# Evaluating HAS and Low-Latency Streaming Algorithms for Enhanced QoE

Syed Uddin *, Michał Grega, Mikolaj Leszczuk, Waqas ur Rahman

Posted Date: 15 April 2025

doi: 10.20944/preprints202504.1114.v1

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

 doi:10.20944/preprints202504.1114.v1



*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, Poland
[2] Birmingham City University, United Kigdom
* Correspondence: uddin@agh.edu.pl
† Current address: Affiliation 3.

**Abstract:** The demand for multimedia traffic over 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 algorithms (ABR) 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. In this paper, we present an evaluation 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. Additionally, we conduct an extensive subjective test to evaluate the impact of video quality variations on QoE. The results show that the algorithms do not perform consistently under different network conditions and content settings. The results indicate that the traditional ABR algorithms outperform low-latency algorithms in stable network conditions. When segment durations are shorter, low-latency algorithms outperform traditional ABR algorithms under variable network conditions. The findings also reveal that the performance of the algorithms drops as the segment duration increases. The dynamic algorithm performs best when the segment duration is increased and under high risk of playback interruptions.

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

## 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]. Adaptive streaming (HAS) HTTP sends dynamic video streams from the server to the client [2].

The MPEG Dynamic Adaptive Streaming over HTTP (DASH) standard is widely used for video streaming. An MPEG DASH streaming scenario is shown in Figure 1. In MPEG-DASH, the video content is encoded at different video bitrate 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 [3]. 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 [4].

The ABR algorithm configured on the client side selects 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 ensure smooth streaming with the initial connection, lowest stalling, and switching events. A quality switch occurs when the two video segments are downloaded with different qualities. Stability is the state when the playback of the media stops because no further segment is available in the client buffer [5,6]. The video sequences are encoded with various bit rates and resolutions. The encoding strategy depends on various parameters, for example, segment size, bitrate, video resolution, and frame rate [7].

 doi:10.20944/preprints202504.1114.v1



Traditional DASH-based ABR algorithms focus on quality of experience metrics such as downloading high quality video content, minimising playback interruptions, and video quality changes [2,8]. 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 seconds 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 minimise end-to-end latency [2,11]. 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 QoE. In the design of traditional ABR algorithms, latency is ignored, and algorithms only simultaneously maximise QoE metrics, while the low-latency algorithms aim to minimise latency while maximising QoE metrics. The aim of this work is to investigate whether the prioritisation of latency by the low-latency algorithms negatively affects other QoE metrics. We analyse the performance of low-latency algorithms and compare it with traditional algorithms using the web-based DASH player, dash.js [12].

This work offers the following contributions.

1. We evaluated, compared, and analysed the performance of low-latency and traditional ABR algorithms. The aim of the comparison is to analyse whether low-latency algorithms simultaneously optimise QoE metrics while prioritising minimising latency.
2. We conducted extensive experiments to evaluate the effect of varying network conditions, segment durations, and different video data sets on the performance of the algorithms.
3. We assess both the strengths and the shortcomings of low-latency algorithms and compare their performance with the traditional ABR algorithms using the DASH player, dash.js.

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

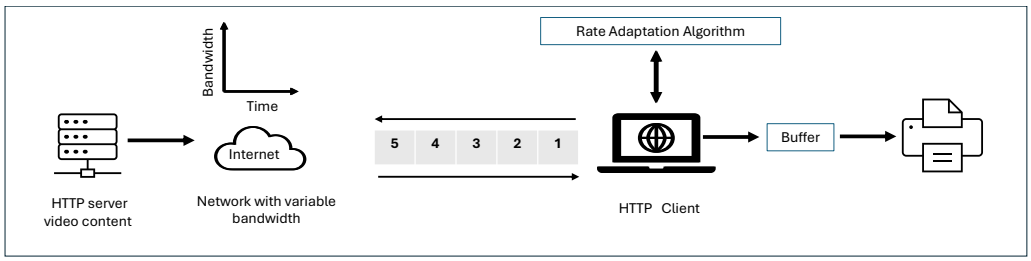


**Figure 1.** MPEG DASH Streaming scenario

## 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 maximise the quality of experience for the end user. The paper [13] highlights the ABR algorithms employed in adaptive streaming of HTTP. These algorithms target maximising the video quality, minimising re-buffering, time spent in re-buffering state, and the quality level switching. 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. Several ABR algorithms are designed to target low

 doi:10.20944/preprints202504.1114.v1



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. In [16] by Karagkioules et al., the algorithm aims to minimise latency and select the video quality for the next segment based on online convex optimisation. The algorithm provided by Bentaleb et al. in [17] uses heuristic and learning-based approaches to minimise latency and optimise QoE. The dynamic algorithm proposed by Spiteri et al. [[18]] is hybrid and employs a throughput algorithm. This algorithm is based on measurement throughput.

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.

An evaluation by Rodrigues et al. [20] is performed to find the trade-off between audio and video quality. The subjective evaluation is carried out in MPEG-DASH streaming environment. The results indicate that the proposed method is suitable for objective video quality evaluation in live streaming environments.

The work of 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 this work is that the authors do not evaluate algorithms in 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. [2] carry out an interesting evaluation with the low-latency algorithm. The evaluation is performed considering various QoE metrics and bandwidth profiles. In this work, the authors only evaluate low-latency algorithms and do not assess their performance against traditional DASH-based algorithms.

### *2.2. QoE Factors*

There are several factors that affect the quality of the experience that can potentially affect the quality of the video. Lebreton et al. [21] highlight that stalling events and quality switches impact the user experience. The authors in [22] 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 [23] considers the user quitting ratio while watching videos in an adaptive streaming scenario. The results demonstrate the factors affecting the user persuasion. Another study by Lebreton et al. [24] 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 behaviour.

Quality switching is also an important factor in quality of experience. The study provided by Babak et al. [22] 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 [25,26], switching between quality levels also negatively impacts Quality of Experience. Switches of quality level and switching time affect user persuasion [6,27]. The research work by Allard et al. [28] investigates the trade-off between buffering delays and playback interruptions. The evaluation results reveal that, compared to the rebuffering events, playback interruptions negatively impact QoE. The findings of the study are used to build a model based on two factors: rebuffering and playback interruptions.

## 3. Experimental Setup

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

 doi:10.20944/preprints202504.1114.v1



### 3.1. Video Dataset

The dataset is acquired from the established DASH database [29]. 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 source videos and their characteristics are described in Table 1 in detail.

