QoE Optimization for HTTP Adaptive Streaming: Performance Evaluation of MEC-assisted and Client-based Methods

Waqas ur Rahman, Muhammad Bilal Amin, Md Delowar Hossain, Choong Seon Hong, and Eui-Nam Huh

Abstract— Seamless streaming of high quality video under unstable network condition is a big challenge. HTTP adaptive streaming (HAS) provides a solution that adapts the video quality according to the network conditions. Traditionally, HAS algorithm runs at the client side while the clients are unaware of bottlenecks in the radio channel and competing clients. The traditional adaptation strategies do not explicitly coordinate between the clients, servers, and cellular networks. The lack of coordination has been shown to lead to suboptimal user experience. As a response, multi-access edge computing (MEC)-assisted adaptation techniques emerged to take advantage of computing and content storage capabilities in mobile networks. In this study, we investigate the performance of both MEC-assisted and client-side adaptation methods in a multi-client cellular environment. Evaluation and comparison are performed in terms of not only the video rate and dynamics of the playback buffer but also the fairness and bandwidth utilization. We conduct extensive experiments to evaluate the algorithms under varying client, server, dataset, and network settings. Results demonstrate that the MEC-assisted algorithms improve fairness and bandwidth utilization compared to the client-based algorithms for most settings. They also reveal that the buffer-based algorithms achieve significant quality of experience; however, these algorithms perform poorly compared with throughput-based algorithms in protecting the playback buffer under rapidly varying bandwidth fluctuations. In addition, we observe that the preparation of the representation sets affects the performance of the algorithms, as does the playback buffer size and segment duration. Finally, we provide suggestions based on the behaviors of the algorithms in a multi-client environment.

Keywords — Quality of experience, DASH, fairness, HTTP adaptive streaming, video

1 Introduction

Multimedia content accounts for the majority of Internet traffic. According to the Cisco Visual Networking Index, global mobile data traffic is expected to reach 82% by 2022 [1]. To handle traffic demands related to multimedia, HTTP adaptive streaming (HAS) solutions are often used. These solutions include Apple HTTP live Streaming (HLS), Adobe HTTP Dynamic Streaming (HDS), Microsoft ISS Smoothing Streaming, and Dynamic Adaptive Streaming over HTTP (DASH) developed under MPEG and standardized by ISO/IEC.

The basic model of HTTP adaptive streaming is shown in Fig. 1. In HTTP adaptive streaming, video content is encoded at different quality levels and fragmented into segments of equal duration. HAS operates by monitoring the network and adjusting the quality of the video stream accordingly. An HAS client initiates a streaming session by downloading a manifest file, which provides a description of the video content available at the HAS server. A HAS algorithm at the client side selects an appropriate segment depending on the received metadata and system conditions, such as the throughput and occupancy of the playback buffer. At the server side, the *content annotation module* provides information about the characteristics of the stored multimedia content. The client initiates the request for the information about the stored content, which is known as metadata. In response to the request from the client, the server sends metadata to the client. The *media preparation module* provides tools for encoding and encapsulation so that the content can be presented and efficiently delivered to the client in the correct format. On the client side, the *scheduler module* is responsible for scheduling the download of upcoming segments. The *bandwidth estimation module* estimates the throughput during the download of the segments. The *adaptation algorithm* selects a suitable bitrate depending on the received metadata and system conditions such as throughput and the occupancy of the playback buffer. The HAS adaptation algorithm attempts to maximize the quality of experience (QoE) by meeting conflicting video quality objectives. These objectives include selecting the highest feasible set of video bit rates, avoiding unnecessary video bit rate changes, assigning equitable video rates among competing video clients, and preserving the buffer level to avoid interruptions in playback [2–5]. It is easier to meet one of the video quality objectives. To maximize the video quality, the users can stream the video at the highest available video rate, which leads to extensive rebuffering in an unstable environment. Then, to minimize the rebuffering, the video can be streamed at the lowest

• *Email: waqas.rahman@gmail.com, bilal.amin@utas.edu.au, {delowar, cshong, johnhuh}@khu.ac.kr*

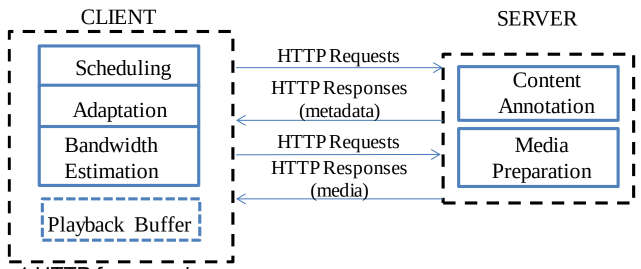


Fig. 1 HTTP framework

available video rate, leading to poor video quality. The aim of an adaptive HAS algorithm is to simultaneously maximize all metrics to improve the QoE.

Traditionally, HAS algorithm runs at the client side [6–11] while the clients are unaware of bottlenecks in the radio channel and competing clients. Decisions are made independently, which may affect the performance of competing clients. Rate adaptation algorithms can be divided into 1) throughput-based and 2) buffer-based methods. Throughput-based algorithms estimate future throughput from past observations to select the video rate for the next segment. Buffer-based algorithms use playback buffer occupancy as an adjustment parameter in addition to the throughput estimation. The segment duration and playback buffer sizes also play an important role in the selection of the video rates. The video streaming services deploy the segment duration differently in their services. Similarly, HTTP clients offer distinct buffer sizes. A client would have more chances to adapt the video rates and accurately estimate the throughput by downloading a smaller segment. In case of sudden fluctuations in the throughput, smaller segment duration may allow the client to quickly adapt the video rates compared to a larger segment. Furthermore, a smaller playback buffer would fill up quickly compared to a larger buffer. Once the playback buffer fills up, a larger playback buffer would allow the client to aggressively select high video quality. The rate adaptive algorithms strive to guarantee QoE under different client settings.

The throughput estimation methods cannot capture the bandwidth fluctuations in cellular networks [12]. It has been shown that they cannot accurately estimate the bandwidth when multiple clients compete for network bottleneck [3] [5]. The authors in [13] proposed that multiple clients cannot achieve fair performance when the bottleneck is the air interface. Furthermore, the unfairness increases as the number of competing clients increase. The clients are unaware of the bottleneck radio channel and cannot coordinate with each other to fairly select video quality [14]. Recently, an edge computing paradigm has been proposed as a promising approach for providing performance superior to that of cloud computing [15–17]. Multi-access edge computing (MEC) [15] offers computation and storage capabilities to the edge of a mobile network by deploying servers within the radio access network. The MEC also provides real-time access to application and RAN information. Moreover, the MEC helps to provide low-latency services to mobile clients requiring intensive computation [18][19]. The user experience could be enriched by moving the video quality adaptation at the MEC. The MEC presents an opportunity to enhance the user experience by centrally adapting the video quality. The advantage of MEC-assisted algorithms is that MECs have the real-time access to information of all clients within a cell. Therefore, a MEC-assisted adaptation algorithm jointly optimizes the video rate selection. Also, MEC-assisted algorithms can exploit the information of the competing clients to fairly assign video quality to competing clients and efficiently utilize the bandwidth to enhance QoE [14][20][21].

Extensive research has been conducted to evaluate the performance of HTTP adaptive streaming algorithms in different experimental scenarios [22–27]. Akashbi et al. [22] evaluated three commercial HAS players and revealed the effectiveness and inefficiencies of the video players. Mueller et al. [12] evaluated multiple HAS systems, including Microsoft Smooth Streaming, Apple HLS, and Adobe HDS in vehicular environments. Thang et al. [23] evaluated throughput- and buffer-based rate adaptation algorithms in the context of live streaming. The authors performed comparisons in terms of bit rate, playback buffer, and perceptual impact on users. Ayad et al. [24] evaluated the operations of different streaming players developed under the MPEG-DASH standard by code level analysis. However, they only investigate the performance of video players within the wired network. Akashbi et al. [5] evaluated the performance of HAS clients competing for network bandwidth. The authors demonstrated that during the steady-state phase, when multiple streams compete for network resources, clients share the bandwidth unfairly. Seufert et al. [25] surveyed quality adaptation in video streaming and discussed its influence on QoE. Kua et al. [26] surveyed key rate adaptation algorithms and classified them based on feedback signals used to select video quality. Stohr et al. [27] provided an evaluation framework to analyze the performance of HAS players.

The above-mentioned studies focus only on client-side rate adaptation algorithms. The authors in [28] surveyed mechanisms that jointly utilize the resources of wireless end devices and the installed MEC to provide services to wireless end devices. This work focused on computation offloading and caching mechanisms to target performance parameters, which include latency minimization, throughput maximization, security enhancement, utility maximization, and energy conservation. Yang el al. [29] implemented the proof-of-concept for the MEC-assisted mobile video streaming services, whereas Martin et al. [30] investigated QoE gains of an MEC-assisted infrastructure. In [31], the authors highlighted the prospects of edge computing for multimedia applications and presented the benefits of using MEC to save cost, bandwidth consumption, energy usage, and latency. Ma et al. analyzed the effectiveness of Wi-Fi and edge content catching solutions for mobile video streaming [32]. This work analyzed the effect of user mobility, cache capacity, content popularity, and caching strategies on the caching performance for video delivery. In [33], we conducted experiments to analyze MEC-assisted rate adaptation algorithms and compared their performance with that of client-based approaches. We only analyzed the effect of varying segment durations, playback buffer sizes,

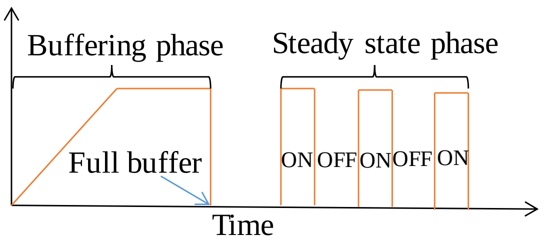


Fig. 2. ON-OFF behavior of HTTP adaptive streaming client.

and number of competing clients. In [21], the authors proposed an MEC-assisted adaptation algorithm and a client to server 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. 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 metrics to compare the performance of the algorithms. This study differs from previous works in the following ways:

- We evaluate, compare, and analyze the performance of MEC-assisted algorithms and conventional client-side rate adaptation algorithms. The aim of the comparison with client-side algorithms is to analyze whether MEC-assisted algorithms achieve the objectives that motivate moving the adaptation intelligence from the client to the edge cloud.
- We evaluate throughput-based and buffer-based algorithms and analyze their performance under different settings.
- We conduct extensive experiments by varying different client and server side parameters, and evaluate the performance of HAS algorithms. We evaluate the effect of varying segment durations, playback buffer sizes, number of competing clients, client moving speeds, client arrival times, and different video datasets on the performance of the algorithms.
- We quantify the benefits and drawbacks of MEC-assisted adaptation and client-side approaches in a multi-client cellular network. We evaluate the algorithms in terms of not only the video rate and dynamics of the playback buffer, but also the fairness and bandwidth utilization.
- Finally, based on the results of the experiments, we suggest guidelines for improving the performance of adaptation algorithms.

The remainder of this paper is organized as follows. Section 2 explains the motivation for shifting the adaptation intelligence to the edge cloud. Section 3 describes the video streaming model and QoE metrics. Section 4 introduces our experimental settings and describes the operation of HAS algorithms. Section 5 evaluates and analyzes the performance of algorithms. Section 6 presents key observations, while Section 7 provides suggestions to improve the performance of the adaptation algorithms. Section 8 discusses future research challenges and directions. Section 9 concludes the paper.

2 Multi-Access Edge Computing-Assisted HAS

Conventional client-side quality adaptation algorithms are widely implemented in the modern streaming systems. Because cellular links are highly dynamic and the underlying TCP is unfair, it is unlikely for an HTTP client to accurately capture the bandwidth share. Standard throughput estimation methods cannot accurately estimate bandwidth fluctuations in the presence of competing video clients [3–5]. The reason for this is that the per-segment throughput cannot estimate the bandwidth share. At the start of a streaming session, an HTTP client downloads a video as quickly as possible to fill the playback buffer. This phase is called the buffering phase. Once the buffer is full, the client enters the ON-OFF phase. This phase is called the steady-state phase. Fig. 2 depicts the ON-OFF behaviors of an HTTP adaptive streaming client. During the OFF state, the client waits for sufficient space in the buffer to download the next segment. The ON state signifies that the client requests the next segment. During the steady-state phase, when multiple clients compete for a network bottleneck, HTTP clients incorrectly estimate the available bandwidth. It has been shown that competing clients result in bandwidth underutilization, unnecessary bit rate switches, and unfair bandwidth sharing. In [4], the authors highlighted that in the presence of competing HTTP clients, the rate adaptation algorithm selects variable and low-quality video, which is undesirable to users. In [5], the authors observed unfair bandwidth sharing among three Microsoft Smooth Streaming clients. This behavior was observed in the presence of competing TCP traffic as well as other competing HTTP clients. Li et al. [3] presented an algorithm Panda (Probe and Adapt) to address fairness and instability of video rate selection. Although the scheme improves the performance of the clients in a wired network, it performed poorly under dynamic cellular links.

Client-side schemes transfer the responsibility of fairness to the underlying TCP; however, the underlying TCP is unfair and unreliable in cellular networks. Moreover, HTTP clients are unaware of other clients within the network. This leads to one client receiving a higher bandwidth share than another client. In this scenario, a client with a better QoE can reduce its bit rate in order to increase the bandwidth share of a competing client with a poor QoE. The client with a poor QoE can then increase its bit rate and improve its QoE. Because the clients are unaware of the performance of competing clients, they cannot decide how aggressively or conservatively they should select their bit rates for fair performance. It is thus logical to shift the adaptation intelligence from the client to the base station. This enables a central controller to jointly optimize the video rate selection. The computational capabilities and storage support of MEC allows for the joint adaptation of multiple clients. Fig. 3 illustrates the edge computing-assisted HAS system for adaptive video streaming over cellular networks. The edge cloud is computationally powerful and can access the RAN information available in the base station. This is feasible

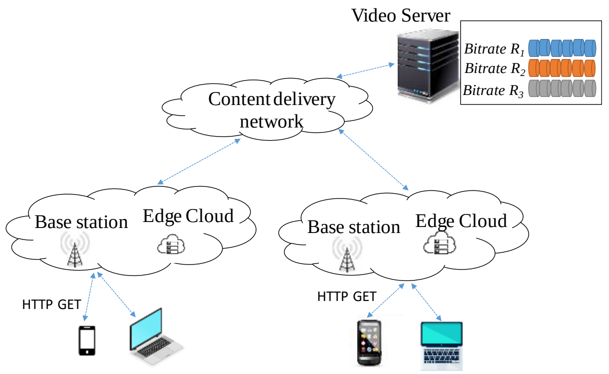


Fig. 3. Streaming architecture for multi-access edge computing-assisted video streaming.

