added to the context vector $b(t)$ as additional information to help the player make reliable decisions.

We develop a QoE-driven reward function with the primary objective of minimizing rebuffering events and increasing playback bitrate. In VoD scenarios, users' QoE is usually less sensitive to network latency, compared with live video streaming, and largely affected by video bitrate and rebuffering events. We define the rebuffering time ratio at time t as follows,

$$\mathrm{RBF}_t = \frac{\sum_{j=1}^{C} t_j}{t}, \tag{5.2}$$

where C is the total number of rebuffering events that happened up to time t and t_j is the time duration for each rebuffering event j. The agent pulls an arm k before downloading each video segment i, and the corresponding reward $r_k(i)$ for video

segment i obtained from path p is defined as,

$$r_k(i) = \begin{cases} -\frac{\mathrm{B}(i)}{\mathrm{B}_{\max}} & \text{if } \mathrm{RBF}_t - \mathrm{RBF}_{t-1} > 0 \\ \frac{\mathrm{B}(i)}{\mathrm{B}_{\max}} & \text{otherwise} \end{cases} \tag{5.3}$$

$\mathrm{B}(i)$ is the video bitrate for video segment i and $\mathrm{B}_{\max}$ is the maximum bitrate available in the MPD file. When $\mathrm{RBF}_t - \mathrm{RBF}_{t-1} > 0$, it indicates that the current selection of bitrate and path introduces interruptions to the playback. Thus, a negative reward is revealed to the agent. The objective of the agent is to pull the best arm which yields the highest expected reward in each round for video segment i, i.e., the video player decides the best combination of network paths and video bitrate which can provide the best QoE for users.

5.3 Limitations

As of October 2023, there are a few active IETF QUIC implementations that support the MPQUIC extension, including *Picoquic* [22], *XQUIC* [2] and *quiche* [11] (work-in-progress)[1]. None of the major web browsers implemented the MPQUIC extension draft [37]. On the other hand, dash.js is based on the Media Source Extensions (MSE) and JavaScript runtime of browsers. It has no access to the underlying transport layer protocol stack. Thus, it is not currently feasible to implement the proposed CMAB-based multipath ABR algorithm in dash.js properly.

However, with the ongoing IETF standardization of MPQUIC, we expect that major browsers will implement the MPQUIC extension and expose the underlying transport layer protocol stack to the application layer, enabling cross-layer information sharing and optimization such that the proposed CMAB-based multipath ABR algorithm can be implemented in dash.js in the future.

[1] `https://github.com/quicwg/multipath/wiki`

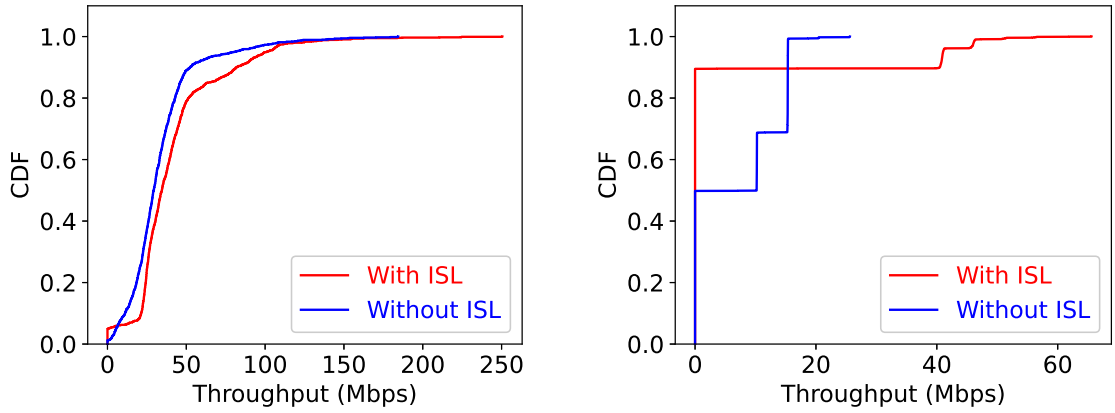
Chapter 6

Conclusions and Future Work

In this thesis, we investigated the problem of low-latency live video streaming over Starlink networks. We presented a novel ABR algorithm for low-latency live video streaming over Starlink networks, based on the unique network characteristics observed from our measurements on the Starlink access networks and the latency target-based analysis of low-latency live video streaming ABR algorithms in dash.js. We implemented an end-to-end prototype of the proposed algorithm in dash.js. The performance evaluation is conducted in both real Starlink networks and a purpose-built emulation testbed. We then discussed a potential solution that can further improve the QoE of adaptive video streaming by utilizing the multipath QUIC extension. In this chapter, we present the open research challenges and discuss future work.