### 3.2. Segment Length and Quality Representations

The dataset chosen for evaluation is encoded using various segment sizes. The segment ranges from 2 seconds to 10 seconds, as recommended in publication [29]. The consideration of segment length is an important factor in video streaming. The short duration segments like 2 seconds 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 segment like 10 seconds may reduce the overhead, but the client would have a smaller number of opportunities to adapt the video rate. In 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 analyse the performance of the algorithms. The bitrates and quality level ladder are shown in Table 2 thoroughly.

**Table 1.** 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 |

**Table 2.** Video sequences encoding and bitrate ladder

| Index | Animated content | Sports content | Movie content |
|---|---|---|---|
| 1 | 50 kbit/s, 320x240 | 100 kbit/s, 320x240 | 50 kbit/s, 320x240 |
| 2 | 200 kbit/s, 480x360 | 250 kbit/s, 480x360 | 200 kbit/s, 480x360 |
| 3 | 600 kbit/s, 854x480 | 900 kbit/s, 854x480 | 600 kbit/s, 854x480 |
| 4 | 1.2 Mbit/s,1280x720 | 2.0 Mbit/s,1280x720 | 1.2 Mbit/s,1280x720 |
| 5 | 2.5 Mbit/s,1920x1080 | 3.0 Mbit/s,1920x1080 | 2.0 Mbit/s,1920x1080 |
| 6 | 3.0 Mbit/s,1920x1080 | 4.0 Mbit/s,1920x1080 | 2.5 Mbit/s,1920x1080 |
| 7 | 4.0 Mbit/s,1920x1080 | 5.0 Mbit/s,1920x1080 | 3.0 Mbit/s,1920x1080 |
| 8 | 8.0 Mbit/s,1920x1080 | 6.0 Mbit/s,1920x1080 | 6.0 Mbit/s,1920x1080 |

 doi:10.20944/preprints202504.1114.v1



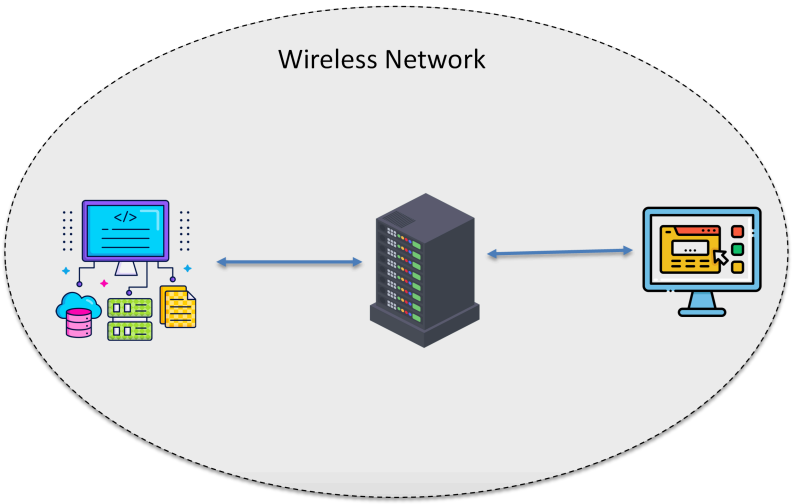


**Figure 2.** Wireless network test bed

### *3.3. Evaluation Test Bed*

Performance evaluation is carried out using dash.js JavaScript-based tool. The library allows streaming 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 and network emulation nodes which are 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.

### *3.4. ABR Algorithms Evaluation*

In this work, the ABR algorithms [18,30,31] 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 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. 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 [32].
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 optimisation principle. The L2A-LL

 doi:10.20944/preprints202504.1114.v1



aims to minimise the video 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 assessment. 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 categorised as stall, rebuffering, switching, and latency. These QoE factors consider various QoE objectives and streaming scenarios [2,8,33].

5. **Low on Latency (LOL+):** This is a heuristic algorithm that uses learning principles to optimise 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-Organising Map) model which takes into consideration various QoE metrics and network variation. The LOL+ playback speed control module is based on a hybrid algorithm which measures latency and buffer level and manages playback speed. The LOL+, QoE evaluation module is responsible for QoE computation based on metrics such as segment bitrate, switching, rebuffer events, latency and playback speed [33,34].

### *3.5. Network Simulation*

Figure 3, Figure 4, Figure 5, and Figure 6, shows 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, (2) abrupt moderate variations in the throughput, and (3) abrupt large variations in the throughput. The motivation behind analysing 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 utilisation. This makes it difficult for algorithms to enhance the user experience under different network conditions.

