Multimedia Tools and Applications
https://doi.org/10.1007/s11042-023-15163-w

Content-aware QoE optimization in MEC-assisted Mobile video streaming

Waqas ur Rahman[1] · Eui-Nam Huh[2]

Received: 31 December 2021 / Revised: 15 September 2022 / Accepted: 13 March 2023

Abstract
The traditional client-based HTTP adaptation strategies do not explicitly coordinate between the clients, servers, and cellular networks. A lack of coordination leads to suboptimal user experience. In addition to optimizing Quality of Experience (QoE), other challenges in adapting HTTP adaptive streaming (HAS) to the cellular environment are overcoming unfair allocation of the video rate and inefficient utilization of the bandwidth under the high-dynamics cellular links. Furthermore, the majority of the adaptive strategies ignore important video content characteristics and HAS client information, such as segment duration, buffer size, and video duration, in the video quality selection process. In this paper, we present a content-aware hybrid multi-access edge computing (MEC)-assisted quality adaptation algorithm by taking advantage of the capabilities of edge cloud computing. The proposed algorithm exploits video content characteristics, HAS client settings, and application-layer information to jointly adapt the bitrates of multiple clients. We design separate strategies to optimize the performance of short and long duration videos. We then demonstrate the efficiency of our algorithm against client-based solutions as well as MEC-assisted algorithms. The proposed algorithm guarantees high QoE, equitably selects video rates for clients, and efficiently utilizes the bandwidth for both short and long duration videos. The results from our extensive experiments reveal that the proposed long video adaptation algorithm outperforms state-of-the-art algorithms, with improvements in average video rate, QoE, fairness, and bandwidth utilization of 0.4%–12.3%, 8%–65%, 3.3%–5.7%, and 60%–130%, respectively. Furthermore, when high bandwidth is available to competing clients, the proposed short video adaptation algorithm improves QoE by 11.1% compared to the long video adaptation algorithm.

✉ Eui-Nam Huh
johnhuh@khu.ac.kr

[1] Digital Media Technology Lab, School of Computing and Digital Technology, Birmingham City University, Birmingham, UK

[2] Department of Computer Science and Engineering, Electronics and Information Building, Room #331, Kyung Hee University, 1732 Deogyeong-daero, Giheung-gu, Yongin-si, Gyeonggi-do 17104, Republic of Korea

Keywords Mobile edge computing · HTTP-based video streaming · Adaptive streaming · Quality of experience · Streaming media

1 Introduction

Multimedia content accounts for the majority of Internet traffic. According to the Cisco Visual Networking Index, 82% of global mobile data traffic will be video traffic by 2022 [8]. To handle traffic demand related to multimedia, HTTP adaptive streaming (HAS) solutions are often used. These solutions include Apple's HTTP Live Streaming (HLS), Adobe's HTTP Dynamic Streaming (HDS), Microsoft's ISS Smoothing Streaming, and Dynamic Adaptive Streaming over HTTP (DASH) developed under MPEG and standardized by the ISO/IEC.

In HAS, video content is encoded at multiple video rates and is stored on an HTTP server. The video content is fragmented into segments of fixed durations. The adaptive bitrate (ABR) algorithms run on the HTTP clients and adapt the video quality according to the network conditions. The HAS clients download the segments into the playback buffer before they are sent to the video player. The objective of the adaptation algorithms is to optimize the user experience by meeting conflicting video quality objectives. These objectives include selecting the highest feasible set of video bitrates, avoiding unnecessary video bitrate switches, and preserving the buffer level to avoid playback interruptions [10, 11, 24, 29, 39].

Traditionally, the ABR algorithms run on the HTTP client, and the clients are unaware of competing clients and radio channels. It has been shown that competing clients cannot achieve fair performance when the air interface is a bottleneck [6]. Furthermore, the unfairness increases as the number of competing clients increase. Similarly, competing clients cannot coordinate with each other to efficiently utilize the bandwidth. Recently, a multi-access edge computing (MEC) paradigm has emerged that offers computation capabilities at the edge of a mobile network by deploying servers within the radio access networks. In addition, MEC provides real-time access to application and radio access network (RAN) information. The computational capabilities of MEC allow for cell-wide central adaptation of multiple clients competing for bandwidths.

In our previous work [36], we analyzed the performance of MEC-assisted and client-based rate-adaptation algorithms under varying client, server, dataset, and network settings. The existing algorithms developed fixed control rules to select the video quality based on the estimated throughput [18, 42], the playback buffer level [14], or a combination of the two parameters [15, 28, 33, 34]. The results in [36] revealed that the algorithms require significant tuning, and performance fluctuates from one network setting to the other. This leads to the algorithms providing inconsistent QoE in different environments. The video streaming services deploy segment durations differently. Microsoft's Smooth Streaming and Adobe's HTTP Dynamic Streaming offer segment durations of 2 seconds and 4 seconds, respectively [1, 47]. With the shorter segment duration, the client has more opportunities to adapt the video rate, compared to the longer duration. In a highly unstable network, the client could adjust the video rate quickly, downloading smaller segments. Similarly, different video players offer different buffer sizes. The buffer-based algorithms adapt the video rates aggressively or conservatively based on the playback buffer level. As the buffer level increases, the algorithms select the video rate aggressively. A smaller buffer would fill up quickly, compared to a larger buffer. This would allow the adaptation

algorithms to aggressively increase the video rate. However, a larger buffer decreases the risk of playback interruption in case of a mismatch between the selected video rate and the available bandwidth. The ABR algorithms should be able to guarantee QoE under different settings. However, the existing adaptation algorithms do not consider buffer sizes and segment durations to adapt the video quality [36]. In this work, in addition to the playback buffer level, we also consider clients' buffer sizes and segment duration to decide video quality.

The results in [36] also reveal that the existing algorithms give precedence to one specific video quality objective over others. This trend is observed in both MEC-assisted and client-based algorithms. It is easier to meet just one of the conflicting video quality objectives. For example, the video can be streamed at the highest available video rate throughout the streaming session. This would increase the risk of playback interruptions in an unstable environment. Similarly, the video can be streamed at the lowest available video rate to minimize the risk of playback interruption. However, this would lead to poor video quality. The aim of this work is to propose an adaptive algorithm that optimizes the QoE by simultaneously maximizing all metrics.

Trends in video content have changed drastically since the advent of social media. The durations of video content from online video-sharing platforms such as YouTube have been drastically reduced over the years [2]. The average duration of the top 10 videos on Facebook was 128 s in 2018 [17]. Similarly, the average duration of movie trailers for over 20,000 movies released between 2000 and 2016 was 114.2 s [13]. In a mobile network, the bandwidth for HAS clients depends on multiple factors, including propagation distance, fading, interference, and user mobility. The throughput may change drastically while downloading the segments. Therefore, the existing ABR algorithms strive to keep the buffer filled to a predefined threshold to minimize the risk of playback interruption [26, 33, 34]. To keep the buffer above a predefined threshold, the algorithms compromise on video quality. This strategy is understandable while streaming a long video, such as a complete movie. However, with a short video, such as a movie trailer, the user expects to watch the complete video at the most feasible video quality. Therefore, to select video rates, it makes sense to design different strategies for short and long videos. To this end, we design separate quality adaptation algorithms for short and long duration videos.

The understanding of MEC-assisted ABR strategies is still in the early stages. In this paper, we present a content-aware edge computing-assisted rate adaptation method for a single cell with multiple clients to centrally optimize the QoE of competing clients. The contributions of this research are as follows.

- We design an integer non-linear programming (INLP) optimization model that jointly optimizes the QoE, fairness, and bandwidth utilization of HAS clients in a cellular network with MEC capabilities.
- Due to the NP-Hardness of the problem, we designed content-aware greedy heuristic algorithms that solve the rate adaptation optimization problem for short and long duration videos. The algorithms consider video duration, segment duration, clients' playback buffer size, estimated throughput, and playback buffer level to jointly select the video rates for HAS clients.
- We conducted extensive experiments to evaluate the performance of the proposed algorithms with varied segment durations, playback buffer sizes, numbers of competing clients, clients' moving speeds, and video durations.

- The results from our extensive experiments show that the proposed algorithm guarantees QoE under varying client, server, dataset, and network settings. The proposed algorithm optimizes QoE by simultaneously enhancing all video quality metrics.
- The results reveal that the proposed long video adaptation algorithm outperforms state-of-the-art algorithms, with average improvements in video rate, QoE, fairness, and bandwidth utilization ranging from 7.3%–12.3%, 8%–28%, 3.3%–5.7%, and 60%–130%, respectively. Additionally, when high bandwidth is available to clients, the proposed short video algorithm downloads 6% higher-quality segments, experiences 45% fewer switches, and improved QoE by 11.1%, compared to the proposed long video adaptation algorithm.

2 Related work

The ABR algorithms can be divided into three methods: 1) throughput-based, 2) buffer-based, and 3) hybrids. Throughput-based algorithms select the video quality based on the throughput observed while downloading segments [36] [5, 23, 27]. It has been shown that they cannot accurately estimate the bandwidth when multiple clients compete against a network bottleneck [22]. Therefore, some ABR algorithms suggest observing only the playback buffer to select the video quality [18, 41]. Multiple researchers have used a combination of playback buffer and estimated throughput to pick video quality [13, 14, 34, 42]. In [20], the authors used segment size in addition to throughput and the playback buffer for video rate adaptation. However, all these algorithms targeted client-side quality adaptation. Moreover, these algorithms do not target fair selection of the video rates in a multi-client environment. FESTIVE [6] uses an approach to improve HAS fairness by using a harmonic bandwidth estimator and randomizing the scheduling of the requested segments. Li et al. [10] presented an algorithm called Probe and Adapt (Panda) that probes for fair bandwidth sharing and adapts the video rate accordingly. Although these algorithms improve the fairness and stability in a wired network, they perform poorly under dynamic cellular links due to TCP unfairness. The Panda probing mechanism follows an additive-increase/multiplicative-decrease (AIMD) strategy. In a cellular network, a client close to the edge of the base station may observe low throughput due to propagation distance. When that client moves closer to the base station, it will observe higher throughput. However, due to Panda's AIMD strategy, it will take multiple segments to increase the client's estimated throughput.

In [9], the authors proposed a server-side scheme using feedback control theory to execute measurement and control at the HAS server. The clients' video qualities are jointly adapted at the server. However, the scheme is not specifically designed for the cellular environment, and does not impose any constraints on radio resources, which might lead to overestimating or underestimating the video rates for adaptation. Petrangeli et al. [30] proposed a method to fairly utilize the bandwidth when multiple clients greedily compete for it. However, their proposed objective function and adaptation scheme do not consider the trade-off between the QoE of the clients and fairness. Other researchers [7, 19, 48] have proposed schemes that combine the designs of quality adaptation and resource allocation in a multi-client cellular environment. However, these schemes require modification of the standard cellular infrastructure.

The concept of MEC has been proposed by the European Telecommunications Standards Institute (ETSI) to satisfy the requirements of 5G. Yang et al. [44] implemented a proof-of-concept for a MEC-assisted mobile video streaming service. Tran et al. jointly utilized the

processing capability of MEC along with edge caching to improve a streaming system [43, 45]. However, the focus of these works was to reduce video delivery latency without considering factors that impact the QoE of the clients. In [25], the authors proposed an MEC-assisted adaptation algorithm and a client to edge server mapping strategy to quantify the benefits of network-assisted solution. The authors compared the effect of network topology and inter-arrival time on the performance of MEC-assisted algorithm and purely client-based adaptation algorithms. The results in [25] showed that the client to edge server mapping mechanism led to clients achieving higher throughput. MEC-assisted algorithm utilized the higher available throughput to download higher quality segments compared to client-based algorithms. The MEC-assisted algorithm outperformed client-based algorithms in some of the video quality objectives when the achievable throughput was moderately high. However, the authors did not discuss the performance of the adaptation algorithms without employing client to serve edge mapping strategy. In addition, the authors did not use QoE and bandwidth utilization metric to compare the performance of the algorithms. Authors used a simple mathematical model to characterize the radio link as a function of distance of client from base station. In cellular networks, radio link depends on multiple factors, including propagation distance, fading, shadowing, and interference. The authors ignore factors such as fading, shadowing and interference in their mathematical model. Moreover, the authors ignored important content information in the design of the adaptation algorithm. In [46], the authors proposed an edge-assisted adaptive video streaming scheme based on a dueling deep Q-learning network. The objective of the video streaming scheme was to optimize QoE by jointly considering the physical layer transmission bandwidth and playback buffer status. In our previous work [35], we presented joint throughput estimation that assists an adaptation algorithm in fairly assigning video rates, as well as MEC-assisted rate adaptation method to enhance the viewing experience. These works [3, 21, 25, 35, 43, 45, 46] focused on jointly optimizing the QoE in a cellular environment. However, they do not focus on efficient utilization of the bandwidth by a HAS client. Moreover, these works do not take into account the video content and HAS client information. Authors in [12] introduced an edge- and SDN-assisted video streaming framework that exploited the capability of Software Defined Network (SDN) and Network Function Virtualization (NFV). This work focused on improving the user experience by improving playback video rate and minimizing playback interruptions. In [36], we showed the impact of segment duration, client buffer size, the number of competing clients, and clients' arrival times on the performance of HAS algorithms. The results revealed that the rate adaptation algorithms must consider these parameters in order to guarantee QoE under different settings.