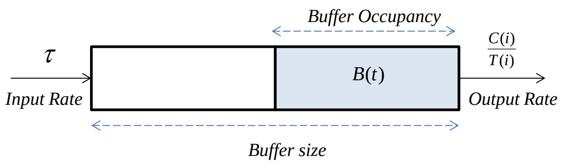


Fig. 4. Dynamics of playback buffer.

and compatible with the centralized resource management in cellular network. 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 adaptation module at the edge cloud can access the channel knowledge of multiple clients for joint adaptation. In the client-based adaptation, the network forwards the client requests to the video server. In case of an MEC-based adaptation, the MEC can intercept the requests and overwrite adaptation decisions based on the high-layer playback information [14]. The client playback information such as buffer size, buffer level, batter percentage, and QoE status can be embedded in the feedback from clients. This is feasible as the 3rd Generation Partnership Project standardized the quality metric reporting process for the clients by using the HTTP POST request carrying the XML formatted metadata [34]. The jointly selected video rates by MEC are then delivered to the video servers to stream the upcoming segment. In this manner, the clients can download the jointly adapted video segments using the standard request-response framework of HAS without any modification in the current infrastructure of cellular schedulers and video servers.

3 Video Streaming Model

For the dynamic adaptation of video streaming, a video is fragmented into τ second segments. Each segment is encoded at different video rates. The set of video rates available for the video stream is denoted by R where $R = \{R_1, R_2, R_3, \ldots, R_N\}$. The video client can choose to download segment i at bitrate $R_k(i)$. The higher the video rate selected, the higher the video quality is perceived by the viewer. Let $q(.) : R \rightarrow \mathbb{R}_+$ be a non-decreasing function which maps video rate to the quality perceived by the viewer $q(R_k(i))$.

The video segments are downloaded into playback buffer. Data of τ seconds is added to the playback buffer when the segment is completely downloaded. The client begins to play the video after the first segment has been completely downloaded. The client selects $R_k(i)$, the k^{th} video rate, from the set of N available video rates for the i^{th} segment. Fig. 4 illustrates the dynamics of the playback buffer. If $R_k(i)$ is the selected video rate of the i^{th} segment, then $\tau \times R_k(i)$ is the size of the segment. The download time of the segment will be $(R_k(i) \times \tau / T(i))$ where $T(i)$ is the throughput observed by the client during the download of the i^{th} segment. Let $B(i) \in [0, B_{max}]$ be the buffer occupancy at the download of the i^{th} segment. During the streaming session, the buffer level remains between 0 and B_{max}. Once segment i is downloaded, the client waits for Δt^i seconds before sending the request for the i^{th}+1 segment. Waiting time is given by:

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

Throughput $T(i)$ during the download of the i^{th} segment is calculated as follows:

$$t^{i+1} = t^i + \frac{R_k(i) \times \tau}{T(i)} + \Delta t^i \tag{2}$$

$$T(i) = \frac{1}{t^{i+1} - t^i - \Delta t^i} \int_{t^i}^{t^{i+1} - \Delta t^i} T_t \, dt \tag{3}$$

At time t^i when the video client selects $R_k(i)$, only the past throughput $\{T_t,\ t < t^i\}$ is available, while future throughput values are not known. If the buffer level at the end of the download of chunk i-1 is $B(i$-1$)$; then, $B(i)$ is given by:

$$B(i) = B(i-1) + \tau - [\tau \times R_k(i) / T(i)] \tag{4}$$

Equation (4) demonstrates that if the selected video rate is greater than the throughput, then the playback buffer is depleted.

3.1 QoE Metrics

The objective of an adaptation algorithm is to optimize the QoE of viewers to achieve long-term user engagement. Several studies have been conducted to determine the factors that affect user engagement. These factors include selecting the highest feasible set of video bit rates, avoiding unnecessary video bit rate switches, and preserving the buffer level to avoid an interruption in playback [38–40].

The average video bitrate over downloaded segments by the client is given by:

$$Q = \frac{\sum_{i=1}^{S} (R_k(i))}{S} \tag{5}$$

where $R_k(i)$ is the k^{th} video rate selected for the i^{th} segment and S is the total number of segments downloaded by the client.

Frequent video rate switches inversely affect the user experience. Magnitude of the changes in the quality from one segment to another is given by:

$$QS = \frac{\sum_{i=1}^{SS} R_k(i) - R_k(i-1)}{Number\ of\ Switches} \quad (6)$$

The client experiences playback interruptions if the download time ($\tau \times R_k(i)\ / T(i)$) is higher than the playback buffer occupancy level. The interruption time during the download of the i^{th} segment, $I^R(i)$, is $(\tau \times R_k(i)\ / T(i) - B(i))_+$. The notation $(x)_+ = \max(x, 0)$ ensures that the term is always positive.

The largest impact on user experience originates from playback interruption due to buffer underflow [41]. One long interruption is preferred to multiple short interruptions [42]. For down switching, abrupt switching impairs the QoE to a larger extent than smooth switching [2]. On average, the minimum quality level is rated 30% better for gradual video rate switching than for instantaneous switching [43]. Experiments have indicated that a small noticeable difference in quality is observed between high- and mid-quality switches during video playback [44]. The authors of [45] suggested that the user experience improves when the video rate is increased aggressively, as it leads the users to believe that the provider is attempting to maximize the QoE. In addition, a long period of high-quality video improves the user experience [46]. In this study, we use the QoE metric used by the authors of [47], which is defined as follows:

$$QoE = \sum_{i=1}^{S} q(R_k(i)) - \mu \sum_{i=1}^{S} I^R(i)$$
$$- \sum_{i=1}^{S} |q(R_k(i)) - q(R_k(i-1))| \quad (7)$$

For a video fragmented into N number of segments, $q(R_k)$ maps the video rate to the quality perceived by the viewer. $I^R(i)$ represents the rebuffering time during the download of the i^{th} segment, while the final term discourages frequent changes in the video rates. The authors of [34] used $q(R_k) = R_k$ and $\mu = 3{,}000$, signifying that a playback interruption of 1 s receives the same penalty as reducing the bit rate of a segment by 3,000 kbps. We consider the same values in our evaluation. In this study, we calculate the average QoE per segment, that is, the total QoE metric divided by the number of segments.

3.2 Fairness and Inefficiency

Rate adaptation algorithms are fairly effective when a client operates alone. In an environment in which multiple clients compete for the bottleneck, the clients are inefficient and select low-quality video rates. Furthermore, bandwidth is shared unfairly among the competing clients.

The inefficiency at time t is given by the following [48]:

$$Inefficiency = \frac{|\sum_x R_{x,t} - W|}{W} \quad (8)$$

where W is the bandwidth, and each client x selects bit rate $R_{x,t}$.

Multiple clients competing at the bottleneck must be able to achieve equitable video rates. To quantify fairness, the Jain fairness index is used [49]. The Jain fairness index of $R_{x,t}$ for all players x is given by:

$$Fairness = \left[\sum_{x=1}^{n} R_{x,t}\right]^2 / n \sum_{x=1}^{n} R_{x,t}{}^2, \quad x \geq 0 \quad (9)$$

Ideally, the inefficiency value should be 0 while the fairness value should be 1. In other words, low values of inefficiency and high value of fairness are desired. A low inefficiency value signifies that the client selects the highest feasible bit rates that are lower than the actual throughput, while a high fairness value signifies that the competing clients achieve equitable video rates.

4 Experimental Framework

In this section, we present our ns-3-based experimental environment, video quality metrics, and details of the rate adaptation algorithms used for evaluation. We modified code available from https://github.com/djvergad/dash to perform our experiments. It provides an MPEG-DASH client-server ns3 module for simulating HAS algorithms. To implement the cellular network, we used ns-3 LTE module by the LENA project [50].

TABLE 1
CELLULAR NETWORK CONFIGURATION

Cell Layout	Single hexagonal cell
UE distribution	Random
Path loss model	Hata Model PCS Extension
BS transmission power	38 dBm
UE distance	1 ~ 500 m
Scheduler	Proportional fairness
UE speed	Pedestrian (3km/h) / Vehicular (75km/h)

4.1 Experimental Settings

We implemented HTTP-based adaptive video streaming in the MEC scenario presented in Fig. 3. An LTE network was used as the underlying cellular network. A detailed configuration of the cellular network is provided in Table 1. The adaptation module at the MEC can access the channel knowledge of multiple clients for joint adaptation. The HAS server shares the media presentation description with the MEC and the clients so that they have knowledge of the video representations stored on the server. In conventional client-side adaptation, the cellular network forwards the client request to the server. Unlike conventional client-side adaptation, MEC-assisted adaptation algorithms jointly select the video rates for the competing clients at the edge cloud.

Table 2 presents the content information used for evaluating the algorithms. The HAS server stores two test sequences Tears of Steel (Dataset 1) and Big Buck Bunny (Dataset 2). Dataset 1 is segmented into three different durations. Microsoft Smooth Streaming, Adobe HTTP Dynamic Streaming, and Apple HTTP Live Streaming offer segment duration of 2, 4 and 10 s respectively [35–37]. In this work, we select segment durations of 2, 4, and 10 s for the evaluation. Dataset 1 was pre-encoded into 12 different video rates, with dataset 1 encoded into higher video rates than dataset 2. In addition, the video rates in dataset 2 were more closely spaced than those in

TABLE 2
CONTENT INFORMATION USED FOR EVALUATION

Dataset	Segment Duration	Bitrates (kbps)	Duration
1	2, 4, 10	184, 380, 459, 693, 1270, 1545, 2000, 2530, 3750, 5379, 7861, 11321 kbps	600s
2	2, 4	45, 88, 128, 177, 217, 255, 323, 378, 509, 577, 782, 887, 1008, 1207, 1473, 2087, 2409, 2944, 3340, 3613, 3936 kbps	600s

dataset 1. Both videos are 600 seconds long. A number of rate adaptation algorithms select video rates higher than the bandwidth at the expense of the playback buffer. However, other algorithms select video rates lower than the bandwidth despite a high playback buffer level. In this paper, we analyze the effect of the video rates on both types of algorithms.

4.2 Tested HAS Algorithms

As explained in Section 1, HAS algorithms can be summarized into throughput-based and buffer-based methods. Throughput-based algorithms select the video rates of the segments based on only the throughput observed during the download of the segments. Buffer-based algorithms observe playback buffer level in addition to the throughput to select the video rate of the future video segments. In addition, some HAS algorithms also consider segment duration and segment size to select the video rates.

In this section, we summarize the operation of the HAS algorithms used for experimentation and analysis. Table 3 presents the properties of the HAS algorithms. We adopt the solutions proposed in [20], [21], [6], [7], [8], [10], [51], and [52] to analyze the performance of these algorithms and refer to them as MECA, ECAA, NALD, SARA, QLSA, AAA, DASH-Google, and DBT, respectively. In addition, we analyze the performance of the algorithm that selects the highest video rate that is lower than the available bandwidth. We refer to this algorithm as Instant.

1) MECA

This algorithm is a hybrid heuristic adaptation algorithm designed to jointly enhance multiple conflicting video quality objectives. Because the MEC is unaware of the client's capabilities, the client provides the MEC with the highest video rate it can playback based on the display capabilities and buffer level. Based on the suggestion from the client, the algorithm jointly selects the video rates of the competing clients on the basis of the estimated throughput. The algorithm selects the lowest available video rate for the first segment. To increase and decrease the video rate, the algorithm calculates the Fairness and Switching indices. The algorithm considers thresholds δ_F and δ_S of fairness and switching, respectively. The fairness threshold δ_F is set equal to 0.7. The fairness index associated with the selected bit rate is computed as $1 - (R_k(i) - R_{avg})/(R_{max} - R_{min})$, which takes the value between 0 and 1. R_{avg}, R_{max}, and R_{min} denote the average of the video rates of the active competing clients, highest available video rate, and lowest available video rate, respectively. The switching threshold δ_S is computed as $|max\{R\} < T(i-1) - \max\{R\} < T(i)|$ where $\max\{R\} < T(i)$ means the highest video rate in the set R that is less than the throughput. The switching index associated with the selected bit rate is computed as $|R_k(i) - R_{prev}|$, where R_{prev} is the video rate selected for the previous segment.

The highest video rate that is lower than the estimated throughput that satisfies the following condition is selected: Fairness index > δ_F. To decrease the video rate, the selected video rate must satisfy the following two conditions: (1) $R_{prev} > R_{avg}$ and (2) Switching index ≤ δ_S.

2) ECAA

ECAA is a heuristic algorithm for efficiently solving video client-to-edge-server mapping and video rate selection. The algorithm allocates a mobile client to the nearest base station or the base station with the highest achievable video rate. Unlike the MECA algorithm, the ECAA algorithm selects video rates independently and does not rely on any guidance from the clients. Once the client has been allocated to the base station, the algorithm selects the highest available video rate for the first segment. For the following segments, the algorithm selects the video rate that results in a low switching level and high fairness value. First, the algorithm calculates the highest video rate that does not result in buffer underflow given the current bandwidth. The highest video rate for the i^{th} segment is selected according to max $\{R\} \le (Thr, B)$, where the buffer level B (in kbit) is calculated as follows:

$$B_i = \begin{cases} B_{i-1} + Thr_i \times D, A_i \le t \le A_i + L_i \\ B_{i-1} + (Thr_i - r_i) \times D, A_i + L_i < t \end{cases} \tag{10}$$

where A_i is the client arrival time, and L_i is the initial buffer delay. It should be noted that the authors in [21] calculate the buffer level in kbit. In this study, we calculate the buffer level in seconds based on the number of segments downloaded into the buffer and the current playback time of the streaming session. The algorithm considers thresholds δ_F and δ_S of fairness and switching, respectively, and selects the video rate that results in a switching level less than δ_S and fairness value greater than δ_F. If no video rate satisfies these conditions, the algorithm selects the video rate that results in a switching level less than δ_S and ignores the fairness condition. If there is still no feasible video rate, it selects the most sustainable video rate. In this study, we adopt the ECAA algorithm for a single-cell scenario in which the MEC allocates video rates to the clients.