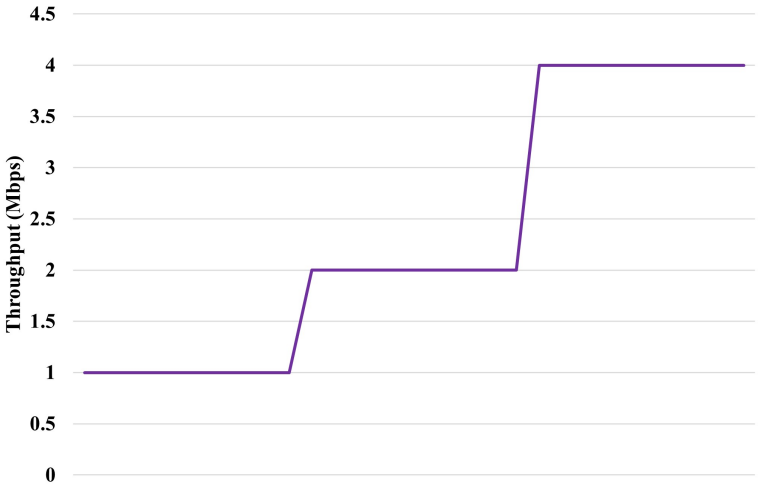


**Figure 3.** Network Profile 1: Bandwidth 1 Mbps - 2 Mbps - 4 Mbps

 doi:10.20944/preprints202504.1114.v1



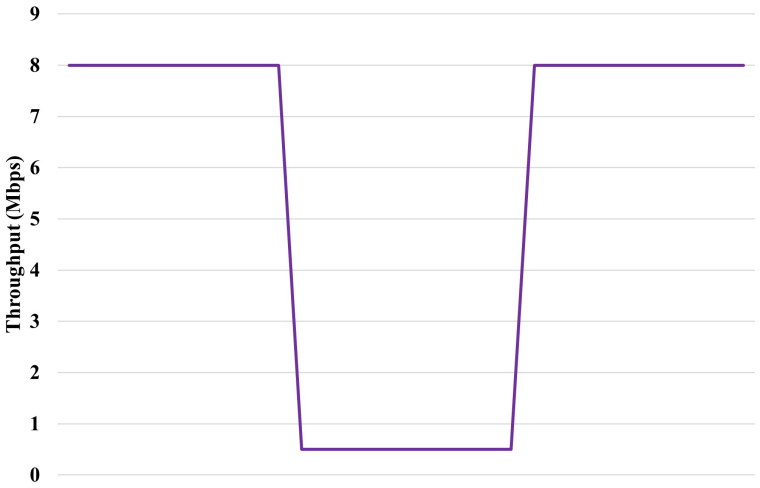


**Figure 4.** Network Profile2: Bandwidth 8 Mbps - 500 kbps - 8 Mbps


**Figure 5.** Network Profile 3: Bandwidth 1 Mbps - 4 Mbps


**Figure 6.** Network Profile 4: Bandwidth 4 Mbps - 1 Mbps

## 4. Results and Discussion

This section discusses the results and analysis of the experiments. This also presents the implications of the results in detail.

 doi:10.20944/preprints202504.1114.v1



### *4.1. Analysis Under Network Profile 1*

Here, we will evaluate the performance of the algorithms in different network profiles. First, we will analyse 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. Figure. 7, illustrates the algorithms' response while streaming Big Buck Bunny. The figure shows that 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 minimise the risk of interruption. When the bandwidth is increased to 4Mbps, all algorithms instantly increase the bitrate to efficiently use the bandwidth. Figure 8, Figure 9, and, Figure 10, shows that the dynamic algorithm has a similar response when streaming all the videos. Figure 11, 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 case of tears of steel and Spark video, it initially increases bitrate but quickly decreases it to avoid playback interruption. The L2A-LL 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 video quality when streaming Tears of Steel and Spark. Figure 11, 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 12, shows that the LoL+ algorithm experienced the lowest number of video rate switches followed traditional DASH algorithms. The L2A-LL algorithm had the highest number of bitrate switches. However, as shown in Figures 7,8,9 and 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.

