electronics

MDPI

Article

Evaluating HAS and Low-Latency Streaming Algorithms for Enhanced QoE

Syed Uddin [1,*], Michał Grega [1], Mikołaj Leszczuk [1] and Waqas ur Rahman [2]

[1] AGH University of Krakow, 30-059 Kraków, Poland; michal.grega@agh.edu.pl (M.G.); mikolaj.leszczuk@agh.edu.pl (M.L.)
[2] College of Computing, Birmingham City University, Birmingham B4 7XG, UK; waqas.rahman@bcu.ac.uk
* Correspondence: uddin@agh.edu.pl

Abstract

The demand for multimedia traffic over the Internet is exponentially growing. HTTP adaptive streaming (HAS) is the leading video delivery system that delivers high-quality video to the end user. The adaptive bitrate (ABR) algorithms running on the HTTP client select the highest feasible video quality by adjusting the quality according to the fluctuating network conditions. Recently, low-latency ABR algorithms have been introduced to reduce the end-to-end latency commonly experienced in HAS. However, a comprehensive study of the low-latency algorithms remains limited. This paper investigates the effectiveness of low-latency streaming algorithms in maintaining a high quality of experience (QoE) while minimizing playback delay. We evaluate these algorithms in the context of both Dynamic Adaptive Streaming over HTTP (DASH) and the Common Media Application Format (CMAF), with a particular focus on the impact of chunked encoding and transfer mechanisms on the QoE. We perform both objective as well as subjective evaluations of low-latency algorithms and compare their performance with traditional DASH-based ABR algorithms across multiple QoE metrics, various network conditions, and diverse content types. The results demonstrate that low-latency algorithms consistently deliver high video quality across various content types and network conditions, whereas the performance of the traditional adaptive bitrate (ABR) algorithms exhibit performance variability under fluctuating network conditions and diverse content characteristics. Although traditional ABR algorithms download higher-quality segments in stable network environments, their effectiveness significantly declines under unstable conditions. Furthermore, the low-latency algorithms maintained high user experience regardless of segment duration. In contrast, the performance of traditional algorithms varied significantly with changes in segment duration. In summary, the results underscore that no single algorithm consistently achieves optimal performance across all experimental conditions. Performance varies depending on network stability, content characteristics, and segment duration, highlighting the need for adaptive strategies that can dynamically respond to varying streaming environments.

Keywords: ABR algorithms; quality of experience; latency; DASH streaming; quality switching

Academic Editor: Kefeng Ji

Received: 12 April 2025
Revised: 15 June 2025
Accepted: 22 June 2025
Published: 26 June 2025

Citation: Uddin, S.; Grega, M.; Leszczuk, M.; Rahman, W.u. Evaluating HAS and Low-Latency Streaming Algorithms for Enhanced QoE. *Electronics* **2025**, *14*, 2587. https://doi.org/10.3390/electronics14132587

1. Introduction

Global internet traffic has exceeded 100 billion GB, and video content represents more than 80% of the total internet traffic in 2021 [1]. Among video delivery technologies, the MPEG Dynamic Adaptive Streaming over HTTP (DASH) standard is the most widely

adopted for streaming applications. MPEG-DASH dynamically adapts the quality of the video based on the available bandwidth of the viewer's network and the performance of the device.

The components of the MPEG-DASH adaptive streaming system are depicted in Figure 1. In MPEG-DASH, the video content is encoded at different video bitrates and divided into small segments. The DASH client downloads the videos segment by segment from the server. The DASH server hosts manifest files that contain information about video segments and quality levels [2]. The user initially requests the manifest file (e.g., mpd for DASH) from the server to retrieve information about available video segments, their quality levels, and timing [3]. The adaptive bitrate (ABR) algorithms are configured on the client side, selecting the appropriate video segment based on the network and the client-side configuration. These algorithms can be classified as throughput-, buffer-, and hybrid-based approaches. The main aim of the ABR algorithms is to select the highest feasible video quality while minimizing playback interruptions and quality switching events. A quality switch occurs when the two video segments are downloaded with different qualities. Playback interruptions is the state when the playback of the media stops because no further segment is available in the client buffer [4,5]. The adaptive bitrate strategy depends on various parameters including the segment size, bitrate, video resolution, and frame rate [6].

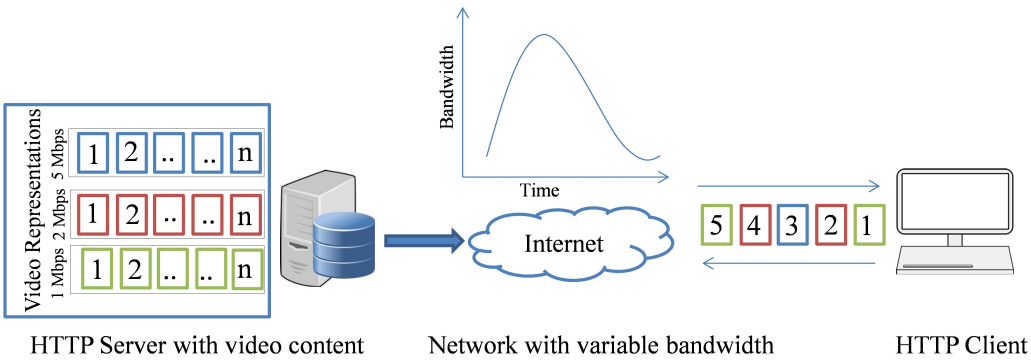

Figure 1. MPEG DASH streaming scenario.

Traditional DASH-based ABR algorithms focus on the quality of the experience metrics such as downloading high-quality video content and minimizing playback interruptions and video quality changes [7,8]. However, they often overlook the need to target low-latency streaming. Compared to terrestrial or satellite transmission, Internet broadcast has been shown to have a significant delay between video capture and playback. For video streaming over IP networks, maintaining end-to-end latency is crucial to achieve a user experience comparable to traditional broadcast TV [9]. There are several factors that influence latency, including video capture, encoding, packaging, video delivery through content delivery networks, segment buffering, and decoding. However, a study on Super Bowl 2024 latency found that streaming platforms over the Internet experienced delays averaging up to 70 s behind real-time action on the field. In contrast, cable and satellite channels are delivered to homes with an average delay of about five seconds behind the live feed. In the video streaming chain, this delay and other factors negatively impact the quality of experience (QoE) of the video [10].

To this end, low-latency ABR algorithms have been proposed to minimize end-to-end latency [7,11]. However, their performance has not been studied under different client- and server-end configurations. It is crucial to find the right balance between latency and the QoE. In the design of traditional ABR algorithms, latency is ignored, and algorithms only simultaneously maximize QoE metrics, while the low-latency algorithms aim to minimize latency while maximizing QoE metrics. The aim of this study is to study the performance of

low-latency algorithms and examine whether the emphasis on minimizing latency in low-latency ABR algorithms compromises the QoE. To this end, we analyze the performance of low-latency algorithms and compare them against traditional DASH-based algorithms using the web-based DASH player dash.js [12].

This work offers the following contributions: First, we evaluated, compared, and analyzed the performance of low-latency and traditional ABR algorithms to determine whether low-latency algorithms can simultaneously optimize QoE metrics while prioritizing latency reduction. Extensive experiments were conducted to evaluate the effects of varying network conditions, segment durations, and different video datasets on the performance of the algorithms. We used both MPEG-DASH and CMAF formats, incorporating chunked encoding and transfer mechanisms to assess their impact on streaming performance. The evaluation considered both objective metrics, such as bitrate, rebuffering, and stability, and subjective user assessments to provide a comprehensive analysis of the quality of experience (QoE). Finally, we assessed the strengths and limitations of low-latency algorithms and compared their performance with traditional ABR algorithms using the DASH player dash.js.

This paper is structured as follows: Section 2 presents the background work. Section 3 outlines the methodology. Section 4 provides the discussion. Section 5 concludes the article.

2. Related Work

This section provides details about the current work in the adaptive streaming domain.

2.1. Adaptive Bitrate Algorithms

The goal of ABR algorithms is to maximize the quality of experience for the end user. The paper [13] highlights the ABR algorithms employed in the adaptive streaming of HTTP. These algorithms target maximizing the video quality and minimizing rebuffering, the time spent in the rebuffering state, and quality-level switching. In this paper the author analyzes a wider range of algorithms including buffer-based, throughput-based, and reinforcement learning approaches and highlights the algorithms' mechanisms, strengths, and limitations. This work outlines major challenges, including accurate bandwidth estimation and maintaining playback smoothness and fairness in multi-user scenarios. The author concludes that no universal solution exists and recommends that future directions should focus on personalized, content-aware adaptation strategies; better subjective QoE integration; and cross-layer optimization. Bentaleb et al. [14] present a bitrate adaptation scheme that takes advantage of the nature of chunk downloads. This scheme uses a sliding window to assess bandwidth and an online linear adaptive filter for bandwidth prediction. This research presents ACTE, an ABR algorithm which is designed for chunk-based low-latency streaming. This paper highlights issue of inaccurate bandwidth estimation that is caused by the idle state between chunk deliveries in low-latency streaming, which possibly impacts the QoE. The ACTE algorithm accurately predicts the bandwidth and selects the optimal bitrates, outperforming the existing ABR algorithms in experiments using real network traces. This work demonstrates that chunk-level adaptation and predictive filtering are essential for maintaining high-quality, low-latency streaming. Several ABR algorithms are designed to target low latency. In [15], the authors present a bandwidth-based algorithm that selects the bitrates based on the moving arithmetic mean and the standard deviation of throughput and latency. The paper presents STALLION, an adaptive bitrate algorithm designed for low-latency video streaming. Stallion incorporates both the mean and standard deviation of throughput and latency measurements to enable stable bitrate decisions under varying bandwidths. The STALLION algorithm aims to balance playback quality and stability while maintaining live latency. In [16], the algorithm aims to minimize latency and select the video quality for the next segment based on online convex optimization.

This work present the Learn2Adapt-LowLatency (L2A-LL) adaptive bitrate algorithm. This algorithm is based on online convex optimization (OCO). The L2A-LL algorithm operates without explicit parameter tuning, throughput estimation, or channel modeling. These features makes L2A-LL robust and adaptive in varying network conditions. The algorithm aims to reduce latency while maintaining a high bitrate and enhancing the QoE. The algorithm provided by Bentaleb et al. in [17] uses heuristic- and learning-based approaches to minimize latency and optimize the QoE. In this research a learning-based ABR strategy is designed for low-latency environments. LoL+ incorporates a self-organizing map (SOM) that dynamically selects the optimal bitrate based on QoE metrics such as rebuffering, latency, and playback quality. LOL+ improves throughput estimation by filtering the idle times using client-side chunk-level data and introduces hybrid runtime weight tuning using k-means++ clustering and greedy search. The LOL+ model also offers QoE-aware playback speed control, which helps maintain target latency while minimizing rebuffering. The dynamic algorithm proposed by Spiteri et al. [18] is a hybrid and employs a throughput algorithm. This algorithm is based on the throughput measurement. The author presents the implementation of three advanced ABR algorithms, namely BOLA-E, DYNAMIC, and FAST SWITCHING. The BOLA-E algorithm is an extension of the existing BOLA algorithm by improving the startup and seek handling using placeholder segments. The dynamic algorithm combines throughput- and buffer-based logic for better adaptation under low-buffer scenarios. Fast switching presents segment replacement by rapidly adjusting to bandwidth fluctuations. These algorithms are integrated into the vendors production environment like BBC.

