StarStream: Live Video Analytics over Space Networking

Miao Zhang
Simon Fraser University
Burnaby, Canada
mza94@sfu.ca

Jiaxing Li
Simon Fraser University
Burnaby, Canada
jla641@sfu.ca

Haoyuan Zhao
Simon Fraser University
Burnaby, Canada
hza127@sfu.ca

Linfeng Shen
Simon Fraser University
Burnaby, Canada
linfengs@sfu.ca

Jiangchuan Liu
Simon Fraser University
Burnaby, Canada
jcliu@sfu.ca

Abstract

Streaming videos from resource-constrained front-end devices over networks to resource-rich cloud servers has long been a common practice for surveillance and analytics. Most existing *live video analytics* (LVA) systems, however, have been built over terrestrial networks, limiting their applications during natural disasters and in remote areas that desperately call for real-time visual data delivery and scene analysis. With the recent advent of space networking, in particular, Low Earth Orbit (LEO) satellite constellations such as Starlink, high-speed truly global Internet access is becoming available and affordable. This paper examines the challenges and potentials of LVA over modern LEO satellite networking (LSN). Using Starlink as the testbed, we have carried out extensive in-the-wild measurements to gain insights into its achievable performance for LVA. The results reveal that the uplink bottleneck in today's LSN, together with the volatile network conditions, can significantly affect the service quality of LVA and necessitate prompt adaptation. We accordingly develop `StarStream`, a novel LSN-adaptive streaming framework for LVA. At its core, `StarStream` is empowered by a Transformer-based network performance predictor tailored for LSN and a content-aware configuration optimizer. We discuss a series of key design and implementation issues of `StarStream` and demonstrate its effectiveness and superiority through trace-driven experiments with real-world network and video processing data.

CCS Concepts

• **Networks** → **Network measurement**; • **Information systems** → **Multimedia streaming**.

Keywords

LEO satellite; live video analytics; network measurement

1 Introduction

Live video analytics (LVA) [2, 13, 21, 43] analyzes online video data from networked cameras for automated knowledge extraction. Given the limited resources in front-end cameras [14], the video data are typically streamed to resource-abundant edge or cloud servers for real-time analytics [4, 14, 15, 42]. To date, most data have been carried over terrestrial networks, which have covered much of the human-inhabited lands and major roads in developed countries. Yet, despite global efforts to connect the uncovered, unserved, or underserved populations to the Internet, one-third of the Earth's population (around 2.9 billion) remains disconnected [25]. Many rural and remote areas, crucial for industries such as mining, fishing, or forestry, have not been covered, or may never be, including places like Alaska in the USA, northern Canada, and central Australia. Additionally, terrestrial networks can be vulnerable to disruptions from extreme weather, natural disasters, or other emergencies.

To bridge this digital gap, recent years have seen the rapid deployments of space networking, particularly those based on Low Earth Orbit (LEO) satellites. Compared to geostationary (GEO) satellites that orbit at 35, 786 km, LEO satellites are much closer to the Earth's surface (below 2, 000 km). This proximity significantly reduces the launch cost as well as the signal travel distance, which further leads to substantially lower network latency (600+ vs. 25 ms [23]). With a massive number of smaller satellites operating in higher frequency bands, an LEO satellite constellation provides truly global coverage with much higher bandwidth than GEO (e.g., 178 vs. 82 Mbps median download throughput [19]). As a key player in this field, Starlink has approximately 6, 241 LEO satellites serving at altitudes of about 550 km up to date [22, 32], with plans to expand to 34, 400 over the long haul [33]. Together, these satellites offer high-speed and affordable Internet access to 3 million subscribers [33], many in remote areas with no terrestrial access [11, 18, 19], making latency- and bandwidth-critical LVA anywhere on the Earth possible.[1]

As the *LEO satellite networking* (LSN) evolves to provide seamless global coverage, LVA applications built upon it will accordingly expand to currently underserved areas, enabling otherwise challenging services, including disaster response and relief, industrial site and wildlife monitoring, and maritime surveillance, to name

---

[1]Starlink's coverage has not yet reached polar areas with its current orbit inclination of about 53°, but it is expected to cover these regions in the coming years.
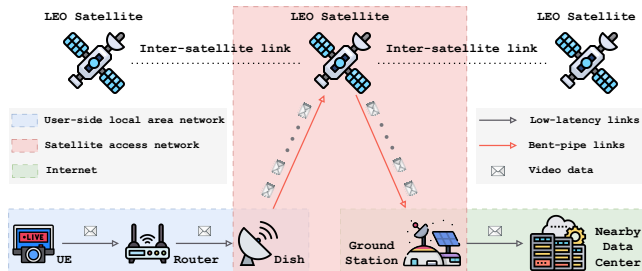
Figure 1: An overview of LSN-enabled LVA.

Table 1: Starlink access network performance (mean ± standard deviation) between the UE and servers.

Network performance metrics	gc-server	aws-server
Download throughput (Mbps)	83.4 ± 60.5	110.1 ± 57.5
Upload throughput (Mbps)	8.1 ± 3.3	8.3 ± 3.5
RTT (ms)	46.9 ± 14.4	40.5 ± 16.4

but a few. Even in urban areas where well-established terrestrial infrastructures exist, LSN can serve as a resilient alternative or even the preferred communication method for latency and cost reasons. For instance, Starlink has recently partnered with cloud providers to install ground stations (GSs) within or near cloud data centers to process user data from space directly at the edge of the LSN [6, 11]. The enhanced integration of communications and computing infrastructures creates new possibilities for realizing resource-efficient LSN-enabled LVA.

Recent measurement studies [11, 19, 45] have confirmed the advantages of LSN over its GEO counterparts and identified its potential in supporting various network-intensive applications, such as video streaming and cloud gaming [17, 19, 45]. Nevertheless, our measurements show that LSN exhibits highly asymmetric uplink and downlink performance[2], with the mean download throughput being more than 10× higher than the upload throughput. The scarcity of uplink resources poses non-trivial challenges for LVA, which primarily consumes the upload bandwidth to ship videos from cameras to remote servers. Furthermore, being susceptible to environmental factors (e.g., precipitation, cloud cover, and temperature [11, 18]) and relative satellite motion, LSN demonstrates wildly fluctuating performance over time. This prevents upper-layer LVA applications from delivering consistent quality of experience (QoE).

This paper closely examines LSN performance from an end-user perspective and sheds light on building LVA services over space networking. We take Starlink as the testbed and discuss the key issues toward designing StarStream, a high-performance and analytics-oriented live video streaming framework over LSN. Our contributions can be summarized as follows:

◇ Based on commercial LSN and cloud computing offerings, we conduct a large-scale measurement study to understand the **access** network performance of LSN. In particular, we identify the scarcity and dynamics of the available uplink resources.
◇ We further carry out extensive **in-the-wild** measurements to examine the achievable LVA performance of the status quo streaming method over today's LSN. Our findings indicate that achieving high-performance LVA streaming remains a challenge.
◇ We propose StarStream, a novel LSN-adaptive LVA streaming framework, to overcome the unique challenges brought by LSN. Specifically, a Transformer-based network performance predictor is designed to adapt the encoding group of pictures (GOP) length to throughput variations while offering throughput predictions

for GOP configuration selection. With the predictions, a content-aware optimizer is further developed to strike a good balance between accuracy and latency via configuration optimization.
◇ Using fine-grained LSN uplink network traces from the real world, we evaluate the predictor's performance and verify its advantages over competitors. By integrating real-world collected video streaming traces, we further examine StarStream's performance. Extensive experiments have validated the framework's effectiveness and superiority.

