Improving quality of experience in adaptive low latency live streaming

Tomasz Lyko[1] · **Matthew Broadbent[2]** · **Nicholas Race[1]** · **Mike Nilsson[3]** · **Paul Farrow[3]** · **Steve Appleby[3]**

© The Author(s) 2023

Abstract

HTTP Adaptive Streaming (HAS), the most prominent technology for streaming video over the Internet, suffers from high end-to-end latency when compared to conventional broadcast methods. This latency is caused by the content being delivered as segments rather than as a continuous stream, requiring the client to buffer significant amounts of data to provide resilience to variations in network throughput and enable continuous playout of content without stalling. The client uses an Adaptive Bitrate (ABR) algorithm to select the quality at which to request each segment to trade-off video quality with the avoidance of stalling to improve the Quality of Experience (QoE). The speed at which the ABR algorithm responds to changes in network conditions influences the amount of data that needs to be buffered, and hence to achieve low latency the ABR needs to respond quickly. Llama (Lyko et al. [28]) is a new low latency ABR algorithm that we have previously proposed and assessed against four on-demand ABR algorithms. In this article, we report an evaluation of Llama that demonstrates its suitability for low latency streaming and compares its performance against three state-of-the-art low latency ABR algorithms across multiple QoE metrics and in various network scenarios. Additionally, we report an extensive subjective test to assess the impact of variations in video quality on QoE, where the variations are derived from ABR behaviour observed in the evaluation, using short segments and scenarios. We publish our subjective testing results in full and make our throughput traces available to the research community.

Keywords CMAF · DASH · ABR algorithm · Live streaming · Low latency · Quality of Experience · Video quality assessment · Database

1 Introduction

Video streaming has been dominating the Internet in terms of traffic volume for years and it continues to grow, this includes live video streaming, as seen in the Cisco report [14]. At the moment, many video streaming services use HTTP Adaptive Streaming (HAS)

Extended author information available on the last page of the article

technologies such as MPEG Dynamic Adaptive Streaming over HTTP (DASH) [1], Apple HTTP Live Streaming (HLS) [32], and Microsoft Smooth Streaming [44].

In HAS, content is split into short segments, encoded at multiple bitrates and then hosted on a standard HTTP server. A manifest file is created that indicates the encoded bitrates and where the content can be obtained. The client requests the manifest file, then makes HTTP requests for consecutive segments of content at bitrates selected by an Adaptive Bitrate (ABR) algorithm, which takes into account measurements of the network conditions to maximise the Quality of Experience (QoE) of the viewer. When using HAS for live content services, the client can only request segments after they become available on the server, as indicated by the manifest file.

Compared to traditional broadcast methods, live streaming using HAS suffers from high end-to-end latency, that is, there is a long delay between when content is captured by a camera and when the content is displayed on the user's screen. Live content delivered using adaptive streaming is typically shown on the user's screen 30–90s later than the same content delivered using terrestrial or satellite transmission. Such high delay can have a significant negative impact on the QoE of live streaming over the Internet. For example, while watching a football match using HAS, the user might hear their neighbour, who is watching the same match via traditional broadcast television, cheering for a goal a minute before they can see the goal on their own screen, potentially ruining their viewing experience.

The end-to-end latency depends on the entire delivery chain, from content capture, encoding, packaging, encryption, and CDN delivery, to buffering and rendering at the client â€ "with buffering being the main contributor to end-to-end latency in HAS. The client buffer holds the already-fetched segments queued for playback, aiming to absorb changes in network conditions to provide a smooth viewing experience where interruption to playback is minimal. To ensure smooth playback, an ABR algorithm is employed, typically at the client, which measures the network conditions and adjusts the quality at which segments are requested, aiming to maximise the QoE of the streaming session.

In live streaming, where the content is generated in real-time, the maximum number of segments that can be queued in the client buffer depends on the player's target latency setting, as only data that has been captured and encoded but not yet presented to the user can be buffered. Therefore, reducing the client's target latency, and in turn the end-to-end latency, requires an ABR algorithm capable of operating with a small client buffer. Recently, the Common Media Application Format (CMAF) [23] has been standardised, enabling segments to be divided into chunks which combined with HTTP/1.1 Chunked Transfer Encoding (CTE) can aid low latency live streaming.

Designing an ABR algorithm specifically for low latency live streaming is a challenging balancing act. The client buffer level must be kept low to achieve the desired low latency, but this severely limits the potential for the ABR algorithm to react. For example, in a scenario where the client is operating only two segments behind live then the potential maximum buffer level is only two segments. If these segments are 2 s in duration, then the ABR algorithm has at most 4 s to detect and respond to changing network conditions. Hence, in order for an ABR algorithm to perform well, it needs to be able to react quickly to adverse network conditions, while being somewhat cautious when network conditions appear to be improving. In our previous work, we have proposed a new ABR algorithm for low latency adaptive streaming, Llama, designed to operate with a small client buffer, and evaluated it against four prominent on-demand ABR algorithms [28].

In this article we present the results of two extensive evaluations, performed to further assess Llama's suitability for low latency conditions. First, we perform a further quantitative evaluation where we compare the performance of Llama against three state-of-the-art

low latency adaptation solutions, in terms of multiple QoE metrics, across multiple client settings, and in various network scenarios based on CDN logs from a commercial live TV service. Second, we perform a qualitative evaluation by conducting subjective testing of video quality variations based on ABR behaviour patterns observed in the quantitative evaluation. The study expands on current literature by testing changes in video quality using short segments and short scenarios. We publish the results of the subjective test in full as a new video quality assessment database. Additionally, we make the 7000 network traces used in the quantitative evaluation available to the research community. To summarise, the main contributions of this article are:

- Comprehensive evaluation of state-of-the-art low latency ABR algorithms across various network scenarios and multiple QoE metrics
- Dataset consisting of 7000 network traces, based on CDN logs of a commercial sports channel which uses HAS to deliver live content online
- Extensive subjective testing of ABR behaviour patterns observed in the first evaluation, using short scenarios to determine their impact on QoE, with full results published as a new video quality assessment database

2 Background and related work

2.1 ABR Algorithms

Bentaleb et al. [10] published a survey of ABR algorithms used in HTTP Adaptive Streaming. They outlined the main goal of an ABR algorithm is to maximise viewer QoE. This involves trying to maximise the average video quality, while trying to minimise the number of rebuffering events, the time spent in the rebuffering state, and the frequency of changes of video quality. They note that most of these goals are in competition with each other, and therefore require a reasonable trade-off. Recently, the following three low latency ABR algorithms have been published. **Stallion** [22] is a bandwidth-based ABR which measures the moving arithmetic mean and standard deviation of throughput, as well as, latency. **Learn2Adapt** [25] (L2A) is based on Online Convex Optimisation and aims to minimise the latency. **LoL+** [12] offers both heuristic and learning-based approaches, optimising for best QoE. In this article, we have used the learning-based variant of LoL+. These ABR algorithms have been designed to operate in low latency scenarios.

2.2 QoE factors

Barman et al. [9] published the most recent survey on QoE factors in HTTP Adaptive Streaming. They concluded that rebuffering events are the most annoying to users, with both the duration (as seen in [5, 6, 30, 34, 43]) and frequency (as seen in [5, 6, 15, 42, 45]) of rebuffering events being relevant, and hence concluded that an ABR algorithm should aim to minimise both. They also noted that some studies suggest that very short rebuffering events are not noticeable, and therefore, are less annoying to users [39]. The survey also mentions switching quality as another important QoE factor, where a high number of switches can have a negative impact on the overall QoE, as seen in studies [20, 37, 40, and]. They also noted that multi-level quality switches, where the quality changes across more than one quality level, can be detrimental to overall QoE [37, 38]. However, the

impact of changes in video quality has been investigated in less detail than rebuffering, as most studies primarily focus on various forms of rebuffering.

2.3 Latency

Shuai et al. [36] found, that the main contributor to latency in adaptive streaming over HTTP is the client buffer. This buffer holds fetched video segments that are queued for playback. Its size is determined by an ABR algorithm's ability to adapt to changing network conditions in a timely manner. It needs to be large enough to give the ABR algorithm enough time to measure network conditions and change video quality before any rebuffering occurs. Lohmar et al. [27] outlined four sources of delay that are specific to HTTP streaming. The most significant is again the client buffer, and the other three are as follows. Asynchronous fetching of media segments, where a client may issue an HTTP GET request for a segment some time after it is made available. HTTP download time, where segment size and available bandwidth determine how fast a segment can be fetched. Finally, segmentation delay, which is the result of video being divided into segments which introduces a delay of at least one segment duration.

2.4 Common Media Application Format

The Common Media Application Format (CMAF) [23] allows a segment to be created as a sequence of chunks. Whereas a DASH segment must be completely written to a server before it is addressable and can be requested, a segment with CMAF chunks can be requested as soon as the first chunk is written to the server. This reduces the minimum latency in live streaming from one segment to one chunk duration, although it is still only possible to change the video bitrate at segment boundaries. HTTP/1.1 Chunked Transfer (CTE) enables subsequent chunks of a segment to be delivered as soon as they become available without additional requests from the player. Essaili et al. [18] and Viola et al. [41] demonstrated the reduction in latency due to CMAF chunks combined with CTE. In our previous work [29], we found that CMAF can improve ABR performance in low latency scenarios, enabling rebuffering to be reduced by 43%–71%. However, we also found that not all of the ABR algorithms we tested performed better with CMAF.

As chunk delivery at the live edge is restricted by the encoder, estimation of network throughput is difficult for applications that have no direct visibility of any idle periods between chunks. Bentaleb et al. [11] attempted to solve this problem by ignoring throughput measurements for chunks that contain idle time in their download times. As our simulation model calculates the delivery time of chunks correctly, we do not consider this issue further.