6.1 Video Ingest 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 6.1, the uplink throughput is significantly lower than the downlink throughput with very high fluctuations. This is potentially due to the different beam placement strategies in downlink and uplink, the limited UT transmition power and the QoS policies employed by SpaceX. Similar to Section 3.2, it is important to note that the utilization of ISLs does not directly impact throughput performance. Instead, it is influenced by Starlink's capacity limitations and QoS policies in different geographical regions.



(a) Downlink throughput CDF

(b) Uplink throughput CDF

Figure 6.1: CDF of Starlink throughputs

In contrast to the "pull"-based model of live video 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 networks 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 QUIC and Extensions

While the MPQUIC extension is under ongoing standardization at IETF [37], the QUIC community also has a few specialized projects to optimize media delivery over QUIC, one of which is Media-Over-QUIC (moq) [13]. The goal of the moq working group is to develop a simple low-latency media delivery solution for media ingest and distribution. This solution addresses use cases including live video streaming, gaming, and media conferencing and will scale efficiently. The solution will be implementable in both browser and non-browser endpoints. It employs a producer-relay-consumer model, and media will be mapped onto underlying QUIC mechanisms (QUIC streams and/or QUIC datagrams) and can be used over raw QUIC or WebTransport. Industry participation such as CDN providers is crucial to the success of this project for developing a next-generation CDN architecture utilizing the "relay" model and providing similar infrastructures as compared to the existing worldwide CDN networks.

6.3 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. It also creates new opportunities to design satellite handover-aware congestion control algorithms such as SaTCP [10] to further improve the TCP performance over 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. Tolouei [58] analyzed the performance of cloud gaming (Nvidia GeForce NOW) over Starlink networks. The results show the uplink performance of Starlink networks has a significant impact on the cloud gaming experience, especially in latency-sensitive scenarios. Games such as first-person shooters are particularly vulnerable to fluctuating latency and frequent satellite handovers, given they also encompass human interactions via Starlink's uplink channels.

Major cloud computing companies such as Amazon and Microsoft are launching Ground Stations as a Service (GSaaS) with AWS Ground Station and Azure Orbital, which enable customers to directly streamline satellite data into their cloud environments to significantly reduce latency. Notably, SpaceX partnered with Google Cloud to host their ground stations using Google's data center infrastructures. This emerging trend suggests a future where the concept of "edge computing" will be redefined in the realm of LEO satellite Internet.

6.4 Future Work

For future works, the technical details and scheduling algorithms of Starlink ISLs require further investigation through collaborative measurements worldwide. 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 video streaming latency and QoE. More work can be done to improve the catch-up policy and reward function with a more detailed analysis of satellite handover patterns and present theoretical regret-bound analysis for the CMAB-based algorithm. VMAF can be used to evaluate the visual quality fluctuation of different ABR algorithms in low-latency live video streaming. It is also worth investigating the uplink video ingestion performance over Starlink networks.

Bibliography

[1] Shipra Agrawal and Navin Goyal. Thompson Sampling for Contextual Bandits with Linear Payoffs. In *Proceedings of the 30th International Conference on Machine Learning*, pages 127–135. PMLR, May 2013.

[2] Alibaba. XQUIC. https://github.com/alibaba/xquic, November 2023.

[3] Amazon. Amazon shares an update on how Project Kuiper's test satellites are performing. https://www.aboutamazon.com/news/innovation-at-amazon/amazon-project-kuiper-test-satellites-space-launch-october-2023-update, October 2023.

[4] Amazon. Amazon's Project Kuiper completes successful tests of optical mesh network in low Earth orbit. https://www.aboutamazon.com/news/innovation-at-amazon/amazon-project-kuiper-oisl-space-laser-december-2023-update, December 2023.

[5] Consumer Technology Association. Web Application Video Ecosystem Interoperability Project. https://github.com/cta-wave/Test-Content.

[6] Abdelhak Bentaleb, May Lim, Mehmet N. Akcay, Ali C. Begen, Sarra Hammoudi, and Roger Zimmermann. Toward One-Second Latency: Evolution of Live Media Streaming, October 2023.