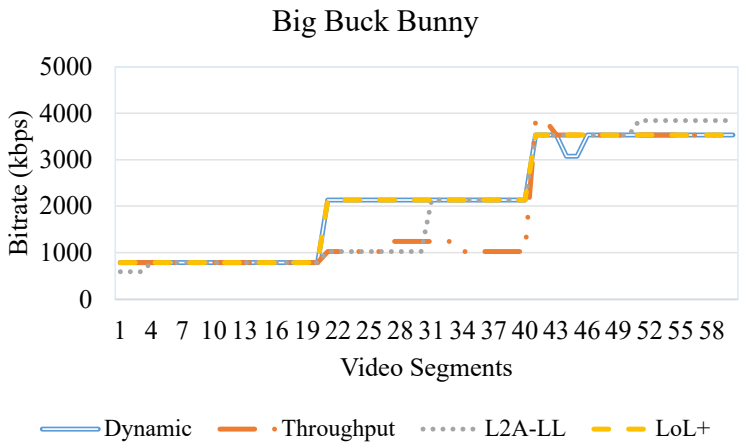


**Figure 7.** Network Profile 1: Bitrates Analysis - Big Buck Bunny

 doi:10.20944/preprints202504.1114.v1



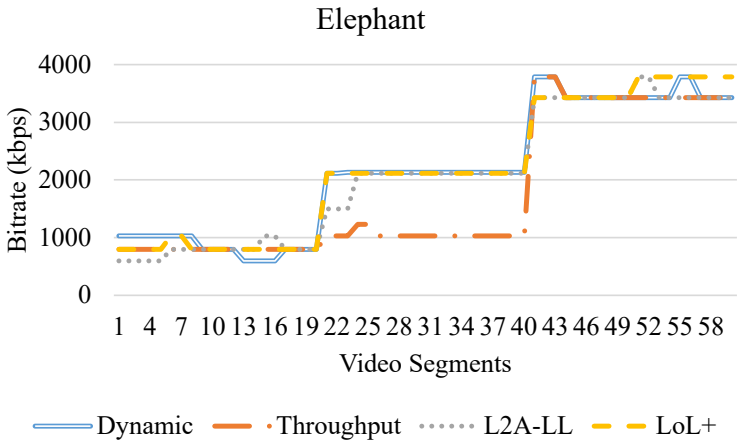


**Figure 8.** Network Profile 1: Bitrates Analysis - Elephant Dream

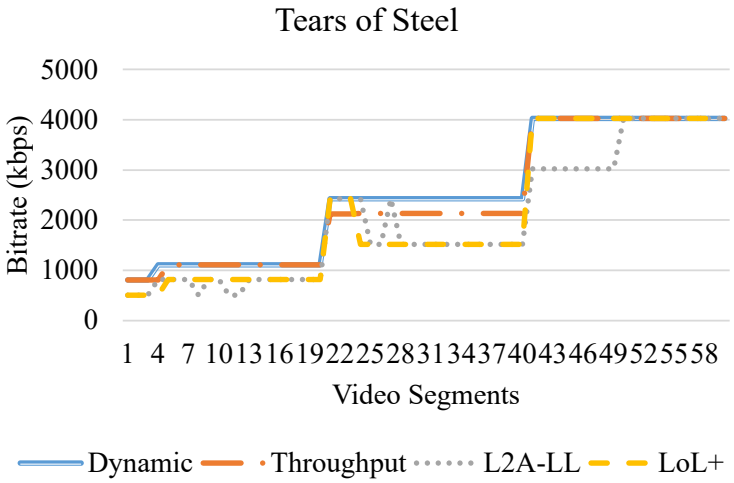


**Figure 9.** Network Profile 1: Bitrates Analysis - Tears of Steel

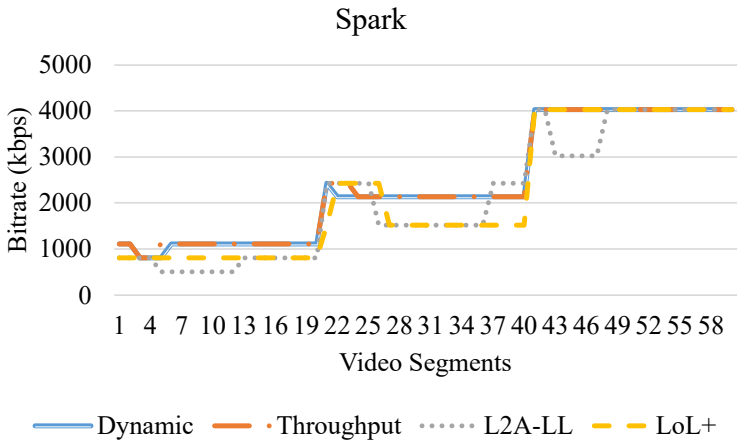


**Figure 10.** Network Profile 1: Bitrates Analysis - Sparks

 doi:10.20944/preprints202504.1114.v1



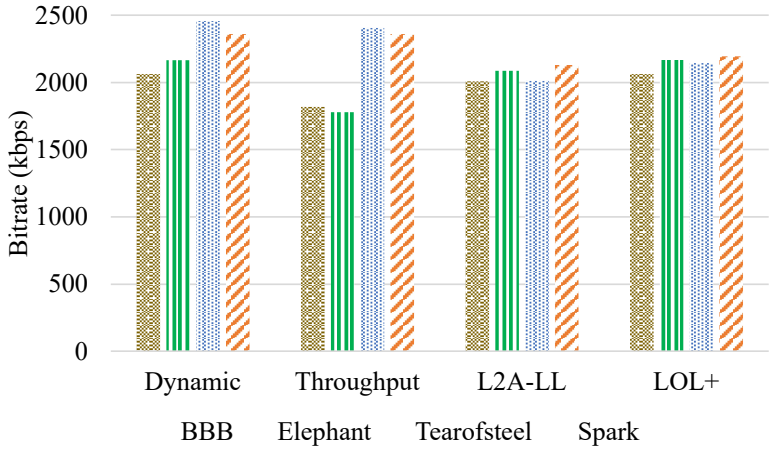


**Figure 11.** Average video bitrates achieved by the algorithm under network profile 1

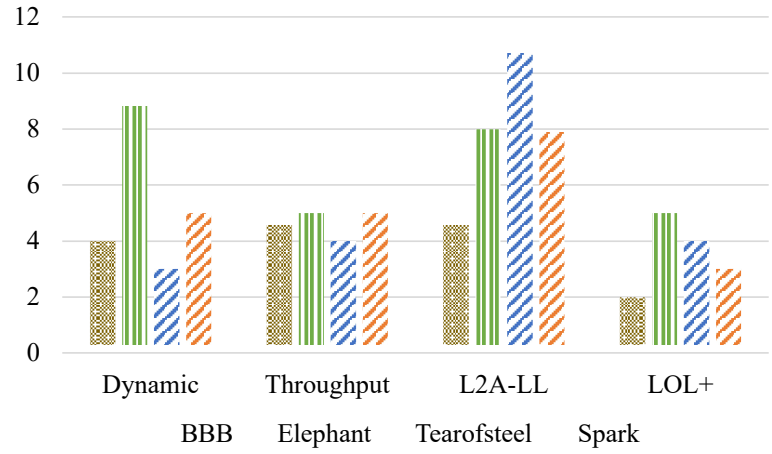


**Figure 12.** Number of switches experienced by the algorithms under network profile 1

*4.2. Analysis Under Network Profile 2*

First, we will analyse 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 aim of evaluating the algorithm under network profile 2, is to analyse their 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 without compromising video quality. 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:

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

Where $(R^k)$ is the bitrate selected for the $k^{th}$ segment and $(T^k)$ is the throughput during the download of the $k^{th}$ segment and $(\tau)$ 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.