In [19], the authors created a data set from various content and video encoders. The data set is based on adaptive bitrate algorithms and performs a subjective evaluation on the data set. The results reveal that there is a correlation between the QoE models and the subjective opinions of the users. This shows that there is a need to improve the existing QoE models and the ABR algorithms.

Rodrigues et al. [20] investigated the impact of varying audio and video quality levels on the perceived QoE in live music streaming over mobile networks using MPEG-DASH. Through subjective testing, they find that video quality has a significantly greater influence on QoE than audio quality. The study proposed a joint parametric model combining audio and video quality metrics, which shows a strong correlation with subjective mean opinion scores (MOSs). The findings support using lower audio bitrates to preserve video quality in adaptive streaming scenarios. Rahman et al. [21] evaluated the effectiveness of multi-access edge computing (MEC)-assisted and client-based adaptation algorithms in optimizing the quality of experience (QoE) for HTTP adaptive video streaming. Through extensive simulations under varying network, buffer, and segment duration conditions, the research compares throughput-based and buffer-based approaches in multi-client cellular environments. However, the authors do not evaluate low-latency algorithms in their study. The work by O'Hanlon et al. [8] presents the evaluation results of the ABR algorithms with respect to a range of latency targets. The evaluation is performed in the dash.js player, and the results reveal that the dynamic algorithm outperforms the low-latency algorithms. Low-latency algorithms achieve higher video quality at the expense of a higher number of playback interruptions. The researchers adjusted the low-latency L2A-LL algorithm and evaluated it under modified settings, revealing promising improvements. A limitation of their work is that the authors do not evaluate algorithms with different video content or investigate the impact of segment duration on algorithm performance. Furthermore, the analysis is based on only three ABR algorithms. Lyko et al. [7] carry out an interesting evaluation with the low-latency algorithm. The evaluation is performed considering various QoE metrics and bandwidth profiles. In their work, the authors only

evaluate low-latency algorithms and do not assess their performance against traditional DASH-based algorithms. Additionally, the evaluation is conducted using ns-3 network simulations rather than a real-world test bed.

Table 1 shows that this study comprehensively addresses multiple key aspects to thoroughly evaluate low-latency video streaming research. Unlike previous studies, which focus selectively on a few components, such as low-latency algorithms or specific network conditions, our work systematically evaluates both low-latency and DASH-based adaptation strategies across a wide range of segment durations, video content, and network profiles. By incorporating objective and subjective QoE assessments and using real-world test bed environments along with CMAF-formatted content, this study ensures practical relevance and empirical robustness. This holistic approach not only bridges gaps left by previous research but also introduces a multidimensional benchmark to assess adaptive streaming performance under real-world constraints.

Table 1. Comparison of related work.

Feature	O'Hanlon et al. [8]	Rahman et al. [21]	Lyko et al. [7]	This Study
Low-latency algorithms	✓	×	✓	✓
DASH algorithms	×	✓	×	✓
Various network conditions	×	✓	×	✓
CMAF format	✓	×	✓	✓
Range of video content	×	×	✓	✓
Various segment lengths	×	✓	×	✓
Objective QoE analysis	✓	✓	✓	✓
Subjective QoE analysis	×	×	✓	✓
Real-World test bed	✓	×	×	✓

2.2. QoE Factors

There are several factors that affect the quality of the experience, which can potentially affect the quality of the video. Lebreton et al. [22] highlight that stalling events and quality switches impact the user experience. The authors in [23] conduct a study on the correlation between media quality and the impact of event stalls on experience quality. The study suggests that if the stall events are very short, they are not noticed by users. The study also mentions that longer stalling events generally impact the user experience the most. The work by Lebreton et al. presented in [24] considers the user quitting ratio while watching videos in an adaptive streaming scenario. The results demonstrate the factors affecting user persuasion. Another study by Lebreton et al. [25] proposes a method to predict the user quitting ratio while watching videos using adaptive streaming. The results reveal that quality, initial buffering, and stalling impact user behavior.

Quality switching is also an important factor in the quality of experience. The study provided by Babak et al. [23] reveals that frequent switching negatively impacts the overall user experience. The results also confirmed that the users preferred quality switching over stall events. However, as demonstrated in various studies [26,27], switching between quality levels also negatively impacts the quality of experience. Switches that affect the quality level and switching time affect user persuasion [5,28]. The research work by Allard et al. [29] investigates the trade-off between buffering delays and playback interruptions. The evaluation results reveal that, compared to the rebuffering events, playback interruptions negatively impact the QoE.

3. Experimental Setup

This section provides details about the methodology of the research work in detail.

3.1. Video Data Set

The data set is acquired from the established DASH database [30]. In the data set quality levels, the configuration between videos is differentiated. It is also required that two sequences have the same configuration. Videos with varying complexities, including high-motion scenes and low-detail content, are used in the evaluation. The video sequences were obtained from the original source (YUV) and run at 24 frames per second (FPS). Each segment is 2 s long, and the total video duration is 2 min (120 s). First, 2880 frames are extracted from the source video. The source videos and their characteristics are described in Table 2 in detail.

Table 2. Selected content and characteristics.

Video	Source Quality	Duration	Genre
Big Buck Bunny	Full HD YUV raw	09:46	Animation
Elephants Dream	Full HD YUV raw	10:54	Animation
Tears of Steel	Full HD YUV raw	12:15	Movie
Sparks	Full HD YUV raw	10:00	Movie

3.2. Segment Length and Quality Representations

The data set chosen for evaluation is encoded using various segment sizes. The segment ranges from 2 s to 10 s, as recommended in publication [30]. The consideration of segment length is an important factor in video streaming. The short-duration segments like 2 s provide more opportunities for the clients to adapt the bitrate. In addition, smaller segment sizes would produce greater overhead, as the client will frequently request the segments. Using longer-duration segments like 10 s may reduce the overhead, but the client would have a smaller number of opportunities to adapt the video rate. In the case of sudden changes in the throughput, there is a higher probability of playback interruption in the case of a longer duration of the segment. In our evaluation, we consider both short- and long-duration segments to analyze the performance of the algorithms. The bitrates and the quality-level ladder are shown in Table 3.

Table 3. Video sequence encoding and bitrate ladder.

Index	Animated Content	Movie Content
1	50 kbit/s, 320 × 240	50 kbit/s, 320 × 240
2	200 kbit/s, 480 × 360	200 kbit/s, 480 × 360
3	600 kbit/s, 854 × 480	600 kbit/s, 854 × 480
4	1.2 Mbit/s, 1280 × 720	1.2 Mbit/s, 1280 × 720
5	2.5 Mbit/s, 1920 × 1080	2.0 Mbit/s, 1920 × 1080
6	3.0 Mbit/s, 1920 × 1080	2.5 Mbit/s, 1920 × 1080
7	4.0 Mbit/s, 1920 × 1080	3.0 Mbit/s, 1920 × 1080
8	8.0 Mbit/s, 1920 × 1080	6.0 Mbit/s, 1920 × 1080

3.3. Evaluation Test Bed

The performance evaluation was conducted using dash.js, a JavaScript-based reference player for MPEG-DASH streaming. The library allows the streaming of MPEG-DASH media in device browsers. In this section, we discuss the details of the evaluation test bed. The architecture of the test bed is illustrated in Figure 2 and comprises four modules: two computers running Ubuntu, connected via Wi-Fi, simulating a video client and a server. The pre-encoded DASH videos were hosted on an Apache Web server. The key component of the architecture includes the bandwidth shaping node which is based on Ubuntu utilities. The bandwidth shaping node manages the client's maximum available bandwidth using

the Linux traffic control system (tc) and the hierarchical token bucket (htb), a class-based queuing discipline (qdisc). The encoded videos are stored on the Apache server. The client streams the video sequences using ABR algorithms, which are deployed on the client side. The detailed architecture is shown in Figure 2.

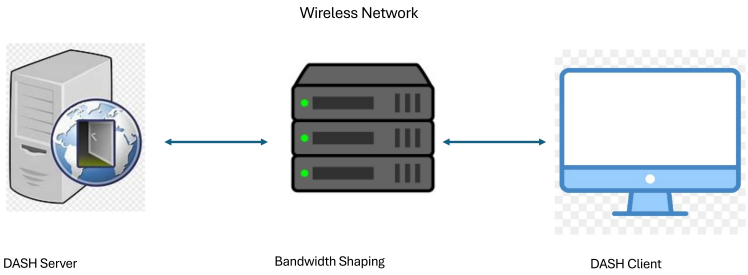

Figure 2. Wireless network test bed.

3.4. ABR Algorithm Evaluation

In this work, the ABR algorithms [18,31,32] are considered on the basis of their working principles. These algorithms range from rate-based algorithms to low-latency algorithms. The dynamic, BOLA, throughput, L2ALL, and LOL+ algorithms are prototyped in a dynamic adaptive streaming framework called dash.js. This framework references the open-source implementation for the MPEG-DASH standard.

1. **Throughput:** This algorithm makes a decision based on the throughput of the network. The algorithm estimates the throughput and decides which segment to download. This algorithm uses the average throughput of the previous video segment that was downloaded and decides on the optimal bitrate for the next video segment to be requested from the server [19].
2. **BOLA (Buffer Occupancy-based Lyapunov Algorithm):** The BOLA algorithm decides which bitrate to download based on the client buffer level. The buffer level is related to the network throughput. This means that this buffer-based algorithm selects a high bitrate in case the buffer fill level is high, and a low bitrate if the buffer level is low. The buffer-based algorithm is chosen by the video streaming provider. The BOLA algorithm is suitable for fluctuation scenarios in the bandwidth [33].
3. **Dynamic:** Dynamic is a hybrid algorithm. This algorithm makes full use of both throughput estimation and buffer levels. This algorithm smoothly switches between BOLA and throughput in real-time streaming. The algorithm addresses the shortcomings of the throughput- and buffer-based algorithms [18].
4. **Learn2Adapt Low Latency (L2A-LL):** L2A-LL is an adaptive bitrate (ABR) algorithm based on low latency. This algorithm uses the online convex optimization principle. The L2A-LL algorithm aims to minimize the video's latency. Compared to other ABR algorithms, L2A-LL provides a robust adaptation strategy. This algorithm does not require parameter tuning, channel model assumptions, or throughput assessments. These characteristics make L2A-LL ideal for users experiencing variations in the channel during streaming. Another feature of this algorithm is its modular architecture, which takes into account more QoE factors. These factors are categorized as stall, rebuffering, switching, and latency. These QoE factors consider various QoE objectives and streaming scenarios [7,8,34].
5. **Low on Latency (LOL+):** This is a heuristic algorithm that uses learning principles to optimize the parameters for the best QoE. In LOL+ each segment boundary is estimated and the highest QoE is predicated. The ABR algorithm which is implemented on a SOM (self-organizing map) model which takes into consideration various QoE metrics and network variations. The LOL+ playback speed control module is

based on a hybrid algorithm that measures latency and the buffer level and manages the playback speed. LOL+ and the QoE evaluation module is responsible for QoE computation based on metrics such as segment bitrate, switching, rebuffer events, latency, and playback speed [17,34].