[7] Abdelhak Bentaleb, Mehmet N. Akcay, May Lim, Ali C. Begen, and Roger Zimmermann. Catching the Moment With LoL$^+$ in Twitch-Like Low-Latency Live Streaming Platforms. *IEEE Transactions on Multimedia*, 24:2300–2314, 2022.

[8] Mike Bishop. HTTP/3. https://datatracker.ietf.org/doc/rfc9114, June 2022.

[9] Rodrigo Blázquez-García, Diego Cristallini, Martin Ummenhofer, Viktor Seidel, Jörg Heckenbach, and Daniel O'Hagan. Experimental Comparison of Starlink

and OneWeb Signals for Passive Radar. In *2023 IEEE Radar Conference (RadarConf23)*, pages 1–6, May 2023.

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

[11] Cloudflare. Quiche. https://github.com/cloudflare/quiche, November 2023.

[12] Laizhong Cui, Dongyuan Su, Shu Yang, Zhi Wang, and Zhong Ming. TCLiVi: Transmission Control in Live Video Streaming Based on Deep Reinforcement Learning. *IEEE Transactions on Multimedia*, 23:651–663, 2021.

[13] Luke Curley, Kirill Pugin, Suhas Nandakumar, and Victor Vasiliev. Media over QUIC Transport. https://datatracker.ietf.org/doc/draft-ietf-moq-transport-00, July 2023.

[14] DASH-IF. Dash.js. https://github.com/Dash-Industry-Forum/dash.js, September 2023.

[15] DASH-IF. Livesim2. https://github.com/Dash-Industry-Forum/livesim2, August 2023.

[16] Ericsson. Ericsson Mobility Report. https://www.ericsson.com/en/reports-and-papers/mobility-report/reports/june-2023, June 2023.

[17] Eutelsat. OneWeb. http://oneweb.net, November 2023.

[18] Reza Farahani, Farzad Tashtarian, Alireza Erfanian, Christian Timmerer, Mohammad Ghanbari, and Hermann Hellwagner. ES-HAS: An Edge- and SDN-Assisted Framework for HTTP Adaptive Video Streaming. In *Proceedings of the 31st ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, NOSSDAV '21, pages 50–57, New York, NY, USA, July 2021. Association for Computing Machinery.

[19] Simone Ferlin, Özgü Alay, Olivier Mehani, and Roksana Boreli. BLEST: Blocking Estimation-based MPTCP Scheduler for Heterogeneous Networks. In *2016 IFIP Networking Conference (IFIP Networking) and Workshops*, pages 431–439, May 2016.

[20] Alan Ford, Costin Raiciu, Mark J. Handley, Olivier Bonaventure, and Christoph Paasch. TCP Extensions for Multipath Operation with Multiple Addresses. https://datatracker.ietf.org/doc/rfc8684, March 2020.

[21] Johan Garcia, Simon Sundberg, Giuseppe Caso, and Anna Brunstrom. Multi-Timescale Evaluation of Starlink Throughput. In *Proceedings of the 1st ACM Workshop on LEO Networking and Communication*, LEO-NET '23, pages 31–36, New York, NY, USA, October 2023. Association for Computing Machinery.

[22] Christian Huitema. Picoquic. https://github.com/private-octopus/picoquic, April 2023.

[23] ISO/IEC. ISO/IEC 23000-19:2020, March 2020.

[24] ISO/IEC. ISO/IEC 23009-1:2022, August 2022.

[25] ITU-T. P.1203 : Parametric bitstream-based quality assessment of progressive download and adaptive audiovisual streaming services over reliable transport. https://www.itu.int/rec/T-REC-P.1203-201710-I/en.

[26] Jana Iyengar and Martin Thomson. QUIC: A UDP-Based Multiplexed and Secure Transport. https://datatracker.ietf.org/doc/rfc9000, May 2021.

[27] Jayasuryan V. Iyer, Khasim Shaheed Shaik Mahammad, Yashodhan Dandekar, Ramakrishna Akella, Chen Chen, Phillip E. Barber, and Peter J. Worters. System and Method of Providing a Medium Access Control Scheduler, December 2022.

[28] Theo Karagkioules, Rufael Mekuria, Dirk Griffioen, and Arjen Wagenaar. Online Learning for Low-Latency Adaptive Streaming. In *Proceedings of the 11th ACM Multimedia Systems Conference*, MMSys '20, pages 315–320, New York, NY, USA, May 2020. Association for Computing Machinery.