 doi:10.20944/preprints202504.1114.v1



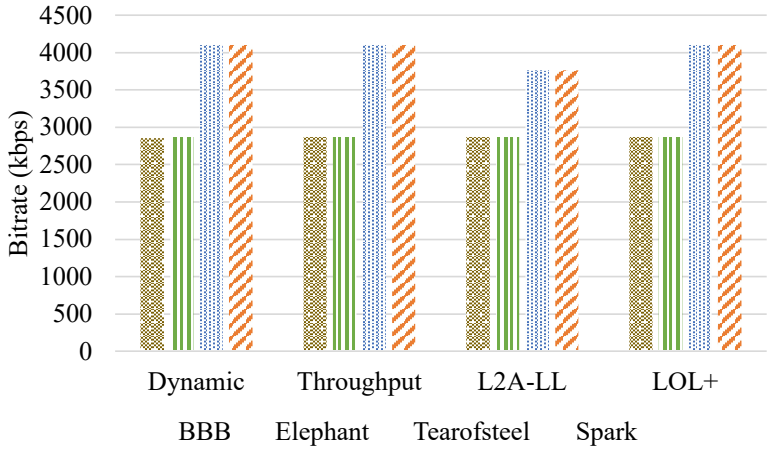


**Figure 13.** Average video bitrates achieved by the algorithm under network profile 2

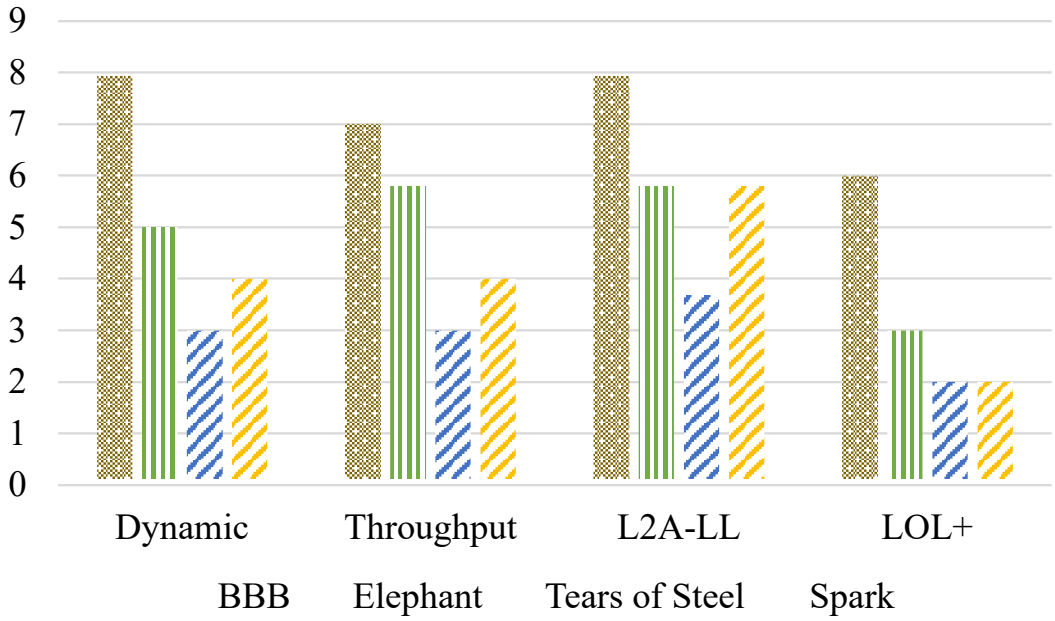


**Figure 14.** Number of switches experienced by the algorithms under network profile 2

 doi:10.20944/preprints202504.1114.v1



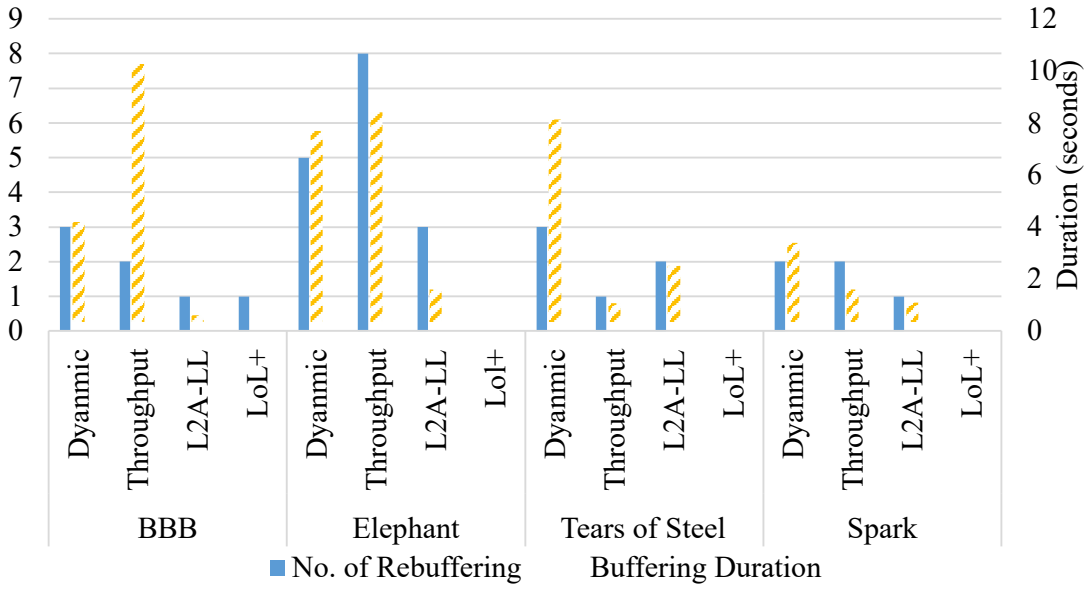


**Figure 15.** Frequency and total duration of rebuffering events encountered by the algorithms when operating under network profile 2

Figure 13, 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 14, 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 15, shows that although the algorithms on average achieved similar video quality, the LoL+ algorithm outperformed the other algorithms by minimising 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.

### *4.3. Impact of Segment Duration*

In this section, we will analyse 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, $\tau$, 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

 doi:10.20944/preprints202504.1114.v1



algorithms, we analyse their behaviour under both 2-second and 10-second segment durations. For each segment setting, we evaluate and compare their performance across network profiles 3 and 4.

1. 2 Seconds Segments
   Here, we will analyse the performance of the algorithms under network profile 3. Figure 16, shows that the BOLA algorithm outperforms the other algorithms by downloading high quality segment 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.

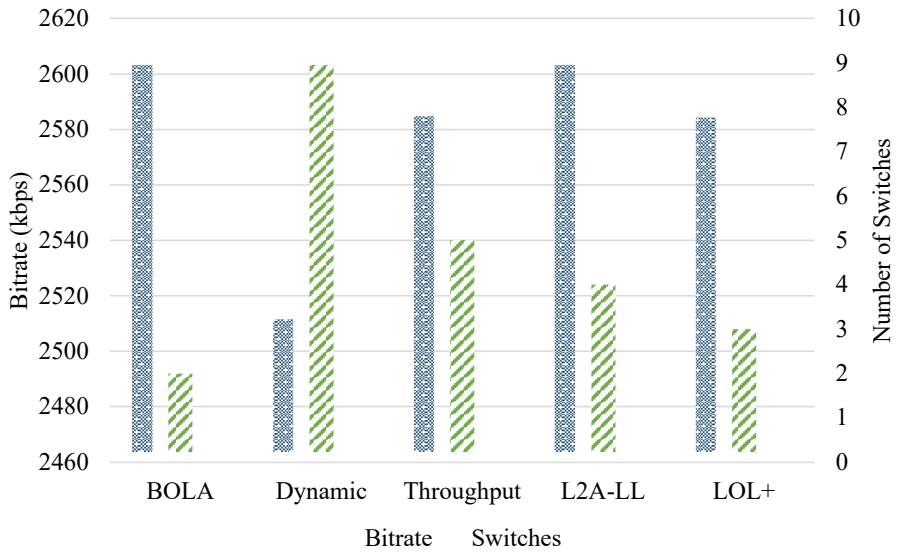


**Figure 16.** Average video rates and bitrate variations observed by the algorithm while streaming 2-second segments under network profile 3

Next, we compare the performance of the algorithms under network profile 4 where the bandwidth drops from 4Mbps to 1Mbps. As evident in the Figure 17, 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 algorithm 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.