3.5. Network Simulation

Figure 3a–d depict the network profiles used to evaluate the performance of the algorithms. We evaluated the algorithms under different network conditions including (1) gradual changes in the throughput (Figure 3a), (2) abrupt moderate variations in the throughput (Figure 3c,d), and (3) abrupt large variations in the throughput (Figure 3b). The motivation behind analyzing the algorithms under these network profiles is that if the algorithm reacts quickly to throughput changes, the algorithm will experience a high number of video rate changes. If the algorithm remains stable during throughput changes, it may respond too slowly to significant fluctuations, which may potentially lead to playback buffering or inefficient bandwidth utilization. For instance, in a cellular network where the user is mobile, the connection may experience both gradual and abrupt fluctuations in bandwidth. These network profiles enable a comprehensive evaluation of algorithm performance across diverse and dynamic environments.

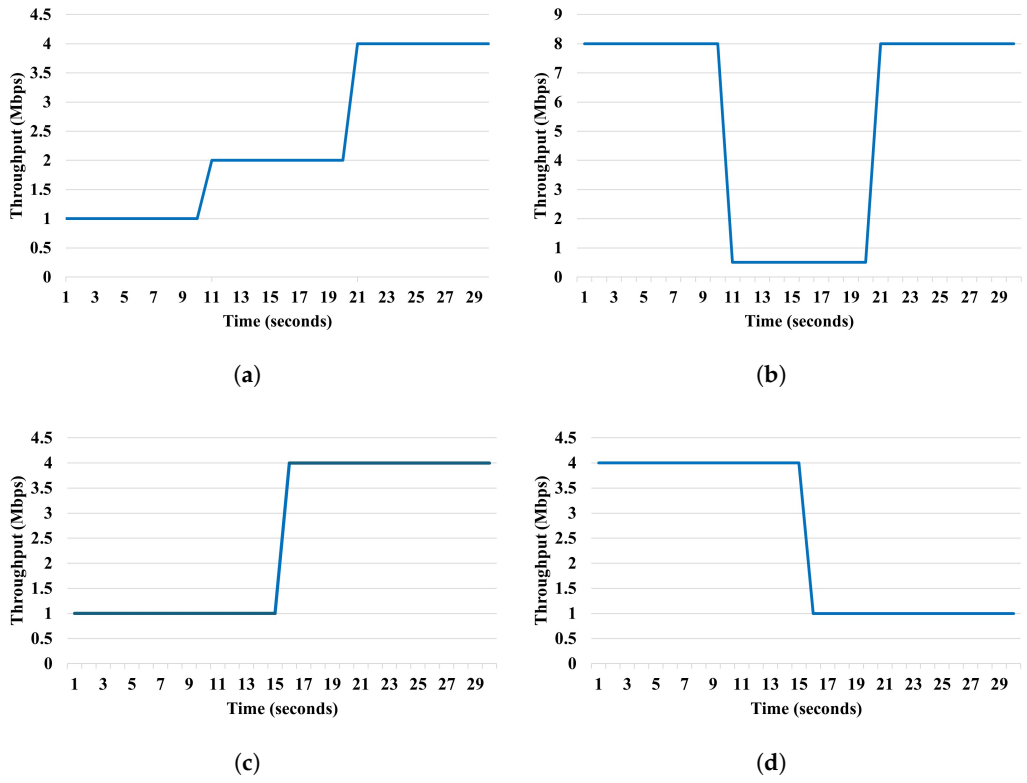

Figure 3. Network Profile Scenarios. (**a**) Network profile 1: Bandwidth 1 Mbps-2 Mbps-4 Mbps. (**b**) Network profile 2: Bandwidth 8 Mbps-500 kbps-8 Mbps. (**c**) Network profile 3: Bandwidth 1 Mbps-4 Mbps. (**d**) Network profile 4: Bandwidth 4 Mbps-1 Mbps.

4. Results and Discussion

This section discusses the results and analysis of the experiments. It also presents the implications of the results in detail. Here, we compare the performance of low-latency ABR algorithms in both DASH (Section 4.1) and CMAF (Section 4.2) modes.

4.1. Dynamic Adaptive Streaming over HTTP

This section presents the performance evaluation of the algorithms operating in DASH mode.

4.1.1. Analysis Under Network Profile 1

Here, we will evaluate the performance of the algorithms under different network profiles. First, we will analyze the performance of the algorithms under network profile 1. We start with a bandwidth of 1 Mbps, then increase it to 2 Mbps, and finally to 4 Mbps. The videos are divided into segments with a duration of 2 s each. Figure 4, illustrates the algorithms' response while streaming Big Buck Bunny. The figure shows that the LoL+ and dynamic algorithms rapidly increase the bitrate when the bandwidth increases from 1 Mbps to 4 Mbps. The L2A-LL and throughput algorithms delay increasing the bitrate to minimize the risk of interruption. When the bandwidth is increased to 4Mbps, all algorithms instantly increase the bitrate to efficiently use the bandwidth. Figures 5–7 show that the dynamic algorithm has a similar response when streaming all the videos. Figure 8 illustrates the fact that the dynamic algorithm achieves the highest video rate in all experiments irrespective of the video. The LoL+ algorithm quickly adapts to the bandwidth changes while streaming Big Buck Bunny and Elephant; however, in the case of the Tears of Steel and Spark video, it initially increases the bitrate but quickly decreases it to avoid playback interruption. The L2A-LL algorithm has a similar response to bandwidth changes in all videos, as it cautiously increases bitrates to small changes to avoid buffer overflow. The throughput algorithm carefully increases the bitrate for Big Buck Bunny and Elephant but more aggressively improves the video quality when streaming Tears of Steel and Spark. Figure 8 shows that the traditional DASH algorithms achieved higher bitrates when streaming Tears of Steel and Spark compared to BBB and Elephant, while the LoL + and L2A-LL algorithms achieved similar average bitrates for all experiments. Figure 9 shows that the LoL+ algorithm experienced the lowest number of video rate switches followed by traditional DASH algorithms. The L2A-LL algorithm had the highest number of bitrate switches. However, as shown in Figures 7–10, most of these changes were minor and would likely go unnoticed by the user. The major bitrate switches were when the bandwidth encouraged the algorithms to increase the bitrate. In general, the dynamic algorithm exhibited a more consistent response to changes in bandwidth across all videos compared to low-latency algorithms.

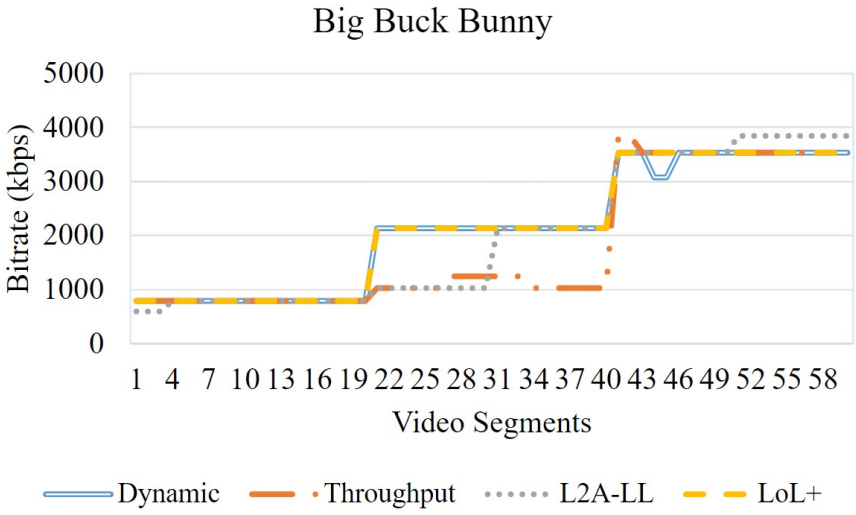

Figure 4. Network profile 1: Bitrate analysis—Big Buck Bunny.

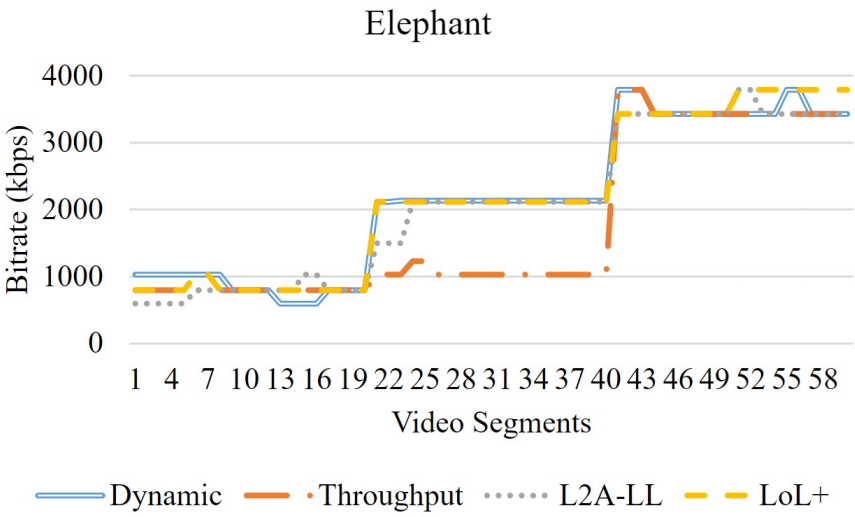

Figure 5. Network profile 1: Bitrate analysis—Elephant Dream.

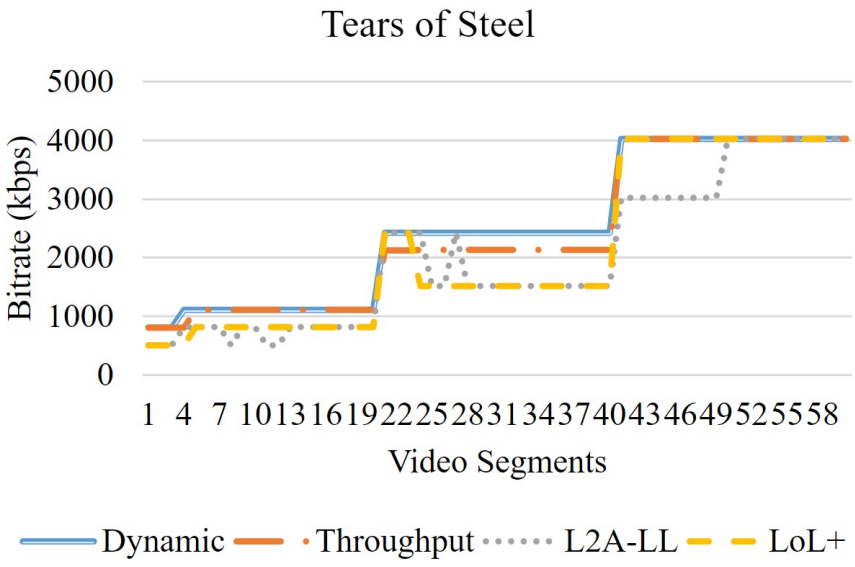

Figure 6. Network profile 1: Bitrate analysis—Tears of Steel.