Different from existing works, the proposed algorithm investigates the impact of content-aware joint optimization of QoE, fairness, and bandwidth utilization for video streaming in MEC environments. The proposed algorithm jointly adapts video rates by exploiting cell-wide HAS clients' information, video content details, and device features. Furthermore, the proposed method uses different strategies to adapt video rates to stream short and long duration videos.

3 Multi-access edge computing assisted streaming

In this section, we describe the proposed system.

3.1 Architecture overview

Traditionally, the adaptation module runs on the HAS client. HAS clients are oblivious to the decisions made by other competing clients. Moreover, clients unfairly utilize the available bandwidth in the presence of competing clients [22]. HAS clients rely on the underlying TCP to fairly and accurately estimate throughput; however, the underlying TCP is inaccurate and unfair in a cellular environment. The edge cloud can access RAN information and is computationally far more powerful than HAS clients. It is thus logical to shift the adaptation module from the client to the edge cloud. The adaptation module at the edge cloud exploits the channel knowledge of multiple streams to jointly adapt the video quality of the clients.

Figure 1 illustrates the MEC HAS system for adaptive video streaming over a cellular network. The HAS server stores video content encoded into a set of m video rates $R = \{R_1, R_2, R_3, \ldots, R_m\}$. Each representation of a video is split into multiple fixed-duration segments, τ. A set of N HAS clients subscribes to HAS services, and each client is indexed by i, where $i = 1, 2, \ldots, N$. The edge cloud is deployed at the base station to enhance the mobile services, and cellular entities such as the cellular scheduler operate in the same way as conventional cellular networks.

The client initiates streaming by requesting information about the stored content from the HAS server. In response, the HAS server sends the media presentation description (MPD) so the adaptation module at the edge cloud and the HAS client have information on the available video representations. Then, the HAS client requests a video segment based on the available application layer information. The request from the HAS client is treated as a suggestion by the adaptation module for the edge cloud. Under conventional client-side adaptation, the cellular network forwards the request to the HAS server. In MEC-assisted adaptation, the edge cloud intercepts the request. The adaptation module overwrites the suggested video rate by the client, Rc, based on the cell-wide optimization of the clients. In addition to the information available

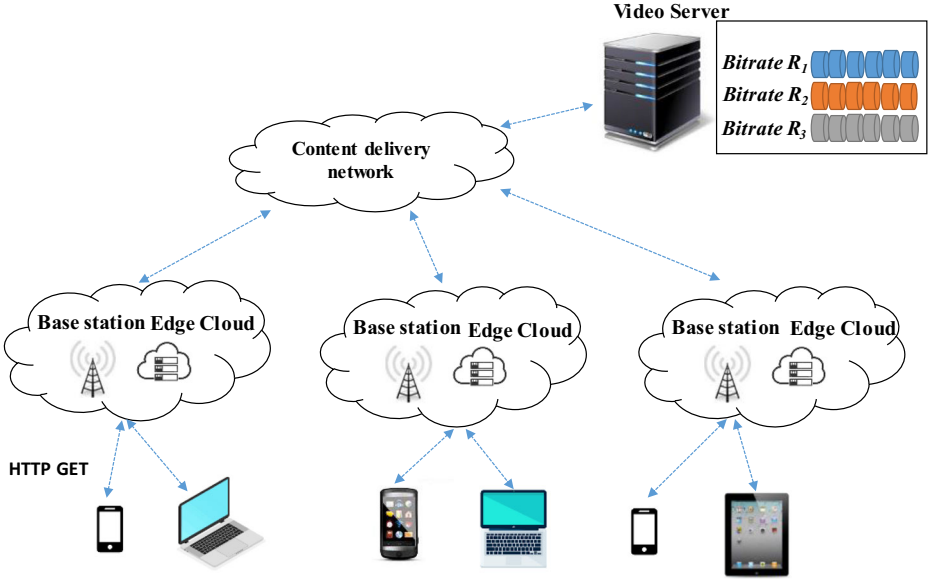


Fig. 1 Streaming architecture for multi-access edge computing–assisted video streaming

in the MPD sent by the server, clients' playback information for adaptation, such as the playback buffer level, device capabilities, observed throughput, and QoE status of the clients, is embedded in the feedback from clients. This is feasible, because the 3rd Generation Partnership Project standardized QoE reporting for clients by using the HTTP POST request carrying XML-formatted metadata in the body [32]. The video rate adaptation results of the adaptation module for the edge cloud are then delivered to the HAS server for streaming the next segment. In this manner, the edge cloud can jointly optimize the user experience of the HAS client without modifying the client or server. The list of parameters and their descriptions have been summarized in Table 1.

The downloaded segments are stored in the playback buffer, which contains the unviewed video. Let $B(t) \in [0, B_{max}]$ be the buffer occupancy at time t. Different video players provide different buffer sizes, B_{max}, depending on the service provider and storage limitations on the player. Figure 2 depicts the dynamics of the playback buffer. At time t^k, the client downloads the k^{th} segment encoded at the i^{th} video rate, R_i^k. The size of the segment is equal to $R_i^k \times \tau$. The download time of the segment will be $(R_i^k \times \tau / T^k)$ where T^k is the throughput observed by the client during the download of the k^{th} segment. Once segment k is downloaded, the client waits for Δt^k seconds before sending the request for the $k^{th} + 1$ segment. Waiting time is given by:

$$\Delta t^k = \begin{cases} 0, & B(t) < B_{max} \\ \tau, & otherwise \end{cases} \tag{1}$$

Table 1 Cellular Network Configuration

N, S	Number of DASH clients and total number of segments downloaded by the client
R, R_i^k	Set of available discrete video rates and k^{th} segment encoded at the i^{th} video rate
R_c	Video rate suggested by the client to edge cloud
R_{max}, R_{min}	Maximum and minimum video rate available
B^k, B^{max}	Buffer occupancy level at the download of k^{th} segment and client's buffer size
SNR	Signal to noise ratio
T^k	Throughput during the download of k^{th} segment
C_j	j^{th} client's channel capacity
m	Number of available video rates
k	Index of the current segment
τ	Segment duration
P_r, N_o	Received power and the power spectral density of additive white Gaussian noise
BW, W_j	Total bandwidth and the bandwidth allocated to the j^{th} client
PL	Path loss
Q_j	Average video bitrate over downloaded segments by the j^{th} client
QS_j	Average magnitude of the changes in the quality from one segment to another
F_j	Fairness contribution by allocating video rate to client j during its streaming session
IE_j	Bandwidth inefficiency by allocating video rates to client j during its streaming session
IR	Total interruption time
x_{ij}	Decision variable that defines number of clients streaming the i^{th} video rate stored in the server
t_{dur},Δt^k	Video duration and time the client waits before sending request for the next $(k+1)$ segment
ρ, β, φ, θ	Adjustable weighting parameters of average bitrate, bitrate switching, fairness, and bandwidth inefficiency, respectively
δ_s, δ_{IE}, δ_F	Threshold videos for switching level, inefficiency, and fairness

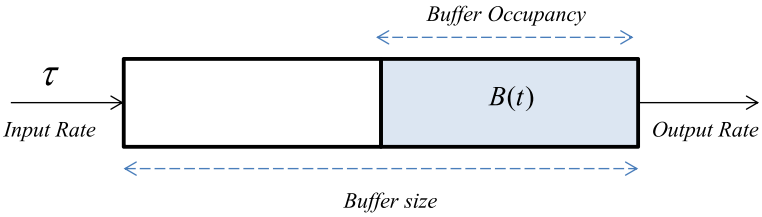


Fig. 2 Dynamics of playback buffer

Throughput T^k at time t is calculated as follows:

$$t^{k+1} = t^k + \frac{R_i^k \times \tau}{T^k} + \Delta t^k \tag{2}$$

$$T^k = \frac{1}{t^{k+1} - t^k + \Delta t^k} \int_{t^k - \Delta t^k}^{t^{k+1}} T_t dt \tag{3}$$

Let B^k be the buffer level before the start of the download of the $k^{th} + 1$ segment; then, B^{k+1} is expressed as:

$$B^{k+1} = \left(B^k + \tau - \left(\tau \times \frac{R_i^k}{T^k} \right) - \Delta t^k \right)_+ \tag{4}$$

The notation $(x)_+ = max\,(x, 0)$ ensures that the term is always positive. Eq. 4 shows that if $B^k < \tau \times R_i^k / T^k$, the buffer will be empty before the video player completely downloads the k^{th} segment. Note that the segment duration plays an important role in the change in buffer occupancy during the download of the segment. A longer segment duration increases the risk of playback interruption in case of a mismatch between the selected video rate and the throughput.

3.1. Channel Model

We consider a cell that consists of N HAS clients that stream video content and are served by a base station. The spatial distributions of the base station and the HAS clients are mutually independent. Ignoring interference among clients, the Signal-to-Noise-Ratio (SNR) of the HAS client can be calculated with:

$$SNR = \frac{P_r}{N_o W} \tag{5}$$

where P_r, N_o, and W denote the received power, the power spectral density of additive white Gaussian noise, and the client's bandwidth, respectively. The received power is related to the path loss and the transmitted power. The path loss, $PL(d)$, is a function of propagation distance [31]:

$$PL(d) = 128.1 + (37.6 * log10(d))(dB) \tag{6}$$

where d is the distance between the HAS client and the base station. Therefore, (5) can be expressed as:

$$SNR = \frac{P_t PL(d)}{N_o W} \tag{7}$$

where P_t denotes the transmitted power.

Assume the total bandwidth is BW, and the bandwidth allocated to the j^{th} client is ω_j. Ignoring interference among the clients, the j^{th} client's channel capacity, C_j, can be calculated according to Shannon's theorem [31]:

$$C_j = W_j log_2(1 + SNR) = W_j log_2\left(1 + \frac{P_t PL(d)}{N_o W_j}\right) \tag{8}$$

The higher the client's channel capacity, C_j, the higher the throughput, T_j, observed by the j^{th} client during the download of a segment. Note that the channel capacity depends on the propagation distance. HAS clients may be located at different distances from the base station. A client close to the base station will achieve higher throughput, compared to a client located at the edge of the cell. On one hand, selecting the highest feasible video rate for both clients would decrease the fairness. On the other hand, selecting a video rate higher than the available throughput for the user at the edge of the circle (in order to improve fairness) would lead to buffer underflow. To this end, selecting a low video rate for the client closer to the base station would improve fairness, but increases bandwidth inefficiency. Therefore, optimizing both bandwidth utilization and fairness for the clients in a cellular environment is a challenging task.

3.2. Quality of Experience

A comprehensive survey on QoE under HAS determined the factors that affect user experience [38]. These factors include selecting the highest feasible set of video bitrates, avoiding unnecessary video bitrate switches, and avoiding playback interruption. Playback interruptions and selection of video bitrates affect the user experience the most [16]. There is a tradeoff between selecting the highest feasible video rate and the risk of playback interruption. We aim to provide optimal QoE based on the abovementioned conflicting criteria.

The average video bitrate over downloaded segments by the j^{th} client is given by:

$$Q_j = \frac{\sum_{k=1}^{S}\left(R_{ij}^k\right)}{S} \tag{9}$$

where R_{ij}^k is the i^{th} video rate assigned to the j^{th} client, k is the segment index, and S is the total number of segments downloaded by the client.