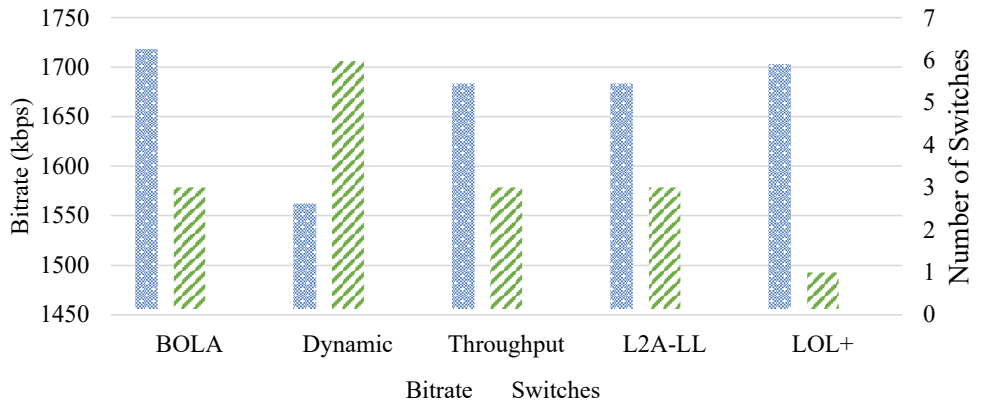


**Figure 17.** Average video rates and bitrate variations observed by the algorithm while streaming 2-second segments under network profile 4

 doi:10.20944/preprints202504.1114.v1



2. 10 Seconds Segments

Next, we increase the segment duration to observe how the algorithms adapt to changes in bandwidth. In the first experiment, we analyse the algorithms under network profile 3. Compared to 2 sec segment experiments, we can clearly observe that the algorithms select lower quality segments. This approach is quite intuitive since the larger segment duration in Eq. (1) increase the risk of playback interruption in the event of a mismatch between throughput and bitrate. The low-latency algorithms outperform the traditional algorithms as the segment duration increases. Figure 18, shows that the BOLA and Throughput algorithms experience frequent bitrate switches; however, these are minor fluctuations between neighbouring bitrates.

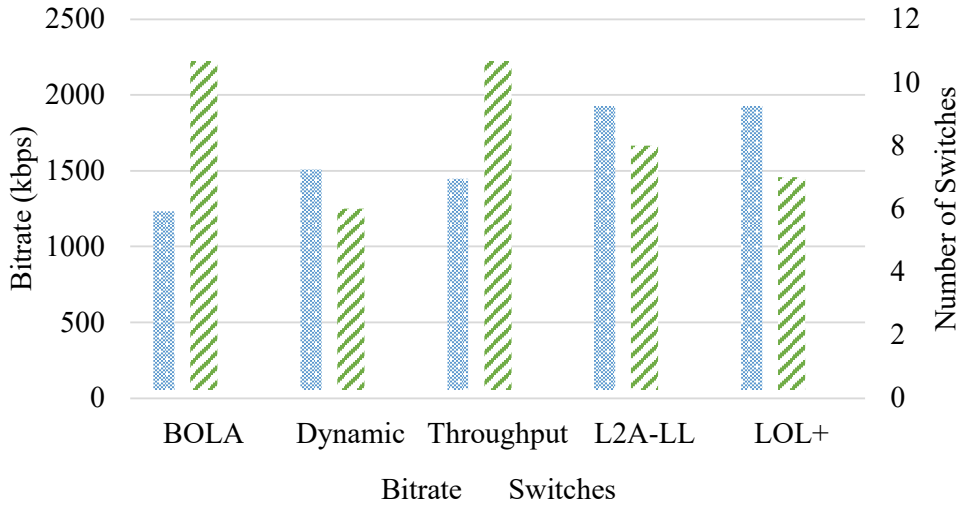


**Figure 18.** Average video rates and bitrate variations observed by the algorithm while streaming 10-second segments under network profile 3

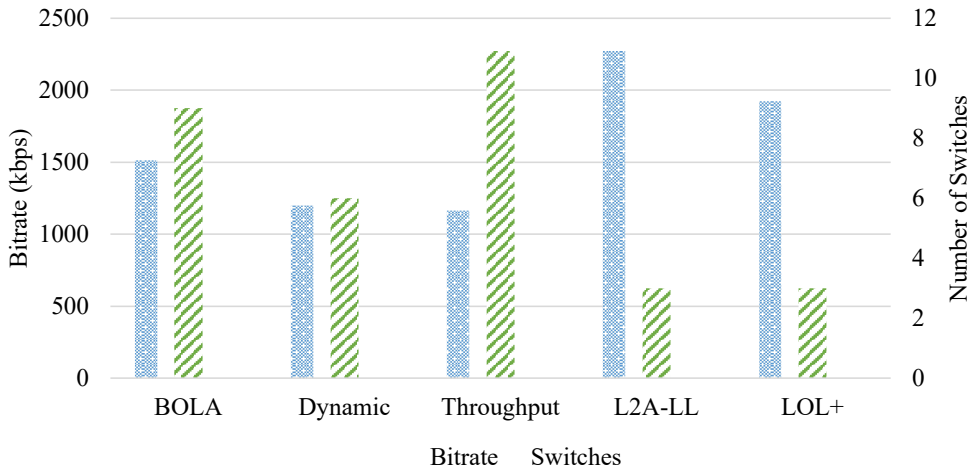


**Figure 19.** Average video rates and bitrate variations observed by the algorithm while streaming 10-second segments under network profile 4

In the next experiment, we analyse the performance of the algorithms under network profile 4. Figure 18, 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 conservative select algorithms. Figure 19, shows that only the Dynamic algorithm is able to avoid any playback interruption, however, at the expense of video quality. Low-latency algorithms achieve higher video quality; however, when throughput drops abruptly, they are unable to prevent the playback buffer from draining. Since the client can only switch bitrates

 doi:10.20944/preprints202504.1114.v1



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 the becomes a challenge for algorithms.

## 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*

The study was carried out to determine the user perception of video quality. The implementation of the study followed the recommendations in [22] and also ITU-T P.910. The Prolific crowd-source platform is used for the evaluation. In the study, 24 video sequences were presented. Of that 4 sequences were of high quality, 4 sequences shown were very low quality. and a total of 16 sequences were presented which are processed using various ABR algorithms. The test sequence has a duration of 1 minute (60 seconds). Each participant was given options to choose from. These were (Bad, Poor, Fair, Good, Excellent). They are required to select the quality of the video according to their perception. The responses are assigned an integer value from (1 to 5). The bad is numbered 1 and the excellent is numbered 5. When test participant watched a sequence, he will respond according to the scale provided and for each sequence 1 MOS score will be recorded.