3 Llama

In this section, we present a brief summary of our low latency ABR algorithm, Llama, which we have previously proposed in [28].

Llama is based on two design principles. The first design principle is to prioritise the avoidance of rebuffering, as rebuffering is the most detrimental factor to overall QoE. Additionally, rebuffering events in live streaming scenarios can cause the end-to-end latency to increase further, especially if there is no catch-up mechanism employed by the player. To achieve this in low latency live streaming, the ABR algorithm needs to act

---

Algorithm 1 Llama

1: $harmonicMeanSize \leftarrow 20$
2: **for** $segment, n$ **do**
3:      $lastThroughput \leftarrow$ Calculate throughput of the previous segment
4:      $harmonicMean \leftarrow$ Calculate harmonic mean of throughput of the previous $harmonicMeanSize$ segments
5:      **if** $lastThroughput < currentQualityBitrate$ **then**
6:          Reduce quality representation by one if possible
7:      **else if** $harmonicMean > nextQualityBitrate$ **and** $lastThroughput > nextQualityBitrate$ **then**
8:          Increase quality representation by one if possible
9:      **else**
10:          Keep the same quality representation
11:     **end if**
12: **end for**

---

on the first signs of deteriorating network conditions and reduce the quality at which segments are requested before the client buffer is depleted. Hence, to satisfy this design principle a short-term throughput estimation method which tracks the available bandwidth closely is required. The second design principle is to stabilise the video quality. While low video quality will significantly reduce the QoE, a high variance in video quality can also be detrimental to overall QoE. This requires the ABR algorithm to take a longer-term view of throughput and not respond instantly to apparent increases in throughput, and hence avoid temporary increases in quality when the throughput increases for only a short time. This design principle requires a long-term throughput estimation method, which will provide a smoothed bandwidth estimate.

Llama was consequently designed to use two throughput metrics, where each metric is independent and optimised for a different purpose: one for decisions about reducing video quality and one for decisions about increasing video quality. This results in Llama being quick to react to worsening network conditions to avoid rebuffering, but also being careful about increasing video quality to improve video quality stability.

Algorithm 1 demonstrates how Llama works. Llama calculates a first throughput metric as the bitrate at which the most recent segment was fetched, and uses this to decide whether to switch to a lower quality representation, switching down to the next lower quality representation if the bitrate of the current representation is higher than the calculated first metric. This allows Llama to satisfy the first design principle by quickly reacting to worsening network conditions and hence reduce rebuffering. Llama calculates a second throughput metric as the harmonic mean of the bitrate at which each of the past twenty segments had been fetched. This smoothed estimate is used to determine whether to request the next segment at a higher quality, doing so when the calculated second metric is higher than the encoded bitrate of the higher quality segment. This results in stable video quality, satisfying the second design principle. Llama has one parameter, *harmonicMeanSize*, which specifies the number of segments that are used for the harmonic mean calculation, by default set to twenty segments, for both DASH and CMAF - where chunks are aggregated into segments. When there are fewer than *harmonicMeanSize* segments available, all available segments are used for the harmonic mean calculation. The more segments that are used, the more smoothing is applied to the throughput measurement, which provides more conservative decisions of when to switch to a higher quality. The use of fewer segments in this calculation would lead to higher mean video quality at the cost of more variation of quality.

Llama operates differently from the other three recently proposed low latency ABR algorithms. It uses two throughput metrics, each serving a different purpose, instead of just one as found in Stallion. LoL+ calculates future QoE by using a linear equation which takes into account multiple QoE factors, making its performance dependent on the accuracy of the QoE estimation. Llama avoids this issue by focusing on the main two QoE factors, prioritising one over the other, enabling the ABR to effectively avoid rebuffering while trying to provide high and stable quality when the network conditions improve. L2A optimises for latency by calculating the response time, future buffer level, and potential rebuffering for each bitrate selection, making its performance dependent on accurate latency prediction. Llama mitigates this issue by prioritising the avoidance of rebuffering in order to minimise the latency.

4 Quantitative evaluation

In this section, we present a further quantitative evaluation of Llama. In our previous work [28], we proposed Llama and reported an evaluation of it and four prominent on-demand ABR algorithms. Recently, three new low latency ABR algorithms have been proposed: Stallion, Learn2Adapt, and LoL+. In this section, we report on the performance of Llama compared to these new low latency solutions and confirm the suitability of Llama for low latency adaptive streaming. We have published the throughput traces and the encoded content [4, 17] used in this evaluation. These can be used along with the already published simulation model [3] to reproduce our results.

4.1 Methodology

In order to perform an extensive evaluation of all four ABR algorithms in reasonable time, we have used a simulation model which allows for faster than real-time evaluation of streaming sessions. A single run of a four-minute video clip takes only 2–3 seconds in the simulation model compared to taking the whole duration of the video clip when using a real DASH player. In this section we describe our simulation model and its configuration parameters, as well as the selected video encoding parameters and QoE metrics used for performance comparison.

4.1.1 Simulation Model

Our simulation model was developed in the discrete-event network simulator NS-3[1] and is available on GitHub [3]. The implementation is based on an existing model which supports on-demand DASH [31]. We have extended the model to support live DASH, CMAF chunks transmitted using HTTP Chunked Transfer Encoding (CTE), latency configuration, additional ABR algorithms, and accurate traffic shaping between the client and the server. We have confirmed the accuracy of our simulation model by comparing it against a real DASH player in our previous work [29]. It has five parameters: Mode, ABR Algorithm, Throughput Profile, Live Delay, and Join Offset.

---

[1] NS-3 - https://www.nsnam.org/

Mode sets the client into DASH or CMAF mode. When in DASH mode, the client requests segments after they have been fully written to the server; and the server then delivers them as whole segments. When in CMAF mode, the client requests segments as soon as the first chunk of the segment has been written to the server; and the server responds by delivering CMAF chunks as soon as they become available, emulating the delivery of CMAF chunks when transmitted using HTTP Chunked Transfer Encoding (CTE). The ABR Algorithm parameter determines which of the four ABR algorithms, Llama, Stallion, L2A, and LoL+, is to be used by the client for segment selection. Llama operates as described above, while the parameters of each of the other three ABR algorithms are set to the default values presented in their respective papers. The ABR algorithm simply provides the player with the quality at which to request the next segment. The Throughput Trace specifies the file that contains timestamps and bandwidth values in kbps, which are used to adapt the bandwidth of the link between the client and the server over time.

The Live Delay parameter specifies which segment the client requests first: a value of one indicates that the most recent segment available on the server is requested first, two indicates the second most recent available segment is the first to be requested, and so on. The Join Offset parameter determines the time at which the client first requests a segment relative to the time at which segments are made available on the server. When set to 0 s, the client requests a first segment as soon as a segment becomes available on the server, although which segment is requested is determined by the Live Delay parameter. When the segment duration is 2 s and the Join Offset is set to 1 s, the client requests its first segment mid-way between consecutive segments being made available on the server. After the first segment has been delivered, the client requests subsequent segments as soon as the previous one has been delivered, or when they become available on the server, whichever is later. The Live Delay and Join Offset parameters affect the end-to-end latency, that is, the time between a segment being encoded and the segment being played out by the client. Live Delay also affects the potential maximum buffer level of the client, as does Join Offset, but only with CMAF and a chunk duration less than Join Offset.

4.1.2 Video encoding

We selected the first four minutes of the BigBuckBunny[2] movie and encoded it using ×264[3] at bitrates of {400, 800, 1200, 2400, 4800} (kbps) with resolutions {426×240, 640×360, 854×480, 1280×720, 1920×1080}. The encoded video was then segmented using MP4Box[4] into 2 s segments for DASH, and into 2 s segments with 0.5 s chunks for CMAF. The sizes of the resulting segments/chunks were used in the simulation model, and are available here [17].
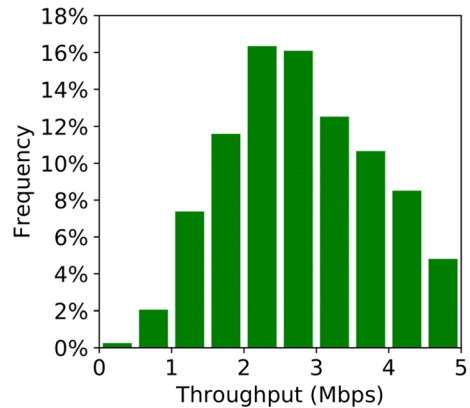
4.1.3 Throughput traces

We used CDN logs from the live BT Sport 1 service to set the bandwidth of the link between the client and server as a function of time. The CDN logs contained the request time, segment size and download time of each segment for every streaming session over

---

[2] BigBuckBunny - https://peach.blender.org/

[3] ×264 - https://www.videolan.org/developers/x264.html

[4] MP4Box - https://gpac.wp.imt.fr/mp4box/

Fig. 1 Histogram showing the distribution of throughput measurements below 5 Mbps in the 7000 throughput traces

a whole day. The streaming sessions included a variety of client devices and connections. From these we produced a throughput trace file for each streaming session in the CDN logs, where each trace file contained pairs of timestamp, equal to the segment request time, and throughput, calculated as the size of the segment divided by its download time.

We discarded trace files shorter than our four-minute video clip and cropped the others to the first four minutes. We discarded trace files that had mean throughput higher than our highest encoding bitrate of 4800 kbps. Additionally, since we wanted to study ABR performance, we discarded trace files where all of the throughput measurements were between the same two encoding bitrates. This left 7000 throughput traces to use in the experiments. The distribution of throughput in these traces is shown as a histogram in Fig. 1, which shows the relative frequencies of throughput measurements below 5 Mbps. About 89% of the measurements are below the highest encoding bitrate of 4.8 Mbps. We have published the 7000 throughput traces on GitHub [4].