2 Starlink Network Access: Background and Measurements

Figure 1 shows a typical Starlink residential setup, where a router connects the user equipment (UE) to the LSN, and a dish (antenna) is responsible for communicating with the LEO satellites. Although Starlink has embarked on the deployment of inter-satellite links [30], single *bent-pipe* communication [11, 18] is still the dominant mode. For example, our thousands of traceroute records suggest that most of the communications involve only one *bent-pipe* (one ground-space hop and one space-ground hop) along the path to the destinations, and Starlink tends to serve as an access network to the Internet.

Given the stringent latency requirements of LVA, we are particularly interested in Starlink's access network performance, i.e., the performance of directly consuming data from space in the proximity of GSs. This is in contrast to existing measurements on Starlink Internet access from different global vantage points [11] or from a single vantage point to access globally distributed servers [18, 19]. As such, we set up two servers near the Starlink GS: one from Amazon Web Services, named aws-server, and the other from Google Cloud Platform, named gc-server. We tried to make the servers as close to the LSN (i.e., the Starlink GS) as possible. Specifically, based on the network latency, servers are rented from the cloud regions with the minimum mean round-trip time (RTT) to the UE. Both servers have Gbps bandwidth for inbound and outbound traffic and will not be the bottleneck in the network performance test. We use iPerf3 [10] utility to measure the TCP throughput and Ping utility to measure the RTT. The test is executed every 30 minutes.

Table 1 shows the statistical results of *1,056 tests* conducted over *22 days*. As shown, the mean network latencies between the UE and the servers are decent and comparable to recently reported LTE network results (i.e., 47.6 ± 8.4 ms [9]) but with higher fluctuations. Moreover, Starlink follows a *download-centric* design, where the mean download throughput can be more than 10× higher than the corresponding upload throughput for both servers. Unfortunately, LVA differs from traditional human-oriented video streaming applications in that it uses the uplink rather than the downlink to

---
[2]In this paper, *uplink* refers to the transmission path of sending data from a piece of user equipment to a server, while *downlink* refers to the opposite path.

Table 2: The video dataset used in this paper

Video Name [source link]	Shooting Scenario	Illumination Conditions	Object Speed	Object Size
hw1 [37]	highway	sunny	high	mostly medium
hw2 [39]	highway	sunny	high	diverse
street [40]	street	night	low	mostly large
beach [41]	beach	cloudy	medium	mostly small

transmit heavy and bursty video data. Consequently, *accommodating LVA can significantly challenge Starlink's asymmetric network link design.* In addition, Starlink's mean upload throughput ($\leq$ 8.3 Mbps) is dramatically lower than that of LTE (53.4 Mbps), let alone 5G mmWave (52.8-157.7 Mbps) [9]. With this upload throughput, it can be challenging for Starlink to live stream Ultra High-definition (UHD, 4K or 8K) videos or multiple Full HD (FHD, 1080p) videos simultaneously. For example, YouTube's recommended live streaming bitrate for a 1080p (4K) video is 3-12 Mbps (8-40 Mbps) [38].

We have analyzed the variations in upload throughput across different days and observed no significant daily patterns. We further divided the measurement results into peak hours (7 AM-11 PM) and off-peak hours (11 PM-7 AM) usage and found that, statistically, Starlink provides better network service during off-peak hours than during peak hours. For instance, the mean off-peak upload throughput from the UE to the `aws-server` is 9.2 Mbps, considerably higher than that of peak hours (8.1 Mbps). When we took a closer look at the variations over the course of a day, we found that the upload throughput fluctuates wildly. For example, it can be as high as 16.5 Mbps (15.6 Mbps) and as low as 2.2 Mbps (1.9 Mbps) for the `gc-server` (`aws-server`) within the same day. A finer examination revealed that the upload throughput is highly volatile from second to second. For example, the upload throughput for both servers can vary from 0 to 18+ Mbps within a minute. LEO satellites move faster relative to the Earth due to their low altitudes, which means that the communication distance and link quality between the UE (even a stationary one) and the serving satellite are constantly changing, and frequent handovers from one satellite to another are unavoidable [11, 18]. Thus, we hypothesize that the observed wild performance fluctuations are inherent to the LSN.

3 When LVA meets LSN: A Reality Check

LVA is recognized as a key technical enabler for a wide range of modern applications, from security surveillance and traffic control to self-driving cars and augmented reality [2]. It relies on advanced learning and computer vision algorithms, such as object detection, semantic segmentation, and human key point detection, to analyze live camera feeds, enabling machines to locate, track, classify, and segment video content of interest. The accuracy of LVA is typically preserved by deep neural network (DNN)-based vision models, which are known to be resource-intensive. This popularizes LVA streaming, which involves transmitting live video content from the capturing camera over networks to an analytics server. This is akin to the first-mile ingestion phase of the canonical live video streaming setup, where the live content is transmitted from the capturing device to a streaming server [47]. Real-time messaging protocol (RTMP) is the state-of-the-practice ingestion protocol utilized by
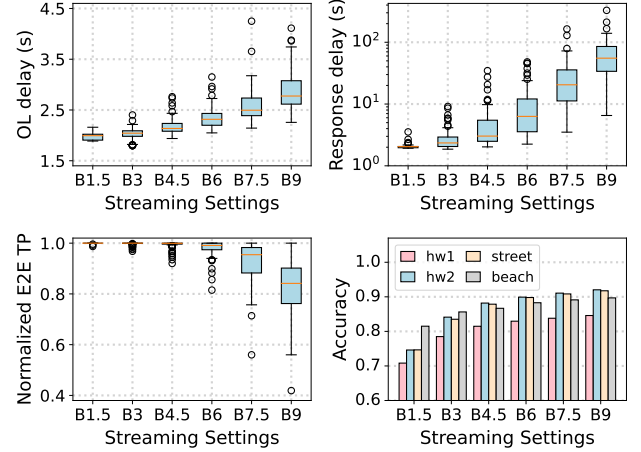
Figure 2: In-the-wild LVA performance over Starlink.

mainstream commercial live streaming platforms such as Twitch [26] and YouTube Live [38], as well as IP cameras [35] for real-time video upload. Therefore, we use RTMP as the streaming protocol to explore in-the-wild LVA performance over today's LSN.