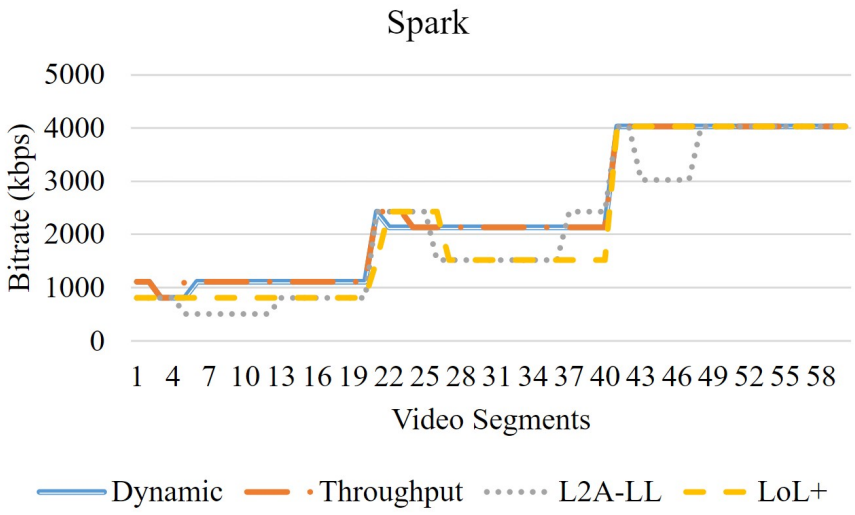

Figure 7. Network profile 1: Bitrate analysis—Sparks.

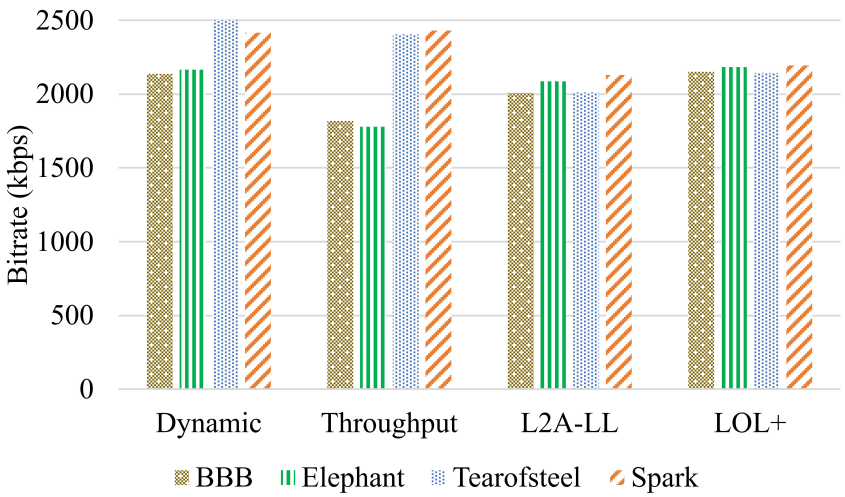

Figure 8. Average video bitrates achieved by the algorithm under network profile 1.

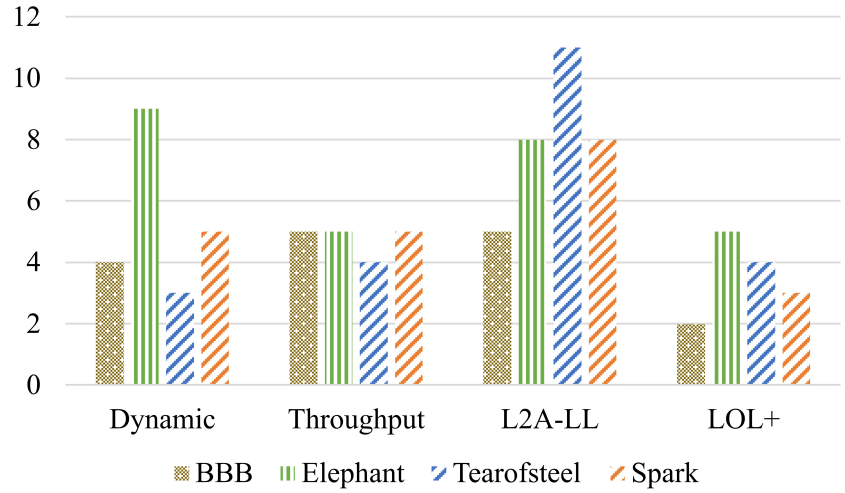

Figure 9. Number of switches experienced by the algorithms under network profile 1.

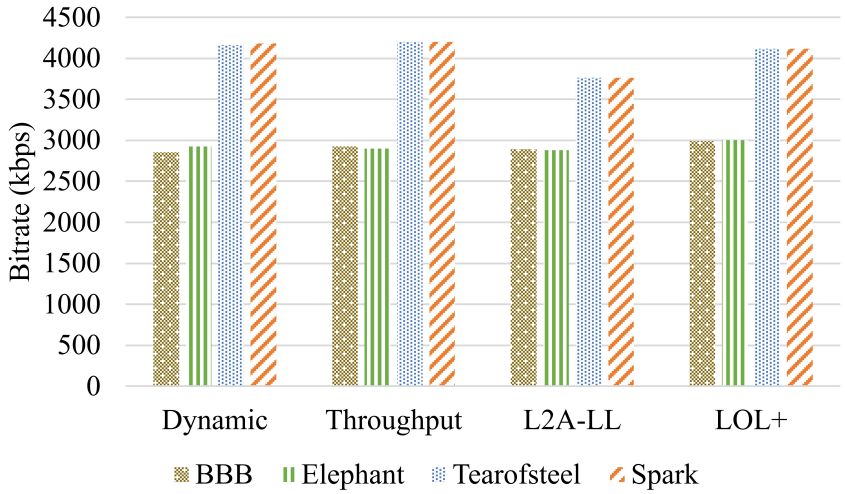

Figure 10. Average video bitrates achieved by the algorithm under network profile 2.

4.1.2. Analysis Under Network Profile 2

Next, we analyze the performance of the algorithms in network profile 2, starting with a bandwidth of 8 Mbps, then dropping it to 500 kbps, and finally increasing to 8 Mbps. The

segment duration is set to 2 s. The aim of evaluating the algorithm under network profile 2 is to analyze its performance under sudden large fluctuations in the throughput. During a significant drop in throughput, it is crucial for ABR algorithms to monitor the available buffer levels and determine how to avoid rebuffering and and minimize degradation of the user experience. However, striking this balance poses a significant challenge for algorithms. Let B^k be the buffer level at the beginning of the download of the k^{th} segment; then the buffer level before the download of the $(k^{th}+1)$ segment will be given by

$$B^{k+1} = B^k + \tau - \left(\tau \times \frac{R^k}{T^k}\right) \quad (1)$$

where (R^k) is the bitrate selected for the k^{th} segment, T^k is the throughput during the download of the k^{th} segment, and τ is the segment duration. When T^k drops, the algorithms must adapt R^k quickly to minimize the risk of B^{k+1} dropping to zero.

Figure 10 illustrates the average bitrates achieved by the algorithms for each video. The figure indicates that the average bitrates downloaded by the algorithms are largely consistent across all videos. However, the L2A-LL algorithm downloaded slightly lower bitrates during the streaming of Tears of Steel and Spark. Overall, the quality of the segment across all algorithms remains similar, with minor differences in these specific cases. Like in the previous experiment, Figure 11 shows that the LoL+ algorithm experienced the lowest number of video rate switches. Most of the fluctuations observed in the other algorithms involved small bitrate changes, which are unlikely to impact the user experience. Figure 12 shows that although the algorithms on average achieved similar video quality, the LoL+ algorithm outperformed the other algorithms by minimizing rebuffering events. The LoL+ algorithm encountered only one rebuffering event during the streaming of the Big Buck Bunny video, while it avoided any rebuffering events when streaming the other videos. The rest of the algorithms experienced rebuffering while streaming all the videos. Compared to the low-latency L2A algorithm, the traditional ABR algorithms, throughput and dynamic, experienced both a higher frequency and a longer duration of rebuffering events. In network profile 1, where the bandwidth gradually increased and the risk of rebuffering was minimal, the throughput and dynamic algorithms achieved higher bitrates. However, in network profile 2, where there was a sudden drop in throughput, these algorithms struggled to quickly reduce bitrates to prevent rebuffering. In contrast, the LoL+ algorithm efficiently selected higher bitrates without compromising playback by proactively managing the risk of interruptions.

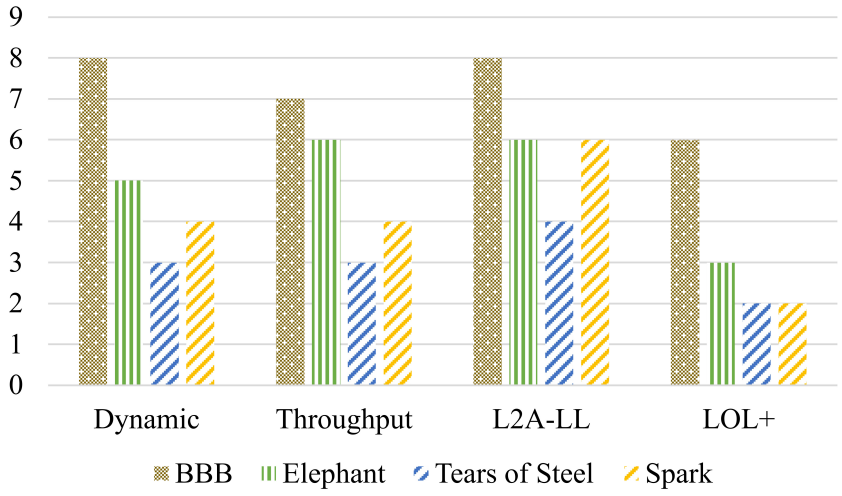

Figure 11. Number of switches experienced by the algorithms under network profile 2.