4.1.4 QoE metrics

We used the following metrics to evaluate the performance of the four ABR algorithms. **Video Quality** is the arithmetic mean of the indices, in the range 0 to 4, of the quality at which the segments are requested. **Quality Variability** is the standard deviation of the encoded bitrates of the requested segments. **Rebuffer Ratio** is the ratio of the total rebuffering time to playback time. **P.1203 MOS** is the overall Mean Opinion Score (MOS) computed using the ITU-T Recommendation P.1203 objective QoE model[5] which combines bitrate, resolution, frame rate and stall duration into a single value between 1 and 5, estimating the subjective quality of a video without the use of participants [35]. **Average Live Latency**, which is the average time between a segment being generated and the segment being played out by the client. In our simulation model, as the client does not use a catch-up mechanism, rebuffering events increase the Live Latency by the rebuffering duration.
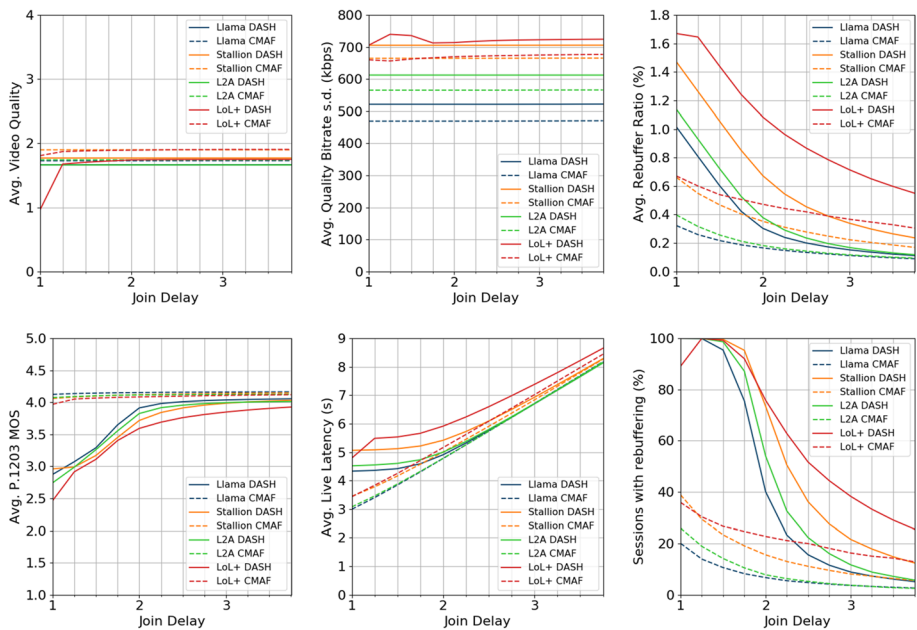
---

[5] https://github.com/itu-p1203/itu-p1203

Fig. 2 The performance of the four low latency ABRs when used in DASH and CMAF clients, in terms of average Video Quality, Quality Variability, Rebuffer Ratio, P.1203 MOS, Live Latency, and the percentage of sessions experiencing rebuffering

4.2 Results

In this section, we present the results of our evaluation. We compare the performance of Llama against low latency ABR algorithms: Stallion, L2A, and LoL+ in both DASH and CMAF modes. We have used the following parameters. The Live Delay was set to values ranging from 1 to 3 segments with the Join Offset set to 0 s, 0.5 s, 1 s, and 1.5 s for each Live Delay setting. All 7000 throughput traces were used for each combination of Live Delay and Join Offset resulting in 84,000 runs per ABR algorithm for one mode of client. In total, the content was delivered to the client 672,000 times. In both modes the segment duration was 2 s. In CMAF mode the chunk duration was 0.5 s, resulting in four chunks per segment.

We report the performance of all ABR algorithms using the QoE metrics previously described, averaged over the 7000 runs with different throughput profiles. We combined Live Delay and Join Offset into a single value termed Join Delay, being equal to the time between the first segment being created and it being requested, measured in segment periods, and calculated as Live Delay added to Join Offset divided by two seconds. Figure 2 shows the performance of all four low latency ABR algorithms when used in DASH and CMAF clients, in terms of average Video Quality, Quality Variability, and Rebuffer Ratio, P.1203 MOS, Live Latency, as well as, the percentage of sessions experiencing rebuffering for each value of Join Delay.

4.2.1 DASH

Buffering performance Llama achieved the lowest average Rebuffer Ratio for all values of Join Delay. At the lowest Join Delay, Llama reduced the average Rebuffer Ratio by 0.45, 0.12, and 0.65 when compared to Stallion, L2A, and LoL+ respectively. As Join Delay increased, Llama's improvement in terms of average Rebuffer Ratio over the other ABRs decreased. At the highest Join Delay, Llama reduced the average Rebuffer Ratio by 0.13, 0.01, and 0.44 when compared to Stallion, L2A, and LoL+ respectively.

Llama outperformed the other ABRs in terms of the percentage of sessions experiencing rebuffering at Join Delay of 1.25 and higher, where on average it had 14.58, 5.03, and 23.93 lower percentage of sessions with rebuffering than Stallion, L2A, and LoL+ respectively. At the lowest Join Delay, LoL+ had the lowest number of sessions experiencing rebuffering at 89%, compared to 100% for the other three ABRs. However, the average Rebuffer Ratio achieved by LoL+ at this setting was significantly higher than for the other three ABRs, suggesting that even though fewer sessions were affected by rebuffering, it was significantly more severe.

Video quality performance Llama, Stallion, and L2A achieved average Video Quality of 1.67, 1.77, and 1.66 respectively across all values of Join Delay. LoL+ achieved average Video Quality of 0.96 at the lowest Join Delay, and 1.68–1.75 for the remaining values of Join Delay. LoL+ was the only ABR for which the average Video Quality varied across different values of Join Delay. Llama and L2A achieved 0.10 and 0.11 respectively lower average Video Quality than Stallion for all values of Join Delay, as well as, on average 0.07 and 0.08 respectively lower average Video Quality than LoL+ for Join Delay values of 1.25 and higher.

Llama, Stallion, L2A, and LoL+ achieved average Quality Variability of 521–522, 705–706, 612, and 706–740 respectively for all values of Join Delay. Llama achieved on average 184, 91, and 201 lower Quality Variability than Stallion, L2A, and LoL+ respectively across all values of Join Delay. Llama did not achieve the highest average Video Quality across all values of Join Delay, however, it did achieve the lowest Quality Variability across all values of Join Delay. This suggests that Llama offered the most stable Video Quality out of all four low latency ABRs at the cost of the reduced average Video Quality.

Overall performance Llama achieved a slightly higher average P.1203 MOS than the other ABRs for Join Delay values of 1.25 and higher, where on average it improved the P.1203 MOS by 0.09, 0.06, and 0.21 over Stallion, L2A, and LoL+ respectively. At the lowest Join Delay, Stallion achieved 0.09, 0.22, and 0.5 better P.1203 MOS than Llama, L2A, and LoL+ respectively. However, it also had 0.73 s, 0.54 s, and 0.27 s higher average Live Latency than Llama, L2A, and LoL+ respectively.

Llama achieved the lowest average Live Latency for Join Delay values of 1–2.75, the joint lowest with L2A for Join Delay values of 3–3.25, and the second lowest average Live Latency, 0.01 s higher than L2A, for Join Delay values of 3.5 and 3.75. Overall, when used in a DASH client at most settings, Llama achieved moderately lower Live Latency, which is crucial in low latency live streaming, and slightly improved P.1203 MOS, which reflects the Quality of Experience.

4.2.2 CMAF

Rebuffering performance Llama achieved the lowest average Rebuffer Ratio for Join Delay values of 1–3.25, and the joint lowest with L2A for Join Delay values of 3.5–3.75. At the lowest Join Delay, Llama achieved average Rebuffer Ratio 0.34, 0.08, and 0.35 lower than Stallion, L2A, and LoL+ respectively. As the Join Delay increased, these differences decreased and L2A achieved the same average Rebuffer Ratio as Llama. At the highest Join Delay, Llama and L2A both achieved 0.08 and 0.21 lower Rebuffer Ratio than Stallion and LoL+ respectively.

In terms of the percentage of sessions experiencing rebuffering, Llama outperformed the other three low latency ABRs at Join Delay values of 1–3, where on average it achieved 10.04, 2.14, and 15.36 lower percentage of sessions experiencing rebuffering than Stallion, L2A, and LoL+ respectively. At Join Delay values of 3.25–3.75, Llama had the second lowest percentage of sessions experiencing rebuffering, 0.03–0.17 higher than L2A.

Video quality performance Llama, Stallion, and L2A achieved average Video Quality of 1.73, 1.9, and 1.74 respectively across all values of Join Delay. LoL+ achieved an average Video Quality of 1.81–1.91 across all values of Join Delay, and was the only low latency ABR with average Video Quality that varied across different values of Join Delay. Llama, on average achieved 0.17, 0.01, and 0.16 lower average Video Quality than Stallion, L2A, and LoL+ respectively across all values of Join Delay.

Llama, Stallion, L2A, and LoL+ achieved average Quality Variability of 469–471, 665, 565–566, and 657–677 respectively for all values of Join Delay. On average, Llama achieved 195, 96, and 200 lower Quality Variability than Stallion, L2A, and LoL+ respectively for all values of Join Delay. Llama did not achieve the highest average Video Quality for all values of Join Delay, however, it did achieve the lowest Quality Variability across all values of Join Delay. Again, just as in the DASH client, Llama offered the most stable Video Quality out of all four low latency ABRs at the cost of the reduced average Video Quality.

Overall performance Llama achieved marginally higher P.1203 MOS than the other ABRs across all values of Join Delay. At the lowest Join Delay, the average P.1203 MOS for Llama, Stallion, L2A, and LoL+ was 4.12, 4.06, 4.07, and 3.97 respectively. As Join Delay increased, Llama's improvement over other ABRs in terms of average P.1203 MOS decreased. At the highest Join Delay, the average P.1203 MOS achieved by Llama, Stallion, L2A, and LoL+ equalled to 4.16, 4.15, 4.13, and 4.12 respectively.