3.1 Measurement Setup

Video dataset. Due to the lack of publicly available video datasets of high quality, high frame rate, and long duration, we use four YouTube videos (details shown in Table 2) as the source to create our video dataset. These source videos cover various scenarios commonly encountered in LVA applications, such as highways, urban environments, and rural areas. They also incorporate a broad spectrum of content characteristics and dynamics. Specifically, for each video source, we extract a 480-second video clip and configure the client to read 1080p frames from the clip at a rate of 15 frames per second (FPS), thereby simulating a scenario where the frames are being captured in real time by a 1080p, 15 FPS camera.

Client and Server. We choose the `aws-server`, a `p3.2xlarge` EC2 instance with an NVIDIA V100 GPU [3], as the analytics server, and an off-the-shelf desktop as the Starlink UE.

Server-side vision task and model. We consider one basic video analytics task, object detection, as the server-side analytics task for proof of concept. In particular, the task is executed by a pre-trained DNN model, `YOLOv8l`, from the YOLOv8 family [27].

Methodology. By default, the H.264 codec is employed to compress raw video frames using a constant bitrate (CBR) encoding scheme, with a keyframe interval of 2 seconds. We explore target encoding bitrates of 1.5, 3, 4.5, 6, 7.5, and 9 Mbps, denoted respectively as `B1.5`, `B3`, etc. In addition to the encoding bitrate, LVA accuracy can also be affected by other encoding parameters, such as resolution and frame rate. Therefore, we evaluate a series of candidate frame rates {1, 3, 5, 15} and resolutions {1920×1080, 1280×720, 640×320} for each target bitrate. We then report the measurement results of the (frame rate, resolution) combination that yields the highest accuracy. For implementation, the client uses the `FFmpeg` C++ libraries [7] to encode and stream frames at the target bitrate, frame rate, and resolution. To further minimize the encoding and streaming latency, we set the encoder to the *ultrafast and zerolatency* mode [8], where
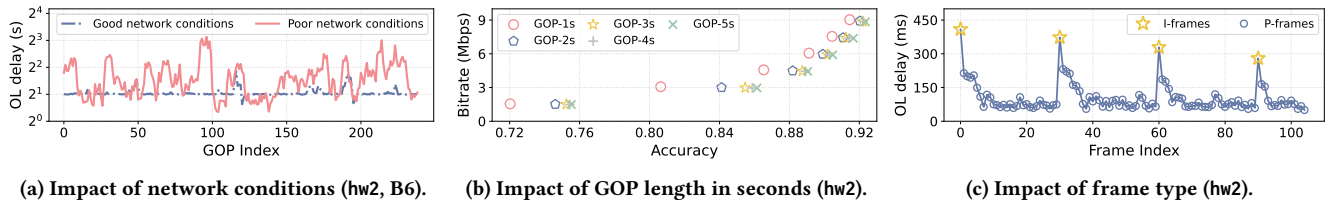
(a) Impact of network conditions (hw2, B6).   (b) Impact of GOP length in seconds (hw2).   (c) Impact of frame type (hw2).

Figure 3: In-depth examination of performance influencing factors.

only I-frames and P-frames are encoded. The server also uses the `FFmpeg` libraries to listen for stream events, receive streams, and decode the compressed video packets into uncompressed frames.
Metrics. We consider the following performance metrics.

◇ *Offloading delay (OL delay).* This measures the time it takes for frames to be encoded by the client, transmitted over the network, and decoded by the server. Specifically, a frame's offloading delay is the elapsed time from when the client starts encoding the frame to when it is prepared for analysis on the server. A GOP's offloading delay is the duration from when the client starts encoding its first frame to when the server finishes decoding its last frame. The average offloading delay of all GOPs within a video is considered as the video's offloading delay.

◇ *Response delay.* A GOP's response delay is the total time elapsed from when the first frame of a GOP is captured to when the analysis results of the last frame in the GOP become available. It includes delays caused by potential queuing, network delivery, and server-side model inference. The average response delay across all GOPs within a video is the video's response delay.

◇ *Normalized end-to-end throughput (E2E TP).* Assume that the entire *capture-streaming-analysis* pipeline transmits and analyzes n frames in t seconds, i.e., the time elapsed from when the camera captures the first frame to when the server finishes analyzing the nth frame is t seconds. Then, the normalized E2E TP is $n/(t \cdot f)$, where f is the frame rate of the stream.

◇ *Accuracy.* Following the best practice in previous studies [14, 42], we regard the detection results of the same model on raw frames as ground truth to eliminate the inaccuracy caused by the object detection model itself. Accuracy is then calculated as the F1 score between the predicted and ground-truth results, as in [42]. A true positive is accepted when the intersection over the union between a predicted bounding box and a ground-truth bounding box is greater than 0.5, and their object categories are the same.

3.2 Measurement Results and Insights

We stream each video under each setting at different times of the day for more than *10 days*. For all settings, the mean frame encoding delay on the client is about 15.83 ms, and the mean frame decoding delay on the server is about 3.73 ms. The server-side model inference delay for the highest resolution (i.e., 1920×1080) is about 62.01 ms on average. This means that the frame encoding, decoding, and model inference can all run at a speed exceeding 15 FPS, i.e., the maximum frame rate. Thus, any delays affecting real-time analytics can be primarily attributed to network transmission.

Figure 2 presents the statistical results of 20 trials conducted in the wild for each video-setting pair. As the offloading delay subfigure suggests, higher streaming bitrates introduce higher delays

during network transmission. When the streaming bitrate exceeds the capacity of the LSN, the offloading delay becomes the latency bottleneck in the entire processing pipeline. This makes it struggle to keep up with the frame arrival rate, leading to progressively accumulating lags. These lags eventually result in exponentially increasing response delays. Overall, it is still challenging for the LSN to support real-time LVA streaming (i.e., normalized E2E TP is 1.0) at bitrates higher than 6 Mbps. While meeting real-time requirements at low to medium bitrates seems promising, the resulting analytics accuracy can be significantly compromised. As a result, simultaneously achieving the low latency and high accuracy goals of LVA remains an open problem over today's LSN.

Another observation from Figure 2 is that the delay-related metrics vary significantly across multiple trials under identical streaming settings. For example, the response delay of video hw2 under setting B6 can be 2.49 seconds in one trial and 48.10 seconds in another, indicating a performance difference of 19.32×. Since the only variable between the trials is the underlying network conditions, this observation reveals the significant influence of the LSN performance on LVA-perceived QoE. *This calls for LSN-adaptive streaming solutions for LVA applications to provide consistent QoE.* Figure 3a further details the GOP offloading delay variations observed in these two trials. As shown, the offloading delay remains stable with only occasional small fluctuations under the good network conditions, suggesting that a coarse-grained adaptation strategy may be adequate. In contrast, under the poor network conditions, the offloading delay fluctuates wildly and frequently, necessitating a more nuanced, fine-grained adaptation approach. These findings uncover the challenges inherent in application-level LSN adaptation and motivate the design of *granularity-variant* adaptation strategies.