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

[30] Ramin Khalili, Nicolas Gast, Miroslav Popovic, and Jean-Yves Le Boudec. MPTCP Is Not Pareto-Optimal: Performance Issues and a Possible Solution. *IEEE/ACM Transactions on Networking*, 21(5):1651–1665, October 2013.

[31] Jonathan Kua, Grenville Armitage, and Philip Branch. A Survey of Rate Adaptation Techniques for Dynamic Adaptive Streaming Over HTTP. *IEEE Communications Surveys & Tutorials*, 19(3):1842–1866, 2017.

[32] *LEO-NET '23: Proceedings of the 1st ACM Workshop on LEO Networking and Communication*, New York, NY, USA, 2023. Association for Computing Machinery.

[33] Li Li, Ke Xu, Tong Li, Kai Zheng, Chunyi Peng, Dan Wang, Xiangxiang Wang, Meng Shen, and Rashid Mijumbi. A Measurement Study on Multi-path TCP with Multiple Cellular Carriers on High Speed Rails. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '18, pages 161–175, New York, NY, USA, August 2018. Association for Computing Machinery.

[34] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A Contextual-Bandit Approach to Personalized News Article Recommendation. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 661–670, New York, NY, USA, April 2010. Association for Computing Machinery.

[35] Yeon-sup Lim, Erich M. Nahum, Don Towsley, and Richard J. Gibbens. ECF: An MPTCP Path Scheduler to Manage Heterogeneous Paths. In *Proceedings of the 13th International Conference on Emerging Networking EXperiments and Technologies*, CoNEXT '17, pages 147–159, New York, NY, USA, November 2017. Association for Computing Machinery.

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

[37] Yanmei Liu, Yunfei Ma, Quentin De Coninck, Olivier Bonaventure, Christian Huitema, and Mirja Kühlewind. Multipath Extension for QUIC. https://datatracker.ietf.org/doc/draft-ietf-quic-multipath-06, October 2023.

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

[39] François Michel, Martino Trevisan, Danilo Giordano, and Olivier Bonaventure. A First Look at Starlink Performance. In *Proceedings of the 22nd ACM Internet Measurement Conference*, IMC '22, pages 130–136, New York, NY, USA, October 2022. Association for Computing Machinery.

[40] Owens Nathan. Unoffical Starlink Global Gateways & PoPs. https://tinyurl.com/starlink-gateway-map, July 2023.

[41] Netflix. VMAF. https://github.com/Netflix/vmaf, October 2023.

[42] Piers O'Hanlon and Adil Aslam. Latency Target based Analysis of the DASH.js Player. In *Proceedings of the 14th Conference on ACM Multimedia Systems*, MMSys '23, pages 153–160, New York, NY, USA, June 2023. Association for Computing Machinery.

[43] OneWeb. OneWeb confirms successful deployment of 16 satellites including next-generation JoeySat. https://oneweb.net/resources/oneweb-confirms-successful-deployment-16-satellites-including-next-generation-joeysat, May 2023.

[44] Jianping Pan, Jinwei Zhao, and Lin Cai. Measuring a Low-Earth-Orbit Satellite Network. In *2023 IEEE 34th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, PIMRC '23, pages 1–6, September 2023.

[45] 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. 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, pages 466–471, New York, NY, USA, June 2018. Association for Computing Machinery.

[46] Crist Ry and Paul Trey. Starlink Explained: What You Need to Know About Elon Musk's Satellite Internet Service. https://www.cnet.com/home/internet/starlink-satellite-internet-explained/, June 2023.

[47] *SatCom '23: Proceedings of the 1st ACM MobiCom Workshop on Satellite Networking and Computing*, New York, NY, USA, 2023. Association for Computing Machinery.

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

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

[50] Kevin Spiteri, Rahul Urgaonkar, and Ramesh K. Sitaraman. BOLA: Near-Optimal Bitrate Adaptation for Online Videos. *IEEE/ACM Transactions on Networking*, 28(4):1698–1711, August 2020.

[51] Gregory Stock, Juan A. Fraire, and Holger Hermanns. Distributed On-Demand Routing for LEO Mega-Constellations: A Starlink Case Study. In *2022 11th Advanced Satellite Multimedia Systems Conference and the 17th Signal Processing for Space Communications Workshop (ASMS/SPSC)*, pages 1–8, Graz, Austria, September 2022. IEEE.

[52] Thomas Stockhammer. Dynamic Adaptive Streaming over HTTP – Standards and Design Principles. In *Proceedings of the Second Annual ACM Conference on Multimedia Systems*, MMSys '11, pages 133–144, New York, NY, USA, February 2011. Association for Computing Machinery.