Frequent video rate switches inversely affect the user experience. Abrupt switching impairs QoE more than smooth switching [11]. Magnitude of the changes in the quality from one segment to another is given by:

$$QS_j = \frac{\sum_{k=2}^{S} R_{ij}^k - R_{ij}^{k-1}}{Number\ of\ Switches} \tag{10}$$

The client experiences playback interruptions if the download time ($\tau \times R_{ij}^k / T^k$) is higher than the playback buffer occupancy level. The total interruption time, IR, is $\sum_{k=1}^{S}\left(\tau \times R_i^k / T^k - B^k\right)_+$.

In this study, we used the same QoE metric used by the authors in [45], which is defined as follows:

$$QoE_j = \sum_{k=1}^{S} q\left(R_{ij}^k\right) - \mu IR_j - \sum_{k=1}^{S} \left| q\left(R_{ij}^k\right) - \mathrm{q}\left(R_{ij}^{k-1}\right) \right| \tag{11}$$

For a video fragmented into N segments, $q(R_i^k)$ maps the video rate to the quality perceived by the viewer. IR_j represents the total rebuffering time during the download of the video, while the final term discourages frequent changes in the video rate. The authors in [45] used $\mathrm{q}(R_{ij}^k) = R_{ij}^k$ and $\mu = 3000$, signifying that a playback interruption of 1 s receives the same penalty as reducing the bitrate of a segment by 3000 kbps. We also consider the same values in our evaluation. In this study, we calculated the average QoE per segment, that is, the total QoE metric divided by the number of segments. In Section 6, we evaluate the QoE of algorithms using Eq. (11).

3.2 Fairness and bandwidth efficiency

Rate adaptation algorithms are fairly effective when a client operates alone. When multiple HAS clients compete for the bandwidth, the clients inefficiently utilize bandwidth and select low-quality video rates [36].

In order to efficiently utilize the bandwidth, we strive to select for the competing clients the best suitable video rates such that their sum has the least difference from the total available bandwidth at the base station. The bandwidth inefficiency at time t is calculated according to:

$$IE_j = \left| R_{ij}^{(t)} + \sum_{v \neq j} R_{iv}^{(t)} - BW^{(t)} \right| \tag{12}$$

where $BW^{(t)}$ is the available bandwidth at time t, $R_{ij}^{(t)}$ is the video rate selected by the j^{th} client at time t, and $\sum_{v \neq j} R_{iv}^{(t)}$ is the sum of the video rates of the competing video clients at time t.

To ensure that the video rates are fairly allocated among the clients, we select for each client the most feasible video rate which has the least difference from the average of video rates allocated to other competing clients. The fairness index at time t is calculated according to:

$$F_j = \left| R_{ij}^{(t)} - R_{avg} \right| \tag{13}$$

where $R_{avg} = \left(\frac{1}{N-1}\right) \sum_{v \neq j} R_{iv}^t$ is the average video rate of the other active streaming clients. Low values for inefficiency and fairness are desired. A low inefficiency value signifies that the client selects the highest feasible bitrates that are lower than the actual throughput, while a low fairness value signifies that the competing clients achieve equitable video rates.

4 Joint optimization problem

The ultimate goal of video quality adaptation is to enhance the QoE of video clients in order to achieve higher long-term user engagement [10]. With the abovementioned system, the utility

maximization problem (jointly maximizing QoE for individual HAS clients, ensuring fairness, and reducing bandwidth inefficiency) is formulated as the following integer non-linear programming (INLP) optimization model.

$$Maximize\ U_j = \rho \times Q_j - \beta \times QS_j - \varphi \times F - \theta \times IE \tag{14}$$

Subject to:

$$t^{k+1} = t^k + \frac{R_{ij}^k \times \tau}{T^k} + \Delta t^k \tag{15}$$

$$T^k = \frac{1}{t^{k+1} - t^k + \Delta t^k} \int_{t^k - \Delta t^k}^{t^{k+1}} T_t dt \tag{16}$$

$$\sum_{j=1}^{N} x_{ij}^{(t)} \geq 0, \forall 1 \leq i \leq \tag{17}$$

$$\sum_{j=1}^{N} x_{ij}^{(t)} \times W_j^{(t)} \leq BW, \forall 1 \leq i \leq m \tag{18}$$

$$B_j^{k+1} = \left(B_j^k + \tau - \left(\tau \times \frac{R_{ij}^k}{T_j^k} \right) - \Delta t^k \right)_+ \tag{19}$$

$$0 \leq B_j^{(t)} \leq B_j^{max} \tag{20}$$

$$R_{ij}^{(t)} \leq R_c, \forall 1 \leq i \leq m, 1 \leq j \leq N \tag{21}$$

$$R_{ij}^{(t)} \in R, \forall 1 \leq i \leq m, 1 \leq j \leq N \tag{22}$$

We define four weighting parameters, $0 \leq \rho, \beta, \varphi, \theta \leq 1$ ($\rho + \beta + \varphi + \theta = 1$), to control the respective video rates, the video-rate switches, fairness, and bandwidth inefficiency. The decision variable x_{ij} defines the number of clients streaming the i^{th} video rate stored on the server. The only decision variables here are integer variables x_{ij} and R_{ij}^k. Variable B_k is a dependent variable whose values depend on the values of the decision variables. The values for the rest of the variables are known in advance.

Objective function (14) aims to jointly optimize the QoE of the j^{th} client, the fairness, and the bandwidth efficiency, given throughput trace $\{T^t,\ t \in [t^l,\ t^{k\ +\ l}]\}$. Constraint (17) specifies that a specific video rate can be streamed by multiple clients. Constraint (18) ensures that the total bandwidth allocated to the clients by the base station does not exceed the instantaneously

available bandwidth at the base station. Constraint (20) guarantees that the clients do not experience any playback interruptions during the whole streaming duration. Constraint (21) ensures that the video rate selected for the j^{th} client at the MEC does not exceed the video rate, R_c, as suggested by the client. And finally, constraint (22) specifies that the discrete video rate downloaded by the client from the server belongs to the set of available video rates.

5 Proposed online algorithm

In this section, we present the algorithms for solving the optimization problem described in Section 4. The algorithms are designed for online execution by the edge cloud. The existence of an integer decision variable in the optimization problem given in Section 4 makes it computationally intractable to solve the problem using exhaustive search methods. The complexity of exhaustive search methods grows exponentially with the increase in the number of clients making it impractical for DASH scheduling at a large scale. Moreover, the deployment of offline solution is unfeasible, since information about the clients is unknown in advance. To reduce complexity, we designed a heuristic online algorithm that is executed using the client data obtained for MEC.

The algorithm selects the i^{th} video rate from set R for the k^{th} segment, denoted as R_{next}. The video rate selected for segment $k-1$ is denoted as R_{prev}. As explained in Section 3.1, the adaptation module for MEC picks the video quality based on the video rate suggested by the client, R_c. The proposed MEC-assisted algorithm is unaware of the client's capabilities. Therefore, the clients share with MEC the highest video rate they can play back, based on the observed throughput and buffer occupancy. Pseudo-code for the client-side adaptation algorithm is given in Algorithm 1, which first checks the current buffer occupancy level. If the buffer level is within the danger zone ($B_k < B_{min}$), the algorithm cautiously selects the video rate. The buffer threshold, B_{min}, is equal to the minimum segment duration and part of the buffer size, as follows:

$$B_{min} = \min\left(\text{Segment duration}, 20\%\text{of buffer size}\right) \tag{23}$$

Algorithm 1 Client-side Adaptation

Input: R: set of available discrete video rates, B: buffer level, T: available throughput, τ: segment duration, m: number of available video rates, k: index of the current segment
Output: R_c: Video rate suggested by the client

```
If B^k ≤ B_min then
    R_c = max{ r ϵ R } < α×T^k
Else If B^k > B_min then
    For each bitrate r ϵ R in increasing order
        If B^k - T^k / r ×τ > B_min
            R_c = r
            break
    End for
```

As explained in Section 1, video streaming services offer segments of different durations. As the segment duration increases, the risk of buffer underflow increases in an unstable environment, as shown in Eq. (4). Therefore, segment duration should be considered in the selection of B_{min}. However, with a long segment and a small buffer, it is not feasible to select

B_{min} based only on the segment length. For example, if the segment duration is 10 s, and the buffer size is 20 s, setting B_{min} equal to the segment duration means the client cautiously selects the video rate most of the time. Therefore, buffer size should also be considered in the selection of B_{min}. To this end, we set B_{min} equal to the minimum segment duration and 20% of the buffer size. Once the buffer level increases above B_{min}, the client aggressively selects R_c while ensuring that buffer occupancy does not enter the danger zone. Given the available throughput and segment duration, the client selects the highest video rate such that the predicted buffer occupancy level upon download of the next segment does not fall below B_{min}. The client then shares the suggested video rate, R_c, with the MEC adaptation module to jointly adapt the video rates of the competing clients. Pseudo-code for the MEC-assisted algorithm is given in Algorithm 2.

Algorithm 2 MEC-assisted Adaptation

Input: N: Number of DASH clients, R: set of available discrete video rates, t_{dur}: video duration
Output: Binary allocations, x_{ij}, and integer bitrate allocation, r_{ij}, for each client, $1 \leq j \leq N$

```
For each client 1 ≤ j ≤ N do
  maxUtility = −∞
  R_next = 0
  If BufferStatus = = False
    Run Subroutine 1: Startup Phase
  Else If BufferStatus = = True
    If t_dur > 120 s
      Run Subroutine 3: Long Video
    Else
      F = 0; φ = 0
      Run Subroutine 2: Short Video
  If t_dur = = End of Streaming Session then
    Utility = Utility + maxUtility
```

Subroutine 1 Startup Phase

```
1: For each bitrate r ϵ R in decreasing order
2:   If Segment Number = = 1
3:     If Client Number = = 1
4:       R_next = R_max
5:     Else
6:       R_next = max {r ϵ R} < R_avg
8:  Else
9:     R_next = max {r ϵ R) < T_i^k
```

The algorithm enters the startup phase (Subroutine 1) when the playback buffer is empty. At the start of a streaming session, MEC does not have any information regarding the throughput observed by the clients during segment download. For the first segment, the algorithm selects the highest available video rate for the first client (line 4). For the rest of the clients, the algorithm selects the highest video rate that is less than the average video rate of the competing clients (line 6). The reason is that as the number of streaming clients increase, the throughput available to each client decreases. Once the segment is downloaded, the client calculates the available throughput using Eq. (3). If the client enters the startup phase due to buffer underflow (line 9), the algorithm picks the highest video rate that is less than the available throughput for the next segment.

Subroutine 2 Long Video Adaptation Algorithm

1: Update $B_i(t)$
2: Compute δ_S, δ_{IE}, δ_F, r_{avg}
3: ***If*** $B^{k-1} < B_{min}$
4: ***For*** each bitrate $r \in R$ in decreasing order
5: ***If*** allocation of r satisfies (18) *and* $r < R_c$
6: $U = \delta \times r - \beta \times |r - R_{prev}| - \varphi \times |r - r_{avg}| - \mu \times |r - T_i|$
7: **If** $U > maxUtility$
8: $maxUtility = U$
9: $R_{next} = r$
10: ***Else***
11: ***For*** each bitrate $r \in R$ in decreasing order
12: ***If*** allocation of r satisfies (18) *and* $r < R_c$
13: ***If*** $|r - R_{prev}| \le \delta_S$ *and* $|r - T_i| \le \delta_{IE}$ *and* $1 - \frac{r - r_{avg}}{R_{max} - R_{min}} > \delta_F$
14: Perform operations in lines 6-9
15: ***If*** $R_{next} = 0$
16: ***For*** each bitrate $r \in R$ in decreasing order
17: ***If*** allocation of r satisfies (18) *and* $r < R_c$
18: ***If*** $|r - T_i| \le \delta_{IE}$ *and* $1 - \frac{r - r_{avg}}{R_{max} - R_{min}} > \delta_F$
19: Perform operations in lines 6-9
20: ***If*** $R_{next} = 0$
21: ***For*** each bitrate $r \in R$ in decreasing order
22: ***If*** allocation of r satisfies (18) *and* $r < R_c$
23: ***If*** $1 - \frac{r - r_{avg}}{R_{max} - R_{min}} > \delta_F$
24: Perform operations in lines 6-9
25: ***If*** $R_{next} = 0$
26: ***For*** each bitrate $r \in R$ in decreasing order
27: Perform operations in lines 6-9
28: Update weighting parameters $\rho, \beta, \varphi,$ *and* θ
29: Compute Video Quality, Switching, QoE, Fairness, and Utility Function according to (9), (10), (11), (12), (13)
30: Update Buffer Level
31: Return U_i

As explained in Section 1, the user expects to watch short videos, such as a movie trailer or sports highlights, at the highest feasible video rate [40]. However, the available throughput fluctuates over time in a cellular environment. While streaming a long video (such as a complete movie or a sports match) in an unstable environment, selecting high-quality video rates throughout the streaming session would increase the risk of buffer underflow. Therefore, separate approaches are required to select the video quality for short and long videos. The next question to answer at this point is how to differentiate between short and long videos in terms of duration. The answer to this question is not available in literature. Based on the average duration of movie trailers, the duration of most videos on Facebook, and the average duration of English Premier League (EPL) highlights, we set the maximum duration of a short video to 120 s [1, 2, 17]. If the video is longer than 120 s, the algorithm runs Subroutine 2 (Long Video Adaptation Algorithm). Otherwise, the algorithm runs Subroutine 3 (Short Video Adaptation Algorithm).