TABLE 3
PROPERTIES OF HAS ALGORITHMS

Algorithm	Adaptation setting	Type	Parameters observed	Video quality objectives	Throughput estimation method
MECA	MEC-assisted 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
ECAA	Hybrid MEC and client-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 download of last segment
NALD	Client-based adaptation	Buffer-based	Buffer level, throughput, segment duration, buffer size	Maximize video quality, minimize switches, minimize playback interruption	Switches between weighted average and moving average based on the network conditions
AAA	Client-based adaptation	Buffer-based	Buffer level, throughput	Maximize video quality, minimize switches, minimize playback interruption	Weighted average method
SARA	Client-based adaptation	Buffer-based	Buffer level, throughput , segment size	Maximize video quality, minimize switches, minimize playback interruption	Weighted harmonic mean method
DBT	Client-based adaptation	Buffer-based	Buffer level, throughput	Maximize video quality, minimize switches, minimize playback interruption, minimize switching magnitude	Weighted average method
Instant	Client-based adaptation	Throughput-based	Throughput	Maximize video quality, minimize switches, minimize playback interruption	Throughput observed over download of last segment
QLSA	Client-based adaptation	Throughput-based	Throughput	Maximize video quality, minimize switches, minimize playback interruption, minimize switching magnitude	Weighted average method
DASH-Google	Client-based adaptation	Throughput-based	Throughput	Maximize video quality, minimize switches, minimize playback interruption	Weighted average method with two different coefficients

3) NALD

The NALD algorithm is a buffer-based algorithm that observes the estimated throughput and playback buffer level to download the video segments. The algorithm maps the video rates to the corresponding buffer thresholds. When the buffer level is less than 20% of the buffer size, the proposed algorithm selects the lowest available video rate. To increase the video rate in response to the increase in the throughput and buffer level, the following two conditions should be satisfied. First, the selected video rate must be lower than the estimated throughput. Second, for a client to select the video rate, the buffer level must be higher than the corresponding buffer threshold. To decrease the video rate, the algorithm remains at the current video rate unless the buffer level decreases below the threshold. When the buffer level decreases below the threshold, irrespective of the estimated throughput, the algorithm selects the highest video rate such that the buffer level at the download of the next segment does not decrease below 20% of the buffer size.

4) SARA

The SARA algorithm considers the sizes of the upcoming segments in addition to the throughput and buffer occupancy to predict the segment download time. The HTTP server sends an enhanced manifest file that contains information about the segment sizes. To accurately predict the download rate of the next segment, the algorithm assigns weights (w_1, w_2…) that are proportional to the segment sizes. To calculate the harmonic mean, these weights and the respective download rates (d_1, d_2…) are used to calculate the harmonic mean download rate.

The harmonic mean download rate for n downloaded segments is calculated as follows:

$$H_n = \sum_{i=1}^{n} w_i / \sum_{i=1}^{n} \frac{w_i}{d_i} \tag{11}$$

The algorithm divides the playback buffer into multiple predefined thresholds: I, B_α, and B_β, where $I < B_\alpha < B_\beta$. When the buffer occupancy is less than I, the client selects the lowest bit rate. When the buffer level is between I and B_α ($I < B_{curr} < B_\alpha$), the bit rate is incremented in single steps. As the buffer level increases and remains between B_α and B_β ($B_\alpha < B_{curr} < B_\beta$), the client selects the most suitable video rate that is greater than or equal to the current video rate. When the buffer level increases above B_β, the most suitable video rate for the current throughput is selected. The request to download the next segment is only sent when the buffer level decreases to B_β.

5) AAA

Miller et al. [10] proposed a method that divides the buffer into multiple predefined thresholds B_{min}, B_{low}, and B_{high}, where $B_{min} < B_{low} < B_{high}$. This method makes different decisions to select video rates when the buffer level remains in different ranges. The streaming session is divided into two phases of operation: a *fast start phase* and a *steady phase*. The *fast start phase* is introduced at the beginning of playback, with the goal to aggressively increase the quality of the downloaded segments. During the *steady phase*, the algorithm conservatively selects the video rate when the buffer level is low. When the buffer level decreases below B_{low}, the algorithm reduces the video rate to avoid playback interruption. As the buffer level falls below B_{min}, the algorithm selects the lowest available video rate. If the buffer level remains between B_{low} and B_{high}, the algorithm remains at the current video rate to mitigate the frequency of playback interruptions. When the buffer level increases above B_{high} and the risk of playback interruption decreases, the algorithm aggressively selects high video rates.

6) DBT

The DBT algorithm applies the buffer-based approach in selecting the video rates. The video rate is increased or decreased depending on the relationship between the current video rate and the estimated throughput. The video rate is decreased only when the estimated throughput is less than the current video rate. The algorithm divides the video rate into three buffer ranges through thresholds. The buffer ranges are named the dangerous range $[0, B_{min}]$, the low range (B_{min}, B_{low}), and the safe range (B_{low}, B_{max}). If the buffer level is within the safe range, the client maintains the current video rate irrespective of the bandwidth to mitigate unnecessary video rate switches. If the buffer level is within the low range, the algorithm decides to decrease the video rate immediately. The switching magnitude is adaptively decided to keep the buffer level above B_{min}. When the buffer level drops below B_{min}, the algorithm selects the lowest available video rate. When the throughput increases, the algorithm does not abruptly increase because when the buffer level is low since future throughput fluctuations may force the client to decrease the video rate. The algorithm maintains the current video rate until the buffer level reaches B_{max}. Once the buffer level reaches B_{max}, the video rate is immediately increased to quickly improve the video quality. The algorithm selects the highest video rate less than the estimated throughput for the next segment.

7) QLSA

The QLSA is a throughput-based algorithm with two key features: insertion of an intermediate video rate and mitigation of video rate switches. The algorithm first calculates $\Delta = l_{tmp} - l_{prev}$, where l_{prev} defines the quality level of the previous segment, and l_{tmp} calculates the highest video rate that is lower than the available bandwidth. The algorithm selects two throughput thresholds: Th_{low} and Th_{high}. If $\Delta < -Th_{low}$, the algorithm gradually decreases the video quality. If $Th_{low} < \Delta < Th_{high}$, the algorithm selects the highest video rate that does not exceed the weighted average of the available throughput. If $\Delta > Th_{high}$, the algorithm selects l_{tmp} as the next video rate.

8) DASH-Google Player

Algorithm 1 provides the pseudo code of Google's MPEG-DASH Media Source demo [51]. The algorithm selects video rates only on the basis of the throughput observed during the download of the segments. The algorithm uses a moving average method with two different bandwidth estimation coefficients to reflect small- and large-scale bandwidth fluctuations. It then selects the minimum of the two as the bandwidth estimate for the next segment. Finally, the algorithm selects the highest available video rate that is lower than the bandwidth estimate.

9) Instant Method

The observed throughput $T(i)$ over the download of the previous segment i is used as the estimated throughput, $T^E(i+1)$, for the next segment.

$$T^E(i+1) = T(i) \tag{12}$$

The Instant throughput method selects the bit rate based on the instant throughput with a small safety margin, μ, as follows [53]:

$$R_k(i+1) = (1-\mu) \times T^E(i+1) \tag{13}$$

Algorithm 1: DASH-Google Player Algorithm

E_{slow}: Bandwidth estimation for small-scale bandwidth fluctuations

E_{fast}: Bandwidth estimation for large-scale bandwidth fluctuations

α_{slow}: Exponential moving average coefficient for E_{slow} (0.99)

α_{fast}: Exponential moving average coefficient for E_{fast} (0.98)

Thr: Bandwidth observed over the download of the previous segment

BW: Estimated bandwidth for the next segment

R_{next}: Next selected video rate

```
while segment is downloaded do
$E_{slow} = \alpha_{slow} \times E_{slow} + (1-\alpha_{slow}) \times Thr$
$E_{fast} = \alpha_{fast} \times E_{fast} + (1-\alpha_{fast}) \times Thr$
$BW = \min(E_{slow}, E_{fast})$

for each bitrate in the set $R$ in the decreasing order
  If $BW >= R_k(i)$ then
    $R_{next} = R_k(i)$
    break
```

5 Performance Evaluation

In this section, we discuss our experiments to evaluate the adaptation methods under varying client, server, dataset and network settings. We evaluate the effect of varying segment durations, playback buffer sizes, number

TABLE 4
EXPERIMENT SETTINGS USED TO EVALUATE THE IMPACT OF SEGMENT DURATION

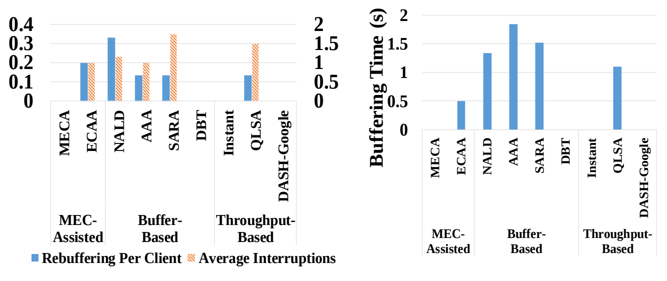
Experiment #	Dataset	Buffer Size	Segment Duration	Mobility	Client's Arrival Pattern	No. of Clients
1	1	15s	2s	75 kmph	Random	10
2	1	15s	4s	75 kmph	Random	10
3	1	60s	4s	75 kmph	Random	10
4	1	60s	10s	75 kmph	Random	10

TABLE 5
PERFORMANCE OF ALGORITHMS WHEN BUFFER SIZE IS 15 S AND SEGMENT DURATION IS 2 S

	MEC-Assisted		Buffer-Based				Throughput-Based		
Adaptation Algorithms	MECA	ECAA	NALD	AAA	SARA	DBT	Instant	QLSA	DASH-Google
Average Video Rate (kbps)	1313.53	1462.57	1254.78	1179.13	1265.09	1223.38	1233.36	1205.04	1216.87
Switching Ratio	0.28	0.34	0.26	0.15	0.67	0.10	0.42	0.32	0.45
Fairness	0.88	0.88	0.86	0.81	0.81	0.87	0.87	0.85	0.86
Inefficiency	0.18	0.10	0.22	0.39	0.27	0.28	0.21	0.23	0.19
Average of Switches (kbps)	642.62	764.61	328.63	821.75	772.17	596.77	647.86	607.66	521.73
QoE	1085.79	1208.49	1073.87	1046.63	735.06	1166.78	1047.33	992.39	930.74

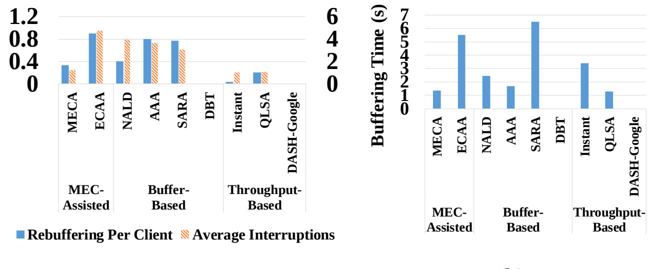
TABLE 6
PERFORMANCE OF ALGORITHMS WHEN BUFFER SIZE IS 15 S AND SEGMENT DURATION IS 4 S

	MEC-Assisted		Buffer-Based				Throughput-Based		
Adaptation Algorithms	MECA	ECAA	NALD	AAA	SARA	DBT	Instant	QLSA	DASH-Google
Average Video Rate (kbps)	1314.88	1430.56	1365.01	1204.11	1290.85	1322.31	1266.12	1183.25	1277.40
Switching Ratio	0.37	0.50	0.24	0.17	0.91	0.14	0.64	0.33	0.59
Fairness	0.88	0.88	0.87	0.83	0.76	0.89	0.85	0.86	0.86
Inefficiency	0.17	0.07	0.17	0.34	0.36	0.21	0.22	0.27	0.24
Average of Switches (kbps)	655.15	505.95	896.33	847.18	1153.13	562.78	640.26	585.93	652.79
QoE	1021.64	631.15	1052.90	897.49	29.03	1211.63	880.25	959.79	813.00



(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 is set to 15 s and the segment duration is set to 2 s.



(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 is set to 15 s and the segment duration is set to 4 s.

of competing clients, client moving speeds, client arrival times, and different video datasets on the performance of the algorithms. A grid based road topology is used to simulate mobility. The experiments are repeated ten 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 [54]. In this work, videos were streamed for 10 minutes for each experiment to evaluate the algorithms.

5.1 Effect of Segment Duration

The video streaming services deploy the segment duration differently in their services. In this section, we describe the effect of varying the segment duration on the performance of the algorithms. Table 4 presents the client and server settings used to evaluate the impact of segment duration. Ten clients competed for bandwidth, and dataset 1 was used to evaluate the algorithms in this section. The client arrival time was uniformly distributed within the first 30 s of the streaming session. Tables 5 and 6 display the performance of the algorithms when the buffer size was set to 15 s and the segment duration was set to 2 and 4 s, respectively. We observed that for both experiments, the MEC-assisted algorithms selected higher video rates while fairly assigning video rates among the users. Similarly, they utilized the bandwidth more efficiently than the client-based algorithms. However, Figs. 5 and 6 indicate that the ECAA algorithm failed to mitigate playback

TABLE 7
PERFORMANCE OF ALGORITHMS WHEN BUFFER SIZE IS 60 S AND SEGMENT DURATION IS 4 S

	MEC-Assisted		Buffer-Based				Throughput-Based		
Adaptation Algorithms	MECA	ECAA	NALD	AAA	SARA	DBT	Instant	QLSA	DASH-Google
Average Video Rate (kbps)	1126.99	1357.90	1264.14	1255.02	1198.18	1055.77	1150.59	1080.90	1225.20
Switching Ratio	0.30	0.34	0.36	0.19	0.80	0.05	0.53	0.32	0.47
Fairness	0.88	0.88	0.87	0.83	0.75	0.84	0.88	0.84	0.89
Inefficiency	0.17	0.11	0.14	0.32	0.29	0.32	0.15	0.32	0.15
Average of Switches (kbps)	614.26	893.88	652.80	757.00	617.16	834.45	539.66	572.24	639.08
QoE	1018.96	1058.27	1052.76	1040.25	582.21	981.41	853.65	849.02	949.86

TABLE 8
PERFORMANCE OF ALGORITHMS WHEN BUFFER SIZE IS 60 S AND SEGMENT DURATION IS 10 S

	MEC-Assisted		Buffer-Based				Throughput-Based		
Adaptation Algorithms	MECA	ECAA	NALD	AAA	SARA	DBT	Instant	QLSA	DASH-Google
Average Video Rate (kbps)	1192.09	1320.40	1161.81	1189.55	1348.74	1139.06	1176.56	1143.38	1169.62
Switching Ratio	0.36	0.54	0.55	0.44	0.42	0.17	0.73	0.60	0.65
Fairness	0.88	0.88	0.83	0.82	0.78	0.82	0.85	0.85	0.85
Inefficiency	0.18	0.17	0.16	0.23	0.38	0.23	0.16	0.22	0.18
Average of Switches (kbps)	702.27	798.68	700.84	696.09	1073.68	982.64	728.45	601.53	693.96
QoE	966.91	899.69	764.02	876.93	-145.51	882.41	598.66	801.57	719.20

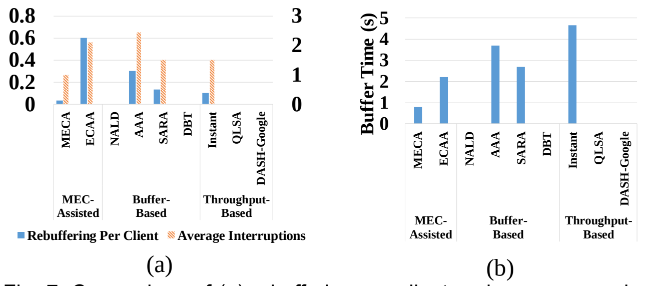


Fig. 7. Comparison of (a) rebuffering per client and average number of interruptions, and (b) average buffering time of the algorithms for a buffer size of 60 s and segment duration of 4 s.