Llama achieved the lowest average Live Latency for Join Delay values of 1–1.75, the joint lowest with L2A for Join Delay value of 2, and the second lowest average Live Latency, only 0.01 s higher than L2A, for Join Delay values of 2.25–3.75. Across all values of Join Delay, Llama achieved on average 0.21 s, 0.01 s, and 0.35 s lower average Live Latency, while achieving 0.03, 0.04, and 0.07 better average P.1203 MOS than Stallion, L2A, and LoL+ respectively. Overall, when used in a CMAF client, Llama moderately outperformed the other low latency ABRs in terms of minimising the Live Latency and maximising the P.1203 MOS.
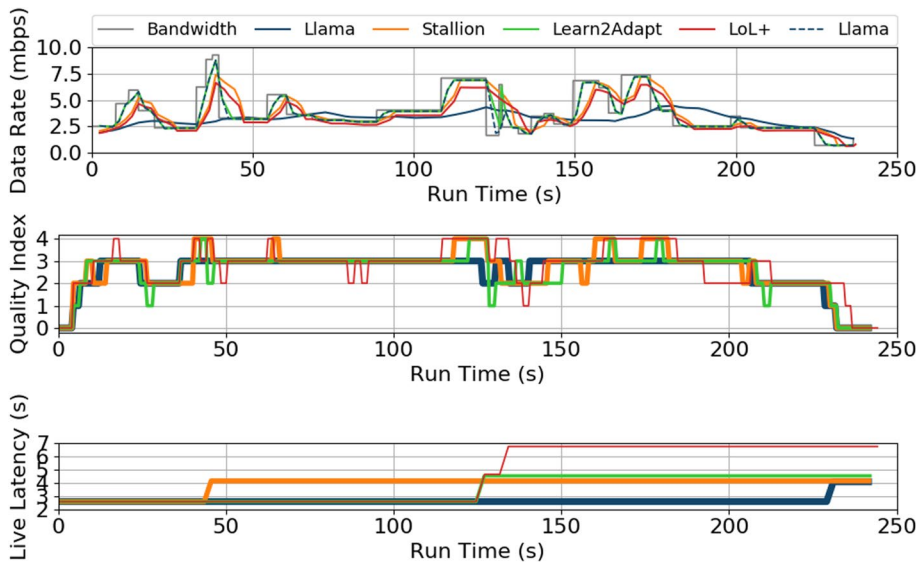
Fig. 3 Performance of the four ABR algorithms in a single scenario in terms of throughput estimation, video quality and live latency

4.2.3 Single scenario

Figure 3 shows the performance of the four ABR algorithms in a single scenario consisting of a CMAF client configured with a Live Delay of 1 segment and Join Offset of 0 s. The top chart shows the available bandwidth and the throughput estimated by each ABR. There are two lines for Llama, the dashed line shows the throughput of the most recent segment and the solid line shows the harmonic mean of 20. The next two charts show the quality and live latency. Since no catch-up mechanism is employed at the client, an increase in live latency indicates a rebuffering event of the same duration.

In this throughput trace, at 33 s the bandwidth increased from 2.3 to 6.2mbps, then 3 s later increased further to 8.8mbps and 2 s later further still to 9.3mbps, and then after only a further 2 s it dropped to 3.3mbps. Stallion overestimated the available bandwidth after this temporary increase occurred and selected the highest quality, causing rebuffering of 1.6 s. L2A increased the quality to the highest, but only briefly, and quickly switched down by two quality indexes to avoid rebuffering. LoL+ chose the highest quality briefly twice, and gradually switched down by two quality indexes to avoid rebuffering. Llama did not increase the quality when the bandwidth improved briefly, and hence did not need to switch down to avoid rebuffering, resulting in constant video quality.

At 123s, the bandwidth dropped from 7.1 to 1.6mbps, after 4 s it briefly increased to 6.5mbps for 1s and then dropped to 2.4mbps. Stallion, operating at higher buffer level due to increased live latency by the previous rebuffering event, gradually reduced quality by two indexes to avoid rebuffering. L2A was too slow to switch down from the highest quality resulting in rebuffering of 1.9 s, after which it reduced the quality by 3 indexes instantly, and slowly increased the quality ignoring the spike in bandwidth. LoL+ was also too slow to reduce the quality resulting in rebuffering of 2 s, after which, it reduced the quality by only one index and when the bandwidth spiked, it again increased the quality

to the highest leading to further rebuffering of 2.1s. Llama avoided rebuffering by quickly reducing the quality by one index, then when the bandwidth spiked, it increased the quality by one index, and avoided the second potential rebuffering event by decreasing the quality by one index again.

At 225 s, the bandwidth dropped from 2.4 to 0.7mbps. Llama decreased the quality by one index but suffered rebuffering anyway. Stallion and L2A did exactly the same, however, but avoided rebuffering as they were already operating at higher buffer level due to previous rebuffering which increased their live latency. LoL+ increased the quality and avoided rebuffering as it was operating at much higher buffer level, with 2.2–4.1 s higher live latency than other ABRs.

Throughout the entire trace, Llama never switched to the highest quality while the three other ABRs frequently tried to reach the highest quality leading to frequent quality switches. In some cases, such as at around 40s, the other three ABRs had to over-correct their decision by reducing the quality to below the pre-increase level to avoid rebuffering, leading to even higher video quality instability. Llama achieved 0.2, 0.3, and 0.4 better P.1203 MOS while having 1.2s, 0.9 s, and 1.9 s lower average Live Latency than Stallion, L2A, and LoL+ respectively.

4.3 Discussion

In our quantitative evaluation we have compared the performance of Llama against three low latency ABR algorithms: Stallion, L2A, and LoL+. In DASH, at Join Delay of 1, Llama achieved 0.19–0.73 s lower average Live Latency than other ABR algorithms. Llama achieved 0.13 and 0.41 better P.1203 MOS than L2A and LoL+, while reducing rebuffering by 11% and 39% respectively. Stallion achieved 0.09 better average P.1203 MOS than Llama, at the cost of 0.73 s higher average Live Latency and 31% more rebuffering. At Join Delay of 2, Llama reduced the average Live Latency by 0.09–1 s, while having 0.08–0.32 better average P.1203 MOS and 21–72% less rebuffering than other ABR algorithms. At Join Delay of 3, Llama reduced the average Live Latency by 0–0.68 s and improved the average P.1203 MOS by 0.04–0.19 while reducing rebuffering by 12–79% when compared to other ABR algorithms.

In CMAF, at the Join Delay of 1, Llama achieved 0.09–0.46 s lower average Live Latency, while having 0.05–0.15 better average P.1203 MOS and 20–52% less rebuffering than other low latency ABR algorithms. At Join Delay of 2, Llama reduced the average Live Latency by 0–0.38s, while improving the average P.1203 MOS by 0.03–0.07 and reducing rebuffering by 6–64% when compared to other ABR algorithms. At Join Delay of 3, Llama achieved 0.13s and 0.31s lower average Live Latency, while also improving the average P.1203 MOS by 0.02 and 0.05 and reducing rebuffering by 50% and 70% when compared to Stallion and LoL+ respectively. L2A achieved 0.01s lower average Live Latency than Llama, at the cost of 0.04 lower average P.1203 MOS and 8% more rebuffering.

In summary, for all values of Join Delay, Llama moderately outperformed the other low latency ABR algorithms, in terms of maximising P.1203 MOS and minimising the Live Latency, in both DASH and CMAF clients. On average across all values of Join Delay, Llama achieved 0.08 better P.1203 MOS and 0.31s lower Live Latency than the three other low latency ABR algorithms. These results indicate that Llama is a suitable ABR algorithm for low latency conditions, as its performance is on a par with the other three low latency solutions.

4.3.1 Behaviour patterns

In our results, and especially in the single scenario analysis, we observed that the tested ABR algorithms exhibited a number of behaviour patterns, five of which we highlight below.

- When the bandwidth improved temporarily, most of the ABR algorithms increased the video quality quickly but only for a very short time. Llama avoided such temporary up-switches as it was slow to increase the video quality.
- When the bandwidth decreased temporarily, in some instances the ABR algorithms would unnecessarily reduce the quality level to avoid rebuffering. As Llama decreases the video quality on the first signs of bandwidth deterioration, it displayed this behaviour more often than the other ABR algorithms.
- In some cases frequent quality switches were observed, causing a higher average bitrate. This was especially apparent in cases where the bandwidth improved for a short period of time. Llama performed fewer quality switches, and achieved the lowest average Quality Variability, but at the cost of a lower average bitrate.
- Multi-level quality switches in some cases were performed instantly, that is, the video quality was changed by more than one level between two consecutive segments. Llama changes the video quality gradually regardless of the direction or magnitude, meaning, the video quality is changed by at most one level every segment duration.
- When the bandwidth reduces, an ABR algorithm may reduce the quality, increase it too soon and then reduce it again, whereas another ABR algorithm may reduce the quality only once, but with the same total duration at the lower quality.

5 Qualitative evaluation

In this section, we present a qualitative evaluation of Llama. We have performed extensive subjective testing to investigate how users perceive various variations in video quality. We have published our results in full [26]. Firstly, we describe the impairment scenarios we constructed from the ABR behaviour patterns presented in the last section. Secondly, we describe the methodology used in the subjective test. Then we present and discuss the results of the subjective test. Finally, we summarise this section and describe how our study fits into the relevant literature.