5.1 Long video bitrate selection

In this section, we discuss the heuristic adaptation algorithm for long videos. The algorithm's objective is to simultaneously optimize QoE, fairness, and bandwidth utilization. Because

throughput can fluctuate in a cellular network for many reasons, selecting the highest feasible video rate could lead to buffer underflow. Therefore, when the buffer level is in the danger zone ($B_k < B_{min}$), we ignore switching, fairness, and bandwidth inefficiency conditions. The algorithm selects the most feasible video rate that is less than R_c with the highest achievable utility objective value as the video rate for the current segment (lines 5–9). When the buffer level increases above B_{min}, the proposed algorithm considers three known threshold values, δ_s, δ_F, and δ_{IE}, for the switching level, fairness, and the bandwidth inefficiency index, respectively. Switching threshold δ_S is computed as $|max\ \{r \in R\} < T_k - max\ \{r \in R\} < T_k\text{-}1|$, where max $\{r \in R\}$ is the highest video rate in set R that is less than the throughput. The switching index associated with the selected rate is computed as $|r - R_{prev}|$. Bandwidth inefficiency threshold δ_{IE} is computed as $|\max\ \{r \in R\} < T_k - T_k\ |$, and the bandwidth inefficiency index is computed as $|r - T_i|$. The fairness index is computed as $1 - (r - R_{avg})/(R_{max} - R_{min})$, which takes a value between 0 and 1, where R_{avg} is the average video rate of streaming clients. Utility objective value (14) is computed for all available video rates less than R_c that satisfy video rate switching, fairness, and bandwidth inefficiency thresholds (lines 11–13). Among the video rates that satisfy these conditions, the video rate that maximizes the utility objective function is allocated to the clients for the next segment (line 14). If no such video rate is available, we compromise on the switching condition. Next, the utility objective function is evaluated for the set of video rates that satisfy the fairness and bandwidth inefficiency thresholds (lines 15–18). The candidate video rate that maximizes the utility objective function is the selected video rate for the next segment (line 19). If no such video rate is available, we compromise on fairness as well. The objective function is computed for the set of video rates that only satisfy the bandwidth inefficiency condition (20–23). Similarly, the most suitable video rate that maximizes the utility value is streamed for the next segment. If none of the video rates satisfy even the bandwidth inefficiency threshold, the most feasible video rate with the highest achievable objective value is streamed for the next segment (lines 16–19).

After the video rate selection, the weighting parameters of the objective function are dynamically computed. A simplified flowchart of the dynamic tuning of the weighting parameters is shown in Fig. 3. At the start of the streaming session, the tuning parameters are set with initial values. Once the streaming session starts, the weighting parameters of the quality factors (average bitrate (ρ), bitrate switching (β), fairness (φ), and bandwidth inefficiency (θ)) at the download of each segment are dynamically computed. The scheme monitors the effect of each video quality factor (video rate, switching magnitude, fairness, and bandwidth inefficiency) on the user experience of the j^{th} client, $\overline{U}_j$, during the download of 5 previous segments.

$$\overline{U}_j = \rho \times \frac{\sum_{k=S'-5}^{S'} R_{ij}^k}{5} - \beta \times \left| \frac{\sum_{k=S'-5}^{S'} R_{ij}^k}{N} - \frac{\sum_{k=S'-10}^{S'-5} R_{ij}^k}{5} \right| - \varphi \times \left| \frac{\sum_{k=S'-5}^{S'} R_{ij}^k}{5} - \sum_{k=S'-5}^{S'} R_{avg}^k \right| - \theta \times \left| \frac{\sum_{k=S'-5}^{S'} \sum_{j=1}^{C} \left(R_{ij}^k - W \right)}{5} \right| \tag{24}$$

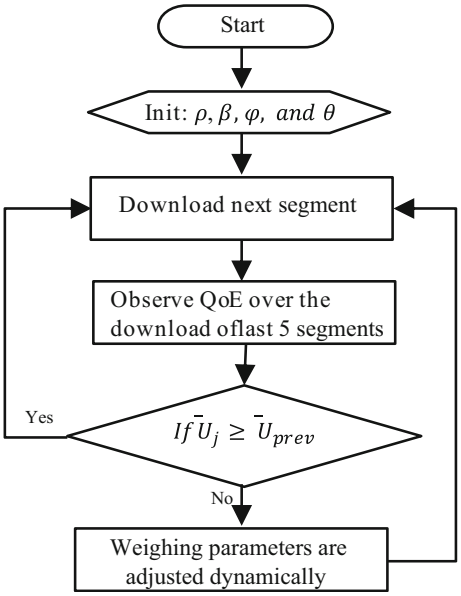


Fig. 3 Flowchart of proposed weighting parameters selection scheme

where S' is the index of current segment. $\overline{U}_{prev}$ represents the value of $\overline{U}_j$ computed for previous segment. *If* $\overline{U}_j \geq \overline{U}_{prev}$, the weighting parameters are not changed. *If* $\overline{U}_j < \overline{U}_{prev}$, the weighting parameters are adjusted by the algorithm. The weighting parameters (δ, β, φ, θ) are calculated based on how far the average of the video rates of the last 5 downloaded segments is from the most feasible bitrates. γ_Q, γ_{QS}, γ_F, and γ_{IE} in Eqs. (25a) to (25d) denote the difference between the selected video rates, and the most feasible video rates for Q_j, QS_j, F_j and IE_j, respectively. $\overline{T}$ denotes the average throughput observed over the download of last 5 segments.

$$\gamma_Q = \left| \frac{\sum_{k=S'-5}^{S'} R_{ij}^k}{5} - \max\{R\} < \overline{T} \right| \tag{25a}$$

$$\gamma_{QS} = \left| \frac{\sum_{k=S'-5}^{S'} R_{ij}^k}{5} - \frac{\sum_{k=S'-10}^{S'-5} R_{ij}^k}{5} \right| \tag{25b}$$

$$\gamma_{QF} = \left| \frac{\sum_{k=S'-5}^{S'} R_{ij}^k}{5} - \sum_{k=S'-5}^{S'} R_{avg}^k \right| \tag{25c}$$

$$\gamma_{IE} = \left| \frac{\sum_{k=S'-5}^{S'} \sum_{j=1}^{N} \left(R_{ij}^k - W \right)}{5} \right| \tag{25d}$$

Next, the weighting parameters are selected according to Eqs. (26a) to (26d):

$$\rho = \frac{\gamma_Q}{\gamma_Q + \gamma_{QS} + \gamma_F + \gamma_{IE}} \tag{26a}$$

$$\beta = \frac{\gamma_{QS}}{\gamma_Q + \gamma_{QS} + \gamma_F + \gamma_{IE}} \tag{26b}$$

$$\varphi = \frac{\gamma_F}{\gamma_Q + \gamma_{QS} + \gamma_F + \gamma_{IE}} \tag{26c}$$

$$\theta = \frac{\gamma_{IE}}{\gamma_Q + \gamma_{QS} + \gamma_F + \gamma_{IE}} \tag{26d}$$

After parameters are updated, the algorithm returns the client's local utility, which was computed using (14).

5.2 Short video bitrate selection

In this section, we discuss the heuristic adaptation algorithm for short videos. For short videos, the objective is to optimize QoE while efficiently utilizing the available bandwidth. Because the throughput in a cellular environment depends on multiple parameters, including propagation distance, client speed, interference, etc., clients located in different regions of the cell would observe different throughput. Therefore, equitably distributing video rates among competing clients means compromising video quality and/or QoE. To this end, fairness is ignored for short videos.

Subroutine 3 Short Video Adaptation Algorithm

1: Update $B_i(t)$
2: Compute δ_S, δ_{IE}, $\bar{r}$
3: **For** each bitrate $r \in R$ in decreasing order
4: **If** allocation of r satisfies (18) *and* $r < R_c$
5: **If** $|r - R_{prev}| \leq \delta_S$ *and* $|r - T| \leq \delta_{IE}$
6: $U = \delta \times r - \beta \times |r - R_{prev}| - \mu \times |r - T_i|$
7: **If** $U > maxUtility$
8: $maxUtility = U$
9: $R_{next} = r$
10: **If** $R_{next} = 0$
11: **For** each bitrate $r \in R$ in decreasing order
12: **If** allocation of r satisfies (18) *and* $r < R_c$
14: **If** $\left|\sum_{j=1}^{C} r_{ij} - W\right| \leq \delta_{IE}$
15: Perform operations in lines 6-9
16: **If** $R_{next} = 0$
17: **For** each bitrate $r \in R$ in the decreasing order
18: Perform operations in lines 6-9
19: Update weighting parameters ρ, β, *and* θ
20: Compute Video Quality, Switching, QoE, Fairness, and Utility Function according to (9), (10), (11), (12), (13)
21: Update Buffer Level
22: Return U_i

The proposed algorithm considers two known threshold values, δ_s and δ_{IE}, for the switching level and the bandwidth inefficiency index, respectively. In decreasing order of video rates, utility objective value (14) is computed for all available video rates less than R_c that satisfy both switching and bandwidth inefficiency thresholds (lines 3–6). The most feasible video rate that has the maximum utility value is then selected as the allocated video rate for the current segment of the client (lines 6–9). If no such video rate is available, the utility objective function is evaluated for the set of video rates less than R_c that only satisfy the bandwidth inefficiency threshold. Here, we compromise on the switching threshold as well (lines 10–15). Similarly, the video rate that maximizes utility objective value (14) is chosen as the video rate for the current segment. If none of the video rates satisfy even the bandwidth inefficiency threshold, the most feasible video rate with the highest achievable objective value is selected as the video rate for the current segment (lines 16–19). After video rate selection, the algorithm updates the weighting parameters as explained in Section 5.1 and returns the client's local utility, which was computed using (14).

5.3 Computational complexity

The computation of estimated throughput during the download of every segment takes $O(\tau)$ time, where τ is the segment duration. Execution of the startup phase results in complexity $O(\tau + |R|)$. The evaluation of switching, and inefficiency thresholds takes $O(|R|)$ units of time. Similarly, the execution of short/long video adaptation algorithms results in complexity of $O(\tau + |R|)$. Putting all the above together gives N competing clients at an overall complexity of $O(N.(\tau + |R|))$. Hence, the overall time complexity for the heuristic algorithm shall be in polynomial time.

6 Performance evaluation

In this section, we implement HTTP–based adaptive video streaming in a multi-access edge computing scenario (as shown in Fig. 1) to evaluate the performance of the proposed algorithm. We implemented the experiments by utilizing the simulation software ns-3. A detailed configuration of the underlying LTE cellular network is shown in Table 2. To achieve adaptive streaming, the HTTP server offers the client 12 presentation levels to adapt video rates, which are 184, 380, 459, 693, 1270, 1545, 2000, 2530, 3750, 5379, 7861, and 11,321 kbps. We assume that all clients can playback the highest available video rate. We set fairness threshold δ_F at 0.6.