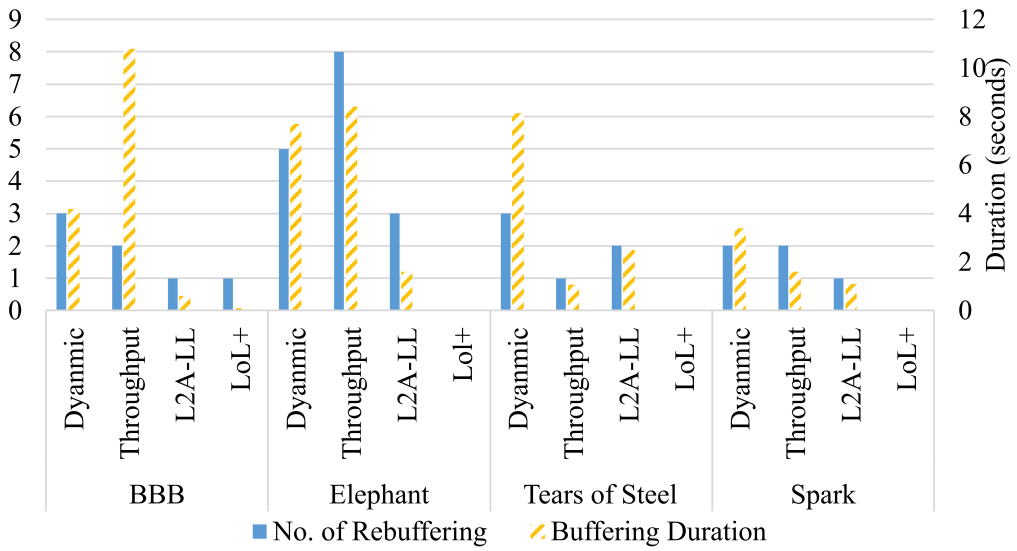

Figure 12. Frequency and total duration of rebuffering events encountered by the algorithms when operating under network profile 2.

4.1.3. Impact of Segment Duration

In this section, we will analyze the performance of the algorithms as the segment duration changes. Video streaming services provide different segment durations. For instance, Adobe's HTTP Dynamic Streaming (HDS) and Apple HTTP Live Streaming (HLS) provide segment durations of four and ten seconds, respectively. Therefore, it is crucial that the algorithms not only adapt the video bitrate efficiently across varying network conditions but also adjust the bitrate seamlessly as the segment durations change to maintain a smooth user experience. Referring to Equation (1), we observe that as the duration of the segment, τ, increases, even a slight mismatch between the selected bitrate and the available throughput can quickly deplete the buffer. Additionally, as the segment duration increases, the throughput is averaged over a longer time. This gives clients fewer opportunities to adapt the video bitrate in response to sudden changes in network conditions. To assess the performance of the algorithms, we analyze their behavior under both 2 s and 10 s segment durations. For each segment setting, we evaluate and compare their performance across network profiles 3 and 4.

1. The 2 s Segments

Here, we will analyze the performance of the algorithms under network profile 3. Figure 13 shows that the BOLA algorithm outperforms the other algorithms by downloading high-quality segments while avoiding unnecessary video rate switches. Although the difference in video bitrates between the algorithms is small, the dynamic algorithm tends to have a slightly lower average bitrate due to experiencing more frequent bitrate switches, where it momentarily drops to a lower bitrate before recovering.

Next, we compare the performance of the algorithms under network profile 4 where the bandwidth drops from 4 Mbps to 1 Mbps. As evident in the Figure 14, the BOLA and LoL+ algorithms achieve higher bitrate while avoiding any playback interruptions. The LoL+ algorithm had only a single bitrate switch due to the sudden drop in the bandwidth, and it avoided any unnecessary bitrate switch. The Dynamic and Throughput algorithms are able to download high quality segments when there is no risk of playback interruption. However, when the bandwidth drops suddenly, the algorithms fail to adapt quickly, leading to rebuffering events. The BOLA and low-latency algorithms avoided playback interruptions in both experiments.

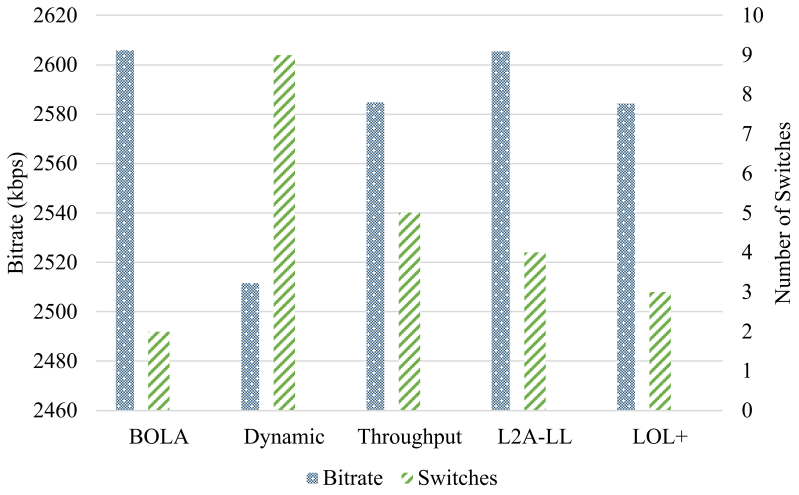

Figure 13. Average video rates and bitrate variations observed by the algorithm while streaming 2 s segments under network profile 3.

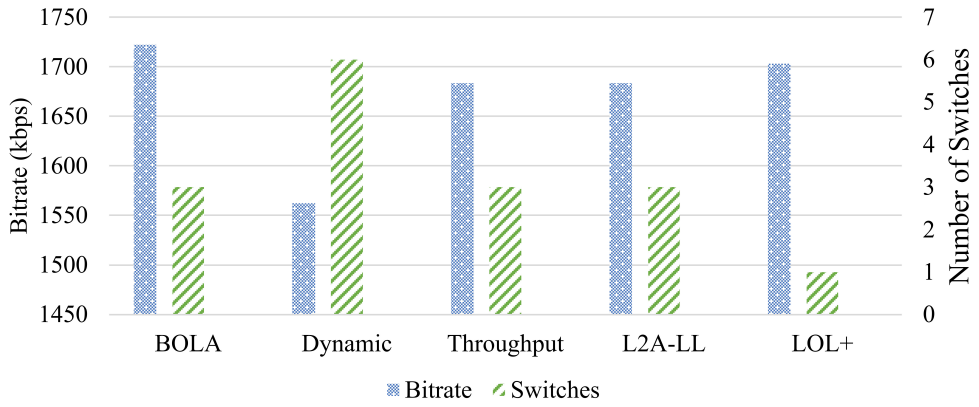

Figure 14. Average video rates and bitrate variations observed by the algorithm while streaming 2 s segments under network profile 4.

2. The 10 s Segments

Next, we increase the segment duration to observe how the algorithms adapt to changes in bandwidth. In the first experiment, we analyze the algorithms under network profile 3. Compared to 2 s segment experiments, we can clearly observe that the algorithms select lower-quality segments. This approach is quite intuitive since the larger segment duration in Equation (1) increases the risk of playback interruptions in the event of a mismatch between the throughput and the bitrate. The low-latency algorithms outperform the traditional algorithms as the segment duration increases. Figure 15 shows that the BOLA and throughput algorithms experience frequent bitrate switches; however, these are minor fluctuations between neighboring bitrates.

In the next experiment, we analyze the performance of the algorithms under network profile 4. Figure 15 shows that, like the previous network profile, the low-latency algorithms outperform traditional algorithms in selecting higher video quality. The BOLA, dynamic, and throughput algorithms conservatively select algorithms. Figure 16 shows that only the dynamic algorithm is able to avoid any playback interruption, but at the expense of video quality. Low-latency algorithms achieve higher video quality; however, when the throughput drops abruptly, they are unable to prevent the playback buffer from draining. Since the client can only switch bitrates at the start of a segment download, if the throughput drops during the segment download, the bitrate cannot be adjusted

mid-segment. Therefore, when streaming longer segments, it is crucial for algorithms to ensure sufficient buffer to prevent rebuffering. However, this often comes at the expense of video quality, creating a trade-off that then becomes a challenge for algorithms.

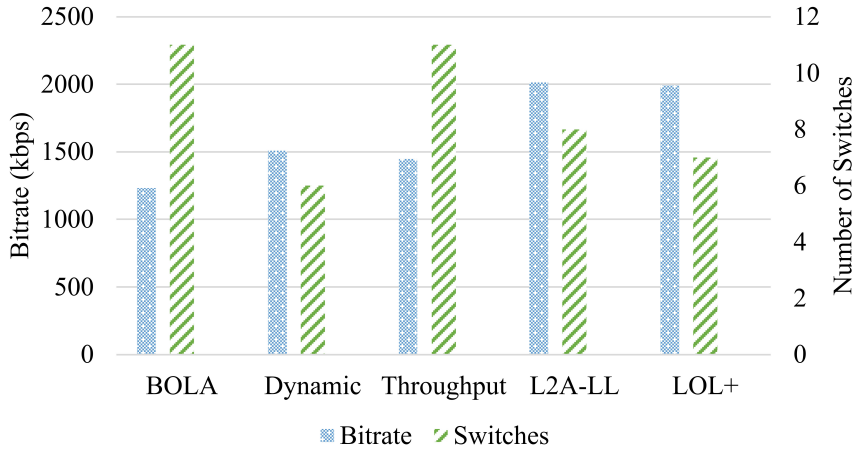

Figure 15. Average video rates and bitrate variations observed by the algorithm while streaming 10 s segments under network profile 3.

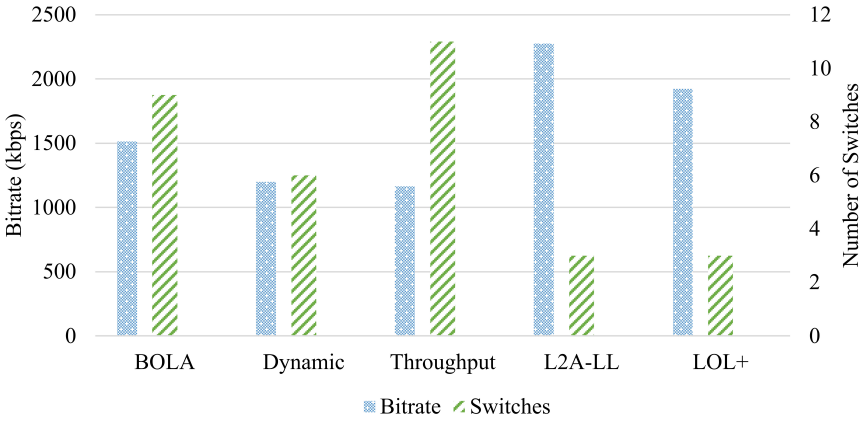

Figure 16. Average video rates and bitrate variations observed by the algorithm while streaming 10 s segments under network profile 4.

As shown in Figures 13–16, a significant difference in the quality of the downloaded segments is observed only in Figure 16, which is where clients download a 10 s segment under network profile 4. To assess the statistical significance of low-latency algorithms outperforming traditional ABR algorithms, we conducted pairwise independent *t*-tests. These tests evaluated the uncorrected *p*-values, *p*-values corrected using the Holm–Bonferroni method, the Bayes Factor (BF) indicating the strength of the evidence, and Hedges' g as a measure of effect size. The complete set of results is presented in Table 4. Notably, the L2A-LL algorithm demonstrated a significant performance advantage over the throughput algorithm, with an uncorrected *p*-value of 0.0053, a Holm–Bonferroni corrected *p*-value of 0.0529, a Bayes Factor (BF_{10}) of 8.147, and a large effect size (Hedges' g = 0.917). Similarly, L2A-LL outperformed the dynamic algorithm, as reflected by an uncorrected *p*-value of 0.0076, a corrected *p*-value of 0.0686, a BF_{10} of 6.122, and a substantial effect size (g = −0.874). Other pairwise comparisons, such as BOLA versus L2A-LL and LOL+ versus throughput, showed moderate evidence of differences, with Bayes Factors slightly above one and effect sizes indicating moderate effects. However, comparisons involving BOLA and dynamic versus LOL+ or throughput did not reach statistical significance

after correction (corrected p-value = 1), accompanied by low Bayes factors and minimal effect sizes. The results confirm that under conditions of longer segment durations and a high probability of rebuffering—a characteristic of live streaming scenarios—low-latency algorithms consistently outperform traditional ABR algorithms.