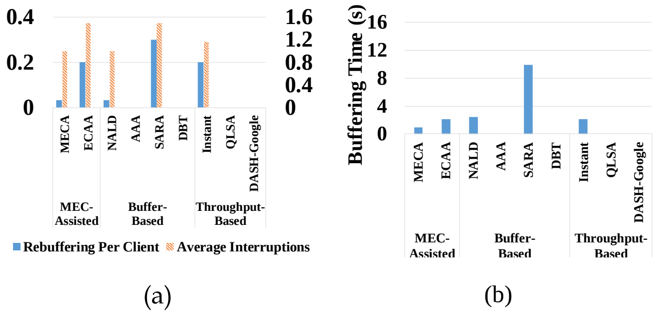
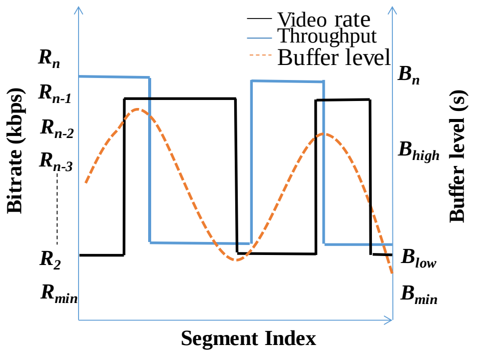


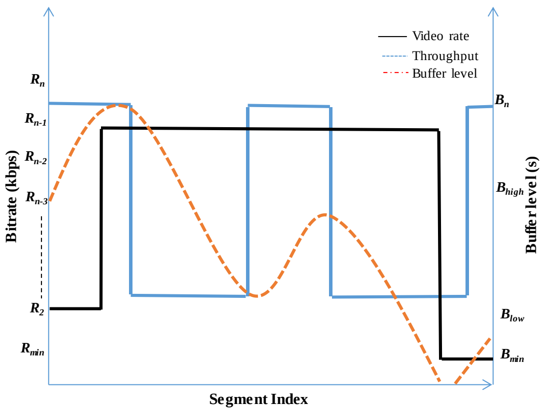
Fig. 8. Comparison of (a) rebuffering per client and average number of interruptions, and (b) average buffering time of the algorithms for a buffer size of 60 s and segment duration of 10 s.

interruptions. The rebuffering per client metric represents the ratio of clients that experienced a playback interruption and the total number of clients, while average interruptions defines the number of times a client experienced playback interruptions. The ECAA algorithm selects the video bit rate for the upcoming segment such that given the current bandwidth, the amount of data consumed during the download of the segment is less than the amount of data available in the playback buffer. However, due to the clients' mobility, which leads to fluctuations in the bandwidth during the download of the upcoming segment, the clients experience buffer underflow. Tables 5 and 6 reveal that when the segment duration is set to 2 s, the ECAA algorithm achieved the highest QoE. However, when the segment duration was increased, the QoE of the ECAA algorithm decreased significantly due to playback interruptions. From Figs. 5 and 6, we see that the playback interruptions increased with the increase in the segment duration. The buffer-based algorithms experienced higher number of buffer underflow events compared to throughput-based algorithms. It should be noted that the ECAA is an MEC-assisted algorithm that selects video rates based on the playback buffer. The buffer-based algorithms aggressively increased the video rate as the playback buffer became full. Because the playback buffer was only 15 s, a larger segment duration increased the risk of playback interruption in case of a mismatch between the selected video rate and available bandwidth. The DBT and AAA algorithms experienced low number of video rate switches. The algorithms avoided switching the video rate unless the buffer level increased above or decreased below predefined thresholds irrespective of fluctuations in bandwidth. This helped the algorithms mitigate the video rate fluctuations; however, it led to inefficient utilization of the bandwidth. Except for the SARA algorithm, buffer-based algorithms were able to mitigate unnecessary video rate switches more effectively than throughput-based algorithms. The SARA algorithm experienced the highest number of fluctuations because it aggressively increased the video rate as the buffer level increased. If the selected video rate was higher than the available bandwidth, the buffer began to deplete. When the buffer level decreased below a predefined threshold, it reduced the video rate aggressively, and this cycle continued throughout the streaming session. The SARA algorithm had the lowest QoE due to the high frequency of video rate switches and playback interruptions. In addition, the throughput-based algorithms had lower QoE due to the larger number of video rate switches.

Next, we increased the buffer level to 60 s. We first set the segment duration to 4 s and then increased the duration to 10 s. The statistics of algorithms are presented in Tables 7 and 8. Tables 5–8 reveal that the MEC-assisted algorithms achieved high video rates while efficiently utilizing bandwidth and fairly electing video rates for competing clients. In contrast, the fairness and bandwidth efficiency of the client-side algorithms varied from experiment to experiment. The reason for this result is that the client-based algorithms lacked knowledge of competing clients and were dependent on the network conditions and available video rates. Table 7 shows that the ECAA algorithm had the highest QoE when the segment duration was set to 4 s. We observed that the buffer-based algorithms achieved higher QoE than throughput-based algorithms. We also observed that when the segment duration was increased to 10 s, the switching ratio of the algorithms increased. As clients move at vehicular speed, they experience rapidly fluctuating bandwidth during the download of video segments. This leads to a high number of video rate switches as the segment duration increases. Table 8 indicates that a larger number of switches decreased the QoE of the algorithms. Tables 5–8 demonstrate that the segment duration did not affect the average switching magnitude. When the segment duration was set to 4 s, the DBT and AAA algorithms experienced the lowest and second lowest number of video rate switches, respectively. When the segment duration was increased to 10 s, the switching ratio of the DBT and AAA algorithms increased 3.4 and 2.31 times, respectively. Figs. 5–8 illustrate that larger segment duration increased the number of playback interruptions. Because a client cannot adapt the video rate during the download of the segment, a mismatch between the bandwidth and selected video rate increases the risk of playback interruption while downloading a larger segment. Interestingly, the AAA algorithm experienced a high number of buffer underflow events when the segment duration was 4 s. However, when the segment duration was increased to 10 s, the number of video rate switches increased but it did not experience playback interruption. Fig. 9(a) depicts the reason for this increase in video rate switches and illustrates how the AAA algorithm avoids playback interruptions when the segment duration is increased to 10 s. Initially, the algorithm increases the video rate from R_2 to R_{n-1} when the buffer level increases above B_{high}. The AAA algorithm decreases the video rate down to R_2 only when the buffer level decreases below Blow. When the playback buffer fills again and increases above B_{high}, the algorithm increases the video rate again. In an unstable environment, if this cycle continues, it increases the number of video rate fluctuations; however, the algorithm can avoid playback interruptions. Fig. 9(b) illustrates why the AAA algorithm experienced a larger number of playback interruptions when smaller segment duration was selected.



(a) The buffer level fluctuates around the buffer threshold, which increases the number of video rate switches.



(b) The buffer level remains close to Blow but does not drop below B_{low}, which would increase the risk of playback interruption
Fig. 9. The challenges for buffer-based rate adaptation algorithms in selecting the buffer thresholds.

When the bandwidth decreased, the buffer level depleted but did not decrease below B_{low}. The AAA algorithm remained at the higher video rate. The video rate cannot be adjusted in the middle of the segment download, which leads to a buffer underflow event. This observation demonstrates the importance of dynamically adjustable buffer thresholds according to the network and client settings. Furthermore, the QoE of the SARA algorithm was affected by playback interruptions in addition to the switches.

The experiments conducted in this section demonstrate that the MEC-assisted algorithms downloaded high-quality segments, equitably distributed video rates among the clients, and efficiently utilized the bandwidth. The client-based algorithms were unable to consistently maintain high fairness and efficient utilization of the bandwidth. Results also demonstrate that the buffer-based algorithms experienced higher rebuffering events compared to throughput-based algorithms. Despite higher rebuffering events, the buffer-based algorithms achieved significant QoE than throughput-based algorithms. Results reveal that the number of playback interruptions and video rate switching ratio increased with the increase in the segment duration. Furthermore, it is suggested that the adjustable buffer thresholds on the basis of

TABLE 9
EXPERIMENT SETTINGS USED TO EVALUATE THE IMPACT OF BUFFER SIZE

Experiment #	Dataset	Buffer Size	Segment Duration	Mobility	Client's Arrival Pattern	No. of Clients
1	1	15s	4s	75 kmph	Random	10
2	1	60s	4s	75 kmph	Random	10
3	1	30s	4s	75 kmph	Random	10

TABLE 10
PERFORMANCE OF ALGORITHMS WHEN BUFFER SIZE IS 30 S AND SEGMENT DURATION IS 4 S

	MEC-Assisted		Buffer-Based				Throughput-Based		
Adaptation Algorithms	MECA	ECAA	NALD	AAA	SARA	DBT	Instant	QLSA	DASH-Google
Average Video Rate (kbps)	1243.71	1246.17	1195.98	1167.02	1322.62	1033.22	1114.94	1269.27	1285.14
Switching Ratio	0.29	0.49	0.41	0.14	0.80	0.13	0.56	0.38	0.44
Fairness	0.88	0.86	0.85	0.82	0.67	0.82	0.87	0.86	0.89
Inefficiency	0.15	0.34	0.17	0.29	0.41	0.33	0.18	0.23	0.19
Average of Switches (kbps)	596.77	684.23	612.22	497.24	988.57	668.30	531.72	551.19	534.21
QoE	1105.26	617.64	915.65	1061.94	-162.15	887.51	829.07	1030.42	1021.27

network conditions could improve the performance of the algorithms.

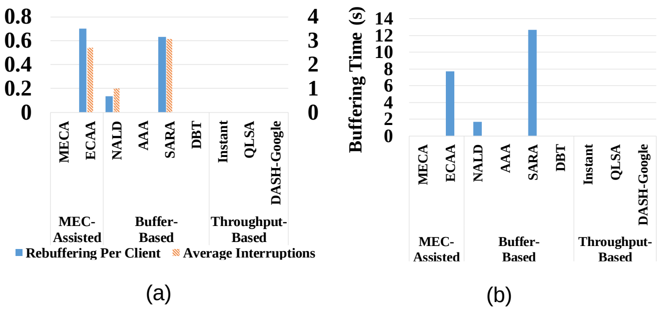


Fig. 10. Comparison of (a) rebuffering per client, average number of interruptions, and (b) average buffering time of the algorithms for a buffer size of 30 s and segment duration of 4 s.

5.2 Effect of Buffer Size

In this section, we analyze the performance of the algorithms as the playback buffer size of the clients was varied. Table 9 shows the experiment settings used to evaluate the impact of buffer size. Tables 6, 7, and 10 present the algorithm statistics as the buffer size was set to 15, 60, and 30 s respectively. In all of these experiments, the segment duration was set to 4 s, 10 clients competed for bandwidth, and the clients moved at vehicular speed within the cell. The clients' arrival time was uniformly distributed within the first 30 s. We use database 1 to analyze the algorithms in this section.

Tables 6, 7, and 10 indicate that the average video rate achieved by the throughput-based algorithms decreased with an increase in the buffer level. The reason for this is that when the buffer level is small, the clients quickly enter the ON-OFF phase. During the OFF state, the client waits for sufficient space in the buffer to download the next segment. During this period, a smaller number of clients compete for the bandwidth. Therefore, the clients in the ON state observe higher throughput. However, the buffer-based algorithms did not exhibit this pattern because they selected video rates on the basis of the buffer level in addition to the throughput. If the buffer level is within the pre-defined buffer thresholds, the throughput fluctuations alone do not result in changes to the video rate. Tables 6, 7, and 10 also reveal that the throughput-based algorithms experienced a larger number of video rate switches when the buffer size was set to 15 s. As in the previous experiments, the MEC-assisted algorithms achieved high fairness values and efficiently utilized the bandwidth irrespective of the buffer size. We observed that the fairness and bandwidth efficiency values of the client-based algorithms fluctuated from one experiment to the other.

Figs. 6, 7, and 10 illustrate that as the buffer size increased, the number of playback interruptions decreased. The buffer-based algorithms experienced a larger number of interruptions than the throughput-based algorithms. The DBT algorithm experienced the lowest number of video rate switches. This helped the algorithm achieve the highest QoE when the buffer size was set to 15 s. However, when the buffer size was increased, the QoE dropped as the algorithm downloaded low quality segments. The ECAA algorithm had a low QoE for smaller buffer sizes due to playback interruptions; however, the number of playback interruptions decreased with an increase in buffer size. The ECAA achieved the highest QoE when the segment duration was increased to 60 s. Similarly, the SARA's QoE improved when then buffer size was increased to 60 s as the rebuffering events decreased. In contrast, the QoE of the Instant algorithm increased with a decrease in buffer size due to a higher video rate. The AAA algorithm achieved higher video rates with an increase in buffer size; however, the QOE decreased as the number of switches increased.