We adopted the algorithms proposed in MECA [46], ECAAS [21], AAA [13], DBT [34], DASH-Google [27], and QLSA [5] as benchmarks in order to demonstrate the efficiency of the proposed algorithm. Table 3 presents the properties of the HAS algorithms. The proposed,

Table 2 Cellular network configuration

Cell Layout	Single hexagonal cell
UE distribution	Uniform
Path loss model	Hata Model PCS Extension
BS transmission power	38 dBm
UE distance	1~500 m
Scheduler	Proportional fairness

Table 3 Properties of HAS algorithms

Algorithm	Adaptation Setting	Type	Parameters Observed	Video Quality Objectives	Throughput Estimation Method
Proposed	Hybrid MEC and client-based adaptation	Buffer-based	Buffer level, throughput, video duration, segment duration, buffer size	Maximize video quality, minimize switches, minimize playback interruption, minimize switching magnitude, fairly distribute video rates, efficiently utilize bandwidth	Weighted average method
MECA	MEC-based adaptation	Buffer-based	Buffer level, throughput	Maximize video quality, minimize switches, minimize playback interruption, minimize switching magnitude, fairly distribute video rates	Joint estimation of the throughput of the competing streams
ECAAS	MEC-based adaptation	Buffer-based	Buffer level, throughput	Maximize video quality, minimize switches, minimize playback interruption, minimize switching magnitude, fairly distribute video rates	Throughput observed over the download of previous segment
AAA	Client-side adaptation	Buffer-based	Buffer level, throughput	Maximize video quality, minimize switches, minimize playback interruption	Weighted average method
DBT	Client-side adaptation	Buffer-based	Buffer level, throughput	Maximize video quality, minimize switches, minimize playback interruption, minimize switching magnitude	Weighted average method
QLSA	Client-side adaptation	Throughput-based	Throughput	Maximize video quality, minimize switches, minimize playback interruption, minimize switching magnitude	Weighted average method
DASH-Google	Client-side adaptation	Throughput-based	Throughput	Maximize video quality, minimize switches, minimize playback interruption	Weighted average method with two different coefficients

ECAAS, and MECA are an MEC-assisted DASH systems for mobile video streaming; AAA and DBT are client-side buffer-based algorithms, whereas DASH-Google and QLSA are client-side throughput-based algorithms. In addition to throughput and buffer level, the proposed algorithm also takes segment duration and client's buffer size into consideration to select the video rate. The motivation is to maintain high QoE, equitably select video rates for video clients, and efficiently utilize bandwidth under different client and server settings. The rest of the algorithms do not consider any client or content characteristics to pick video quality. Since the client-based algorithms are oblivious to the decisions made by other competing clients, they do not target bandwidth efficiency and fairness. In addition, the proposed greedy heuristic algorithms solve the rate adaptation optimization problem for short and long duration videos. To the best of our knowledge, this is the first work to design algorithms for both short and long duration videos.

6.1 Long videos

In this section, we evaluate the performance of the proposed long video adaptation algorithm. In our previous work [36], we showed that the performance of existing algorithms struggles to meet conflicting QoE objectives under different client/server settings. The reason is that algorithms employ fixed control laws, even though meeting different video quality objectives requires different strategies. In this work, we evaluated the algorithms under varying client speeds, network conditions, video durations, buffer sizes, and segment durations. A grid-based road topology is used to simulate mobility. The clients remain within a single cell throughput the streaming session. We analyzed the algorithms for the settings mentioned in Table 4. The experiment was repeated 10 times for each setting and the average of the results is presented in this section. The average YouTube video in 2018 was 11.7 minutes [4]. For this section's experiments, a video was streamed for 12 minutes to evaluate the algorithms. As the experiments given in Table 2 were repeated 10 times and a video of 12 minutes was streamed during each run, the performance of each algorithm was analyzed for 120 minutes. The initial values of the tuning parameters for the objective function in (14) were set to $\rho = 0.4$, $\beta = 0.4$, $\varphi = 0.1$, and $\theta = 0.1$. In the following results, we use Jain's fairness index [37] to quantify fairness and bandwidth inefficiency at time t is calculated by $\frac{\left|\sum_j R_{ij}^{(t)} - W\right|}{W}$ [6].

Table 4 Experiment settings used to evaluate the algorithms

Experiment	Buffer Size	Segment Duration	Mobility	Client's Arrival Pattern	No. of Clients
1	15 s	2 s	75 km/h	Random	10
2	15 s	4 s	75 km/h	Random	10
3	30s	4 s	75 km/h	Random	10
4	60s	4 s	75 km/h	Random	10
5	15 s	4 s	3 km/h	Random	10
6	15 s	4 s	75 km/h	Simultaneous	10

6.1.1 Effect of segment duration

Figure 4 displays the performance of the algorithms when the buffer size was set to 15 s and the segment duration was set to 2 s and 4 s, respectively. During both experiments, the client arrival time was uniformly distributed within the first 30 s of the streaming session. In the first experiment, the segment duration was set to 2 sec. Figure 4a shows that the proposed algorithm achieved the highest average video rate among the compared algorithms, followed by ECAAS. Similarly, the proposed algorithm selected video rates fairly while efficiently utilizing the available bandwidth. Figure 5 shows that the proposed algorithm avoided unnecessary playback interruptions. The rebuffering per client metric represents the ratio of clients that experienced a playback interruption to the total number of clients, while average interruptions is the number of times a client experienced a playback interruption. Figures 4 and 5 also reveal that the selection of high video rates and avoiding playback interruptions led to

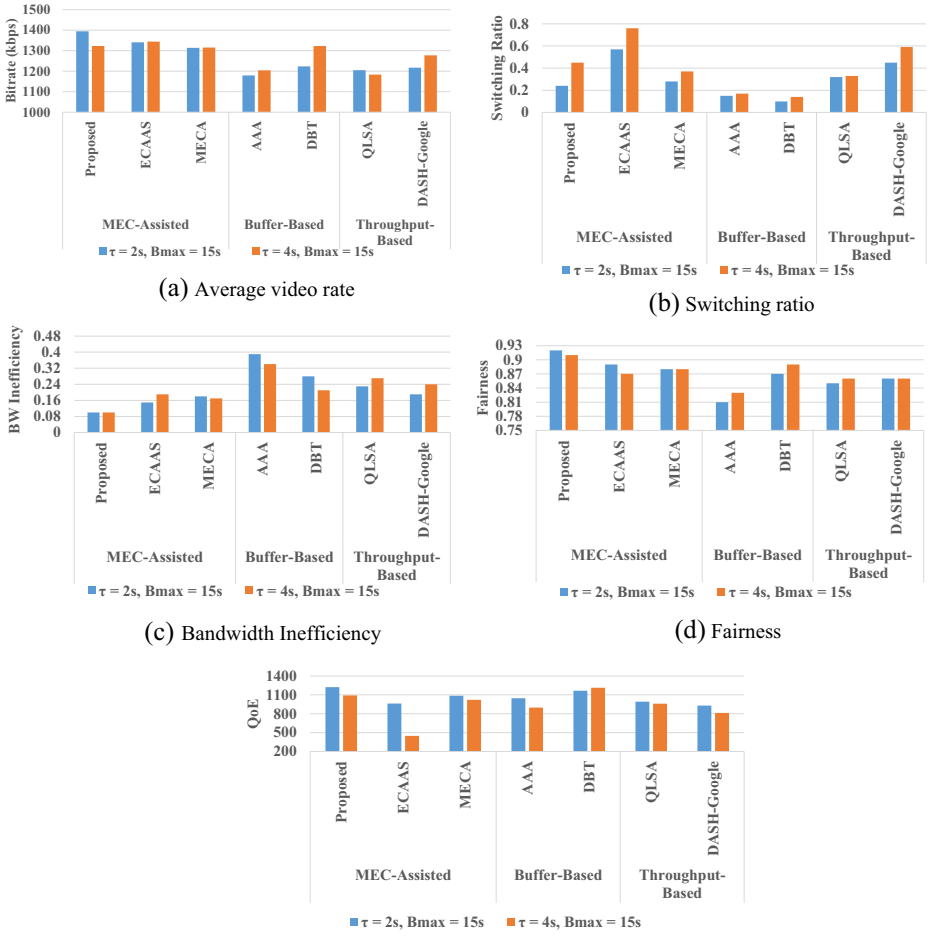
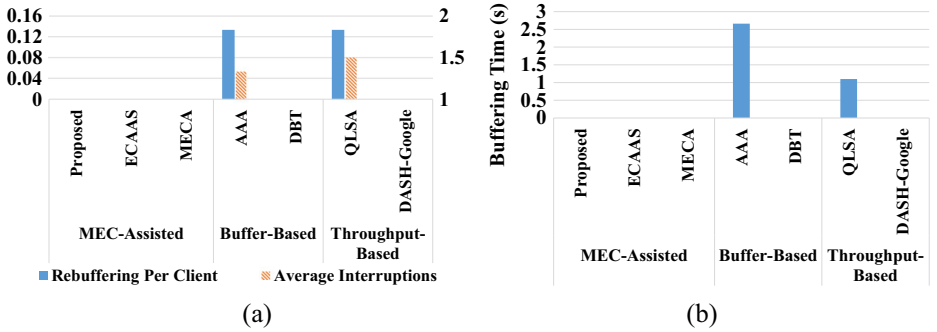


Fig. 4 The effect of segment duration on the performance of the algorithms

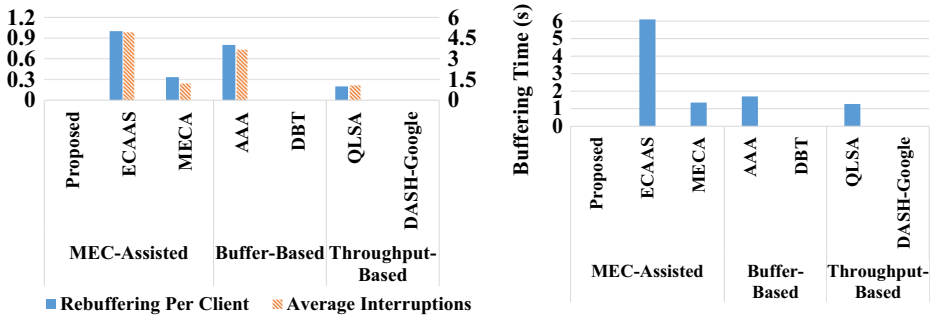


(a) (b)

Fig. 5 Comparison of (**a**) rebuffering per client and average number of interruptions, and (**b**) average buffering time of the algorithms when the buffer size was 15 s and segment duration was 2 s

the highest QoE for the proposed algorithm. The DBT and AAA algorithms experienced fewer video rate switches. The reason is that the algorithms avoided switching video rates unless the buffer level increased above or decreased below predefined thresholds, irrespective of fluctuations in bandwidth. This led to mitigating unnecessary video rate switches, but also compromised video quality. On the other hand, the ECAAS algorithm downloaded high quality segment at the expense of high number of video rate switches.

Next, we increased the segment duration to 4 s. Figure 4 shows that, like the previous experiment, the proposed algorithm achieved the highest video rate. However, the proposed algorithm experienced a high number of video rate switches. Because a longer segment takes more time to download, the proposed algorithm reacted aggressively to reduce the risk of playback interruption. This led to a higher frequency of switches. The DBT algorithm achieved an average video rate similar to the proposed algorithm; however, it achieved the highest QoE due to a low frequency of video rate switches and avoided playback buffer underflow. Figure 6 shows that the proposed, DBT, and DASH-Google algorithms were able to avoid playback interruptions. Furthermore, Fig. 4c and d show that the proposed algorithm achieved the highest fairness and the lowest bandwidth inefficiency values. The reason is that the proposed algorithm jointly optimizes fairness and ensures bandwidth is efficiently utilized. The ECAAS and MECA algorithms achieved better fairness and bandwidth inefficiency, compared to the client-based algorithms.