Table 4. Pairwise comparison of ABR algorithms.

A	B	p-unc	p-corr	BF_{10}	Hedges' g
BOLA	Dynamic	0.3027	1	0.476	0.324
BOLA	L2A-LL	0.0862	0.517	1.036	−0.546
BOLA	LOL+	0.3464	1	0.443	−0.296
BOLA	Throughput	0.2400	1	0.542	0.370
Dynamic	L2A-LL	0.0076	0.0686	6.122	−0.874
Dynamic	LOL+	0.0630	0.4407	1.284	−0.594
Dynamic	Throughput	0.8679	1	0.312	0.052
L2A-LL	LOL+	0.4742	1	0.380	0.224
L2A-LL	Throughput	0.0053	0.0529	8.147	0.917
LOL+	Throughput	0.0480	0.384	1.553	0.633

Another important factor is bitrate stability, which measures how stable or variable the bitrate is between segments of a video stream. To this end, we use the coefficient of variation (CV) to measure the variable in video quality. The CV is given by

$$\mathrm{CV} = \frac{\sigma}{\mu}$$

where σ = the standard deviation of segment bitrates and μ = the mean bitrate of the segments.

Figure 17 presents the CV values for the ABR algorithms, providing information on bitrate stability. For 10 s segments, under network profile 3, BOLA exhibits the lowest variability (CV $\approx$ 0.596), suggesting the most stable bitrate decisions, while throughput-based adaptation has the highest variability (CV $\approx$ 0.812), indicating more frequent fluctuations in bitrate. Interestingly, in network profile 4, the throughput algorithm becomes the most stable (CV $\approx$ 0.541), and LOL+ becomes the least stable (CV $\approx$ 0.798), demonstrating how network conditions can significantly affect algorithm performance.

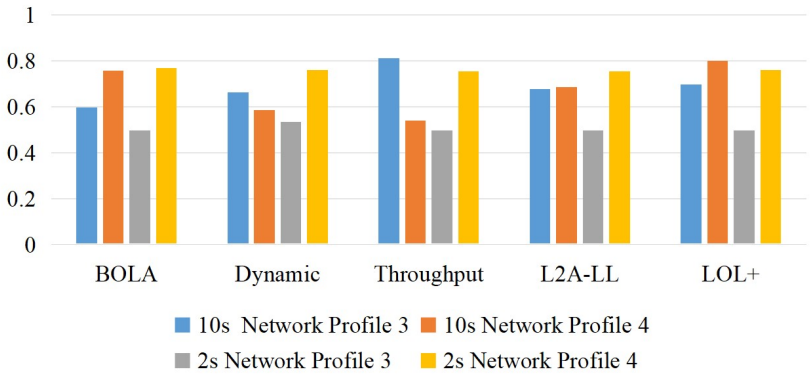

Figure 17. Coefficient of variation (CV) of per-segment bitrate for different ABR algorithms across two network profiles and segment durations (10 s and 2 s).

For 2 s segments, all algorithms exhibit much lower CVs under network profile 3 (ranging from 0.495 to 0.534), indicating more stable bitrate behavior likely due to faster adaptation in short segments. However, under network profile 4, CVs rise sharply across all

algorithms (CVs around 0.753 to 0.767), showing increased instability due to a higher risk of playback interruption as the throughput abruptly drops. Overall, the figure highlights that the segment length and network profile significantly influence bitrate stability, and no single algorithm is universally the most stable across all scenarios.

Figure 18 illustrates the time-varying average bitrates selected by the ABR algorithms across all experiments during the streaming session. Figure 19 compares the ABR algorithms based on their average video bitrate and the coefficient of variation (CV) over all experimental trials. The results clearly show that low-latency ABR algorithms deliver a more favourable trade-off between video quality and stability, outperforming conventional ABR methods in both the average bitrate and consistency of playback. L2A-LL achieves the highest average bitrate (2149.47 kbps) and also boasts the lowest CV (0.6390) among the group. This suggests a high-performance adaptation mechanism capable of efficiently utilizing the available bandwidth while ensuring stable quality levels. LOL+ closely follows with a bitrate of 2054.21 kbps and a relatively low CV of 0.6462, showcasing a strong balance between high video quality and bitrate consistency. Although slightly more variable than L2A-LL, its performance remains robust and well-suited to fluctuating network environments.

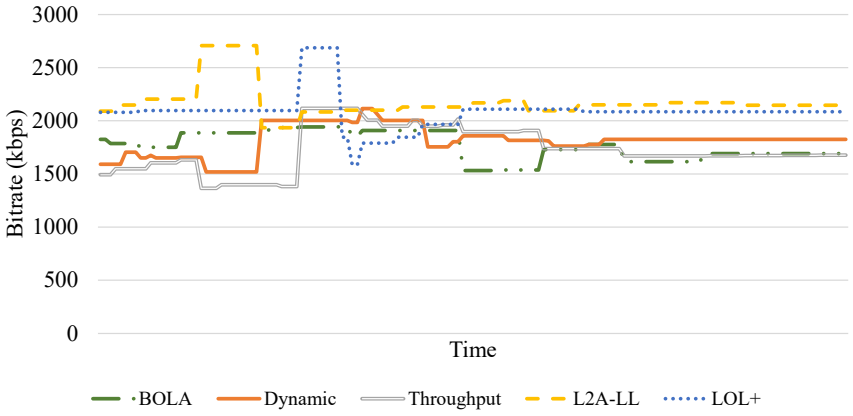

Figure 18. Average bitrate selected by clients during streaming sessions across all experiments.

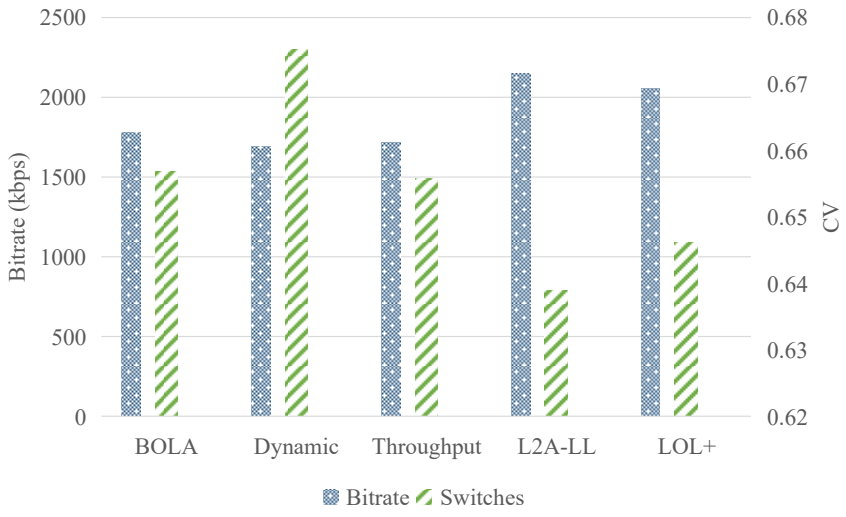

Figure 19. Average bitrate and coefficient of variation (CV) for ABR strategies.

4.2. CMAF

The The Common Media Application Format (CMAF) allows media segments to be further divided into smaller chunks, which can be delivered using Chunked Transfer Encoding (CTE). In this section, we evaluate the performance of adaptive bitrate (ABR) algorithms when streaming content encoded in the CMAF. CTE enables the transfer of data with an unknown or dynamically generated size as a sequence of length-delimited chunks that are streamed progressively as they become available on the server. This means that the client does not need to wait for an entire segment to be downloaded before beginning playback. Instead, playback can start as soon as the first few chunks are received, thereby reducing startup delay and contributing to lower overall end-to-end latency. In this section, we evaluate the performance of the algorithm when using Chunked Transfer Encoding (CTE), examining whether it aids in not only minimizing latency but also enhancing the user experience and improving the overall QoE.

Analysis Under Network Profile 1

The CMAF-based data set is prepared using the same video sequences employed in the objective evaluation. Encoding is performed using the FFmpeg tool, and MP4Box is used to package the encoded sequences into CMAF-compliant chunks. Each segment has a duration of 2 s and is further divided into four chunks.

Figure 20 presents the average video bitrate achieved by the evaluated algorithms under network profile 2 while streaming the Elephant video. The primary focus of this experiment is to investigate the performance of content encoded in the Common Media Application Format (CMAF) using Chunked Transfer Encoding (CTE). CTE allows clients to start playing without waiting for the complete download of an entire segment. Instead, playback can start as soon as the initial chunks are received. This mechanism significantly reduces startup delay and contributes to lower end-to-end latency.

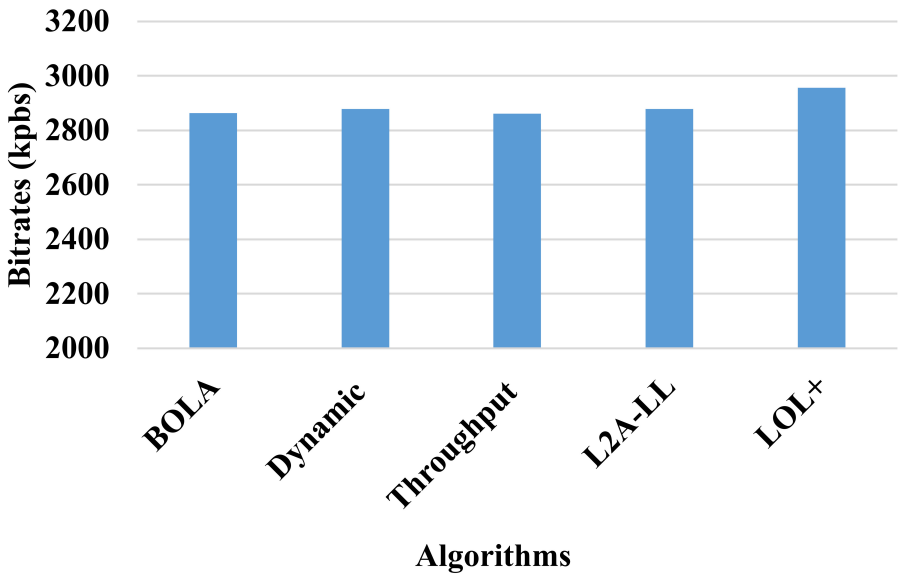

Figure 20. Average video bitrate achieved by the evaluated algorithms under network profile 2.

The results show that none of the algorithms experienced playback interruptions even during large and abrupt fluctuations in throughput. In comparison, in the traditional DASH format, only the LOL+ algorithm was able to prevent buffer depletion under similar network conditions. The advantage observed with the CMAF is attributed to the chunked delivery mechanism, where clients can start downloading individual chunks as soon as

they are available on the server, thereby minimizing latency and the risk of playback stalls. This behavior is particularly beneficial in live streaming scenarios where buffer sizes are kept small to maintain low latency close to the live edge.