5.1 Impairment scenarios

We have created 28 quality impairment scenarios based on ABR behaviour highlighted in the last section. Figure 4 shows all 28 quality impairment scenarios used in the subjective testing. Each graph shows a single scenario, with the video quality index of each segment plotted and the resulting average bitrate shown in the yellow box. The video quality index corresponds to bitrates of {400, 800, 1200, 2400, 4800} (kbps) with encoding resolutions {426×240, 640×360, 854×480, 1280×720, 1920×1080}. The duration of each scenario was 14s, comprising seven 2s segments.

We divided the scenarios into seven groups, each labelled according to the ABR behaviour pattern tested. Group 'Spike' includes scenarios with different magnitudes of
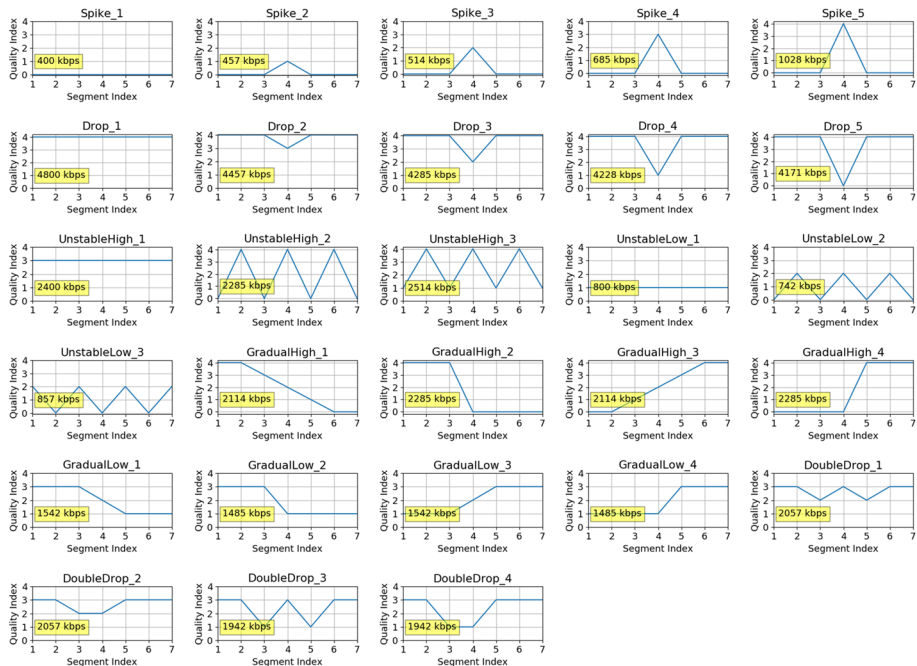
Fig. 4 The quality impairment scenarios used in the subjective test. Each graph shows, for a single scenario, the video quality index for each segment, and, in the yellow box, the resulting average bitrate

temporary increase in video quality. Group 'Drop' includes different magnitudes of temporary decrease in video quality. Groups 'UnstableHigh' and 'UnstableLow', for two levels of average video quality, include constant video quality and quality variations with two magnitudes. Groups 'GradualHigh' and 'GradualLow' include sudden and gradual changes in video quality, with the former having a greater magnitude of change. Group 'DoubleDrop' includes a single quality drop and two shorter quality drops with the same total duration of quality drop.

5.2 Methodology

Testing procedure We have performed the subjective test using an Absolute Category Rating (ACR) method described in ITU-T Recommendation P.910 [33]. In ACR, also called the single stimulus method, participants are presented with one test clip at a time. The participants are asked to rate each test clip, immediately after it has been presented, on the following five-level scale: {Bad, Poor, Fair, Good, Excellent}. We have processed the responses using the methodology described in ITU-R Recommendation BT.500–13 [24], by firstly mapping each response to the integer scale 1–5 with bad being mapped to 1, then calculating, for each test clip: a Mean Opinion Score (MOS) as the arithmetic mean of all scores for the clip, and a (95%) Confidence Interval.

Experiment set-up The subjective test was performed remotely due to the social restrictions introduced in response to the global COVID-19 pandemic. We developed an online

Fig. 5 Spatial and Temporal Information of each source clip, calculated as specified in ITU-T Rec. P.910
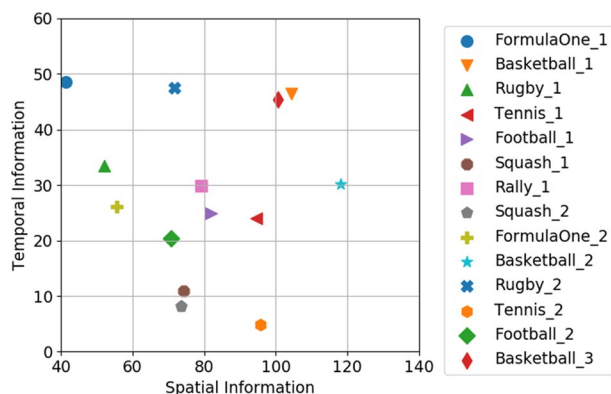
survey in which the participant was presented with one question per page, with each question containing a test clip and the five-level scale to record the participant's response. The test clip was fully downloaded before the participant could begin playback to ensure that no unintended rebuffering occurred during playback. Once the test clip was fetched, the participant needed to click on the playback button to begin watching the clip in full-screen mode. After the playback finished, full-screen mode was exited and the scoring scale was displayed. The participant could watch the test clip again, or rate it and proceed to the next test clip, after which the participant could not go back. All responses were anonymous and each participant was restricted to only one response per test clip.

Content selection We selected a range of source clips for our subjective testing. All clips were captured from the live BT Sport service which offers a variety of live sports content. The following set of 14 source clips was selected: {Tennis_1, Tennis_2, FormulaOne_1, FormulaOne_2, Rugby_1, Rugby_2, Football_1, Football_2, Squash_1, Squash_2, Rally_1, Basketball_1, Basketball_2, Basketball_3}. We have selected sports content as it is the most common type of content transmitted using live streaming. Each source clip was 14 s long and encoded at 25 FPS. Figure 5 shows Temporal and Spatial Information of each source clip calculated as specified in ITU-T Recommendation P.910.

Test clips For each impairment scenario, test clips were created by concatenating DASH segments of appropriate video quality, without re-encoding, using FFmpeg.[6] Source clips were encoded and segmented into DASH 2s segments using the same methodology as in Section 4.1.2, with video quality levels corresponding to the following bitrates {400, 800, 1200, 2400, 4800} (kbps) and resolutions {426×240, 640×360, 854×480, 1280×720, 1920×1080}, using VBR encoding with maxBitrate and vbvBuffer parameters set to the bitrates. Additionally, each segment was scaled back to the highest resolution to avoid playback issues in some web browsers.

Test sequences We created 6 test sequences by using 3 sets of test clips, each with 2 orders of presentation. Each participant observed only one test sequence. We created 3 sets of impaired content, A, B, and C, each including scenarios from every impairment

---

[6] FFmpeg - https://ffmpeg.org/

Table 1 Table showing the source clips assigned to each impairment group within each impairment set. Each participant will watch test clips from only one impairment set, where all scenarios within each group are based on a single source clip

Scenario	Impairment Set A	Impairment Set B	Impairment Set C
Spike_1–5	FormulaOne_1	Squash_2	Tennis_1
Drop_1–4	Basketball_1	FormulaOne_2	
Drop_5	Basketball_1	FormulaOne_2	Tennis_1, FormulaOne_1, Basketball_2
UnstableHigh_1–3	Rugby_1	Basketball_2	FormulaOne_1
UnstableLow_1–3	Tennis_1	Rugby_2	Basketball_2
GradualHigh_1–4	Football_1	Tennis_2	Rally_1
GradualLow_1–4	Squash_1	Football_2	Football_1
DoubleDrop_1–4	Rally_1	Basketball_3	Squash_1

Fig. 6 Age distribution of the recruited participants
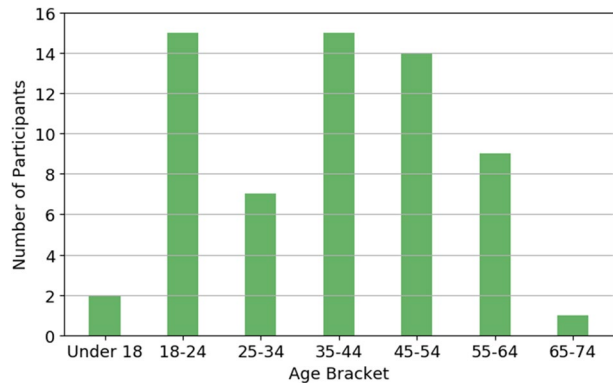
group. We randomly created 2 different orders of presentation for these sets, with the single constraint that clips from the same impairment group were not shown consecutively, and hence that the participant never saw the same source clip consecutively. Table 1 shows the source clips assigned to each impairment group and set. To allow comparisons to be made between scenarios within each impairment group, two unique source clips were assigned to each of the seven groups and presented with each scenario in that group. In each group, one of these source clips was allocated to a first set of impaired content (A), and the other to a second set (B). A viewer therefore observed all of the scenarios of one group with the same source clip.

To allow some comparison between scenarios of different impairment groups, we defined one more set of impaired content (C) by creating two super groups, first based on impairment groups {Spike, Drop_5, UnstableHigh, and UnstableLow}, and the second on groups {GradualHigh, GradualLow, DoubleDrop}. Within each super group, each impairment group was assigned an additional source clip, which corresponds to a source clip from impairment set A or B of another impairment group within the same super group. For example, group DoubleDrop was assigned source clip Squash_1 which belongs to the impairment set A of group GradualLow, allowing for comparison of MOS between all 8
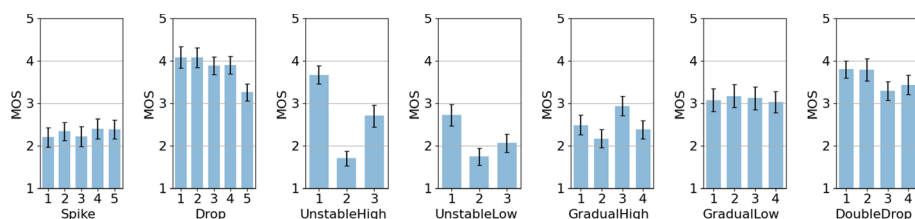
Fig. 7 Mean Opinion Score of common test clips within each impairment group with Confidence Interval of 95% plotted in the error bars

scenarios from the two groups. Preliminary testing suggested that there was little benefit in more extensive comparisons of scenarios Drop_1–4, and hence we decided to include only scenario Drop_5 within the first super group. Drop_5 was assigned 3 additional source clips, each corresponding to source clips assigned to impairment groups {Spike, Unstable-High, and UnstableLow}.