GOP is the natural encoding unit for bitrate control and network adaptation, given its self-contained structure. We thus fix all the other variables and vary only the GOP length to investigate its impact on accuracy. Figure 3b shows that for the same target encoding bitrate, increasing GOP length can improve the analytics accuracy, and the improvements are particularly obvious for low target bitrates. We identify that this is because a shorter GOP results in a higher frequency of I-frames. For a given bitrate budget, the codec has to increase the mean quantization parameter to accommodate more relatively large I-frames, ultimately leading to diminished overall image quality that adversely affects accuracy. Additionally, since a P-frame has to wait for its reference frame to arrive before it can be decoded at the server, the large size of an I-frame affects not only its own offloading delay but also the offloading delay of its subsequent P-frames as shown in Figure 3c. This observation encourages the adoption of longer GOP lengths, when network conditions permit, to benefit both accuracy and delay stability.
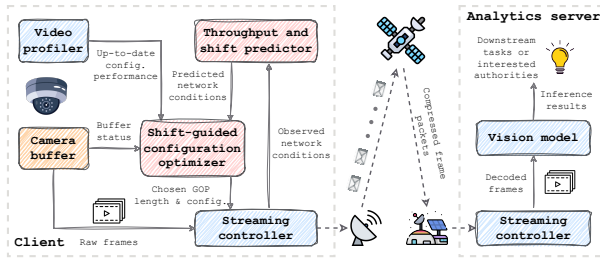
Figure 4: Overview of StarStream.

4 Toward LSN-Adaptive LVA Streaming

Inspired by our measurement insights, we propose StarStream, a novel adaptive LVA streaming framework specifically designed for LSNs. Figure 4 presents an overview of StarStream, where the client continually streams captured frames over LSN to an analytics server. The client adapts video encoding GOP lengths and other configurations, such as bitrate, to the dynamic uplink network conditions for both accuracy and latency optimization. The core adaptation component is the *shift-guided configuration optimizer*, which decides how to encode and stream incoming video content according to predicted network throughputs and their shifts, up-to-date configuration performance, and camera buffer status, to strike a balance between accuracy and latency.

Specifically, the future network conditions are derived from the *throughput and shift predictor*, which predicts not only the future upload throughputs but also the timing of significant changes in throughput. The up-to-date configuration performance is estimated based on the information provided by the *video profiler*, which estimates the reference configuration performance by offline profiling representative frames and captures recent content characteristics by online compact model inference. The camera buffer status is obtained by querying the *camera buffer*, which caches captured video frames that cannot be promptly transmitted due to poor network conditions. After the *shift-guided configuration optimizer* makes the encoding configuration decision, the *streaming controller* component will encode and stream the corresponding frames according to the chosen GOP length and configuration.

4.1 LSN Uplink Performance Prediction

Network throughput prediction serves as a foundational step in building network-adaptive multimedia applications. A multitude of throughput predictors based on historical observations (e.g., harmonic mean [36] and moving average [15]), classical machine learning models (e.g., decision tree [16] and random forest [1]), or deep learning models (e.g., fully connected network [34], long short-term memory [12], and Seq2seq [20]), have been developed for various terrestrial networks.

Compared to terrestrial networks, throughput prediction for LSN is much more challenging due to its inherent characteristics, e.g., vulnerabilities to exogenous factors. Recent years have witnessed the success of time series prediction models based on *Transformers* [28], which have outperformed other time series prediction methods in various domains [31]. Their strengths in capturing long-term

dependencies, modeling complex temporal interactions, and easily incorporating external information align well with the properties of an ideal uplink performance predictor for LSN. As a result, we propose a Transformer-based throughput and shift predictor to fully utilize the uplink resources of LSN while mitigating its challenges. In particular, the predictor's design is based on Informer [46], an efficient Transformer-based time series forecasting model, but with unique designs tailored to the characteristics of LSN.

Given the notable instability in LSN's uplink performance, understanding the timing of significant throughput changes can help make judicious decisions. For instance, if the throughput is predicted to remain stable in a future time horizon, we can choose a long GOP length to enhance accuracy. Conversely, if the throughput is expected to experience frequent variations, we can choose a short GOP length to allow for flexible configuration adjustments at GOP boundaries, thereby accommodating fine-grained throughput variations. As such, aside from a throughput prediction head, the proposed predictor also integrates a *shift prediction head*, as shown in Figure 5. Formally, let b_t denote the throughput at time step t. We define that a throughput shift occurs at time step t if the difference between b_t and b_{t-1} is greater than a predefined threshold δ. The *shift prediction head* then outputs a binary shift indicator for each future time step, indicating whether a throughput shift is expected to occur. Both prediction heads are attached directly to the output layer of the Informer decoder.

As Figure 5 shows, the proposed predictor takes a sequence of network observations and their corresponding timestamps from time step $t - m + 1$ to t as input, to predict the future network performance from time step $t + 1$ to $t + n$. Specifically, the Informer encoder receives the entire input sequence, while the decoder's input is crafted by concatenating two sequences: one is a subsequence of the input sequence, starting from time step $t - p + 1$ to time step t (where $p \leq m$), and the other is the target sequence to be predicted, where unknown values are padded with zeros. Note that unlike traditional encoder-decoder style models that recursively generate the output for each time step one at a time, the decoder generates outputs for all time steps at once in a generative way.

In addition to the positional embedding that encodes the relative position information within the input sequence, the input to the predictor also integrates the outputs of the following three embedding layers. *1) Observable variables (OV) embedding layer:* This layer embeds observable variables for network performance prediction. Apart from the historical observations of throughput and its shifts, we also consider variables related to the underlying TCP connection, such as the retransmit times, the sending congestion window size, the smoothed RTT estimate, and the RTT variation, given their proven effectiveness in throughput prediction [16, 34]. *2) Date embedding layer:* Our measurements confirm that the wall-clock time can have certain influences on the LSN network performance (recall the peak and off-peak hours performance in §2). Consequently, the date embedding layer is introduced to encode the global time information. *3) Handover embedding layer:* Starlink schedules satellite-UE associations every 15 seconds, suggesting that handovers may occur as frequently as every 15 seconds [5, 24]. To account for the potential influence of handovers on network performance, a handover embedding layer is used to encode the current second's position in the 15-second scheduling window.
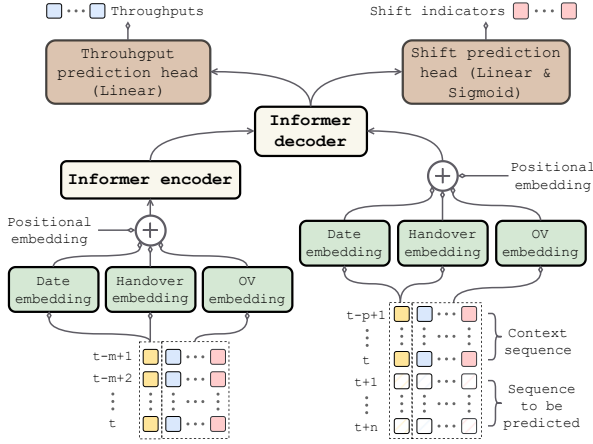
Figure 5: Architecture of the proposed predictor.