Results of this section reveal that the MEC-assisted algorithms achieved high fairness value and low

TABLE 11
EXPERIMENT SETTINGS USED TO EVALUATE THE IMPACT OF NUMBER OF CLIENTS

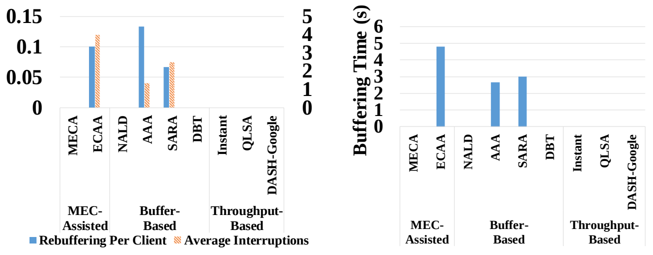
Experiment #	Dataset	Buffer Size	Segment Duration	Mobility	Client's Arrival Pattern	No. of Clients
1	1	15s	4s	75 kmph	Simultaneous	10
2	1	15s	4s	75 kmph	Simultaneous	8
3	1	15s	4s	75 kmph	Simultaneous	6
4	1	15s	4s	75 kmph	Simultaneous	4

TABLE 12
PERFORMANCE OF ALGORITHMS WHEN 4 CLIENTS COMPETE FOR THE BANDWIDTH

	MEC-Assisted		Buffer-Based				Throughput-Based		
Adaptation Algorithms	MECA	ECAA	NALD	AAA	SARA	DBT	Instant	QLSA	DASH-Google
Average Video Rate (kbps)	2718.56	3368.08	3329.27	3103.34	3357.80	2499.34	3061.16	2657.38	3005.37
Switching Ratio	0.39	0.29	0.26	0.11	0.95	0.16	0.55	0.22	0.56
Fairness	0.94	0.98	0.96	0.95	0.86	0.95	0.96	0.96	0.95
Inefficiency	0.18	0.14	0.12	0.24	0.28	0.29	0.15	0.23	0.19
Average of Switches (kbps)	979.89	1283.97	1241.50	1132.69	2478.26	798.64	1066.37	1001.73	1041.92
QoE	2167.95	2965.04	3006.67	2946.76	737.74	2319.43	2451.36	2139.38	2356.78

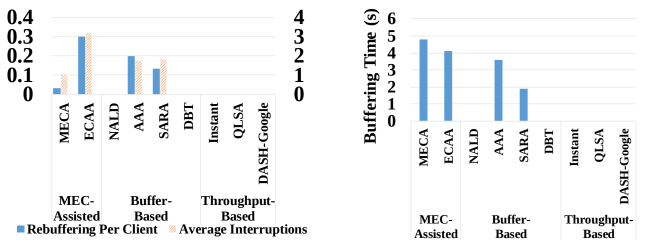
TABLE 13
PERFORMANCE OF ALGORITHMS WHEN 6 CLIENTS COMPETE FOR THE BANDWIDTH

	MEC-Assisted		Buffer-Based				Throughput-Based		
Adaptation Algorithms	MECA	ECAA	NALD	AAA	SARA	DBT	Instant	QLSA	DASH-Google
Average Video Rate (kbps)	1887.34	2100.06	2140.23	1866.32	2084.64	1764.51	1889.17	2059.79	1640.90
Switching Ratio	0.35	0.52	0.31	0.24	0.93	0.17	0.63	0.11	0.62
Fairness	0.96	0.96	0.95	0.90	0.86	0.97	0.91	0.93	0.88
Inefficiency	0.16	0.14	0.11	0.31	0.25	0.26	0.17	0.20	0.22
Average of Switches (kbps)	808.12	1013.22	954.84	872.37	1661.95	549.89	390.41	638.67	402.36
QoE	1736.24	1185.55	1865.94	1625.72	464.20	1636.49	1455.98	1794.57	1136.83



(a) (b)

Fig. 11. Comparison of (a) rebuffering per client, average number of interruptions, and (b) average buffering time of the algorithms when 4 compete for bandwidth. All clients start streaming simultaneously.



(a) (b)

Fig. 12. Comparison of (a) rebuffering per client, average number of interruptions, and (b) average buffering time of the algorithms when 6 compete for bandwidth. All clients start streaming simultaneously

inefficiency value in each experiment. The throughput-based algorithms downloaded lower quality segments with the increase in buffer size. However, the QoE of the throughput-based algorithms improved as the algorithms mitigated video rate switches and avoided the rebuffering event. The buffer-based algorithms achieved higher QoE than throughput-based algorithms despite experiencing more playback interruptions.

5.3 Effect of Number of Clients

In this section, we analyze the effect of the number of the competing clients on the performance of the algorithms. Table 11 shows the experiment settings used to evaluate the impact of buffer size. We set the buffer size and segment duration to 15 and 4 s, respectively. The clients moved inside the cell at vehicular speed. At the beginning of the streaming session, all clients started streaming simultaneously.

Tables 12–15 demonstrate that with a decrease in the number of streaming clients, the clients achieved higher video rates. Similarly, the fairness increased with a decrease in the number of competing clients. However, we did not observe the same effect on the switching ratio and inefficiency value. We also observed an increase in the average switching magnitude with a decrease in the number of competing clients. The reason for this increase

TABLE 14
PERFORMANCE OF ALGORITHMS WHEN 8 CLIENTS COMPETE FOR THE BANDWIDTH

	MEC-Assisted		Buffer-Based				Throughput-Based		
Adaptation Algorithms	MECA	ECAA	NALD	AAA	SARA	DBT	Instant	QLSA	DASH-Google
Average Video Rate (kbps)	1462.68	1474.29	1509.76	1329.95	1,507.98	1389.91	1202.61	1314.72	1413.81
Switching Ratio	0.31	0.43	0.31	0.26	0.48	0.14	0.61	0.37	0.46
Fairness	0.91	0.93	0.93	0.83	0.89	0.96	0.90	0.90	0.88
Inefficiency	0.18	0.12	0.19	0.39	0.20	0.26	0.25	0.32	0.22
Average of Switches (kbps)	601.44	708.20	872.46	787.07	993.45	449.89	595.87	613.41	578.39
QoE	1313.27	949.46	1031.12	978.56	559.34	1298.75	699.11	1008.19	1034.51

TABLE 15
PERFORMANCE OF ALGORITHMS WHEN 10 CLIENTS COMPETE FOR THE BANDWIDTH

	MEC-Assisted		Buffer-Based				Throughput-Based		
Adaptation Algorithms	MECA	ECAA	NALD	AAA	SARA	DBT	Instant	QLSA	DASH-Google
Average Video Rate (kbps)	1209.78	1182.34	1216.31	1109.63	1010.12	1035.37	1051.05	1096.60	1006.44
Switching Ratio	0.34	0.42	0.22	0.20	0.89	0.15	0.55	0.31	0.32
Fairness	0.89	0.90	0.88	0.85	0.81	0.91	0.86	0.88	0.89
Inefficiency	0.15	0.10	0.19	0.33	0.24	0.27	0.19	0.27	0.33
Average of Switches (kbps)	581.19	603.44	830.10	560.87	848.45	434.77	562.37	524.54	549.77
QoE	979.66	756.22	814.78	823.81	-71.77	948.06	790.30	894.41	811.91

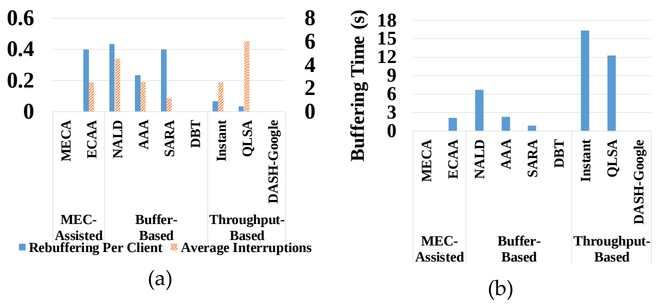


Fig. 13. Comparison of (a) rebuffering per client, average number of interruptions, and (b) average buffering time of the algorithms when 8 clients compete for bandwidth. All clients start streaming simultaneously.

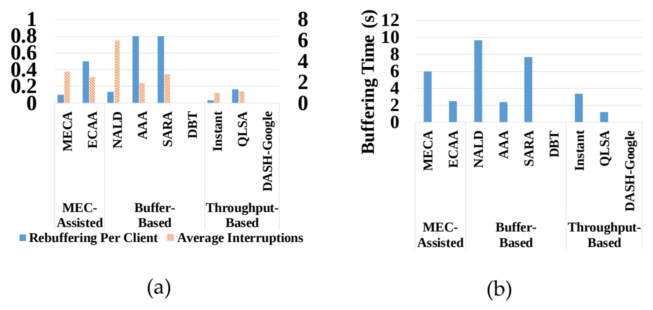


Fig. 14. Comparison of (a) rebuffering per client, average number of interruptions, and (b) average buffering time of the algorithms when 10 clients compete for bandwidth. All clients start streaming simultaneously.

is that the clients were able to aggressively increase the video rates as higher bandwidth became available to each client, and a higher video rate increases the risk of playback interruption in case the bandwidth suddenly decreases. This led the algorithm to aggressively decrease the video rate as the bandwidth decreased. The MEC-assisted algorithms were able to maintain high fairness and low bandwidth inefficiency values for all experiments. In case of 10 clients competing for the bandwidth, we observe that the MECA and NALD algorithms achieved similar video rates; however, the MECA algorithm had a higher QoE owing to lower playback interruptions. Tables 14 and 15 reveal that the MECA algorithm achieved the highest QoE as the algorithm achieved high video rate and avoided unnecessary playback interruptions. Table 12 illustrates that the NALD algorithm achieved a higher QoE as it achieved high video rate and mitigated playback interruptions. Furthermore, Table 12 also reveals that when the numbers of clients are decreased to 4, the buffer-based algorithms achieve higher video rate than the throughput-based algorithms. The throughput-based algorithms are unaware of the playback buffer; therefore, they conservatively select the video rates in order to avoid playback interruption. Whereas, the buffer-based algorithms aggressively increase the video quality as the buffer level increases. When the number of clients is decreased, the buffer fills up quickly. This allows the algorithms to quickly increase the video quality. Figs. 11–14 reveal that the clients experienced a larger number of playback interruptions as the number of competing clients increased, and the buffer-based algorithms experienced a larger number of interruptions than the throughput-based algorithms. The buffer-based algorithms select video rates higher than the throughput at the expense of the playback buffer, which leads to buffer underflow in an unstable environment. Because the highest available video rate in dataset 1 was 11,321 kbps, which was significantly higher than the throughput available to each client, selecting the higher video rates resulted in playback interruptions. A tradeoff can be

TABLE 16
EXPERIMENT SETTINGS TO EVALUATE THE IMPACT OF CLIENTS' ARRIVAL TIME

Experiment	Dataset	Buffer Size	Segment Duration	Mobility	Client's Arrival Pattern	No. of Clients
1	1	15s	4s	75 kmph	Simultaneous	10
2	1	15s	4s	75 kmph	Uniformly distributed	10
3	1	15s	4s	75 kmph	Exponentially distributed	10

TABLE 17
PERFORMANCE OF ALGORITHMS WHEN CLIENTS' ARRIVAL TIME IS EXPONENTIALLY DISTRIBUTED

	MEC-Assisted		Buffer-Based				Throughput-Based		
Adaptation Algorithms	MECA	ECAA	NALD	AAA	SARA	DBT	Instant	QLSA	DASH-Google
Average Video Rate (kbps)	1299.11	1358.31	1221.56	1255.22	1310.26	1390.50	1188.73	1099.33	1303.09
Switching Ratio	0.31	0.45	0.21	0.20	0.89	0.10	0.59	0.35	0.61
Fairness	0.89	0.90	0.87	0.82	0.83	0.90	0.82	0.84	0.85
Inefficiency	0.13	0.10	0.22	0.37	0.35	0.18	0.21	0.26	0.24
Average of Switches (kbps)	612.21	558.83	721.77	696.87	534.83	640.36	733.54	576.12	511.03
QoE	1099.34	833.18	921.55	917.41	248.59	1284.70	849.82	917.76	843.17

observed between selecting a higher video rate and protecting the buffer from emptying. The ECAA algorithm selected high video rates in each experiment; however, higher video rate switches and playback interruptions affected the QoE.

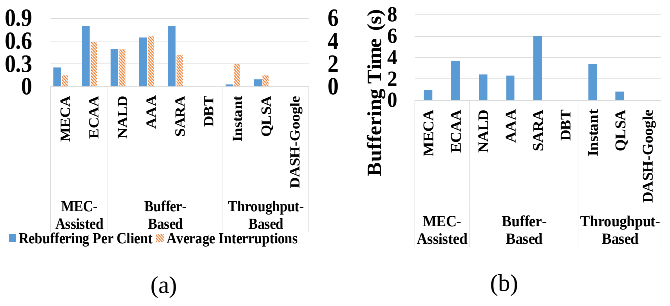


Fig. 15. Comparison of (a) rebuffering per client, average number of interruptions, and (b) average buffering time of the algorithms when clients' arrival time is exponentially distributed

The experiments conducted in this section demonstrate that the clients achieved higher video rates and fairly distributed video rates as the number of competing clients decreased. Similarly, the switching magnitude also increased with the decrease in number of clients. However, the changes in the number of competing clients did not have the same effect on switching ratio and bandwidth inefficiency. Results highlight that the buffer-based algorithms download higher quality segment when the throughput improves compared with the throughput-based algorithms. Furthermore, a compromise between selecting higher video rates and avoiding playback interruption is highlighted.

5.4 Effect of Clients' 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, and (3) the client arrival time was exponentially distributed within the first 30 s of the streaming session. Table 16 shows the experiment settings used to evaluate the impact of clients' arrival time. We set the buffer size and segment duration to 15 and 4 s, respectively. The clients moved inside the cell at a speed of 75 km/h.