Participant recruitment We recruited 63 participants, with the median age group of 35–44, mean age of 39.6, and standard deviation of 14.7. The age distribution of participants is shown in Fig. 6. 42 participants were male, 20 were female, and 1 preferred not to say. Additionally, we asked the participants how often they watch live content online, with 35 reporting daily consumption, 16 once a week, 9 once a month, 2 once a year and 1 never. Participation in the study was voluntary. We did not offer rewards for completing the survey to try to maximise the reliability of the responses given. 14% of responses were completed on a mobile device, and 86% on desktop or laptop. We have screened the responses using the method described in ITU-R Recommendation BT.500–13 [24] which resulted in one response being rejected.

5.3 Results and Discussion

In this section we present and discuss the results of our extensive subjective testing. First, we compare the MOS of the impairment scenarios within each impairment group. Later, we compare the MOS of the impairment scenarios within each super group.

5.3.1 Comparison of impairment scenarios within groups

Figure 7 shows the MOS of clips for each impairment scenario, divided by impairment group. The error bars show the Confidence Interval of 95%, used to determine the statistical significance of differences in MOS, where overlapping confidence intervals are perceived as statistically insignificant. As shown in Table 1, the same three source clips were used for each scenario within an impairment group, except for group 'Drop' for which only two source clips were used. A further three source clips were used with scenario 'Drop_5', but these results are excluded from Fig. 7.

Group Spike contains impairment scenarios which experience a temporary increase in video quality of varying magnitude halfway through the clip. In scenario 1, the video quality is constantly at the lowest level, while in scenarios 2–5, the video quality is at the lowest level except for the middle segment which has different higher levels of video quality. From the Spike chart in Fig. 7 we can observe that the differences in MOS were insignificant,

with the range equal to only 0.2 - lower than the Confidence Interval of each scenario. Scenario 5 had an average bitrate of 1028 kbps, more than twice as high as scenario 1 with 400 kbps. This suggests that a temporary increase in video quality, which can significantly increase the average bitrate, does not improve QoE.

Group Drop focuses on impairment scenarios with a temporary decrease in video quality of varying magnitude halfway through the clip. In scenario 1, the video quality remains constantly at the highest level throughout the clip, while in scenarios 2–5 the video quality of the middle segment is at different lower levels. We can observe that were was no significant difference between scenarios 1–4, with the range in MOS of 0.2 being lower than the Confidence Interval for scenarios 1–4. However, we can observe a significant difference between scenarios 1 and 5, with the MOS of scenario 5 being 0.82 lower than that of scenario 1. These results suggest that a temporary decrease in video quality does not have a significant negative impact on QoE, unless that temporary drop is to the lowest quality.

Group UnstableHigh contains impairment scenarios which compare stable and unstable video quality. Scenario 1 offers constantly the second highest video quality. In scenario 2, the video quality oscillates between the highest and the lowest quality every segment duration, while in scenario 3 it oscillates between the highest and the second lowest quality. The average bitrate of the latter two scenarios is within 5% of that of scenario 1. We can observe that the MOS for scenarios 1, 2, and 3, was 3.7, 1.7, and 2.7 respectively. This suggests that stable video quality, presented in scenario 1, is much preferred over unstable video quality, presented in scenarios 2 and 3; additionally, the smaller amplitude of oscillations in scenario 3 is preferred to the larger oscillations of scenario 2.

Group UnstableLow also contains impairment scenarios that compare stable and unstable video quality. Scenario 1 offers constantly the second lowest video quality, while in scenarios 2 and 3, the video quality oscillates every segment duration between the middle and the lowest quality. However, scenario 2 begins and ends at the lowest quality, whereas scenario 3 begins and ends at the middle quality. The MOS for scenarios 1, 2, and 3 was 2.7, 1.7, and 2.1 respectively. This suggests again that stable video quality is better. Scenario 3 was scored 0.4 higher than scenario 2, however, the difference was statistically insignificant.

Group GradualHigh focuses on impairment scenarios which compare gradual and instant multi-level quality switches. In scenario 1, the video quality gradually switches from the highest to the lowest level, while in scenario 2 it switches instantly. In scenario 3, the video quality switches gradually from the lowest to the highest level, while in scenario 4 it switches instantly. The gradual quality switches of scenarios 1 and 3 (MOS 2.5 and 2.9) were preferred to the instant quality switches of scenarios 2 and 4 (MOS 2.2 and 2.4). However, the difference between MOS of scenarios 1 and 2 was statistically insignificant.

Group GradualLow also contains impairment scenarios which compare gradual and instant multi-level quality switches, but with a lower magnitude of switches in video quality. In scenario 1, the video quality gradually switches from second highest to second lowest, while in scenario 2 this quality change is instant. In scenario 3, the video quality gradually switches from the second lowest to the second highest, while in scenario 4 this quality change is instant. There was no significant difference in MOS between these scenarios. This suggests that when there are small changes in quality, the rate of change, whether instant or gradual, has little impact on QoE.

Group DoubleDrop focuses on impairment scenarios which compare two quality drops with a single quality drop of the same duration. In scenarios 1 and 2, the quality drops from the second highest quality to the third highest, twice and once respectively. In scenarios 3 and 4, the quality drops from the second highest quality to the fourth highest, twice and
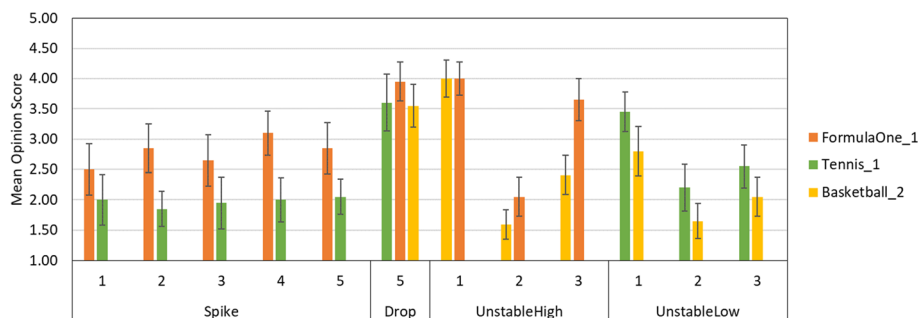
Fig. 8 Mean Opinion Score of all test clips for scenarios within the first super group with the 95% Confidence Interval plotted in the error bars. Each source clip is plotted in a different colour

once respectively. There was no statistically significant difference in MOS between one or two quality drops in either of these pairs of scenarios.

5.3.2 Comparison of impairment scenarios within super groups

In order to compare the scores between scenarios in different groups we created two super groups: {Spike, Drop_5, UnstableHigh, and UnstableLow}, and {GradualHigh, Gradual-Low, DoubleDrop}. Within each super group, every group has a source clip in common with another group allowing for comparison of scenarios in different groups.

Figure 8 shows the MOS of clips for all scenarios within the first super group with the 95% Confidence Interval plotted in the error bars, which is used to determine the statistical significance of differences in MOS, where overlapping confidence intervals are perceived as statistically insignificant. Each column illustrates the MOS for a single source clip. MOS can only be fairly compared between scenarios using the same source clip.

Comparison of scenario Drop_5 with the other scenarios using common clips Scenario Drop_5, with an average bitrate of 4171 kbps, and with six segments at the highest quality level and one at the lowest quality, has two clips in common with each of the other scenarios in the first super group. Considering Drop_5 has a much higher average bitrate than the other scenarios, it is not surprising that it achieves higher MOS in many cases. The exceptions are of most interest. There is no significant difference with UnstableHigh_1, where all segments are at the second highest quality and the average bitrate equals 2400 kbps. UnstableHigh_3 with FormulaOne_1 has similar MOS, but much lower MOS for Basketball_2, while UnstableLow_1 with an average bitrate of only 800 kbps and all segments at the second lowest quality achieves only slightly lower MOS. This suggests that the benefit in Drop_5 of having six segments at the highest quality, is lost by having one segment at the lowest quality, and that more effective use of bandwidth could be made by having all segments at the same quality, even if it would result in lower average bitrate.

Comparison of scenarios Spike_1–5 and UnstableHigh_1–3 using the common clip FormulaOne_1 Each scenario in group UnstableHigh had an average bitrate of about 2400 kbps, while those in group Spike had average bitrates in the range of 400 to 1028 kbps. Compared to the scenarios in group Spike, UnstableHigh_1 with all segments at the second highest quality, used the higher bitrate to achieve higher MOS,
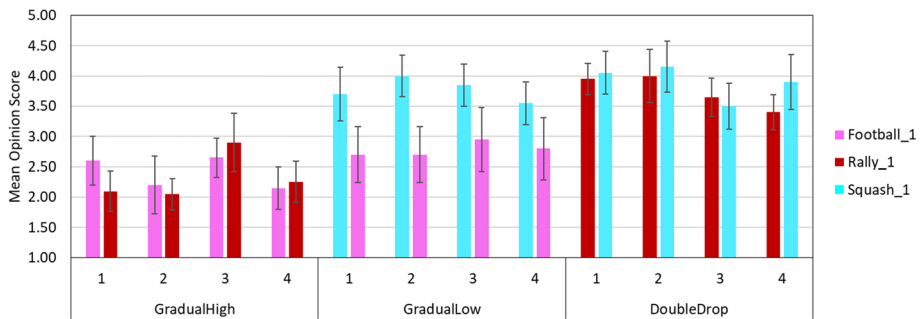
Fig. 9 Mean Opinion Score of all test clips for scenarios within the second super group with the 95% Confidence Interval plotted in the error bars. Each source clip is plotted in different colour

UnstableHigh_2, with quality varying every segment between the lowest and highest quality, achieved lower MOS than all of the scenarios of group Spike despite having a much higher bitrate, although the difference in MOS was statistically significant in only three of the five scenarios. This suggests that delivering large variations of quality is an inefficient use of bandwidth.