4.2 Shift-Guided Configuration Optimization

GOP Length Selection. In our problem, the GOP length determines the encoding and decision-making granularity. Instead of setting the GOP length to a fixed value, `StarStream`'s configuration optimizer dynamically adjusts the GOP length under the guidance of the predicted throughput shift indicators. For instance, assume that the interval between two consecutive time steps is one second, and the lookahead window size n is 3. If the predicted throughputs and shift indicators are $[\hat{b}_{t+1}, \hat{b}_{t+2}, \hat{b}_{t+3}]$ and $[0, 0, 1]$, respectively, the chosen GOP length for the next GOP will be 2 seconds, and the corresponding predicted throughput for the GOP will be calculated as $(\hat{b}_{t+1} + \hat{b}_{t+2})/2$. Then, with the chosen GOP length and the mean predicted throughput, the optimizer further chooses the encoding configuration (e.g., bitrate) for the GOP.

Problem Formulation. The performance goals of configuration selection are to maximize the analytics accuracy and minimize the lag, where lag is defined by the amount of queued-up frames that wait to be processed [43]. Without loss of generality, we assume that for the entire "capture-encoding-transmission-decoding-inference" processing pipeline, the primary cause of lag is network transmission. In continuous streaming scenarios, if the system's E2E processing speed can keep up with the frame capture speed, there will be no lag; otherwise, the newly captured frames will queue up at the camera buffer, resulting in an accumulation of lag. We then quantify the lag using the camera buffer queue length and formulate the performance optimization problem from GOP $k1$ to GOP $k2$ ($k1 \leq k2$) as follows:

$$\underset{c_{k1},\cdots,c_{k2}}{\arg\max} \quad \sum_{k=k1}^{k2} \alpha A_k(c_k) - \beta Q_k$$

$$s.t. \quad \begin{cases} \bar{b}_k = \frac{1}{t_k - t_{k-1}} \int_{t_{k-1}}^{t_k} b_t \, dt \\ t_k = t_{k-1} + \sum_j e_j(c_k) + \frac{\sum_j d_j(c_k)}{\bar{b}_k} + \Delta t_k \\ Q_k = Q_{k-1} + (t_k - t_{k-1}) - L_k \\ c_k \in C, \quad \forall k = k1, \cdots, k2 \end{cases} \quad (1)$$

where $A_k(c_k)$ is the server-side analytics accuracy for GOP k encoded with configuration c_k. Q_k is the camera buffer queue length (in seconds) when the client finishes transmitting the last frame in GOP k. α and β are positive weighting parameters used to trade off accuracy against lag. t_k denotes the time when the client finishes transmitting the last frame of GOP k. $e_j(c_k)$ and $d_j(c_k)$ are the encoding delay and the compressed frame size of the jth frame in GOP k encoded with configuration c_k, respectively. Note that in live streaming scenarios, frames are compressed and transmitted sequentially as they are captured. This means that the frame compression cannot be completed in advance; instead, compression and transmission occur in an interleaved manner. $\bar{b}_k$ is the average upload throughput between t_{k-1} and t_k. Δt_k denotes the total wait time between t_{k-1} and t_k. A wait can happen when the frame upload speed is faster than the frame capture speed, and the client has to wait for the next frame to arrive to resume processing. L_k is the chosen GOP length, and C is the candidate configuration set.

Content-Aware Configuration Performance Estimation. Since we use CBR encoding for video streaming [38], the data sizes and encoding delays of GOPs with the same length and configuration will not have significant differences. Thus, the *video profiler* component estimates $e_j(c_k)$ and $d_j(c_k)$ by offline profiling representative video content. The key challenge here is estimating $A_k(c_k)$, which is known to be related to video content dynamics. Existing solutions tend to combine offline and online profiling to adapt to changing video content [42]. However, online profiling requires sending raw frames to the server to acquire the ground truth results for accuracy calculation. This places a heavy burden on network transmission, preventing its practical use in LSNs with scarce uplink resources.

Let $A(c)$ denote the offline profiled accuracy of configuration c. As the video content varies, using $A(c)$ to estimate the actual configuration accuracy may result in either an overestimate or an underestimate, depending on the relative analysis difficulty of the current content compared to the offline profiled content. The immediate consequence of such inaccurate estimates on our optimization objective is that they bias the accuracy-lag tradeoff and configuration comparisons. Our workaround is to define a proxy for the *relative content analysis difficulty*, namely γ, and use it to scale $A(c)$, thereby minimizing bias. Specifically, when the current content becomes more difficult to analyze than the profiled content, γ will take on a value greater than 1, increasing the accuracy gaps between configurations and placing greater emphasis on accuracy relative to lag. Conversely, when the current content is easier to analyze, γ will be less than 1. The rationale is as follows: for easy-to-analyze content, the accuracy gaps between different configurations are small, while for challenging content, the gaps are significant.

To update γ, the *video profiler* runs a compact model (`YOLOv8n` by default) online to periodically analyze newly captured frames. The analysis difficulty of new content is then inferred from the *confidence scores* output by the compact model. Intuitively, if the content is harder to analyze, there will be more detections with low confidence scores, indicating that more information is needed to ensure high accuracy. Specifically, a detection is considered uncertain if its confidence score is lower than 0.5. We then define an uncertainty metric u, calculated as the ratio of the number of uncertain detections to the total number of detections. Let u_n denote

the uncertainty of new content and u_p denote the uncertainty of the profiled content. With new content analyzed by the compact model, γ is accordingly updated as u_n/u_p.

Configuration Optimization to Balance Accuracy and Lag. Since the camera captures frames at a constant frame rate, the future frame capture times can be known in advance. Consequently, once the future upload throughputs and video-related variables are determined, Δt_k can also be determined. This makes us ready to solve Problem (1). However, as relatively accurate predictions can only be guaranteed for a short future time horizon, the *optimizer* solves the problem over a finite time horizon (3 GOPs by default), following the model predictive control (MPC) [34, 36] paradigm.

In practice, evaluating every possible combination of frame rate, resolution, and bitrate can significantly prolong the online configuration optimization time. Fortunately, we analyzed the accuracy variations of all configurations and found that for all given bitrates, the best-performing (frame rate, resolution) combination is always one of three candidates, and their overall accuracies are very similar for a given bitrate. Considering the additional communication cost of changing the frame rate and resolution in the middle of the stream, we propose a profiling-based configuration pruning method. Using the offline profiling results provided by the *video profiler*, it simply selects the (frame rate, resolution) combination that most frequently hits the top-3 performance under all candidate bitrates as the frame rate and resolution for online streaming. As such, the optimizer only needs to choose the bitrate for each GOP online. We further design an efficient dynamic programming (DP) algorithm to solve the optimization problem.

5 Evaluation

5.1 Evaluation of Network Predictor

Network Traces. We collected real-world LSN upload traces with the same client and server setup introduced in §3.1. The dataset comprises 504 network traces measured from two different geographical locations, collected at different times over 17 days across two years and covering various weather conditions. Each trace has a duration of 10 minutes with a granularity of 1 second, including timestamps, upload throughput, and TCP connection information. We also added a *shift* column to indicate whether a shift occurs, with δ set to 2.5 Mbps. The dataset is randomly divided into a training set (70%), a validation set (10%), and a testing set (20%).