Table 6 displays the performance of the algorithms when clients' arrival time is uniformly distributed, Table 15 displays the performance when the clients started the streaming session simultaneously, while Table 17 displays the performance of the algorithms when clients' arrival time is exponentially distributed. Tables 6, 15, and 17 indicate that the algorithms achieved slightly lower video rates and fairness when the clients joined the streaming session simultaneously. The reason for this result is that when all clients stream at the same time, there is a tug-of-war between greedy clients to obtain the bandwidth share from the beginning. However, the algorithms experienced lesser number of playback interruptions when the clients joined the streaming session simultaneously. The MEC-assisted algorithms were able to fairly assign video rates to the clients and efficiently utilizing the available bandwidth in both scenarios. The DBT algorithm has the highest fairness value in both scenarios. Among the client-based algorithms, the DBT and AAA algorithms avoided unnecessary video rate switches, whereas the SARA algorithm had the highest frequency of switches. Furthermore, the QLSA algorithm had a high bandwidth inefficiency value. The QLSA algorithm focused on minimizing the sharp decrease in video quality and minimizing the frequency of video rate switches. However, in a multi-client environment, this resulted in inefficient utilization of the bandwidth. Figs. 6, 14, and 15 depict the statistics of playback interruption events for both scenarios. The figures indicate that only the DASH-

TABLE 18
EXPERIMENT SETTINGS USED TO EVALUATE THE IMPACT OF CLIENTS' SPEED

Experiment	Dataset	Buffer Size	Segment Duration	Mobility	Client's Arrival Pattern	No. of Clients
1	1	15s	4s	75 kmph	Random	10
2	1	15s	4s	3 kmph	Random	10
3	1	15s	4s	60 kmph	Random	10
4	1	60s	4s	75 kmph	Random	10
5	1	60s	4s	3 kmph	Random	10
6	1	60s	4s	60 kmph	Random	10

TABLE 19
PERFORMANCE OF ALGORITHMS WHEN CLIENTS ARE MOVING AT 60KMPH. THE BUFFER SIZE IS SET TO 15 S

	MEC-Assisted		Buffer-Based				Throughput-Based		
Adaptation Algorithms	MECA	ECAA	NALD	AAA	SARA	DBT	Instant	QLSA	DASH-Google
Average Video Rate (kbps)	1467.44	1402.32	1311.08	1291.11	1333.75	1280.50	1381.23	1123.19	1322.87
Switching Ratio	0.29	0.45	0.28	0.15	0.92	0.10	0.61	0.35	0.62
Fairness	0.89	0.88	0.87	0.84	0.78	0.90	0.87	0.85	0.84
Inefficiency	0.17	0.17	0.16	0.29	0.36	0.18	0.20	0.26	0.24
Average of Switches (kbps)	721.04	822.79	951.14	774.37	933.20	640.36	768.12	753.13	621.86
QoE	1211.84	688.43	1022.03	977.34	110.95	1194.70	1027.88	940.73	854.55

TABLE 20
PERFORMANCE OF ALGORITHMS WHEN CLIENTS ARE MOVING AT PEDESTRIAN SPEED. THE BUFFER SIZE IS SET TO 15 S

	MEC-Assisted		Buffer-Based				Throughput-Based		
Adaptation Algorithms	MECA	ECAA	NALD	AAA	SARA	DBT	Instant	QLSA	DASH-Google
Average Video Rate (kbps)	1546.01	1572.16	1615.81	1456.35	1608.45	1390.50	1594.81	1212.33	1400.71
Switching Ratio	0.18	0.38	0.11	0.12	0.95	0.12	0.15	0.40	0.48
Fairness	0.90	0.89	0.88	0.88	0.79	0.89	0.92	0.84	0.85
Inefficiency	0.16	0.10	0.15	0.25	0.35	0.20	0.13	0.26	0.26
Average of Switches (kbps)	950.22	695.20	1054.45	696.87	2277.83	640.36	729.54	616.22	522.94
QoE	1380.69	722.03	1478.52	1300.41	9.59	1284.70	1416.82	941.18	1029.25

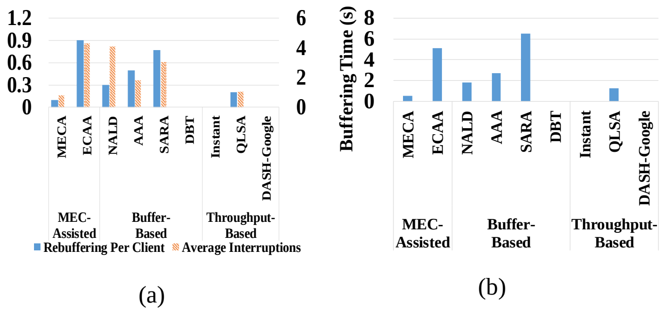


Fig. 16. Comparison of (a) rebuffering per client, average number of interruptions, and (b) average buffering time when the clients move at 60kmph. The buffer size for this experiment is set to 15s.

Google and DBT algorithms were able to avoid playback interruption during both experiments. We observed that the buffer-based algorithms experienced a larger number of playback interruptions than the throughput-based algorithms. Except, for the ECAA, the adaptation methods achieve higher QoE when the clients randomly join the streaming session. The ECAA algorithm achieves low QoE scores due to larger number of playback interruptions. DASH-Google has a similar QoE score in both scenarios.

Results of this section reveal that the client's arrival time affects the performance of the algorithms. It is shown that the algorithms achieved significant video rate, fairly distributed video rates, and experienced higher number of rebuffering events when the clients joined the streaming session randomly. Except for the ECAA algorithm, the higher video rate leads to higher QoE when the clients joined the streaming session randomly.

5.5 Effect of Clients' Speed

In this section, we compared the performance of the algorithms for the following scenarios: (1) the clients moved at 75km/h, (2) the clients moved at 60 km/h, (3) the clients moved at pedestrian speed (3km/h). Table 18 shows the experiment settings used to evaluate the impact of clients' speed.

In this first experiment, the buffer size and segment duration were set to 15 and 4 s, respectively, and 10 clients competed for bandwidth. Tables 6, 19 and 20 present the performance of the algorithms as the clients moved at 75km/h, 60 km/h, pedestrian speed respectively. We observed that the clients achieved higher video rate,

TABLE 21
PERFORMANCE OF ALGORITHMS WHEN CLIENTS ARE MOVING AT 60KMPH. THE BUFFER SIZE IS SET TO 60 S

	MEC-Assisted		Buffer-Based				Throughput-Based		
Adaptation Algorithms	MECA	ECAA	NALD	AAA	SARA	DBT	Instant	QLSA	DASH-Google
Average Video Rate (kbps)	1297.08	1409.63	1252.08	1185.87	1355.14	1177.35	1144.99	1097.45	1251.80
Switching Ratio	0.27	0.38	0.29	.21	0.77	0.07	0.36	0.35	0.41
Fairness	0.89	0.88	0.84	.85	0.80	0.86	0.88	0.87	0.87
Inefficiency	0.16	0.12	0.14	0.33	0.27	0.31	0.13	0.25	0.16
Average of Switches (kbps)	662.74	772.19	576.04	697.57	805.22	571.32	687.61	722.11	587.41
QoE	1167.46	1063.41	1096.91	1067.44	723.67	1099.23	923.07	855.98	1008.38

TABLE 22
PERFORMANCE OF ALGORITHMS WHEN CLIENTS ARE MOVING AT PEDESTRIAN SPEED. THE BUFFER SIZE IS SET TO 60 S

	MEC-Assisted		Buffer-Based				Throughput-Based		
Adaptation Algorithms	MECA	ECAA	NALD	AAA	SARA	DBT	Instant	QLSA	DASH-Google
Average Video Rate (kbps)	1320.58	1376.94	1309.48	1199.69	1436.68	1141.76	1305.49	1179.04	1283.11
Switching Ratio	0.15	0.40	0.28	0.12	0.62	0.05	0.25	0.32	0.45
Fairness	0.90	0.90	0.87	0.86	0.83	0.84	0.89	0.88	0.86
Inefficiency	0.14	0.15	0.14	0.28	0.24	0.28	0.13	0.19	0.16
Average of Switches (kbps)	722.23	736.00	683.11	723.66	678.56	705.00	814.27	577.99	180.60
QoE	1236.59	861.81	1131.94	1094.38	1048.70	1076.83	1169.66	1005.91	1012.80

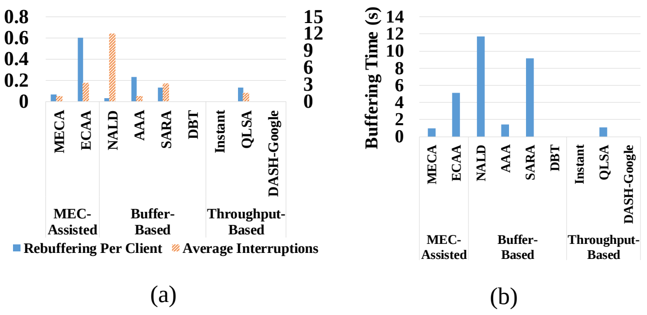


Fig. 17. Comparison of (a) rebuffering per client, average number of interruptions, and (b) average buffering time when the clients move at pedestrian speed. The buffer size for this experiment is set to 15s.

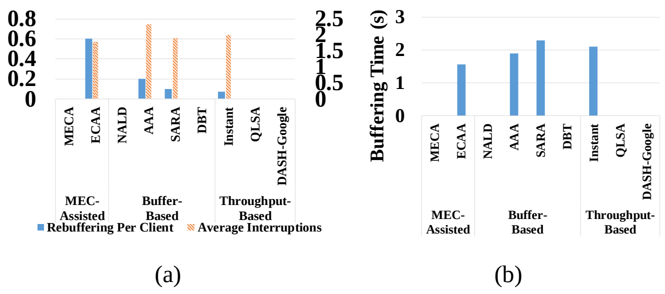


Fig. 18. Comparison of (a) rebuffering per client, average number of interruptions, and (b) average buffering time when the clients move at 60kmph. The buffer size for this experiment is set to 60s.

fewer playback interruptions and a fair distribution of video rates when moving at pedestrian speed. Consequently, the algorithms achieved higher QoE when moving at pedestrian speed. Tables 6, 19 and 20 indicate that the MEC-assisted algorithms achieved higher video rates and were able to efficiently and fairly assign video rates among the clients. All selected algorithms except for SARA and QLSA achieved a smaller number of video rate switches when moving at pedestrian speed than at vehicular speeds. The number of playback interruptions decreased with the decrease in the clients speed. The buffer-based algorithms experienced higher number of playback interruptions compared to throughput-based algorithms.

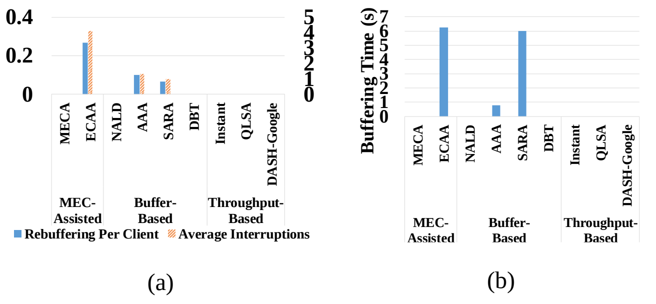


Fig. 19. Comparison of (a) rebuffering per client, average number of interruptions, and (b) average buffering time when the clients move at pedestrian speed. The buffer size for this experiment is set to 60s.

In the next experiment, we set the buffer size and segment duration to 60 and 4 s, respectively. Similar to the last experiment, Tables 21 and 22 show that the MEC-assisted algorithms were also able to achieve high video rates by efficiently utilizing bandwidth and equitably assigning video rates to the clients. A comparison of Tables 7, 21 and 22 reveals that the QoE of throughput-based algorithms improved with the decrease in the clients' speed. Except for the ECAA algorithm, the QoE of buffer-based algorithms also improved with the decrease in the clients' speed. A comparison of Figs. 7, 18, and

TABLE 23
EXPERIMENT SETTINGS USED TO EVALUATE THE IMPACT OF DATASETS

Experiment #	Dataset	Buffer Size	Segment Duration	Mobility	Client's Arrival Pattern	No. of Clients
1	1	15s	2s	75 kmph	Random	10
2	1	15s	4s	75 kmph	Random	10
3	1	60s	4s	75 kmph	Random	10
4	2	15s	2s	75 kmph	Random	10
5	2	15s	4s	75 kmph	Random	10
6	2	60s	4s	75 kmph	Random	10

TABLE 24
PERFORMANCE OF ADAPTIVE METHODS WHEN BUFFER SIZE IS 15 S AND SEGMENT DURATION IS 2 S FOR DATASET 2

	MEC-Assisted		Buffer-Based				Throughput-Based		
Adaptation Algorithms	MECA	ECAA	NALD	AAA	SARA	DBT	Instant	QLSA	DASH-Google
Average Video Rate (kbps)	1342.48	1423.48	1400.19	1367.88	1297.27	1229.03	1389.13	1380.94	1394.02
Switching Ratio	0.32	0.46	0.43	0.13	0.73	0.13	0.48	0.42	0.48
Fairness	0.90	0.87	0.88	0.86	0.78	0.88	0.91	0.89	0.89
Inefficiency	0.14	0.15	0.20	0.26	0.38	0.25	0.15	0.19	0.14
Average of Switches (kbps)	474.45	462.72	636.93	642.90	931.73	427.44	384.42	494.58	384.51
QoE	1200.23	1304.51	1287.93	1291.51	646.17	1093.30	1189.92	1205.33	1135.70

19 shows that buffer-based algorithms experienced higher number of playback interruptions irrespective of the speed at which the client moved. The SARA algorithm experienced the highest number of interruptions at vehicular speed, whereas, the ECAA algorithm experienced the highest number of interruptions at the pedestrian speed. At the pedestrian speed, the ECAA algorithm also experienced the highest number of video rate switches. As in the previous experiments, the SARA algorithm achieved high video rates but also experienced the highest number of video rate switches. This algorithm also had a high bandwidth inefficiency value. The reason for this result is that sum of the downloaded video rates of the competing clients was higher than the total bandwidth available to the clients. This also led the SARA algorithm to experience buffer underflow, as depicted in Figs. 6, 7, 16, 17, 18, and 19. The DBT and AAA algorithms had high bandwidth inefficiency values. Both algorithms wait for the playback buffer to increase above the predefined threshold before they increase or decrease the video rates irrespective of the available bandwidth irrespective of the selected video rate and available bandwidth. This leads to inefficient utilization of the bandwidth.

Results obtained in this section demonstrate that the clients' speed inversely affect the performance of the algorithms. The clients achieved higher video rates and experienced lesser number of rebuffering when the clients moved at pedestrian speed compared to vehicular speed. Consequently, the clients achieved higher QoE at the pedestrian speed. Moreover, the bandwidth is more efficiently utilized and video rates are fairly distributed at pedestrian speed. Irrespective of the speed, the buffer-based algorithms experienced higher number of playback interruptions.