Comparison of scenarios Spike_1–5 and UnstableLow_1–3 using the common clip Tennis_1 Each of the scenarios in group UnstableLow had an average bitrate of about 800 kbps, while those in group Spike had average bitrates in the range of 400 to 1028 kbps. From group UnstableLow, only UnstableLow_1 with constant quality at the second lowest level achieved consistent statistically significant improvements over the scenarios of group Spike with all segments at the lowest quality except the middle one at higher quality. This suggests that consistent quality, even at the second lowest level, is preferred to variable quality. UnstableLow_2–3 suggested some benefit from having fewer segments at the lowest quality level.

Comparison of scenarios UnstableHigh_1–3 and UnstableLow_1–3 using the common clip Basketball_2 The average bitrate of clips in group UnstableHigh is about 2400 kbps, and in group UnstableLow about 800 kbps. The first scenario in each group has constant quality, and statistically significantly higher MOS than the other scenarios in the same group, again suggesting that consistent quality is an efficient use of bandwidth. The other scenarios in group UnstableHigh, with high magnitude variations in quality, show little benefit in terms of MOS over the other scenarios in group UnstableLow, with variations of lower magnitude, despite having about three times the average bitrate. This suggests that large variations in quality cause lower MOS, even when the average bitrate is higher. And UnstableLow_1 with consistent quality encoded at 800 kbps significantly outperforms UnstableHigh_2 with large variations in quality requiring three times the bandwidth.

Figure 9 shows the Mean Opinion Score of all clips for scenarios within the second super group with the 95% Confidence Interval plotted in the error bars, which is used to determine the statistical significance of differences in MOS, where overlapping confidence intervals are perceived as statistically insignificant. As with Fig. 8, MOS can only be fairly compared between scenarios using the same source clip.

Comparison of scenarios GradualHigh_1–4 and GradualLow_1–4 using the common clip Football_1 Group GradualHigh has scenarios with either a rapid or a slow transition from either the highest quality or the lowest quality to the other, with each scenario having an encoded bitrate of about 2200 kbps. Group GradualLow has corresponding scenarios between the second highest and second lowest quality levels, with average bitrates of about 1500 kbps. Each scenario in group GradualLow has higher MOS than the corresponding scenario in group GradualHigh despite having approximately 32% lower bitrate, although none of the differences are statistically significant. This suggests that quality variations of small magnitude can make more effective use of bandwidth than larger variations.

Comparison of scenarios GradualHigh_1–4 and DoubleDrop_1–4 using the common clip Rally_1 Group DoubleDrop has scenarios with one or two quality drops from the second highest level to either the middle level (scenarios 1 and 2) or the second lowest level (scenarios 3 and 4), with average encoded bitrates about 10% lower than the scenarios of group GradualHigh which have a single transition between the highest and lowest quality levels. All the scenarios in DoubleDrop had higher MOS than all of the scenarios of GradualHigh. All pairwise comparisons were statistically significant except for GradualHigh_3 which had overlapping confidence intervals with scenarios DoubleDrop_3 and DoubleDrop_4. This again suggests that the more stable quality of DoubleDrop scenarios with only small quality variations around a mid-level are preferred to a quality change over the whole range of quality levels, even though the average bitrate is 10% lower.

Comparison of scenarios GradualLow_1–4 and DoubleDrop_1–4 using the common clip Squash_1 Both of these groups contain scenarios with small variations in quality around the mid-level. Although the scenarios in GradualLow have approximately 25% lower average bitrate than the scenarios in DoubleDrop, there are no statistically significant differences between any of the scenarios in these groups.

5.4 Summary

We have performed a subjective study to investigate how users perceive changes in video quality in adaptive streaming, using impairment scenarios based on ABR behaviours observed in our quantitative evaluation. Considering the predominant use case of low latency streaming is live sport [2], we limited the content used in our experiment to sports content, and hence do not claim our findings would necessarily apply to all types of content. We highlight the following findings.

- Spike and Drop Scenarios: short temporary changes in quality level, whether an increase or a decrease, have no significant impact on QoE. Llama is quick to switch down to avoid rebuffering and hence may at times switch down unnecessarily causing a temporary drop in quality, but this has no significant impact on QoE. Furthermore, other ABR algorithms may at times increase video quality temporarily, but again, without any gain in QoE.
- Unstable High and Low Scenarios: stable quality was preferred even when the average bitrate was 42% lower. We found that the use of the lowest quality level had a significant negative impact on QoE. Llama reduces video quality on the first signs of network deterioration, but does so gradually which can minimise the time spent at lowest video quality.

- Gradual High and Low Scenarios: gradual quality variation was preferred to instant changes when the magnitude of change was high and the quality increased, but there was no significant difference when the magnitude of change was lower or when the quality decreased. Llama always changes the video quality gradually, regardless of the magnitude or direction of the planned change. Based on this finding, it may be possible to improve Llama by allowing the quality to increase by two levels instantly, and to decrease by any number of levels instantly.
- Double Drop Scenario: no significant difference in QoE between two quality drops separated by a return to higher quality and a single quality drop of the same total duration. Llama is slow to increase video quality which might result in less frequent but longer temporary drops in quality, but without significant impact on QoE.

5.4.1 Related work

We have conducted a subjective study consisting of short segments and video clips, using the ACR method to test a variety of quality bitrate patterns. Additionally, we made our results database available to the research community. In the relevant literature, there are only 7 publicly available video quality assessment databases. The study presented in [21] only tested various forms of rebuffering. Long videos (25–300s) were used to evaluate quality switches, alone in [13], as well as alongside rebuffering in [7, 8, 19], by employing a continuous testing procedure. Various quality patterns were presented in the evaluated videos, however, scores for partial patterns cannot be isolated due to the primacy and recency effects. [15, 16] evaluated quality switches, with and without rebuffering respectively, using short videos (4–10s). Our study expands on the relevant literature by testing a wide range of short video quality variations, which were isolated and evaluated using short video clips.

The study presented in [37] evaluated a subset of the impairment scenarios tested in our study using the continuous evaluation method. The authors found that, when compared to gradual quality changes, instant changes of small magnitude had no impact on QoE, whereas instant switches of large magnitude had a significant negative impact on QoE. In our results, instant decreases in video quality, regardless of magnitude, did not have a significant negative impact on QoE when compared to gradual decreases. They also observed that fluctuation between medium and high qualities had no significant impact on QoE. From our results we can observe that quality fluctuations between lower-than-medium and high qualities have a significant negative impact on QoE.

6 Conclusion

In this article, we have presented a comprehensive evaluation of state-of-the-art low latency ABR algorithms as well as the results of extensive subjective testing of observed ABR behaviour patterns. Both experiments were designed to further assess the suitability of Llama for low latency conditions, building on our earlier work in which we both proposed Llama, an ABR algorithm we designed specifically for low latency live adaptive streaming, and evaluated its performance against four prominent on-demand ABR algorithms.

First, in a quantitative manner, we evaluated Llama by comparing it against three low latency ABR algorithms, Stallion, L2A, and LoL+, with multiple client settings and 7000 throughput traces. Their performance was presented using a range of QoE metrics,

including average Live Latency, the average time between a segment being generated and played by the client, and the ITU-T Recommendation P.1203 QoE model which combines multiple QoE factors into a single metric. When compared to the other low latency ABR algorithms, for most settings, Llama achieved the lowest Live Latency by a moderate margin, which is crucial in low latency live streaming, and achieved slightly improved P.1203 MOS, which reflects the Quality of Experience.

Secondly, we have presented the results of a subjective experiment designed to investigate how users perceive changes of video quality in adaptive streaming, using impairment scenarios based on observations from our quantitative evaluation. We concluded the following from the subjective test. Consistent quality was preferred over variable quality, even when the average bitrate was 42% lower. A short temporary change in quality level had no significant impact on QoE. The lowest quality level had a significant negative impact on QoE. Gradual quality variation was preferred to instant changes when the magnitude of quality change was high, but only when increasing the video quality.

We have published the simulation model along with the throughput traces and the encoded content used in this quantitative evaluation to allow the research community to verify our results and perform further studies. We have also published the subjective test results in full.

Data Availability The datasets generated during and/or analysed during the current study are available in the github repository [26].