### *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 20, 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 20, 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, whereas its quality improves for Tears of Stell and Spark. This aligns with the results in (Section 4.1) Figure 11, as the performance algorithm streams segments of the BBB and Elephant videos at lower bitrates. The dynamic algorithm delivers a consistent viewing experience maintaining a MOS score of around 3 across all four videos. Figure 21, shows that the dynamic algorithm 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.

 doi:10.20944/preprints202504.1114.v1



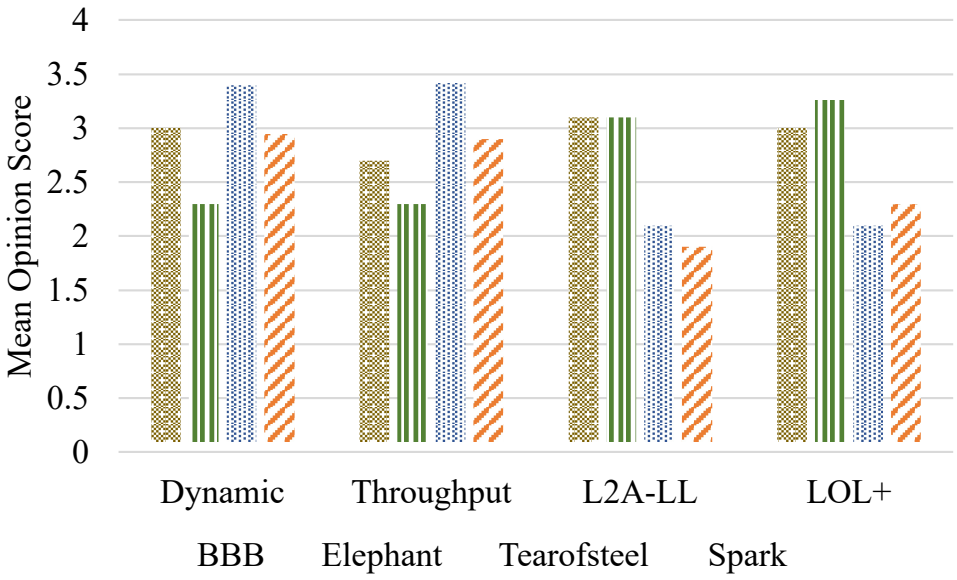


**Figure 20.** Mean Opinion Score of the algorithms for network profile 1

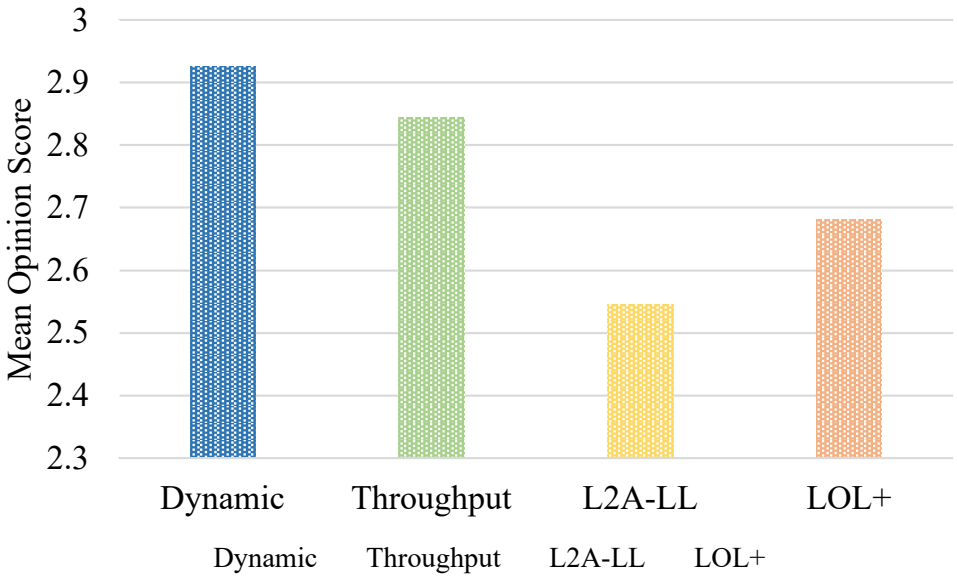


**Figure 21.** Average Mean Opinion Score of the algorithms for network profile 1

## 6. Conclusions

In this research work, evaluation results of adaptive bitrate algorithms (ABR) are presented. The objective assessment is performed to measure the impact on the 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 1MBIT to 4MBIT. The L2A-LL and throughput algorithms delayed increasing the bitrate, which minimises the chances of playback interruptions. The dynamic algorithm outperformed other algorithms by achieving a high video rate. The evaluation is carried out with longer segment duration (10 seconds). The results demonstrate that when streaming longer segments, it is crucial for algorithms to ensure sufficient

 doi:10.20944/preprints202504.1114.v1



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 (MOS) for some set 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 considering (a) additional QoE metrics, (b) incorporating more source videos, and (c) deploying CMAF for low-latency computation.