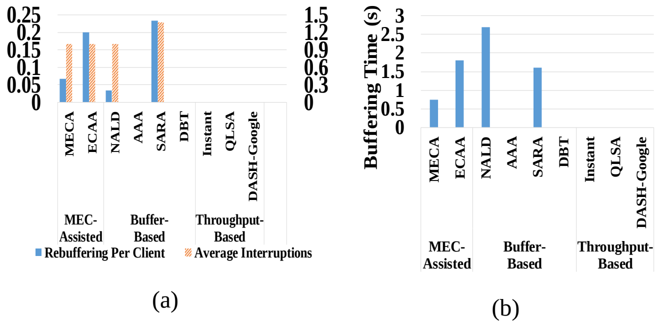


Fig. 20. Comparison of (a) rebuffering per client, average number of interruptions and (b) average buffering time when the clients download data set 2. The buffer size and segment duration are set to 15 and 2 s, respectively.

5.6 Effect of Datasets

In this section, we compare the performance of the algorithms for the two datasets presented in Table 2. Table 23 shows the experiment settings used to evaluate the impact of datasets. The aim of this comparison is to analyze how the selection of a representation set affects the performance of the algorithms. The adaptation algorithm has no control over the representation set; therefore, it should adapt the video rates dynamically such that it achieves high user experience irrespective of the dataset.

In the first experiment, we set the buffer size and segment duration to 15 and 2 s, respectively. Comparison of Tables 5 and 24 shows that algorithms achieved higher average video rate while downloading dataset 2. The video rates in dataset 2 were close to each other, which helped the algorithms improve the fairness and reduce bandwidth inefficiency. However, we observed that for dataset 2, all algorithms experienced a larger number of video rate switches. A comparison of Figs. 5 and 20 indic-

TABLE 25
PERFORMANCE OF ADAPTIVE METHODS WHEN BUFFER SIZE IS 15 S AND SEGMENT DURATION IS 4 S FOR DATASET 2

	MEC-Assisted		Buffer-Based				Throughput-Based		
Adaptation Algorithms	MECA	ECAA	NALD	AAA	SARA	DBT	Instant	QLSA	DASH-Google
Average Video Rate (kbps)	1355.38	1385.00	1302.65	1279.83	1316.02	1091.73	1273.18	1293.08	1380.85
Switching Ratio	0.47	0.34	0.23	0.24	0.46	0.19	0.67	0.47	0.68
Fairness	0.90	0.91	0.87	0.85	0.81	0.81	0.90	0.86	0.89
Inefficiency	0.13	0.12	0.14	0.29	0.26	0.30	0.16	0.22	0.15
Average of Switches (kbps)	460.24	534.53	589.60	482.70	950.53	507.93	448.73	614.87	489.85
QoE	1118.27	1191.73	994.68	977.27	423.93	966.48	1000.20	1001.08	951.10

TABLE 26
PERFORMANCE OF ADAPTIVE METHODS WHEN BUFFER SIZE IS 60 S AND SEGMENT DURATION IS 4 S FOR DATASET 2

	MEC-Assisted		Buffer-Based				Throughput-Based		
Adaptation Algorithms	MECA	ECAA	NALD	AAA	SARA	DBT	Instant	QLSA	DASH-Google
Average Video Rate (kbps)	1237.55	1256.41	1144.86	1109.00	1111.38	1043.21	1217.56	1272.98	1152.71
Switching Ratio	0.30	0.59	0.31	0.14	0.83	0.10	0.62	0.45	0.68
Fairness	0.91	0.88	0.87	0.87	0.72	0.77	0.89	0.87	0.90
Inefficiency	0.13	0.26	0.23	0.30	0.41	0.31	0.13	0.24	0.14
Average of Switches (kbps)	427.01	491.54	484.25	468.90	515.35	644.35	430.43	525.30	412.12
QoE	1135.13	701.76	1033.43	1075.37	709.88	953.00	977.08	1006.99	905.28

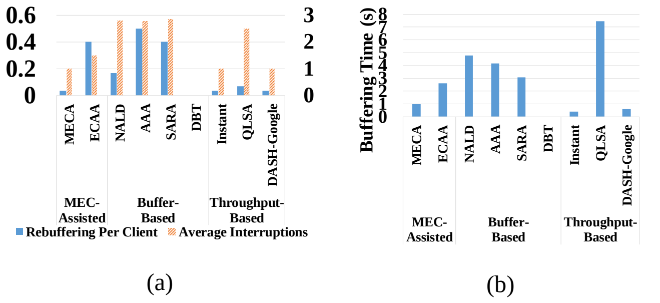


Fig. 21. Comparison of (a) rebuffering per client, average number of interruptions, and (b) average buffering time when the clients download dataset 2. The buffer size and segment duration are set to 15 and 4 s, respectively.

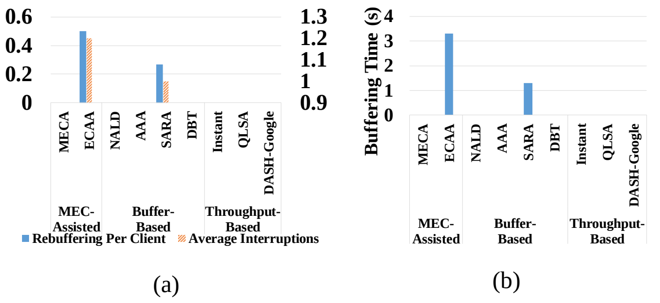


Fig. 22. Comparison of (a) rebuffering per client, average number of interruptions, and (b) average buffering time when the clients download dataset 2. The buffer size and segment duration are set to 60 s and 4 s, respectively.

ates that the algorithms experienced fewer playback interruptions when downloading dataset 2. The higher data rates in dataset 1 allowed the algorithms to select high video quality but also increased the playback interruptions.

In the next experiment, we increased the segment duration to 4 s. Table 25 reveals that the MEC-assisted algorithms efficiently and fairly selected video rates for the clients. A comparison of Tables 6 and 25 demonstrates that the algorithms utilized the bandwidth more efficiently and fairly when the clients downloaded dataset 2. However, the number of video rate switches increased when dataset 2 was used. Furthermore, a comparison of Figs. 6 and 21 reveals that the total number of playback interruptions decreased when dataset 2 was used. This is due to the availability of higher video rates in dataset 1 and lower video rates in dataset 2. The MEC-assisted algorithms had the highest QoE score. The ECAA algorithm had a low QoE score when it downloaded dataset 1. In case of dataset 2, the QoE score of the ECAA algorithm increased 1.88 times, as the algorithm experienced smaller number of rebuffering events.

Next, we increased the buffer size to 60 s. Similar to Section 5.2, Table 26 demonstrates that the average video rate decreased as the buffer size was increased. The MECA algorithm achieved high video rate, fairly distributed video rates and efficiently utilized the bandwidth. Consequently, the algorithm achieved the highest QoE. Fig. 22 reveals that a larger buffer size reduced the number of buffer underflow events. However, the ECAA algorithm experienced a larger number of video rate switches and playback fluctuations despite the larger buffer size. To determine the reason for the large number of playback interruptions despite the larger buffer size, we examined the buffer level of the clients. We discovered that the buffer level never increased above 30% for any client. The ECAA algorithm selects the video rates aggressively, which leads to a low buffer level. On one hand, this helps the algorithm achieve higher video quality. On the other hand, it increases the risk of playback interruption in an unstable environment. A low frequency of video rate

TABLE 27
PERFORMANCE OF ADAPTIVE METHODS AVERAGED OVER ALL EXPERIMENTS

	MEC-Assisted		Buffer-Based				Throughput-Based		
Adaptation Algorithms	MECA	ECAA	NALD	AAA	SARA	DBT	Instant	QLSA	DASH-Google
Average Video Rate (kbps)	1431.48	1536.86	1470.29	1385.24	1469.23	1306.32	1390.63	1333.45	1395.88
Switching Ratio	0.31	0.43	0.30	0.19	0.75	0.12	0.53	0.36	0.53
Fairness	0.90	0.90	0.88	0.85	0.79	0.88	0.88	0.87	0.88
Inefficiency	0.16	0.14	0.17	0.31	0.32	0.26	0.17	0.24	0.20
Average of Switches (kbps)	659.52	725.29	752.18	715.06	1059.28	626.78	629.39	622.19	548.50
QoE	1232.07	1037.55	1208.51	1176.42	415.81	1193.26	1068.24	1079.10	1030.25

TABLE 28
PERFORMANCE OF METHODS AVERAGED OVER ALL EXPERIMENTS WHERE 10 CLIENTS COMPETE FOR THE BANDWIDTH

	MEC-Assisted		Buffer-Based				Throughput-Based		
Adaptation Algorithms	MECA	ECAA	NALD	AAA	SARA	DBT	Instant	QLSA	DASH-Google
Average Video Rate (kbps)	1304.76	1370.30	1286.84	1232.10	1300.33	1182.41	1249.12	1188.34	1262.13
Switching Ratio	0.30	0.43	0.30	0.19	0.76	0.11	0.51	0.38	0.52
Fairness	0.89	0.89	0.87	0.84	0.78	0.86	0.88	0.86	0.87
Inefficiency	0.15	0.15	0.17	0.31	0.33	0.26	0.17	0.24	0.20
Average of Switches (kbps)	630.17	666.04	694.16	668.86	934.32	632.63	617.64	594.53	521.56
QoE	1123.41	895.59	1045.78	1032.01	349.17	1073.62	968.12	957.33	927.59

switches helped the AAA algorithm achieve high QoE. However, the algorithm utilized the bandwidth inefficiently. The DASH-Google, Instant, and QLSA algorithms had higher average video rates than the AAA algorithm, but they achieved lower QoE due to a larger number of video rate switches.

Experiments conducted in this section reveal that the available set of video rates play an important role in the performance of the algorithms. A dataset with high video rates can help the algorithms in improving user experience. Results reveal that it also resulted in higher number of playback interruptions in an unstable environment. Similarly, a dataset with closely spaced data rates allows the algorithms to improve fairness and efficiently utilize bandwidth. Furthermore, it is demonstrated that the closely spaced video rates resulted in higher number of video rate switches, which inversely affects the user experience.

6 KEY OBSERVATIONS

The experiments conducted in this study led to numerous insights regarding the compared algorithms.

- **Fairness and bandwidth utilization:** First, we observed that the MEC-assisted algorithms were able to achieve high fairness and efficiently utilize bandwidth in most settings. The MEC-assisted algorithms had a bird's-eye view of the network and were able to jointly assign video rates to the clients. This allowed them to efficiently select video rates. The performance of the client-side adaptation algorithms varied as the client and server settings changed.

- **Fixed control laws:** These results indicate that the client-side adaptation algorithms employ fixed control laws and were designed with certain client/server and network settings in mind. There is a need for content and context-aware adaptation algorithms that consider network settings, HAS clients' information, video content details and device features to adapt video quality.

- **Tradeoff between selecting high quality video and mitigating playback interruption:** The understanding of MEC-assisted ABR strategies is still in the early stages. The results in this work show tradeoffs between selecting high quality video and protecting playback buffer from draining. Table 27 presents the performance of the algorithms averaged over all experiments. The MEC-assisted algorithms selected high-quality video rates for clients. In improving the video quality, however, these algorithms were unable to protect the buffer from emptying. The hybrid algorithm (MECA) was more effective in protecting the playback buffer. The MECA algorithm on average had the highest QoE among the competing algorithms. The ECAA algorithm was unable to protect the playback buffer from emptying, which resulted in degraded performance in some experiments. Analysis of the ECAA algorithm reveals that it selected video rates higher than the available throughput as the buffer level filled. This approach helped the algorithm select high-quality video in a stable network; however, in an unstable network, this approach led to higher rebuffing events. The results indicate that a better coordination between the clients and edge cloud further improve the performance of MEC-assisted algorithms.

- **Tradeoff between minimizing video rate switches and mitigating playback interruption** Throughput-based algorithms are dependent on bandwidth in selecting the video rates. They do not have knowledge of the buffer level; therefore, they do not take the risk of conservatively reacting to changes in bandwidth. Interestingly, our results demonstrate that the buffer-based algorithms performed more poorly than throughput-based algorithms in protecting the buffer. However, the buffer-based algorithms performed better than the throughput-based algorithms in minimizing unnecessary video rate switches. Throughput-based algorithms take the average of past throughput observations to smooth out the throughput fluctuations. The throughput-based algorithms failed to mitigate the switching ratio, which degraded their QoE.

- **Buffer-based algorithms aggressively selecting high quality as buffer level grows**: Table 28 presents the performance of the algorithms' average over experiments where only 10 clients competed for the bandwidth. The experiment performed in Section 5.3 showed that the algorithms achieve higher QoE as the number of competing clients decrease. A comparison of Tables 27 and 28 illustrates that except the SARA algorithm, the buffer-based algorithms improve their QoE more than the throughput-based algorithms as the number of clients decreases. Once the buffer level increases above the buffer threshold, the buffer-based algorithms do not decrease the video rate even if the throughput fluctuates. This allows the buffer-based algorithms to maintain higher video rates and minimize video rate fluctuations.

- **Prioritizing video quality objectives:** Results also indicate that each algorithm prioritized different video quality objectives. This trend is observed in both MEC-assisted and client-based algorithms. In case of buffer-based algorithms, the playback buffer level plays an important role in prioritizing video quality objective. When the buffer level is low, the algorithms prioritize avoiding playback interruptions. As the buffer level fills up, the algorithms give precedence to improving the video quality. The selection of buffer thresholds dictates when the algorithms shift the focus between improving the video quality and mitigating playback interruption and video rate switches. The throughput-based algorithms have limited options as they do not observe buffer level. The algorithms minimize small-scale bandwidth fluctuations but react quickly in case of large and high bandwidth fluctuations. This helps the algorithms in improving the video quality when the bandwidth improves and mitigating the risk of playback interruptions when the bandwidth drops.

- **Segment duration:** Our results reveal that selecting a larger segment duration increases the risk of buffer underflow in case of a mismatch between the selected video rate and available throughput to the client. The results show that a smaller segment duration achieves a higher QoE in an unstable environment.

- **Set of available video rates:** The available set of video rates in the datasets also plays an important role in the performance of the algorithms. In this study, two datasets were used with three main differences. The highest available video rate was much higher in dataset 1 than in dataset 2, while the lowest available video rate was much lower in dataset 2 than in dataset 1. In addition, the video rates in dataset 2 were more closely spaced than in dataset 1. The results reveal that dataset 2 resulted in a higher frequency of video rate switches; however, it reduced the number of buffer underflow events and improved fairness and bandwidth efficiency.