Baselines. We consider the following methods: harmonic mean (HM), moving average (MA), random forest (RF), fully connected network (FCN), long short-term memory (LSTM), and Seq2seq. Since these methods are designed to only predict the throughput, we calculate the throughput shift indicators by differencing the predicted throughputs and then comparing the differences to δ.

Evaluation metrics. We use Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), Mean Absolute Percentage Error (MAPE), and R-Squared (R^2) to evaluate throughput prediction, while using Accuracy and F1 score for shift indicator prediction. For MAE, RMSE, and MAPE, smaller values are better, while larger values are better for the other metrics.

Result Analysis. Table 3 presents the performance of different predictors, where the context sequence length p of our method is 15. As shown, there exists a large performance gap between the two

Table 3: Comparison of different network predictors (lookback window size $m = 60$; lookahead window size $n = 15$).

Methods	Throughput				Shift indicator	
	MAE	RMSE	MAPE	R^2	Accuracy	F1
HM [36]	4.019	5.275	57.095	−0.709	0.670	0.074
MA [15]	3.166	4.045	52.173	−0.005	0.671	0.065
RF [1]	2.577	3.388	42.695	0.295	0.682	0.025
FCN [34]	2.493	3.302	41.042	0.330	0.684	0.040
LSTM [12]	2.472	3.281	40.513	0.339	0.684	0.041
Seq2seq [20]	2.463	3.274	40.383	0.342	0.685	0.053
Ours	2.435	3.248	39.244	0.352	0.706	0.467

naive prediction methods (i.e., HM and MA) and the other methods. This implies that simple models relying on historical throughput information can hardly capture the complex variations in LSN upload throughput. RF, FCN, and LSTM are able to capture the complex relationships between input features and hit competitive performance. However, they treat the throughput at different future time steps equally as output features to be predicted and may not fully exploit the temporal relationships between multi-step outputs. Seq2seq overcomes this deficiency by using a decoder to recursively predict the output at each time step and achieves better performance.

Our predictor achieves the best performance on all evaluation metrics. This can be attributed to the introduction of the attention mechanism and embeddings of exogenous information. Additionally, the performance of using predicted throughputs to estimate the throughput shifts is notably worse. We find the reason is that prediction models tend to generate *smoothed* multi-step throughput forecasts, which renders the shift indicator rarely equal to 1. In practical deployments, continual learning [29] can be integrated into our framework to maintain the predictor's performance as network conditions evolve.

5.2 Evaluation of StarStream

Methodology. We use the videos shown in Table 2 for StarStream's performance evaluation. We consider the same frame rate, resolution, and bitrate candidates as in §3.1, and 5 GOP length candidates: {1, 2, 3, 4, 5} seconds. In the offline stage, the *video profiler* profiles the first 20-second of each video to obtain each configuration's performance and processing costs (including encoding/decoding/inference delays and compressed frame sizes). Then, it selects the streaming frame rate and resolution for each video based on the proposed pruning method. γ is updated every 30 seconds online by profiling 5 seconds of newly captured video content. For a fair comparison, we implement a trace-driven simulator based on the test network traces introduced in §5.1 and video processing traces collected offline. Specifically, we encode and stream each video with different configurations and GOP lengths from the client to the server (same as that in §3.1), while recording the corresponding compressed frame sizes, encoding delays, decoding delays, and inference delays, to construct the video processing traces dataset. The traces for the same video are well aligned to facilitate flexible GOP and configuration switching during the online stage. By default, α and β in Problem (1) are set to 1 and 0.02, respectively.

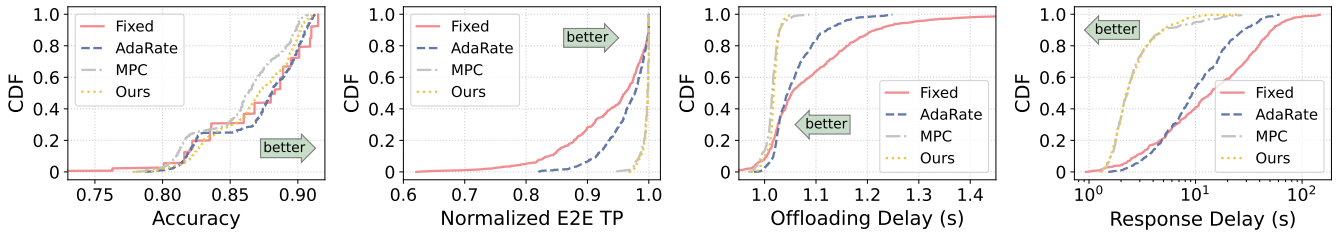**Baselines.** All of the baselines use a fixed GOP length of 2 seconds.

Miao Zhang, Jiaxing Li, Haoyuan Zhao, Linfeng Shen, and Jiangchuan Liu

Figure 6: Overall performance of different solutions. The CDF figures show statistics on all video-trace pairs.

◇ `Fixed`: This non-adaptive solution streams videos at a fixed bitrate, selected as the highest bitrate that is below the mean throughput observed 1 minute before the streaming begins.

◇ `AdaRate`: This is a pure rate-based adaptive streaming solution. It employs the same network predictor as `StarStream` but simply chooses the maximum bitrate below the predicted throughput.

◇ `MPC`: It uses MPC [34, 36] to optimize the objective of Problem (1) over 3 future GOPs. The future throughputs are estimated using harmonic means of past 5 GOPs. The video-related variables are estimated by offline profiling the first 20-second video content.

Performance Metrics. We consider the *accuracy, normalized E2E TP, OL delay* and *response delay* defined in §3.1 as the metrics. Note that since different videos can be streamed at different frame rates and the GOP length can also change within a stream, the OL delay and response delay in this evaluation are uniformly defined for per-second video content, rather than for per frame or GOP.

Performance Improvement. Figure 6 shows the overall performance of different methods. As shown, `Fixed` cannot achieve real-time E2E processing (i.e., normalized E2E TP is 1.0) for most video-trace pairs because this rigid method cannot adapt to the ever-changing network conditions. In comparison, `AdaRate` improves the overall normalized E2E TP, OL delay, and response delay by dynamically adjusting each GOP's bitrate based on the predicted network throughput. Yet, it inevitably suffers from performance degradation due to imperfect predictions. Also, it has no mechanism to automatically recover from the previously made bad decisions, which can lead to accumulated lags that eventually reduce the throughput and increase delays. By integrating the network predictions along with the camera buffer queue status, both `MPC` and `StarStream` achieve near real-time E2E processing for almost all video-trace pairs. By strategically optimizing over multiple future GOPs to balance accuracy and lag, these methods can timely recover from previously made bad decisions and control the response delays within a reasonable range (i.e., < 10 seconds). `StarStream` further benefits from the more flexible GOP length configuration and more accurate configuration accuracy estimation, achieving noticeable accuracy improvements over `MPC`.