Although the LOL+ algorithm achieved a slightly higher average bitrate, its impact on the QoE was limited. The ability to maintain continuous playback without interruptions had a greater influence on perceived user experience. This highlights the importance of using CMAF-encoded video content in combination with CTE to minimize latency while simultaneously enhancing the QoE.

5. Qualitative Study

This section provides details about the subjective evaluation of the ABR algorithms. This section includes the setup and evaluation procedure of the study.

5.1. Subjective Evaluation Method

This study was carried out to determine the user perception of video quality. The implementation of this study followed the recommendations in [23] and also ITU-T P.910. The Prolific crowd-source platform is used for the evaluation. The test participants were recruited through online advertisements. A total of 70 participants were selected. Both males and females participated in the subjective study. The 25 participants are included in the analysis. The Pearson correlation (0.75) is set on the sample. In this study, 24 video sequences were presented. Among them, four sequences (source) were of high quality, while another four were of very low quality. The remaining 16 video sequences (distorted version) were the outputs of the evaluated ABR algorithms. Each test sequence had a duration of 1 min (60 s). Participants were asked to evaluate the perceived video quality by selecting one of the following options: bad, poor, fair, good, or excellent. These qualitative ratings were mapped to integer values ranging from 1 to 5, where 'Bad' corresponded to 1 and 'Excellent' to 5. After viewing each sequence, participants provided a rating based on this scale, and a single mean opinion score (MOS) was recorded for each sequence.

5.2. Subjective Evaluation Results

Next, we will perform a subjective evaluation analysis for network profile 1. The objective of the subjective evaluation is to determine whether the results align with our quantitative analysis. Figure 21 presents the mean opinion scores of the algorithms for network profile 1 in various videos. In network profile 1, the algorithms initially select a lower quality followed by two incremental quality improvements. As a result, users tend to form an overall impression of the viewing experience based on initial quality. Figure 21 shows that the low-latency algorithms have a better MOS for the BBB and Elephant videos. However, their quality drops when streaming Tears of Stell and Spark. The throughput algorithm displays low quality for the BBB and Elephant videos, whereas the quality improves for Tears of Stell and Spark. This aligns with the results in (Section 4.1) Figure 8, as the performance algorithm streams segments of the BBB and Elephant videos at lower bitrates. The dynamic algorithm delivers a consistent viewing experience while maintaining a MOS score of around three across all four videos. Figure 22 shows that the dynamic algorithm achieves the best overall viewing experience followed by the throughput algorithm. The low-latency algorithm demonstrates inconsistent performance across different videos, leading to a lower overall QoE compared to traditional algorithms.

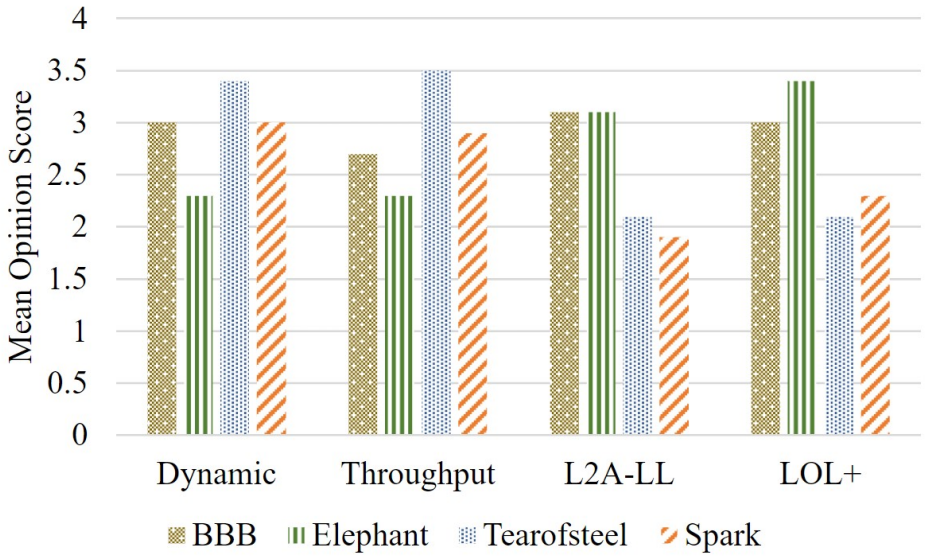

Figure 21. Mean Opinion Score of the algorithms for network profile 1.

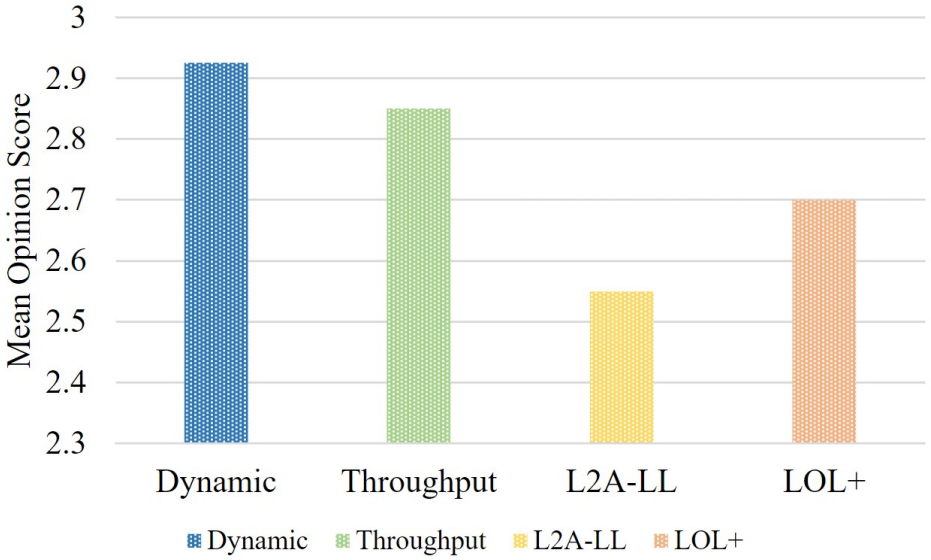

Figure 22. Average mean opinion score of the algorithms for network profile 1.

6. Analysis and Future Work

The primary goal of this work is to evaluate the effectiveness of low-latency ABR algorithms in maintaining high user experience while reducing playback latency. Given the growing dominance of multimedia traffic and the significant delays observed in live video streaming, this study investigates whether the emphasis on minimizing latency by low-latency algorithms compromises other key QoE metrics such as video quality, stability, and smoothness.

To achieve this, we compare the performance of low-latency algorithms with traditional DASH-based ABR algorithms. We conduct extensive experiments to assess their performance under diverse network conditions, segment durations, and content types. The aim is to provide a comprehensive comparison that reveals both the strengths and limitations of low-latency algorithms in real-world streaming scenarios.

The key insights and comparative observations are summarized in Table 5.

Table 5. Summary of evaluation insights.

Category	Observation
Content Robustness	Low-latency algorithms maintained consistently high video quality across different video types, whereas traditional DASH algorithms showed content-dependent performance variability.
Performance Under Unstable Conditions	In highly unstable network environments, low-latency algorithms achieved the highest QoE and effectively minimized playback interruptions—an essential feature for live streaming scenarios where buffer sizes are intentionally kept small to stay close to the live edge.
Segment Duration Independence	Low-latency algorithms sustained high video quality regardless of segment duration. In contrast, the performance of traditional algorithms degraded as segment duration increased. For example, BOLA achieved the highest bitrate with 2 s segments but fell to one of the lowest at 10 s (see Figure 15).
Stability of Playback Quality	Traditional algorithms demonstrated performance fluctuations across different network conditions. For instance, the dynamic algorithm performed well under network profile 1 but significantly degraded under other profiles.
Overall Suitability for Live Streaming	Despite being optimized primarily for reducing latency, low-latency algorithms consistently delivered comparable or superior video quality across varying content types and network conditions, making them highly suitable for live video streaming applications.
Adaptation Speed	LoL+ and L2A-LL demonstrated faster bitrate adaptation in response to abrupt bandwidth drops, reducing rebuffer events more effectively.
Quality vs. Bitrate Trade-off	Traditional algorithms (e.g., throughput) consumed more bandwidth but delivered only marginally better video quality, reducing overall efficiency.
Consistency Across Content	The dynamic algorithm maintained relatively consistent performance across both animation and movie content, whereas L2A-LL was more affected by content complexity.

In future work, we aim to extend this study in the following directions:

- **Evaluation with Realistic Network Traces:** We plan to use real-world cellular network throughput traces to evaluate the performance of ABR algorithms under more realistic and dynamic conditions, such as those described in [35].
- **Comprehensive Subjective Analysis:** While this work includes limited subjective evaluations, future efforts will involve subjective assessments of all experiments conducted, including those run on cellular network traces.
- **Exploration of Latency Target Variability:** In our current setup, the latency target (i.e., delay from the live edge) is fixed at 6 s. Future work will investigate the following:
  - How varying latency targets impact the user's quality of experience (QoE);
  - The performance of different ABR algorithms in `dash.js` under various latency targets;
  - Strategies for dynamically selecting an optimal latency target based on client-side network conditions and device capabilities.

7. Conclusions

In this research work, the evaluation results of adaptive bitrate (ABR) algorithms are presented. The objective assessment is performed to measure the impact on video

quality. In the experiment, both short- and long-duration segments are considered. In this work, various bandwidth profiles are considered. The MPEG.js environment is used for evaluation. The ABR algorithms are evaluated under various network conditions. The HAS algorithm is compared to the low-latency algorithms. The performance was carried out using QoE metrics like bitrate, buffering events, and buffering duration. The ITU-T guidelines were followed in the objective evaluation. The results demonstrate that the LOL+ and dynamic algorithms performed well when the bandwidth is increased from 1 MBIT to 4 MBIT. The L2A-LL and throughput algorithms delayed increasing the bitrate, which minimizes the chances of playback interruptions. The dynamic algorithm outperformed other algorithms by achieving a high video rate. The evaluation is carried out with a longer segment duration (10 s). The results demonstrate that when streaming longer segments, it is crucial for algorithms to ensure sufficient buffering to prevent rebuffering. However, this often comes at the expense of video quality, creating a trade-off that becomes a challenge for algorithms. The results provide evidence that there are needs for the framework for video quality in a streaming environment. This research provides a foundation for the building of next-generation video streaming solutions.

In this article a qualitative study is carried out, and preliminary results are presented. The purpose of the qualitative study is to collect the opinions of the users and estimate the perception of the users. The results reveal that low-latency algorithms have better mean opinion scores (MOSs) for some sets of videos. But the quality of the low-latency algorithms drops for one set of videos. These preliminary results align with our quantitative results of the experiment. According to the qualitative study, the dynamic algorithm shows promising viewing experience, maintaining a consistent MOS score across all presented video sequences. According to the results, the low-latency algorithms demonstrate inconsistent performance across all video streams. This leads low-latency algorithms to a lower overall QoE compared to traditional HAS algorithms. In future work, we will extend the experiment and perform further studies by considering (a) additional QoE metrics, (b) incorporating more source videos, and (c) deploying CMAF for low-latency computation.