## Reference


1. Kim, S.; Baek, H.; Kim, D.H. OTT and live streaming services: Past, present, and future. *Telecommunications Policy* **2021**, *45*, 102244. https://doi.org/https://doi.org/10.1016/j.telpol.2021.102244.
2. Lyko, T.; Broadbent, M.; Race, N.; Nilsson, M.; Farrow, P.; Appleby, S. Improving quality of experience in adaptive low latency live streaming. *Multimedia Tools and Applications* **2023**. https://doi.org/10.1007/s11042-023-15895-9.
3. Vlaovic, J.; Rimac-Drlje, S.; Žagar, D.; Filipović, L. Content dependent spatial resolution selection for MPEG DASH segmentation. *Journal of Industrial Information Integration* **2021**, *24*, 100240. https://doi.org/10.1016/j.jii.2021.100240.
4. 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. *International journal of electrical and computer engineering systems* **2021**, *12*, 1–9. https://doi.org/10.32985/ijeces.12.1.1.
5. İren, E.; Kantarci, A. Content Aware Video Streaming with MPEG DASH Technology. *TEM Journal* **2022**, *11*, 611–619. https://doi.org/10.18421/TEM112-15.
6. Alsabaan, M.; Alqhtani, W.; Taha, A. An Adaptive Quality Switch-aware Framework for Optimal Bitrate Video Streaming Delivery. *International Journal of Advanced Computer Science and Applications* **2020**, *11*, 570 – 579.
7. 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), 2022, pp. 1–6. https://doi.org/10.23919/SoftCOM55329.2022.9911338.
8. O'Hanlon, P.; Aslam, A. Latency Target based Analysis of the DASH.js Player. In Proceedings of the Proceedings of the 14th ACM Multimedia Systems Conference, New York, NY, USA, 2023; MMSys '23, p. 153–160. https://doi.org/10.1145/3587819.3590971.
9. Erfanian, A. Optimizing QoE and Latency of Live Video Streaming Using Edge Computing and In-Network Intelligence. In Proceedings of the Proceedings of the 12th ACM Multimedia Systems Conference, New York, NY, USA, 2021; MMSys '21, p. 373–377. https://doi.org/10.1145/3458305.3478459.
10. Xie, G.; Chen, H.; Yu, F.; Xie, L. Impact of playout buffer dynamics on the QoE of wireless adaptive HTTP progressive video. *ETRI Journal* **2021**, *43*. https://doi.org/10.4218/etrij.2020-0184.
11. Taraghi, B.; Hellwagner, H.; Timmerer, C. LLL-CAdViSE: Live Low-Latency Cloud-based Adaptive Video Streaming Evaluation Framework. *IEEE Access* **2023**, *PP*, 1–1. https://doi.org/10.1109/ACCESS.2023.3257099.
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 Proceedings of the 1st Mile-High Video Conference, New York, NY, USA, 2022; MHV '22, p. 32–38. https://doi.org/10.1145/3510450.3517311.

 doi:10.20944/preprints202504.1114.v1




13. Bentaleb, A.; Taani, B.; Begen, A.; Timmerer, C.; Zimmermann, R. A Survey on Bitrate Adaptation Schemes for Streaming Media over HTTP. *IEEE Communications Surveys & Tutorials* **2018**, *21*, 1–1. https://doi.org/10.1109/COMST.2018.2862938.
14. Bentaleb, A.; Timmerer, C.; Begen, A.C.; Zimmermann, R. Bandwidth prediction in low-latency chunked streaming. In Proceedings of the Proceedings of the 29th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video, New York, NY, USA, 2019; NOSSDAV '19, p. 7–13. https://doi.org/10.1145/3304112.3325611.
15. Gutterman, C.L.; Fridman, B.; Gilliland, T.; Hu, Y.; Zussman, G. Stallion: video adaptation algorithm for low-latency video streaming. *Proceedings of the 11th ACM Multimedia Systems Conference* **2020**.
16. Karagkioules, T.; Mekuria, R.; Griffioen, D.; Wagenaar, A. Online learning for low-latency adaptive streaming. In Proceedings of the Proceedings of the 11th ACM Multimedia Systems Conference, New York, NY, USA, 2020; MMSys '20, p. 315–320. https://doi.org/10.1145/3339825.3397042.
17. Bentaleb, A.; Akcay, M.; Lim, M.; Begen, A.; Zimmermann, R. Catching the Moment with LoL+ in Twitch-Like Low-Latency Live Streaming Platforms. *IEEE Transactions on Multimedia* **2021**, *PP*, 1–1. https://doi.org/10.1109/TMM.2021.3079288.
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*. https://doi.org/10.1145/3336497.
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**, *abs/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. *Multimedia Tools and Applications* **2020**. https://doi.org/10.1007/s11042-020-09047-6.
21. Taraghi, B.; Haack, S.Z.; Timmerer, C. Towards Better Quality of Experience in HTTP Adaptive Streaming. *2022 16th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)* **2022**, pp. 608–615.
22. 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**, *PP*, 1–1. https://doi.org/10.1109/ACCESS.2021.3107619.
23. Lebreton, P.; Yamagishi, K. Quitting Ratio-Based Bitrate Ladder Selection Mechanism for Adaptive Bitrate Video Streaming. *IEEE Transactions on Multimedia* **2023**, *PP*, 1–14. https://doi.org/10.1109/TMM.2023.3237168.
24. Lebreton, P.; Yamagishi, K. Predicting User Quitting Ratio in Adaptive Bitrate Video Streaming. *IEEE Transactions on Multimedia* **2020**, *PP*, 1–1. https://doi.org/10.1109/TMM.2020.3044452.
25. 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), 2023, pp. 87–90. https://doi.org/10.1109/QoMEX58391.2023.10178491.
26. 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), 2023, pp. 131–134. https://doi.org/10.1109/QoMEX58391.2023.10178443.
27. 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), 2020, pp. 17–22. https://doi.org/10.1109/SST49455.2020.9264069.
28. 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 Proceedings of the 18th International Conference on Advances in Mobile Computing & Multimedia, New York, NY, USA, 2021; MoMM '20, p. 153–160. https://doi.org/10.1145/3428690.3429173.
29. Lederer, S.; Müller, C.; Timmerer, C. Dynamic adaptive streaming over HTTP dataset. In Proceedings of the Proceedings of the 3rd Multimedia Systems Conference, New York, NY, USA, 2012; MMSys '12, p. 89–94. https://doi.org/10.1145/2155555.2155570.
30. Sani, Y.; Mauthe, A.; Edwards, C. Adaptive Bitrate Selection: A Survey. *IEEE Communications Surveys & Tutorials* **2017**, *PP*, 1–1. https://doi.org/10.1109/COMST.2017.2725241.
31. 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

 doi:10.20944/preprints202504.1114.v1



Conference on Intelligent Technologies (CONIT), 2021, pp. 1–7. https://doi.org/10.1109/CONIT51480.2021.9498318.

32. Marx, E.; Yan, F.Y.; Winstein, K. Implementing BOLA-BASIC on Puffer: Lessons for the use of SSIM in ABR logic. *CoRR* **2020**, *abs/2011.09611*, [2011.09611].
33. 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 Proceedings of the 11th ACM Multimedia Systems Conference, New York, NY, USA, 2020; MMSys '20, p. 321–326. https://doi.org/10.1145/3339825.3397043.
34. Bentaleb, A.; N. Akcay, M.; Lim, M.; C. Begen, A.; Zimmermann, R. Catching the Moment With LoL$^+$ in Twitch-Like Low-Latency Live Streaming Platforms. *IEEE Transactions on Multimedia* **2022**, *24*, 2300–2314. https://doi.org/10.1109/TMM.2021.3079288.