Ablation Study. We consider two variants: one disables the γ updates, named V1, and the other replaces the network predictor with a Seq2seq predictor, named V2. We find that, compared to `StarStream`, the mean response delay of V1 increases by 8.6% while the accuracy remains almost the same, and the mean response delay of V2 increases by 10.1% while the mean accuracy decreases from 0.867 to 0.864. V1 is unaware of configuration performance fluctuations caused by varying video content and misses many opportunities to choose a low-latency configuration with decent

accuracy. V2 cannot accurately predict when throughput shifts will happen and tends to choose longer GOP lengths that cannot adapt to second-to-second throughput variations. This is exacerbated by less accurate throughput predictions, leading to sub-optimal configuration choices. This confirms the effectiveness of `StarStream`'s key components.

System Overheads. The system overheads primarily stem from the predictor, the DP algorithm, and the online content profiling. We benchmark these overheads using our measurement client, equipped with a 6-core Intel i7-6850K CPU and an Nvidia GeForce GTX 1080 Ti GPU. Notably, the DP algorithm is highly efficient, solving the problem in just 0.63 ± 0.35 ms on the CPU. Profiling 5-second raw frames on the GPU takes about 1.44 seconds, and the fast profiling speed ensures that the profiled video content is fresh. The inference delay of the predictor on the GPU is about 13.0 ± 5.1 ms, which is insignificant, especially given that network prediction and video streaming can be conducted in parallel in our design.

6 Related Work

Shipping video data over networks for real-time online analytics has been a hot research topic in recent years [13–15, 21, 42, 44]. To handle the tremendous bandwidth requirements, various content-driven, model-driven, and compression-driven strategies have been proposed to reduce network data transfer without significantly compromising accuracy. These strategies include controlling video encoding parameters (e.g., resolution) [42], frame filtering [14], frame masking [15], and frame partitioning [44]. While promising, most of them prioritize reducing network consumption over network adaptation since they are designed for relatively stable terrestrial networks where adaptation is not always necessary. Network-adaptive streaming solutions like `AWStream` [42] exist but rely on simple network probing and bandwidth estimation methods that are insufficient for LSNs. In contrast, the core designs of `StarStream` are network-driven, and to the best of our knowledge, it is the first attempt to bridge the gap between LSNs and existing LVA solutions.

7 Conclusion

This paper closely investigated the performance of emerging LSNs in supporting LVA applications. Through extensive measurements, we found that the uplink resources of today's LSNs are still scarce, and their wild network performance fluctuations can prevent upper-level LVA applications from providing consistently satisfactory QoE. We further proposed a novel adaptive LVA streaming framework, `StarStream`, to improve the performance of LSN-enabled LVA applications. Extensive trace-driven experiments verified the effectiveness and superiority of our proposed solution.

Acknowledgments

References

[1] E. Alimpertis, A. Markopoulou, C. Butts, and K. Psounis. 2019. City-Wide Signal Strength Maps: Prediction with Random Forests. In *The World Wide Web Conference (WWW'19)*. 2536–2542.

[2] G. Ananthanarayanan, P. Bahl, P. Bodík, K. Chintalapudi, M. Philipose, L. Ravindranath, and S. Sinha. 2017. Real-Time Video Analytics: The Killer App for Edge Computing. *IEEE Computer* 50, 10 (2017), 58–67.

[3] AWS. 2024. Amazon EC2 P3 Instances. https://aws.amazon.com/ec2/instance-types/p3/. [Online; accessed 6 August 2024].

[4] C. Canel, T. Kim, G. Zhou, C. Li, H. Lim, D. G. Andersen, M. Kaminsky, and S. R. Dulloor. 2019. Scaling Video Analytics on Constrained Edge Nodes. In *Proceedings of the 2nd SysML Conference (SysML'19)*.

[5] X. Cao and X. Zhang. 2023. SaTCP: Link-Layer Informed TCP Adaptation for Highly Dynamic LEO Satellite Networks. In *IEEE Conference on Computer Communications (INFOCOM'23)*. IEEE, 1–10.

[6] CNBC. 2021. Google wins cloud deal from Elon Musk's SpaceX for Starlink internet connectivity. https://www.cnbc.com/2021/05/13/google-cloud-wins-spacex-deal-for-starlink-internet-connectivity.html. [Online; accessed 6 August 2024].

[7] FFmpeg. 2022. FFmpeg Documentation. https://ffmpeg.org/doxygen/4.4/index.html. [Online; accessed 6 August 2024].

[8] FFmpeg. 2023. H.264 Video Encoding Guide. https://trac.ffmpeg.org/wiki/Encode/H.264. [Online; accessed 6 August 2024].

[9] M. Ghoshal, Z J. Kong, Q. Xu, Z. Lu, S. Aggarwal, I. Khan, Y. Li, Y C. Hu, and D. Koutsonikolas. 2022. An In-Depth Study of Uplink Performance of 5G mmWave Networks. In *Proceedings of the ACM SIGCOMM Workshop on 5G and Beyond Network Measurements, Modeling, and Use Cases (5G-MeMU'22)*. 29–35.

[10] iPerf. 2024. iPerf - The ultimate speed test tool for TCP, UDP and SCTP. https://iperf.fr/iperf-download.php. [Online; accessed 6 August 2024].

[11] M. M Kassem, A. Raman, D. Perino, and N. Sastry. 2022. A Browser-side View of Starlink Connectivity. In *Proceedings of the 22nd ACM Internet Measurement Conference (IMC'22)*. 151–158.

[12] J. Lee, S. Lee, J. Lee, S. Sathyanarayana, H. Lim, J. Lee, X. Zhu, S. Ramakrishnan, D. Grunwald, K. Lee, et al. 2020. PERCEIVE: Deep Learning-based Cellular Uplink Prediction Using Real-Time Scheduling Patterns. In *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services (MobiSys'20)*. 377–390.

[13] C. Li, R.-X. Zhang, T. Huang, L. Jia, and L. Sun. 2023. Concerto: Client-server Orchestration for Real-Time Video Analytics. In *Proceedings of the 31st ACM International Conference on Multimedia (MM'23)*. 9215–9223.

[14] Y. Li, A. Padmanabhan, P. Zhao, Y. Wang, G. H. Xu, and R. Netravali. 2020. Reducto: On-Camera Filtering for Resource-Efficient Real-Time Video Analytics. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication (SIGCOMM'20)*. 359–376.

[15] S. Liu, T. Wang, J. Li, D. Sun, M. Srivastava, and T. Abdelzaher. 2022. AdaMask: Enabling Machine-Centric Video Streaming with Adaptive Frame Masking for DNN Inference Offloading. In *Proceedings of the 30th ACM International Conference on Multimedia (MM'22)*. 3035–3044.

[16] G. Lv, Q. Wu, W. Wang, Z. Li, and G. Xie. 2022. Lumos: Towards Better Video Streaming QoE through Accurate Throughput Prediction. In *IEEE Conference on Computer Communications (INFOCOM'22)*. IEEE, 650–659.