(a) (b)

Fig. 6 Comparison of (**a**) rebuffering per client and average number of interruptions, and (**b**) average buffering time of the algorithms when the buffer size was 15 s and segment duration was 4 s

6.1.2 Effect of buffer size

In this section, we describe the effect on the performance of the algorithms from varying the buffer size. Figure 7 displays the performance of the algorithms when the segment was 4 s long and the buffer size was 15 s, 30 s and 60 s, respectively. Figure 7 shows that the proposed algorithm streamed higher-quality video irrespective of buffer size. The ECAAS algorithm also downloaded high quality segments, however, it also experienced the highest number of video rate switches. The proposed algorithm equitably allocated video rates to clients, and efficiently utilized bandwidth. Figure 7d shows that the proposed algorithm had the highest QoE value when the buffer size was increased to 30 s and 60 s. The MECA algorithm achieved a similar QoE when the buffer size was 30 s, but its QoE degraded when the buffer size was 60 s, because it downloaded low quality segments. We also observe that the DBT algorithm achieved low QoE and inefficiently utilized the bandwidth when the buffer size increased. The reason is that the algorithm downloads low-quality segments when the buffer size increases. Unlike the proposed algorithm, the DBT algorithm does not adapt the playback buffer

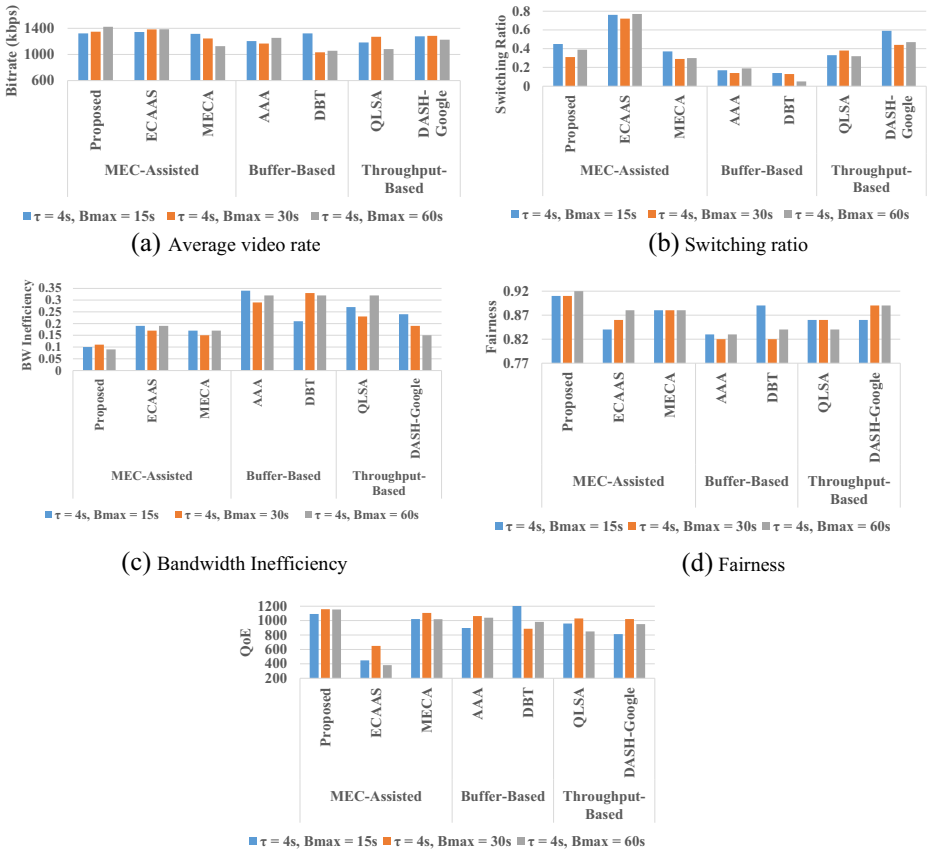


Fig. 7 The effect of buffer size on the performance of the algorithms

thresholds as the buffer size changes. The QoE of throughput-based algorithms fluctuated from one experimental setting to the other.

Figures 6, 8, and 9 show that the proposed algorithm avoided playback interruptions in all the experiments. Figures 6, 8, and 9 also show that only the ECAAS, MECA, AAA, and QLSA algorithms experienced playback interruptions, and the ECAAS algorithm had the most. The reason is that the ECAAS algorithm selects high quality segments at the expense of depletion of playback buffer. In case of a large drop in the throughput in the middle of a segment download, this approach increases the risk of playback interruption. The AAA algorithm also experiences long interruption durations because the algorithm waits for the buffer level to decrease below a pre-defined threshold before it adapts the video quality. Because the video rate cannot adapt in the middle of a segment download, the algorithm failed to protect the buffer when there was a sudden drop in throughput.

6.1.3 Effect of client speed

Here, we compared the algorithms for the following scenarios: (1) the clients moved at vehicular speed (75 km/h), and (2) the clients moved at pedestrian speed (3 km/h). Figure 10 shows that when clients moved at pedestrian speed, the proposed algorithm achieved the highest video rate and fairness value, and the lowest inefficiency value. The table also reveals that the proposed, ECAAS and the MECA algorithms guaranteed high QoE when the clients operated at pedestrian speed. However, the ECAAS algorithm had a lower value when the clients operated at vehicular speed. The result shows that the performance of ECAAS algorithm degrades in case of large fluctuations in throughput. Under stable network condition, the ECAAS algorithm performs better. Figure 10 indicates that the algorithms achieved higher QoE and fairness, and efficiently utilized the bandwidth at pedestrian speed, compared to vehicular speed. Figure also reveals that DBT had the best QoE among the compared algorithms when the clients moved at vehicular speed; however, it underutilized bandwidth when the clients moved at pedestrian speed, whereas the proposed algorithm efficiently utilized bandwidth and downloaded high-quality segments during both experiments. The proposed algorithm did not experience any rebuffering when the clients moved at pedestrian speed, as shown in Fig. 11. The DBT algorithm also avoided playback interruptions at the expense of video quality. The MECA, ECAAS and AAA algorithms experienced buffer underflow while streaming high-quality videos. If a higher weight is given to playback

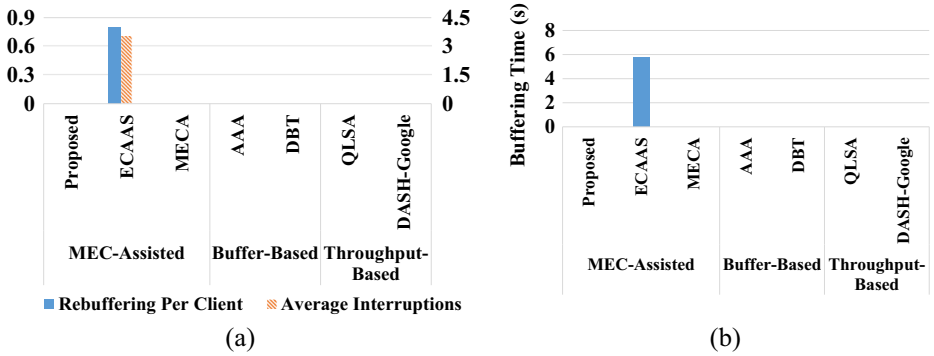
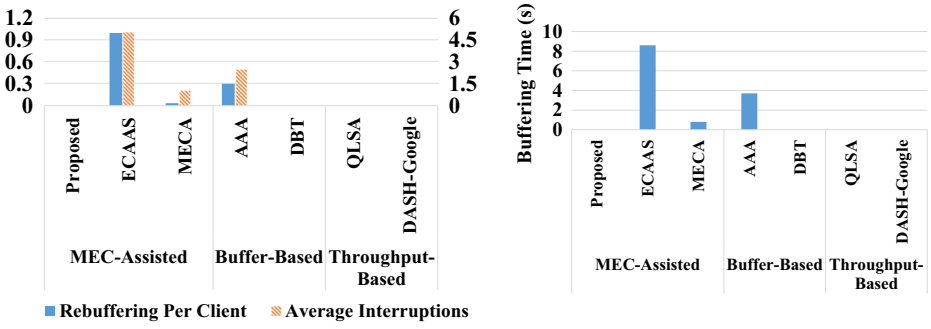
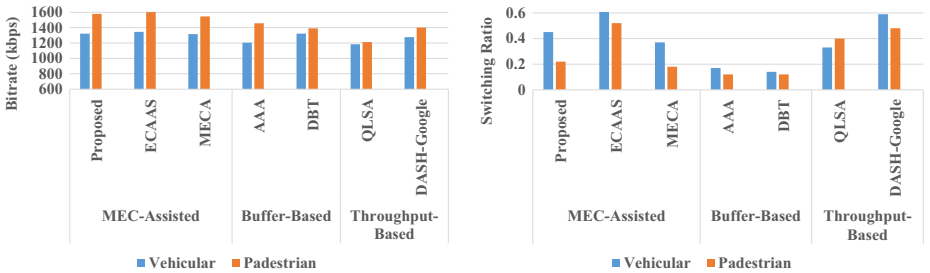


Fig. 8 Comparison of (**a**) rebuffering per client and average number of interruptions, and (**b**) average buffering time of the algorithms when the buffer size was 30 s and the segment duration was 4 s

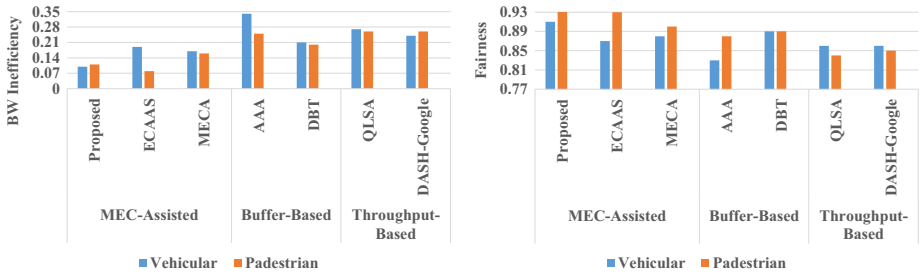


(a) (b)

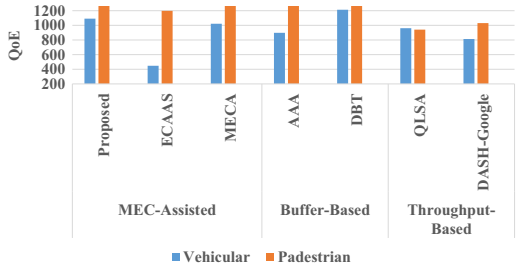
Fig. 9 Comparison of (**a**) rebuffering per client and average number of interruptions, and (**b**) average buffering time of the algorithms when the buffer size was 60 s and the segment duration was 4 s



(a) Average video rate (b) Switching ratio



(c) Bandwidth Inefficiency (d) Fairness



(e) QoE

Fig. 10 The effect of client speeds on the performance of the algorithms

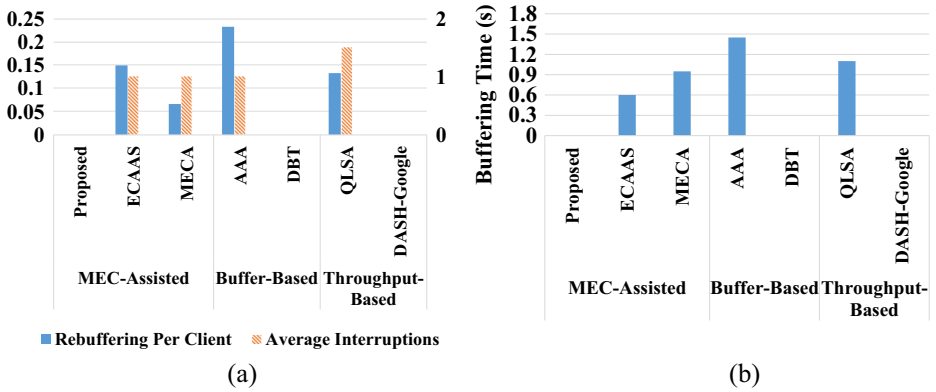


Fig. 11 Comparison of (**a**) rebuffering per client and average number of interruptions, and (**b**) average buffering time of the algorithms when the clients moved at pedestrian speed

interruptions in Eq. (11), the QoE of the ECAAS, MECA and AAA algorithms would decrease further. The DASH-Google and QLSA algorithms are throughput-based methods that are unaware of the client buffer levels and competing clients. Therefore, they reacted aggressively to any changes in throughput, resulting in higher video rate switches and unfair selection of video rates.