Declarations

References

1. 23009–1, I. (2019) Information technology-dynamic adaptive streaming over http (dash)-part 1: Media presentation description and segment formats
2. 61% of Livestreaming Viewers Watch Sports, the Most Popular Type of Livestreamed Content. https://www.parks associates.com/blog/article/05092022 Accessed May 9, 2022
3. A simulation model for Live DASH with CMAF chunks support (2021). https://github.com/tomlyko/ns3-dash-cmaf-model
4. ABR Throughput Traces derived from CDN logs of BT Sport 1 service (2022). https://github.com/lancs-net/ABR-Throughput-Traces

5. Allan B, Nilsson M, Kegel I (2019) A subjective comparison of broadcast and unicast transmission impairments. SMPTE Motion Imag J 128(6):1–15

6. Bampis CG, Bovik AC (2017) Learning to predict streaming video qoe: Distortions, rebuffering and memory. arXiv preprint arXiv:1703.00633

7. Bampis CG, Li Z, Moorthy AK, Katsavounidis I, Aaron A, Bovik AC (2017) Study of temporal effects on subjective video quality of experience. IEEE Trans Image Process 26(11):5217–5231. https://doi.org/10.1109/TIP.2017.2729891

8. Bampis CG, Li Z, Katsavounidis I, Huang T-Y, Ekanadham C, Bovik AC (2018) Towards perceptually optimized end-to-end adaptive video streaming. arXiv preprint arXiv:1808.03898

9. Barman N, Martini MG (2019) Qoe modeling for http adaptive video streaming–a survey and open challenges. IEEE Access 7:30831–30859

10. Bentaleb A, Taani B, Begen AC, Timmerer C, Zimmermann R (2019) A survey on bitrate adaptation schemes for streaming media over http. IEEE Commun Surv Tutor 21(1):562–585

11. Bentaleb A, Timmerer C, Begen AC, Zimmermann R (2019) 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, pp. 7–13

12. Bentaleb A, Akcay MN, Lim M, Begen AC, Zimmermann R (2021) Catching the moment with lol+ in twitch-like low-latency live streaming platforms. IEEE Trans Multimed 1–1. https://doi.org/10.1109/TMM.2021.3079288

13. Chen C, Choi LK, de Veciana G, Caramanis C, Heath RW, Bovik AC (2014) Modeling the time—varying subjective quality of http video streams with rate adaptations. IEEE Trans Image Process 23(5):2206–2221. https://doi.org/10.1109/TIP.2014.2312613

14. Cisco V (2018) Cisco visual networking index: Forecast and trends, 2017–2022. White Paper 1

15. Duanmu Z, Zeng K, Ma K, Rehman A, Wang Z (2017) A quality-of experience index for streaming video. IEEE J Select Topics Sign Proc 11(1):154–166. https://doi.org/10.1109/JSTSP.2016.2608329

16. Duanmu Z, Ma K, Wang Z (2018) Quality-of-experience for adaptive streaming videos: An expectation confirmation theory motivated approach. IEEE Trans Image Process 27(12):6135–6146. https://doi.org/10.1109/TIP.2018.2855403

17. Encoded Content Input (2022). https://github.com/tomlyko/ns3-dash-cmaf-model/tree/master/content

18. Essaili AE, Lohmar T, Ibrahim M (2018) Realization and evaluation of an end-to-end low latency live dash system. In: 2018 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB), pp. 1–5

19. Eswara N, Manasa K, Kommineni A, Chakraborty S, Sethuram HP, Kuchi K, Kumar A, Channappayya SS (2018) A continuous qoe evaluation framework for video streaming over http. IEEE Trans Circuits Syst Video Technol 28(11):3236–3250. https://doi.org/10.1109/TCSVT.2017.2742601

20. Garcia MN, Robitza W, Raake A (2015) On the accuracy of short-term quality models for long-term quality prediction. In: 2015 Seventh International Workshop on Quality of Multimedia Experience (QoMEX), pp. 1–6. https://doi.org/10.1109/QoMEX.2015.7148123

21. Ghadiyaram D, Pan J, Bovik AC (2019) A subjective and objective study of stalling events in mobile streaming videos. IEEE Trans Circuits Syst Video Technol 29(1):183–197

22. Gutterman C, Fridman B, Gilliland T, Hu Y, Zussman G (2020) Stallion: Video adaptation algorithm for low-latency video streaming. In: Proceedings of the 11th ACM Multimedia Systems Conference. MMSys'20, pp. 327–332. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3339825.3397044

23. Hughes K, Singer D (2017) Information technology–multimedia application format (mpeg-a)–part 19: Common media application format (cmaf) for segmented media. ISO/IEC, 23000–19

24. ITU, R.I.-R.B. (2012) 500–13: Methodology for the subjective assessment of the quality of television pictures. International Telecommunications Union, Technical report

25. Karagkioules T, Mekuria R, Griffioen D, Wagenaar A (2020) Online learning for low-latency adaptive streaming. Association for Computing Machinery, New York, NY, USA

26. Llama QoE Database (2022). https://github.com/tomlyko/Llama-QoE-Database

27. Lohmar T, Einarsson T, Fröjdh P, Gabin F, Kampmann M (2011) Dynamic adaptive http streaming of live content. In: 2011 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, pp. 1–8. https://doi.org/10.1109/WoWMoM.2011.5986186

28. Lyko T, Broadbent M, Race N, Nilsson M, Farrow P, Appleby S (2020) Llama - low latency adaptive media algorithm. In: 2020 IEEE International Symposium on Multimedia (ISM), pp. 113–121. https://doi.org/10.1109/ISM.2020.00027

29. Lyko T, Broadbent M, Race N, Nilsson M, Farrow P, Appleby S (2020) Evaluation of cmaf in live streaming scenarios. In: Proceedings of the 30th ACM Workshop on Network and Operating Systems Support for

Digital Audio and Video. NOSSDAV '20, pp. 21–26. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3386290.3396932

30. Mok RKP, Chan EWW, Chang RKC (2011) Measuring the quality of experience of http video streaming. In: 12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops, pp. 485–492. https://doi.org/10.1109/INM.2011.5990550

31. Ott, H., Miller, K.,Wolisz, A. (2017) Simulation framework for http-based adaptive streaming applications. In: Proceedings of the Workshop on Ns-3. WNS3 '17, pp. 95–102

32. Pantos R, May W (2017) Http live streaming. Technical report

33. Recommendation, I. (1999) 910,"subjective video quality assessment methods for multimedia applications," recommendation itu-t p. 910. ITU Telecom. Standardization Sector of ITU

34. Robitza W, Garcia M-N, Raake A (2017) A modular http adaptive streaming qoe model—candidate for itu-t p. 1203 ("p. nats"). In: 2017 Ninth International Conference on Quality of Multimedia Experience (QoMEX), pp. 1–6. IEEE

35. Robitza W, Göring S, Raake A, Lindegren D, Heikkilä G, Gustafsson J, List P, Feiten B, Wüstenhagen U, Garcia M-N, Yamagishi K, Broom S (2018) HTTP Adaptive Streaming QoE Estimation with ITU-T Rec. P.1203 – Open Databases and Software. In: 9th ACM Multimedia Systems Conference, Amsterdam. https://doi.org/10.1145/3204949.3208124

36. Shuai Y, Herfet T (2018) Towards reduced latency in adaptive live streaming. In: 2018 15th IEEE Annual Consumer Communications Networking Conference (CCNC), pp. 1–4. https://doi.org/10.1109/CCNC.2018.8319262

37. Staelens N, De Meulenaere J, Claeys M, Van Wallendael G, Van den Broeck W, De Cock J, Van de Walle R, Demeester P, De Turck F (2014) Subjective quality assessment of longer duration video sequences delivered over http adaptive streaming to tablet devices. IEEE Trans Broadcast 60(4):707–714. https://doi.org/10.1109/TBC.2014.2359255

38. Tavakoli S, Egger S, Seufert M, Schatz R, Brunnström K, García N (2016) Perceptual quality of http adaptive streaming strategies: Crossexperimental analysis of multi-laboratory and crowdsourced subjective studies. IEEE J Select Areas Commun 34(8):2141–2153. https://doi.org/10.1109/JSAC.2016.2577361

39. Tran HTT, Ngoc NP, Pham AT, Thang TC (2016) A multi-factor qoe model for adaptive streaming over mobile networks. In: 2016 IEEE Globecom Workshops (GC Wkshps), pp. 1–6. https://doi.org/10.1109/GLOCOMW.2016.7848818

40. Tran HTT, Vu T, Ngoc NP, Thang TC (2016) A novel quality model for http adaptive streaming. In: 2016 IEEE Sixth International Conference on Communications and Electronics (ICCE), pp. 423–428. https://doi.org/10.1109/CCE.2016.7562674

41. Viola R, Gabilondo A, Martin A, Mogollón JF, Zorrilla M, Montalbán J (2019) Qoe-based enhancements of chunked cmaf over low latency video streams. In: 2019 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB), pp. 1–6

42. Wang F, Fei Z, Wang J, Liu Y, Wu Z (2017) Has qoe prediction based on dynamic video features with data mining in lte network. Sci China Inform Sci 60(4):042404

43. Yamagishi K, Hayashi T (2017) Parametric quality-estimation model for adaptive-bitrate-streaming services. IEEE Trans Multimed 19(7):1545–1557. https://doi.org/10.1109/TMM.2017.2669859

44. Zambelli A (2009) Iis smooth streaming technical overview. Microsoft Corp 3:40

45. Zegarra Rodríguez D, Lopes Rosa R, Costa Alfaia E, Issy Abrahão J, Bressan G (2016) Video quality metric for streaming service using dash standard. IEEE Trans Broadcast 62(3):628–639. https://doi.org/10.1109/TBC.2016.2570012

Authors and Affiliations

Tomasz Lyko[1] · **Matthew Broadbent[2]** · **Nicholas Race[1]** · **Mike Nilsson[3]** ·
Paul Farrow[3] · **Steve Appleby[3]**

✉ Tomasz Lyko
  t.lyko@lancaster.ac.uk

  Matthew Broadbent
  m.broadbent@napier.ac.uk

  Nicholas Race
  n.race@lancaster.ac.uk

  Mike Nilsson
  mike.nilsson@bt.com

  Paul Farrow
  paul.farrow@bt.com

  Steve Appleby
  steve.appleby@bt.com

[1] School of Computing and Communications, Lancaster University, Lancaster, UK

[2] School of Computing, Edinburgh Napier University, Edinburgh, UK

[3] Applied Research, British Telecommunications, Martlesham, UK