Author Contributions: Methodology, S.U.; Software, S.U.; Investigation, S.U.; Resources, M.L.; Data curation, S.U.; Writing—review & editing, M.G., M.L. and W.u.R.; Visualization, S.U.; Supervision, M.G. and M.L.; Project administration, M.L. and W.u.R. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Polish Ministry of Science and Higher Education with the subvention funds of the Faculty of Computer Science, Electronics and Telecommunications of AGH University.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kim, S.; Baek, H.; Kim, D.H. OTT and live streaming services: Past, present, and future. *Telecommun. Policy* **2021**, *45*, 102244. [CrossRef]
2. Vlaovic, J.; Rimac-Drlje, S.; Žagar, D.; Filipović, L. Content dependent spatial resolution selection for MPEG DASH segmentation. *J. Ind. Inf. Integr.* **2021**, *24*, 100240. [CrossRef]
3. Vlaovic, J.; Žagar, D.; Rimac-Drlje, S.; Vranjes, M. Evaluation of objective video quality assessment methods on video sequences with different spatial and temporal activity encoded at different spatial resolutions. *Int. J. Electr. Comput. Eng. Syst.* **2021**, *12*, 1–9. [CrossRef]
4. İren, E.; Kantarci, A. Content Aware Video Streaming with MPEG DASH Technology. *TEM J.* **2022**, *11*, 611–619. [CrossRef]
5. Alsabaan, M.; Alqhtani, W.; Taha, A. An Adaptive Quality Switch-aware Framework for Optimal Bitrate Video Streaming Delivery. *Int. J. Adv. Comput. Sci. Appl.* **2020**, *11*, 570–579.

6. Klink, J.; Brachmański, S. An Impact of the Encoding Bitrate on the Quality of Streamed Video Presented on Screens of Different Resolutions. In Proceedings of the 2022 International Conference on Software, Telecommunications and Computer Networks (SoftCOM), Split, Croatia, 22–24 September 2022; pp. 1–6. [CrossRef]
7. Lyko, T.; Broadbent, M.; Race, N.; Nilsson, M.; Farrow, P.; Appleby, S. Improving quality of experience in adaptive low latency live streaming. *Multimed. Tools Appl.* **2024**, *83*, 15957–15983. [CrossRef]
8. O'Hanlon, P.; Aslam, A. Latency Target based Analysis of the DASH.js Player. In Proceedings of the 14th ACM Multimedia Systems Conference (MMSys'23), Vancouver, BC, Canada, 7–10 June 2023; Association for Computing Machinery: New York, NY, USA, 2023; pp. 153–160. [CrossRef]
9. Erfanian, A. Optimizing QoE and Latency of Live Video Streaming Using Edge Computing and In-Network Intelligence. In Proceedings of the 12th ACM Multimedia Systems Conference (MMSys'21), Istanbul, Turkey, 28 September–1 October 2021; Association for Computing Machinery: New York, NY, USA, 2021; pp. 373–377. [CrossRef]
10. Xie, G.; Chen, H.; Yu, F.; Xie, L. Impact of playout buffer dynamics on the QoE of wireless adaptive HTTP progressive video. *ETRI J.* **2021**, *43*, 447–458. [CrossRef]
11. Taraghi, B.; Hellwagner, H.; Timmerer, C. LLL-CAdViSE: Live Low-Latency Cloud-based Adaptive Video Streaming Evaluation Framework. *IEEE Access* **2023**, *11*, 25723–25734. [CrossRef]
12. Silhavy, D.; Pham, S.; Arbanowski, S.; Steglich, S.; Harrer, B. Latest advances in the development of the open-source player dash.js. In Proceedings of the 1st Mile-High Video Conference (MHV'22), Denver, CO, USA, 1–3 March 2023; Association for Computing Machinery: New York, NY, USA, 2022; pp. 32–38. [CrossRef]
13. Bentaleb, A.; Taani, B.; Begen, A.; Timmerer, C.; Zimmermann, R. A Survey on Bitrate Adaptation Schemes for Streaming Media over HTTP. *IEEE Commun. Surv. Tutor.* **2018**, *21*, 562–585. [CrossRef]
14. Bentaleb, A.; Timmerer, C.; Begen, A.C.; Zimmermann, R. Bandwidth prediction in low-latency chunked streaming. In Proceedings of the 29th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'19), Amherst, MA, USA, 21 June 2019; Association for Computing Machinery: New York, NY, USA, 2019; pp. 7–13. [CrossRef]
15. Gutterman, C.L.; Fridman, B.; Gilliland, T.; Hu, Y.; Zussman, G. Stallion: Video adaptation algorithm for low-latency video streaming. In Proceedings of the 11th ACM Multimedia Systems Conference, Istanbul, Turkey, 8–11 June 2020.
16. Karagkioules, T.; Mekuria, R.; Griffioen, D.; Wagenaar, A. Online learning for low-latency adaptive streaming. In Proceedings of the 11th ACM Multimedia Systems Conference (MMSys'20), Istanbul, Turkey, 8–11 June 2020; Association for Computing Machinery: New York, NY, USA, 2020; pp. 315–320. [CrossRef]
17. Bentaleb, A.; Akcay, M.N.; Lim, M.; Begen, A.C.; Zimmermann, R. Catching the Moment with LoL+ in Twitch-Like Low-Latency Live Streaming Platforms. *IEEE Trans. Multimed.* **2022**, *24*, 2300–2314. [CrossRef]
18. Spiteri, K.; Sitaraman, R.; Sparacio, D. From Theory to Practice: Improving Bitrate Adaptation in the DASH Reference Player. *ACM Trans. Multimedia Comput. Commun. Appl.* **2019**, *15*, 1–29. [CrossRef]
19. Duanmu, Z.; Liu, W.; Li, Z.; Chen, D.; Wang, Z.; Wang, Y.; Gao, W. Assessing the Quality-of-Experience of Adaptive Bitrate Video Streaming. *arXiv* **2020**, arXiv:2008.08804.
20. Rodrigues, R.; Počta, P.; Melvin, H.; Bernardo, M.; Pereira, M.; Pinheiro, A. Audiovisual Quality of Live Music Streaming over Mobile Networks using MPEG-DASH. *Multimed. Tools Appl.* **2020**, *79*, 24595–24619. [CrossRef]
21. Rahman, W.u.; Huh, E.N. Content-aware QoE optimization in MEC-assisted Mobile video streaming. *Multimed. Tools Appl.* **2023**, *82*, 42053–42085. [CrossRef]
22. Taraghi, B.; Haack, S.Z.; Timmerer, C. Towards Better Quality of Experience in HTTP Adaptive Streaming. In Proceedings of the 2022 16th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS), Dijon, France, 19–21 October 2022; pp. 608–615.
23. Taraghi, B.; Nguyen, M.; Amirpour, H.; Timmerer, C. Intense: In-Depth Studies on Stall Events and Quality Switches and Their Impact on the Quality of Experience in HTTP Adaptive Streaming. *IEEE Access* **2021**, *9*, 118087–118098. [CrossRef]
24. Lebreton, P.; Yamagishi, K. Quitting Ratio-Based Bitrate Ladder Selection Mechanism for Adaptive Bitrate Video Streaming. *IEEE Trans. Multimed.* **2023**, *25*, 8418–8431. [CrossRef]
25. Lebreton, P.; Yamagishi, K. Predicting User Quitting Ratio in Adaptive Bitrate Video Streaming. *IEEE Trans. Multimed.* **2020**, *23*, 4526–4540. [CrossRef]
26. Nguyen, M.; Vats, S.; Van Damme, S.; Van Der Hooft, J.; Vega, M.T.; Wauters, T.; Timmerer, C.; Hellwagner, H. Impact of Quality and Distance on the Perception of Point Clouds in Mixed Reality. In Proceedings of the 2023 15th International Conference on Quality of Multimedia Experience (QoMEX), Ghent, Belgium, 20–22 June 2023; pp. 87–90. [CrossRef]
27. Vats, S.; Nguyen, M.; Van Damme, S.; van der Hooft, J.; Vega, M.T.; Wauters, T.; Timmerer, C.; Hellwagner, H. A Platform for Subjective Quality Assessment in Mixed Reality Environments. In Proceedings of the 2023 15th International Conference on Quality of Multimedia Experience (QoMEX), Ghent, Belgium, 20–22 June 2023; pp. 131–134. [CrossRef]

28. Vlaović, J.; Žagar, D.; Rimac-Drlje, S.; Filipović, L. Comparison of representation switching number and achieved bit-rate in DASH algorithms. In Proceedings of the 2020 International Conference on Smart Systems and Technologies (SST), Osijek, Croatia, 14–16 October 2020; pp. 17–22. [CrossRef]
29. Allard, J.; Roskuski, A.; Claypool, M. Measuring and modeling the impact of buffering and interrupts on streaming video quality of experience. In Proceedings of the 18th International Conference on Advances in Mobile Computing & Multimedia (MoMM'20), Chiang Mai, Thailand, 30 November–2 December 2020; Association for Computing Machinery: New York, NY, USA, 2021; pp. 153–160. [CrossRef]
30. Lederer, S.; Müller, C.; Timmerer, C. Dynamic adaptive streaming over HTTP dataset. In Proceedings of the 3rd Multimedia Systems Conference (MMSys'12), Chapel Hill, NC, USA, 22–24 February 2012; Association for Computing Machinery: New York, NY, USA, 2012; pp. 89–94. [CrossRef]
31. Sani, Y.; Mauthe, A.; Edwards, C. Adaptive Bitrate Selection: A Survey. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 2985–3014. [CrossRef]
32. Tadahal, S.S.; Gummadi, S.V.; Prajapat, K.; Meena, S.M.; Kulkarni, U.; Gurlahosur, S.V.; Vyakaranal, S. A Survey on Adaptive Bitrate Algorithms and Their Improvisations. In Proceedings of the 2021 International Conference on Intelligent Technologies (CONIT), Hubli, India, 25–27 June 2021; pp. 1–7. [CrossRef]
33. Marx, E.; Yan, F.Y.; Winstein, K. Implementing BOLA-BASIC on Puffer: Lessons for the use of SSIM in ABR logic. *arXiv* **2020**, arXiv:2011.09611.
34. Lim, M.; Akcay, M.N.; Bentaleb, A.; Begen, A.C.; Zimmermann, R. When they go high, we go low: Low-latency live streaming in dash.js with LoL. In Proceedings of the 11th ACM Multimedia Systems Conference (MMSys'20), Istanbul, Turkey, 8–11 June 2020; Association for Computing Machinery: New York, NY, USA, 2020; pp. 321–326. [CrossRef]
35. Müller, C.; Lederer, S.; Timmerer, C. An evaluation of dynamic adaptive streaming over HTTP in vehicular environments. In Proceedings of the 4th Workshop on Mobile Video (MoVid'12), Chapel Hill, NC, USA, 24 February 2012; Association for Computing Machinery: New York, NY, USA, 2012; pp. 37–42. [CrossRef]