6.1.4 Effect of client arrival time

In this experiment, we compared the performance of the algorithms for the following scenarios: (1) all clients simultaneously start streaming, (2) the client arrival time was uniformly distributed within the first 30 s of the streaming session. The results of scenarios (1) and (2) are shown in Fig. 12. Similar to the previous experiment, Fig. 12 shows that the proposed algorithm achieved the highest video rate and guarantees the highest QoE. Furthermore, the proposed algorithm equitably selects video rates and efficiently utilizes bandwidth. Figure 12 also reveals that the algorithms achieved slightly higher video rates and fairness when the clients joined the streaming session randomly. The reason is that when all clients start streaming at the same time, there is a tug-of-war between greedy clients to obtain bandwidth share. The comparison also shows that the proposed algorithm achieved similar QoE in both experiments. However, the QoE of the rest of the algorithms degraded when the clients started the streaming session simultaneously. Figure 13 shows that the proposed, DBT, and DASH-Google algorithms were able to avoid buffer underflow. The ECAA and MECA algorithms downloaded higher quality segments and more efficiently utilized bandwidth, compared to the DBT algorithm, but they achieved low QoE due to a higher number of playback interruptions and higher frequency of switches.

6.2 Small videos

In this section, we compare the performance of the algorithms for short and long video bitrate selection. The initial tuning parameters in objective function (14) were set to $\rho = 0.6$, $\beta = 0.3$, and $\theta = 0.1$. As explained in Section 5.2, fairness was ignored. For short videos, we gave more weight to ρ in Eq. (14), since the objective is to select high quality throughout the streaming session.

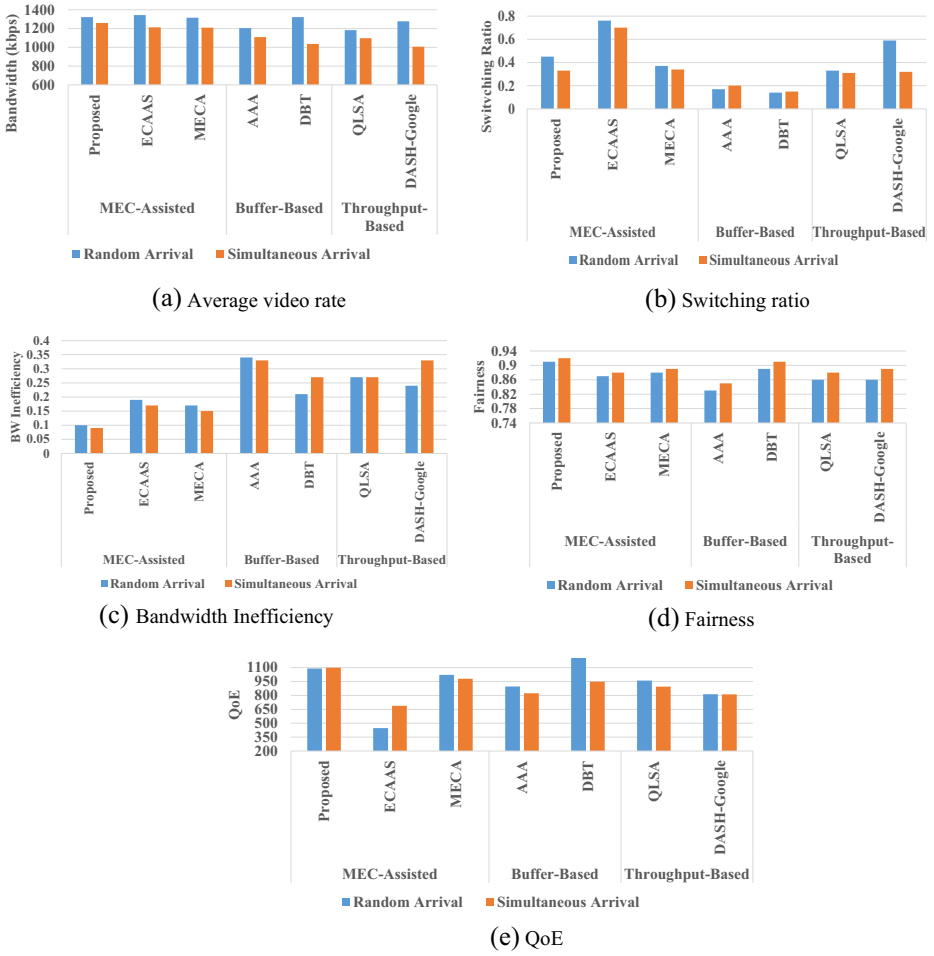


Fig. 12 The effect of client arrival time on the performance of the algorithms

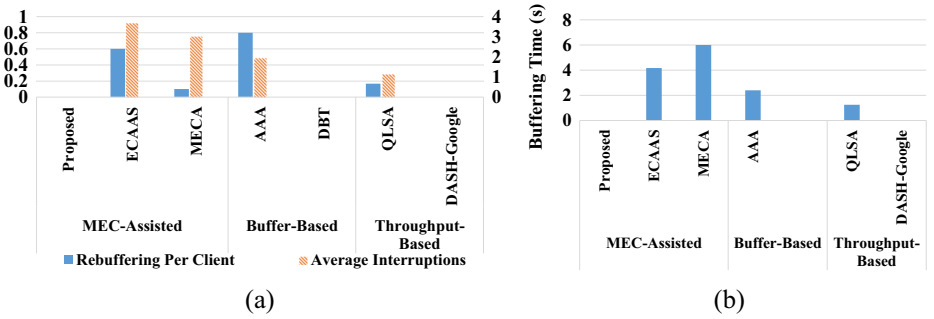


Fig. 13 Comparison of (**a**) rebuffering per client and average number of interruptions, and (**b**) average buffering time of the algorithms when the clients joined the streaming session simultaneously

In a multi-client, bandwidth-constrained environment, a short video adaptation algorithm does not bring any notable advantage over a long video adaptation algorithm while streaming short duration videos (less than 120 s). Due to space limitations, we omit the comparison of the proposed short and long video adaptation algorithms for a bandwidth-constrained environment. Here, we compare the algorithms for a scenario where four clients compete through a bottleneck. The experiment settings used to evaluate the algorithms are given in Table 4. As each experiment was repeated 10 times and a video of 2 minutes was streamed during each run, the performance of each algorithm was analyzed for 20 minutes. Table 5 shows the initial tunings parameters used to evaluate the algorithms for experiments given in Table 6. We compare the following three strategies: (1) short video algorithm, (2) long video algorithm with same initial tuning parameters as used for short video algorithm (long video(SP)), and (3) long video algorithm with tuning initial parameters used in Section 6.1.

Figure 14 (a) displays the average video bitrate and switching ratio experienced by the clients while employing the algorithms and tunings parameters given in Table 5. Figure 14a shows that the short duration adaptation algorithm outperformed the long video adaptation algorithm in both experiments irrespective of the tuning parameters. Even when an adaptation algorithm optimizes QoE, it is important to understand the distributions of the underlying parameters given in Eq. (11). The short video algorithm achieved a higher video rate and experienced fewer video rate switches. Figure 14b shows the short video achieved higher QoE when the achievable throughput was high. The short video algorithm is able to achieve higher QoE as it downloaded high quality video segments, mitigated unnecessary video rate switches and playback interruption. Figure 14b shows that when segment duration is set to 4 seconds, the long video(SP) algorithm performs worse than long video algorithm despite assigning higher weightage to video quality. Because the playback buffer was only 15 s, larger segment duration increased the risk of playback interruption in case of mismatch between the select video rate and available bandwidth. It forced the algorithm to conservatively select video rates to avoid playback interruption. Although the short video adaptation algorithm prioritizes higher video quality at the expense of fairness, Fig. 14c reveals that the short video algorithm achieved similar fairness and bandwidth inefficiency values compared to long video algorithms for both experiments.

6.3 Summary

The results in Section 6.1 reveal that the proposed algorithm guaranteed high QoE irrespective of buffer size, segment duration, client speed, number of clients, and client arrival times. However, the performance of other state-of-the-art algorithms varied from one setting to the other. The reason is that these algorithms employ fixed control strategies, even though optimizing different QoE objectives and experiment settings required different adaptive strategies. The ECAAS algorithm achieved high QoE under stable network conditions.

Table 5 Initial tuning parameters used to evaluate the adaptation algorithms for experiments given in Table 6

Algorithm	Initial tuning parameters in objective function (14)
Short video	= 0.6, β=0.3, and θ=0.1
Long video(SP)	= 0.6, β=0.3, and θ=0.1 (same as short video)
Long video	= 0.4, β=0.4, φ=0.1, and θ=0.1

Table 6 Experiment settings used to compare short and long video adaptation algorithms

	Buffer Size	Segment Duration	Mobility	Client's Arrival Pattern	No. of Clients
1	15 s	2 s	75 km/h	Random	4
2	15 s	4 s	75 km/h	Random	4

However, it experienced high number of playback interruptions in an unstable environment, which degraded its QoE. The DBT algorithm achieved high QoE when the buffer size was small. However, the performance of the algorithm degraded when the buffer size increased, or if all clients started a streaming session simultaneously. Similarly, the performance of the AAA algorithm degraded when the segment duration increased or when all clients joined the streaming session together. Similarly, the algorithms did not meet all the conflicting video-quality objectives. The MECA algorithm streamed high-quality video but at the expense of playback interruptions. The results also indicate that the buffer-based algorithms performed better than throughput-based algorithms.

Table 7 displays the average performance of the adaptation algorithms over all the experiments and demonstrates that the proposed algorithm achieved the highest video rate and the lowest bandwidth inefficiency; and guaranteed the highest QoE among the state-of-the-art algorithms. Furthermore, the proposed algorithm avoided unnecessary playback interruptions during all experiments. However, the proposed algorithm experienced slightly more video rate switches. The reason is that in a highly unstable environment where the users are moving at a high speed, it is important to quickly react to changes in throughput and buffer occupancy levels to avoid unnecessary buffer underflow. The ECAAS algorithm streams high quality video at the expense of high number of switches and playback interruptions. Most of

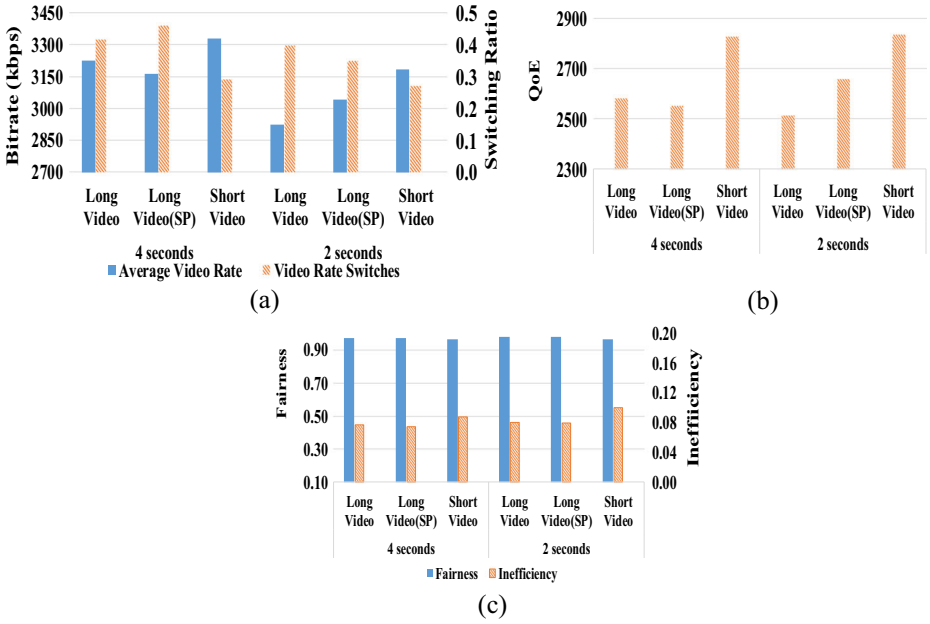


Fig. 14 Comparison of (**a**) rebuffering per client and average number of interruptions, and (**b**) average buffering time of the algorithms when the clients join the streaming session simultaneously

Table 7 Average performance of the adaptive methods over all experiments

Adaptation Algorithms	Proposed	ECAAS	MECA	AAA	DBT	QLSA	DASH-Google
Average Video Rate (kbps)	1387.95	1381.79	1292.48	1228.54	1176.76	1174.57	1235.29
Switching Ratio	0.32	0.67	0.29	0.16	0.12	0.34	0.46
Fairness	0.92	0.89	0.89	0.84	0.87	0.86	0.87
Bandwidth Inefficiency	0.10	0.16	0.16	0.32	0.27	0.26	0.23
QoE	1186.91	718.11	1098.67	1028.42	1080.02	944.54	926.01

the buffer-based algorithms, including DBT and AAA, waited for buffer occupancy to increase above (or decrease below) predefined thresholds. That minimized the switching ratio, but compromised video quality. Furthermore, this approach led to inefficient utilization of bandwidth. The throughput-based algorithms do not have information on the playback buffers; therefore, they did not take the risk of conservatively reacting to changes in bandwidth.