[53] Emily Strong, Bernard Kleynhans, and Serdar Kadıoğlu. MABWISER: Parallelizable Contextual Multi-armed Bandits. *International Journal on Artificial Intelligence Tools*, 30(04):2150021, June 2021.

[54] Meesha Sundarum. Update on 5G Non-Terrestrial Networks Briefing Paper. Technical report, 5G Americas, July 2023.

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

[56] The Pyodide development team. Pyodide. https://github.com/pyodide/pyodide, September 2023.

[57] Yannis Thomas, Merkourios Karaliopoulos, George Xylomenos, and George C. Polyzos. Low Latency Friendliness for Multipath TCP. *IEEE/ACM Transactions on Networking*, 28(1):248–261, February 2020.

[58] Pouria Tolouei. Analysing the Performance of Cloud Gaming over a Low-Earth Orbit Satellite Network, September 2023.

[59] Shangguang Wang and Qing Li. Satellite Computing: Vision and Challenges. *IEEE Internet of Things Journal*, pages 1–1, 2023.

[60] Yufei Wang, Lin Cai, and Jun Liu. High-Reliability, Low-Latency, and Load-Balancing Multipath Routing for LEO Satellite Networks. In *2023 Biennial Symposium on Communications (BSC)*, pages 107–111, July 2023.

[61] Hongjia Wu, Özgü Alay, Anna Brunstrom, Simone Ferlin, and Giuseppe Caso. Peekaboo: Learning-Based Multipath Scheduling for Dynamic Heterogeneous Environments. *IEEE Journal on Selected Areas in Communications*, 38(10):2295–2310, October 2020.

[62] Yitao Xing, Kaiping Xue, Yuan Zhang, Jiangping Han, Jian Li, and David S. L. Wei. An Online Learning Assisted Packet Scheduler for MPTCP in Mobile Networks. *IEEE/ACM Transactions on Networking*, pages 1–16, 2023.

[63] Wang Yang, Jing Cao, and Fan Wu. Adaptive Video Streaming with Scalable Video Coding using Multipath QUIC. In *2021 IEEE International Performance, Computing, and Communications Conference (IPCCC)*, pages 1–7, October 2021.

[64] Hyunho Yeo, Chan Ju Chong, Youngmok Jung, Juncheol Ye, and Dongsu Han. NEMO: Enabling Neural-Enhanced Video Streaming on Commodity Mobile Devices. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, MobiCom '20, pages 1–14, New York, NY, USA, September 2020. Association for Computing Machinery.

[65] Haoyuan Zhao, Hao Fang, Feng Wang, and Jiangchuan Liu. 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, pages 43–49, New York, NY, USA, June 2023. Association for Computing Machinery.

[66] Jia Zhao, Jiangchuan Liu, Cong Zhang, Yong Cui, Yong Jiang, and Wei Gong. MPTCP+: Enhancing Adaptive HTTP Video Streaming over Multipath. In *2020 IEEE/ACM 28th International Symposium on Quality of Service (IWQoS)*, pages 1–6, Hangzhou, China, June 2020. IEEE.

[67] Jinwei Zhao and Jianping Pan. QoE-driven Joint Decision-Making for Multipath Adaptive Video Streaming. In *2023 IEEE 42nd Global Communications Conference*, GLOBECOM '23, Kuala Lumpur, Malaysia, December 2023. IEEE.

Appendix A

Publications

The following list of publications shows the outcome of my research with my supervisor and collaborators during my Master's program at the University of Victoria.

A.1 Conference Papers

- Jinwei Zhao, Jianping Pan. Low Latency Live Video Streaming over a Low-Earth-Orbit Satellite Network with DASH. Accepted by *2024 ACM 15th Multimedia Systems Conference (MMSys'24)*.
- Jinwei Zhao, Jianping Pan. QoE-driven Joint Decision-Making for Multipath Adaptive Video Streaming. In *2023 IEEE 42nd Global Communications Conference (GLOBECOM'23)*.
- Jianping Pan, Jinwei Zhao, Lin Cai. Measuring a Low-Earth-Orbit Satellite Network. In *2023 IEEE 34th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC'23)*

A.2 Open Dataset

- Jinwei Zhao, Jianping Pan. Starlink Latency and Downlink Throughput Measurement Dataset. Zenodo. `https://doi.org/10.5281/zenodo.10020034`.