[17] S. Ma, Y. C. Chou, M. Zhang, H. Fang, H. Zhao, J. Liu, and W. I Atlas. 2024. LEO Satellite Network Access in the Wild: Potentials, Experiences, and Challenges. *IEEE Network* (2024).

[18] S. Ma, Y. C. Chou, H. Zhao, L. Chen, X. Ma, and J. Liu. 2023. Network Characteristics of LEO Satellite Constellations: A Starlink-Based Measurement from End Users. In *IEEE Conference on Computer Communications (INFOCOM'23)*. IEEE, 1–10.

[19] F. Michel, M. Trevisan, D. Giordano, and O. Bonaventure. 2022. A First Look at Starlink Performance. In *Proceedings of the 22nd ACM Internet Measurement Conference (IMC'22)*. 130–136.

[20] A. Narayanan, E. Ramadan, R. Mehta, X. Hu, Q. Liu, R. A. Fezeu, U. K. Dayalan, S. Verma, P. Ji, T. Li, et al. 2020. Lumos5G: Mapping and Predicting Commercial mmWave 5G Throughput. In *Proceedings of the ACM Internet Measurement Conference (IMC'20)*. 176–193.

[21] A. Padmanabhan, N. Agarwal, A. Iyer, G. Ananthanarayanan, Y. Shu, N. Karianakis, G. H. Xu, and R. Netravali. 2023. Gemel: Model Merging for Memory-Efficient, Real-Time Video Analytics at the Edge. In *Proceedings of the 20th USENIX Symposium on Networked Systems Design and Implementation (NSDI'23)*. 973–994.

[22] SpaceX. 2024. High-Speed Internet by SpaceX. https://www.starlink.com/. [Online; accessed 6 August 2024].

[23] Starlink. 2024. How Starlink Works. https://www.starlink.com/technology. [Online; accessed 6 August 2024].

[24] H. B. Tanveer, M. Puchol, R. Singh, A. Bianchi, and R. Nithyanand. 2023. Making Sense of Constellations: Methodologies for Understanding Starlink's Scheduling Algorithms. In *Companion of the 19th International Conference on Emerging Networking EXperiments and Technologies (CoNEXT'23)*. 37–43.

[25] TelecomReview. 2024. The Race for Space Internet Builds Up Pace. https://www.telecomreview.com/articles/reports-and-coverage/7713-the-race-for-space-internet-builds-up-pace. [Online; accessed 6 August 2024].

[26] Twitch. 2024. Broadcast Guidelines. https://help.twitch.tv/s/article/broadcast-guidelines. [Online; accessed 6 August 2024].

[27] Ultralytics. 2024. YOLOv8: A New State-of-the-Art Computer Vision Model. https://yolov8.com/. [Online; accessed 6 August 2024].

[28] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A N Gomez, Ł. Kaiser, and I. Polosukhin. 2017. Attention Is All You Need. *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS'17)* (2017).

[29] L. Wang, X. Zhang, H. Su, and J. Zhu. 2024. A Comprehensive Survey of Continual Learning: Theory, Method and Application. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 46, 8 (2024), 5362–5383.

[30] WCCFtech. 2022. Starlink Turns On Laser Satellites For Region With Four Months Long Night. https://wccftech.com/starlink-turns-on-laser-satellites-for-region-with-fourth-month-long-night/. [Online; accessed 6 August 2024].

[31] Q. Wen, T. Zhou, C. Zhang, W. Chen, Z. Ma, J. Yan, and L. Sun. 2023. Transformers in Time Series: A Survey. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence (IJCAI'23)*. 6778–6786.

[32] Wikipedia. 2024. List of Starlink and Starshield launches. https://en.wikipedia.org/wiki/List_of_Starlink_and_Starshield_launches. [Online; accessed 6 August 2024].

[33] Wikipedia. 2024. Starlink. https://en.wikipedia.org/wiki/Starlink. [Online; accessed 6 August 2024].

[34] F. Yan, H. Ayers, C. Zhu, S. Fouladi, J. Hong, K. Zhang, P. Levis, and K. Winstein. 2020. Learning in situ: a randomized experiment in video streaming.. In *Proceedings of the 17th USENIX Symposium on Networked Systems Design and Implementation (NSDI'20)*, Vol. 20. 495–511.

[35] J. Yi, M. R. Islam, S. Aggarwal, D. Koutsonikolas, Y C. Hu, and Z. Yan. 2020. An Analysis of Delay in Live 360° Video Streaming Systems. In *Proceedings of the 28th ACM International Conference on Multimedia (MM'20)*. 982–990.

[36] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli. 2015. A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication (SIGCOMM'15)*. 325–338.

[37] YouTube. 2024. 4K Road traffic video for object detection and tracking. https://www.youtube.com/watch?v=QuUxHIVUoaY. [Online; accessed 20 March 2024].

[38] YouTube. 2024. Choose live encoder settings, bitrates, and resolutions. https://support.google.com/youtube/answer/2853702. [Online; accessed 6 August 2024].

[39] YouTube. 2024. Highway sounds cars trucks passing by for sleeping relaxing. https://www.youtube.com/watch?v=TW3EH4cnFZo. [Online; accessed 20 March 2024].

[40] YouTube. 2024. Live Street Webcam. https://www.youtube.com/watch?v=VR-x3HdhKLQ. [Online; accessed 20 March 2024].

[41] YouTube. 2024. Mori Point, Pacifica CA 4K Live. https://www.youtube.com/watch?v=xf0NCn-1rws. [Online; accessed 20 March 2024].

[42] B. Zhang, X. Jin, S. Ratnasamy, J. Wawrzynek, and E. A Lee. 2018. AWStream: Adaptive Wide-Area Streaming Analytics. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication (SIGCOMM'18)*. 236–252.

[43] H. Zhang, G. Ananthanarayanan, P. Bodik, M. Philipose, P. Bahl, and M. J. Freedman. 2017. Live Video Analytics at Scale with Approximation and Delay-Tolerance. In *Proceedings of the 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI'17)*. 377–392.

[44] W. Zhang, Z. He, L. Liu, Z. Jia, Y. Liu, M. Gruteser, D. Raychaudhuri, and Y. Zhang. 2021. ELF: Accelerate High-resolution Mobile Deep Vision with Content-aware Parallel Offloading. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking (MobiCom'21)*. 201–214.

[45] H. Zhao, H. Fang, F. Wang, and J. Liu. 2023. Realtime Multimedia Services over Starlink: A Reality Check. In *Proceedings of the 33rd Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'23)*. 43–49.

[46] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang. 2021. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. In *Proceedings of the AAAI conference on artificial intelligence (AAAI'21)*, Vol. 35. 11106–11115.

[47] X. Zhu, S. Sen, and Z M. Mao. 2021. Livelyzer: Analyzing the First-Mile Ingest Performance of Live Video Streaming. In *Proceedings of the 12th ACM Multimedia Systems Conference (MMSys'21)*. 36–50.