In the following summary, consecutive numbers represent the results from ECAAS, MECA, AAA, DBT, QLSA and Google-Dash, in that order.

The proposed algorithm

1) increased the average video rate by 0.4%, 7.3%, 12.9%, 18%, 18.1%, and 12.3%
2) outperformed existing schemes in terms of fairness by 3.3%, 3.3%, 9.5%, 5.7%, 6.9%, and 5.7%
3) improved bandwidth utilization by 60%, 60%, 220%, 170%, 160%, and 130%
4) outperformed other schemes in terms of QoE by 65%, 8%, 15.4%, 9.8%, 25.6%, and 28%

Only the proposed and DBT algorithms avoided experiencing any playback interruptions. In comparison with the DBT algorithm, the proposed algorithm achieved an 18% higher video rate, delivered 9.8% better QoE, 5.7% higher fairness, and 170% lower bandwidth inefficiency. In addition, the results showed that when high bandwidth is available to competing clients, the proposed short video adaptation algorithm streams a 6% higher-quality segment, performed 45% fewer switches, and improved QoE by 11.1%, compared to the proposed long video adaptation algorithm.

7 Conclusion

In this paper, we presented a context-aware hybrid MEC-assisted quality-adaptation algorithm that exploits video content characters, client-side settings, and application-layer information to achieve multiple objectives including: 1) Jointly optimize the user experience of multiple HAS clients in a cellular environment; 2) Guarantee QoE under varying client, server, dataset, and network settings; 3) Simultaneously meet conflicting video-quality objectives to optimize QoE, while fairly selecting video rates for competing clients, and efficiently utilizing bandwidth. To achieve these objectives, we designed a solution for content-aware MEC-assisted adaptation which considers the joint weighted maximization of QoE, bandwidth utilization, and fairness. Simulation results revealed that the proposed MEC-assisted algorithm outperformed state-of-the-art MEC-assisted and purely client-based algorithms. The results demonstrated that the proposed algorithm guaranteed improved user experience, irrespective of client playback buffer size, segment duration, the number of competing clients, client

movement speed, and client arrival times. The proposed algorithm, on average, improved video quality by over 11%, fairness by over 6%, bandwidth efficiency by over 57%, and QoE by over 22%. Moreover, we presented separate strategies for short and long duration video content based on user expectations. The results showed that the proposed short video adaptation strategy achieved higher QoE and utilized bandwidth more efficiently than the long video strategy when achievable throughput was moderately high.

Acknowledgments This work was partly supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT)(No.2202-0-00047, Development of Microservices Development/Operation Platform Technology that Supports Application Service Operation Intelligence).

Data availability statement The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Adobe (n.d.) Configure HTTP Dynamic Streaming and HTTP Live Streaming. [Online]. Available: https://helpx.adobe.com/adobe.../configure-dynamic-streaming-live-streaming.html
2. Aggarwal V, Jana R, Pang J, Ramakrishnan KK, Shankaranarayanan NK (2011) Characterizing fairness for 3G wireless networks. IEEE Workshop Local Metropolitan Area Netw. pp1–6
3. Aguilar-Armijo J, Timmerer C, Hellwagne H (2021) EADAS: edge assisted adaptation scheme for HTTP adaptive streaming. IEEE Conference on Local Computer Networks
4. Average YouTube video length as of December 2018, by category (n.d.) [Online]. Available: https://www.statista.com/statistics/1026923/youtube-video-category-average-length/
5. Azumi M, Kurosaka T, Bandai M (2015) A QoE-aware quality-level switching algorithm for adaptive video streaming. In: Proc. of IEEE Global Commun. pp 1–5
6. Che X, Ip B, Lin L (2015) A survey of current youtube video characteristics. IEEE MultiMed 2(22):56–63
7. Chen J, Mahindra R, Khojastepour MA, Rangarajan S, Chiang M (2013) A scheduling framework for adaptive video delivery over cellular networks. In Proc Conf Mobile Comput Netw:389–400
8. Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2017–2022 White Paper, Feb 2019. Accessed: Dec. 20, 2020 [Online]. Available:https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html
9. De Cicco L, Mascolo S, Palmisano V (2011) Feedback control for adaptive live video streaming. Proc Conf Multimedia Syst. pp 145–156
10. Dobrian F, Awan A, Joseph D, Ganjam A, Zhan J, Sekar V, Stoica I, Zhang H (2013) Understanding the impact of video quality on user engagement. SIGCOMM Comput Commun Rev 41(4):91–99
11. Egger S, Gardlo B, Seufert M, Schatz R (2014) The impact of adaptation strategies on perceived quality of http adaptive streaming In: Proc. ACM Workshop Design Quality Deployment Adaptive Video Streaming. pp 31–36

12. Farahani R, Tashtarian F, Erfanian A, Timmerer C, Ghanbari M, Hellwagner H (2021) ES-HAS: an edge- and SDN-assisted framework for HTTP adaptive video streaming. In: Proc ACM Workshop Netw Operating Syst Support for Digital Audio and Video. pp 50–57
13. How long is the average movie trailer? (2017) [Online]. Available:https://stephenfollows.com/long-average-movie-trailer/
14. Huang TY, Johari R, McKeown N, Trunnell M, Watson M (2015) A buffer-based approach to rate adaptation: evidence from a large video streaming service. ACM SIGCOMM Comput Commun Rev 44(4):187–198
15. Hung LT, Ngoc NP, Truong CT (2018) Bitrate adaptation for seamless on-demand video streaming over mobile networks. Signal Process: Image Commun, 65: 154–64
16. Huynh-Thu Q, Ghanbari M (2008) Temporal aspect of perceived quality in mobile video broadcasting. IEEE Trans Broadcasting 54(3):641–651
17. Is a shorter video length more engaging on Facebook? (2019) [Online]. Available: https://www.newswhip.com/2019/03/video-length-engaging-facebook/.
18. Jiang J, Sekar V, Zhang H (2014) Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive. IEEE/ACM Trans Netw 22(1):326–340
19. Joseph V, de Veciana G (2014) NOVA: QoE-driven optimization of DASH-based video delivery in networks. In: Proc. IEEE Conf. Comput. Commun. pp 82–90
20. Juluri P, Tamarapalli V, Medhi D (2015) SARA: Segment aware rate adaptation algorithm for dynamic adaptive streaming over HTTP. In: Proc. of IEEE Int. Conf. on Commun. Workshop. pp 1765–1770
21. Kim M, Chung K (2020) Edge computing assisted adaptive streaming scheme for mobile networks. IEEE Access 25(9):2142–2152
22. Li Z, Zhu X, Gahm J, Pan R, Hu H, Begen AC, Oran D (2014) Probe and adapt: rate adaptation for HTTP video streaming at scale. IEEE J Sel Areas Commun 32(4):719–733
23. Liu C, Bouazizi I, Gabbouj M (2011) Rate adaptation for adaptive HTTP streaming. In Proc. MMSys. pp 169–174
24. Liu Y, Dey S, Gillies D, Ulupinar F, Luby M (2013) User experience modeling for DASH video. In: Proc. IEEE Packet Video Workshop. pp 1–8
25. Mehrabi A, Siekkinen M, Ylä-Jääski A (2018) Edge computing assisted adaptive mobile video streaming. IEEE Trans Mobile Comput 18(4):1–17
26. Miller K, Quacchio E, Gennari G, Wolisz A (2012) Adaptation algorithm for adaptive streaming over HTTP. In: Proc. PV. pp 173–178
27. MPEG-DASH / Media Source demo (n.d.) http://dash-mse-test.appspot.com/
28. Muller S Leder C, Timmerer An evaluation of dynamic adaptive streaming over HTTP in vehicular environments. In: Proc. Workshop Mobile Video. pp 37–42
29. Ni P, Eg R, Eichhorn A, Griwodz C, Halvorsen P (2011) Flicker effects in adaptive video streaming to handheld devices. In: Proc. ACM Int. Conf. Multimedia. pp 463–472
30. Petrangeli S, Jeroen F, Claeys M, Latré S, De Turck F (2015) QoE driven rate adaptation heuristic for fair adaptive video streaming. ACM Trans Multimed Comput Commun Appl 12(2):1–15
31. Physical layer aspect for evolved Universal Terrestrial Radio Access (UTRA), document 3GPP TSG-RAN WG1 R1–081483, 2008. [Online] Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=1247
32. Progressive Download and Dynamic Adaptive Streaming Over HTTP, document 3GPP TS 26.247 V12.1.0, 2013. [Online]. Available: http://goo.gl/4EJbvd.
33. Rahman W, Chung K (2017) A novel adaptive logic for dynamic adaptive streaming over HTTP. J Vis Commun Image Represent 49:433–446
34. Rahman W, Chung K (2018) SABA: segment and buffer aware rate adaptation algorithm for streaming over HTTP. Multimed Syst 24(5):509–529
35. Rahman WU, Hong CS, Huh E (2019) Edge computing assisted joint quality adaptation for Mobile video streaming. IEEE Access 7(1):129082–129094
36. Rahman WU, Amin MB, Hossain MD, Hong CS, Huh EN (2021) QoE Optimization for HTTP Adaptive Streaming: Performance Evaluation of MEC-assisted and Client-based Methods. Journal Visual Commun. Image Representation, vol. 82
37. Rajendra J et al (1984) A quantitative measure of fairness and discrimination for resource allocation i n shared computer system. Technical Report, December
38. Seufert M, Egger S, Slanina M, Zinner T, Hoßfeld T, Tran-Gia P (2014) A survey on quality of experience of HTTP adaptive streaming. IEEE Commun Survey Tuts 17(1):469–492
39. Shen Y, Yitong L, Yang H, Yang D (2015) Quality of experience study on dynamic adaptive streaming based on HTTP. IEICE Tran Commun 98(1):62–70
40. Short vs. Long Videos: What Is The Best Explainer Video Length? (n.d.) [Online]. Available: https://www.yumyumvideos.com/short-vs-long-videos-explainer-video-length-wp/

41. Spiteri K, Urgaonkar R, Sitaraman RK (2018) BOLA: near-optimal bitrate adaptation for online videos. IEEE Trans Netw 28(4):1–9
42. Sun Y et.al (2016) CS2P: Improving video bitrate selection and adaptation with data-driven throughput prediction. In: Proc. ACM SIGCOMM Conf. pp 272–285
43. Tran TX, Pandey P, Hajisami A, Pompili D (2017) Collaborative multi-bitrate video caching and processing in mobile-edge computing networks. In: Proc.Conf. Wireless On-demand Netw. Systems Services. pp 165–172
44. Yang SR, Tseng YJ, Huang CC, Lin WC (2018) Multi-access edge computing enhanced video streaming: Proof-of-concept implementation and prediction/QoE models. IEEE Trans Veh Technol 68(2):1888–1902
45. Yin X, Jindal A, Sekar V, Sinopoli B (2015) A control-theoretic approach for dynamic adaptive video streaming over HTTP. ACM SIGCOMM Comput Commun Rev 45(4):325–338
46. Yu J, Wen H, Pan G, Zhang S, Chen X, Xu S (2022) Quality of experience oriented adaptive video streaming for edge assisted cellular networks. IEEE Wireless Communications Letters
47. Zambelli A (n.d.) Microsoft Corporation. IIS smooth streaming technical overview. [Online]. Available: http://www.microsoft.com/enus/download/details.aspx?id=17678
48. Zhao M, Gong X, Liang J, Wang W, Que X, Cheng S (2015) QoE-driven cross-layer optimization for wireless dynamic adaptive streaming of scalable videos over HTTP. IEEE Trans Circuits Syst Video Technol 25(3):451–465