- **Tradeoff between fairness and bandwidth utilization:** In a wired network, owing to TCP fairness, each client deserves an equal share of the available bandwidth. In a cellular network, TCP achieves unfair throughput due to a poor interaction between TCP congestion control and PF scheduling. In addition, the throughput observed by video clients depends on multiple factors, including propagation distance, fading, interference, and user mobility. For example, a user close to the base station observes higher throughout than a user at the edge of the cell. Selecting a low video rate for the client closer to the base station to improve fairness increases bandwidth inefficiency. Similarly, selecting a video rate higher than the available bandwidth for the user at the edge of the circle leads to buffer underflow.

7 Suggestions For Future Algorithms Designs

- **Adaptable buffer thresholds:** We observed that buffer-based algorithms selected video rates higher than the available throughput once the buffer filled and increased above predefined buffer thresholds, and decreased the video rates once the buffer level depleted and decreased below the predefined buffer threshold. In a rapidly fluctuating environment, we observed that the buffer-based algorithms were also unable to protect the buffer from depleting. Currently, the algorithms employ fixed buffer thresholds. The results suggest that adaptable buffer thresholds on the basis of network conditions and client settings can provide a solution.

  For instance, the video client can buffer a larger number of smaller segments, compared to larger segments, as shown in Fig. 23. Let us assume the current buffer size is twenty seconds, and the segment duration (τ) is two seconds. This means the total number of segments the buffer can hold is ten. If the segment duration is four seconds, then only five segments can fit in the buffer. If throughput is less than the video rate of the segment being downloaded, there is a greater risk of buffer underflow when downloading larger segments. As the buffer-based algorithms increase or decrease video quality based on buffer-thresholds, the algorithms should consider

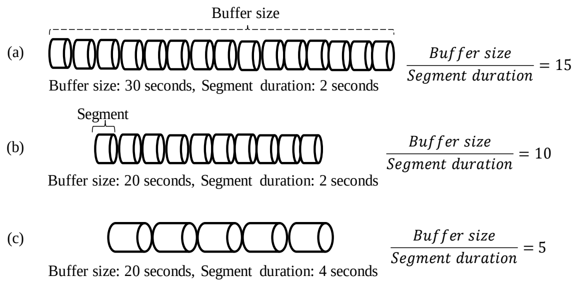

Buffer size
(a)
$\frac{Buffer\ size}{Segment\ duration} = 15$
Buffer size: 30 seconds, Segment duration: 2 seconds
Segment
(b)
$\frac{Buffer\ size}{Segment\ duration} = 10$
Buffer size: 20 seconds, Segment duration: 2 seconds
(c)
$\frac{Buffer\ size}{Segment\ duration} = 5$
Buffer size: 20 seconds, Segment duration: 4 seconds

Fig. 23. Comparing (a) and (b) shows that the larger buffer size allows the client to buffer more segments. Comparing (b) and (c) shows that longer segment durations decrease the number of segments that can be buffered.

adapting the buffer thresholds as the segment duration, client's playback buffer size and network conditions vary. This would lead to a more consistent performance by algorithms.

- **Selection of segment duration:** The results show that a smaller segment duration achieves a higher QoE in an unstable environment. Therefore, it is desirable to fragment the playback video into smaller duration segments in an unstable environment.

  The advantage of a larger segment duration is that the client does not react to small changes in throughput. A larger segment duration signifies a smaller number of segments in the playback buffer. Although the switching ratio increases with an increase in the segment duration, the total number of video rate switches decreases as the total number of downloaded segments decreases. Therefore, a larger segment duration is recommended for a stable network.

- **Selection of video rates:** In a stable environment, a dataset with the properties of dataset 1 are preferable. Whereas, in an unstable network, a dataset with the properties of dataset 2 would lead to better QoE. These dataset properties can also help both throughput- and buffer-based algorithms in designing algorithms. For instance, in case of dataset with closely spaced video rates, the algorithm can restrict the changes in the video rate selection unless the client observes large variations in the throughput. In our previous work [55], we designed a method to distinguish between large and small-scale fluctuations in the throughput. The algorithms can utilize such methods to conservatively or aggressively select the video rate based on the properties of datasets.

- **Adaptive algorithm control laws:** The existing HAS algorithms strive to keep the buffer filled to a predefined threshold to minimize the risk of playback interruption. 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.

- **Context-aware adaptation:** In addition to the information utilized by the algorithms, some context data, such as the clients' position, trajectory, and location inside a cell with poor coverage, can help the algorithms improve their decision-making. The client's location inside the cell can be acquired through usual built-in sensors. If a client receiving high bandwidth enters a location with poor coverage, this information can help the algorithms adapt their decision-making accordingly. In this scenario, the algorithm can focus on filling the buffer in advance to reduce the risk of buffer underflow events as the client enters a poor coverage area.

8 Challenges and Potential Research Directions

In this section, the cost, complexity, and future research challenges and directions are discussed for MEC-assisted video streaming.

(1) Energy Consumption

MEC offers computation and storage capabilities to the edge of a mobile network. This allows edge cloud to respond to video client requests and it can reduce wastage of returning resources. However, computing and communication between MEC servers and users can lead to significant energy consumption. To this end, it is important to establish energy efficient resource consumption in order to schedule computing and communication resources. Therefore, joint optimization of QoE and energy consumption is great challenge for future.

(2) Backhaul traffic and QoE tradeoff

The objective of dynamic adaptive streaming is to explore effective approaches to optimize QoE. Moreover, video caching and delivery at the MEC allows Internet service providers (ISPs) to reduce backhaul and inter-ISP traffic. In HAS, the video content caching at the MEC is challenging since for the download of each segment, not only the requested segment but also its video rate must be available in the cache. Due to the unstable wireless network, the video caching at the MEC for HAS needs to be efficiently handled to reduce the backhaul traffic and improve the QoE of video users. For instance, by fetching low quality cached segment than that sustainable by network, the backhaul traffic can be reduced by avoiding downloading not yet cached high quality segment from the video server.

(3) Multi-MEC Collaboration

The storage capacity and computational capability of an MEC is limited. The deployment of MEC's at the edge of the network is distributed, so that the storage and computational resources are available in different location within the network. In case an MEC server gets overloaded, the tasks can be offloaded to the neighboring MEC servers. The sharing of resources among MEC servers is an important research issue. For instance, if the bandwidth available at the current MEC server cannot provide high quality video, how to select an optimal server among other MEC servers that can stream high quality video to the video client. Moreover, when the current MEC server is computationally overloaded, how to offload the tasks to the neighboring servers. The resource sharing methods between collaborating MEC servers need to be studied to optimize resource utilization and QoE in the future.

9 Conclusion

In this paper, we perform an in-depth analysis of both MEC-assisted and client-side rate adaptation algorithms using a multi-client cellular network under varying client, server, dataset, and network settings. Evaluations and comparisons were performed in terms of conflicting video quality metrics. The experimental results revealed that MEC-assisted algorithms fairly selected video rates for competing HTTP clients and efficiently utilized resources compared to client-based algorithms for most of the experimental settings. The performance of client-based algorithms in terms of fairness and bandwidth inefficiency was inconsistent. Results suggest that the reason is that the algorithms were designed based on specific client, server, and network settings. It is also demonstrating that both MEC-assisted and client-based algorithms give precedence to a specific video quality objective over others. This also leads to the algorithms providing inconsistent QoE in different environments. Results demonstrate that buffer-based adaptation algorithms selected higher video rates and experienced lesser video rate switches than throughput-based algorithms in a stable network; however, they were unable to protect the playback buffer in an unstable environment. The unnecessary video rate switch affected the QoE of throughput-based algorithms the most. In addition, results revealed that the buffer size and segment duration affected the performance of the algorithms. The algorithms performed better while downloading smaller segments compared with larger segments. This effect was more prominent for buffer-based algorithms. It is suggested that rather than predefined buffer thresholds, algorithms need to select adjustable buffer threshold on the basis of client, server, and network settings. Our results demonstrate that the selection of the representation set for algorithms should be carefully considered. The results also suggest that the algorithms should utilize context data including HAS clients' information and device features to select the video quality.

In future work, we will focus on designing context-aware MEC-assisted algorithms that consider factors such as network settings, HAS clients' information, video content details and device features in addition to the bandwidth and playback buffer dynamics to select video rates.

ACKNOWLEDGEMENT

This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the Grand Information Technology Research Center support program(IITP-2021-2015-0-00742) supervised by the IITP(Institute for Information & communications Technology Planning & Evaluation)

REFERENCES

[1] "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2017-2022 White Paper," Feb 2019. Accessed: Apri. 27, 2019 [Online]. Available:https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html

[2] S. Egger, B. Gardlo, M. Seufert, and R. Schatz, "The impact of adaptation strategies on perceived quality of http adaptive streaming," *in Proc. ACM Workshop Design Quality Deployment Adaptive Video Streaming*, 2014, pp. 31-36.

[3] Z. Li, X. Zhu, J. Gahm, R. Pan, H. Hu, A.C. Begen, and D. Oran, "Probe and adapt: Rate adaptation for HTTP video streaming at scale," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 4, pp. 719-733, 2014.

[4] T. Y. Huang, H. Nikhil, H. Brandon, M. Nick, and J. Ramesh, "Confused, timid, and unstable: picking a video streaming rate is hard," *in Proc. of ACM Int. conf. on Internet Meas.*, 2012, pp. 225-238.

[5] S. Akhshabi, L. Anantakrishnan, A.C. Begen, and C. Dovrolis, "What happens when HTTP adaptive streaming players compete for bandwidth?," *in Proc. of ACM Workshop Netw. Operating Syst. Support for Digit. Audio Video*, 2012, pp. 9-14.

[6] W. U. Rahman and K. Chung, "A novel adaptive logic for dynamic adaptive streaming over HTTP," *J. Vis. Commun. Image Representatio*, vol. 49, pp. 433-46, 2017.

[7] P. Juluri, V. Tamarapalli, and D. Medhi: SARA: Segment aware rate adaptation algorithm for dynamic adaptive streaming over HTTP," *in Proc. of IEEE Int. Conf. on Commun. Workshop*, 2015, pp. 1765-1770.

[8] M. Azumi, T. Kurosaka, M. Bandai, "A QoE-aware quality-level switching algorithm for adaptive video streaming," *in Proc. of IEEE Global Commun.*, 2015, pp. 1-5.

[9] W. U. Rahman and K. Chung, ''A multi-path-based adaptive scheme for multi-view streaming over HTTP,'' *IEEE Access*, vol. 6, pp. 77869-77879, 2019.

[10] K. Miller, E. Quacchio, G. Gennari, and A. Wolisz, "Adaptation algorithm for adaptive streaming over HTTP," *in Proc. PV*, Munich, Germany, 2012, pp. 173-178.

[11] T. Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A buffer-based approach to rate adaptation: Evidence from a large video streaming service," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, 187-198, 2015.

[12] C. Muller, S. Leder, C. Timmerer, "An evaluation of dynamic adaptive streaming over HTTP in vehicular environments," *in Proc. Workshop Mobile Video*, 2012, pp. 37-42.

[13] V. Aggarwal, R. Jana, J. Pang, K. K. Ramakrishnan, and N. K. Shankaranarayanan, "Characterizing fairness for 3G wireless

networks," *in Proc. IEEE Workshop Local Metropolitan Area Netw.*, 2011, pp. 1–6.

[14] Z. Yan, J. Xue, and C. W. Chen, "Prius: Hybrid edge cloud and client adaptation for HTTP adaptive streaming in cellular networks," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 27, no. 1, pp. 209–222, Jan. 2017.

[15] T. X. Tran, A. Hajisami, P. Pandey, and D. Pompili, "Collaborative mobile edge computing in 5G networks: New paradigms, challenges and scenarios," *IEEE Commun. Magazine*, vol. 55, no. 4, pp. 54-61, 2017.

[16] S. Yi, Z. Hao, Q. Zhang, Q. Zhang, W. Shi, and Q. Li, "Lavea: Latencyaware video analytics on edge computing platform," *in Proc. IEEE Int. Conf. Distrib. Comput. Syst*., 2017, pp. 1-13.

[17] D. Wang, Y. Peng, X. Ma, W. Ding, H. Jiang, F. Chen, and J. Liu, "Adaptive wireless video streaming based on edge computing: Opportunities and approaches," *IEEE Trans. Services Comput.*, vol. 12, no. 5, pp, 685-697, 2018.

[18] S. Nunna, A. Kousaridas, M. Ibrahim, M. Dillinger, C. Thuemmler, H. Feussner, and A. Schneider, ''Enabling real-time context-aware collaboration through 5G and mobile edge computing,'' *in Proc. Int. Conf. Inf. Technol. New Gener.*, 2015, pp. 601–605.

[19] Z. Xiang, F. Gabriel, E. Urbano, G. T. Nguyen, M. Reisslein, and F. H. P. Fitzek, "Reducing latency in virtual machines: Enabling tactile Internet for human-machine co-working,'' *IEEE J. Sel. Areas Commun*., vol. 37, no. 5, pp. 1098–1116, 2019.

[20] W. U. Rahman, C. S. Hong and E. Huh, "Edge Computing Assisted Joint Quality Adaptation for Mobile Video Streaming," *IEEE Access*, vol.7, no. 1, pp. 129082-129094, 2019.

[21] A. Mehrabi, M. Siekkinen, and A. Ylä-Jääski, "Edge computing assisted adaptive mobile video streaming," *IEEE Trans. Mobile Comput*., vol. 18, no. 4, pp. 1-17, 2018.

[22] S. Akhshabi, A. C. Begen, and C. Dovrolis, "An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP," *in Proc. ACM Conf. on Multimedia Syst*., 2011, pp. 157-168.

[23] Thang, Truong Cong, et al. "An evaluation of bitrate adaptation methods for HTTP live streaming," *IEEE J. Sel. Areas Commun*., vol. 32, no. 4, pp. 693-705, 2014.

[24] Ayad, Ibrahim, et al. "A practical evaluation of rate adaptation algorithms in http-based adaptive streaming." *Comput. Netw.* vol. 133, pp. 90-103, 2018.

[25] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman, "BOLA: Near-optimal bitrate adaptation for online videos." *in Proc. IEEE Conf. Comput. Commun.*, 2016, pp. 1-9.

[26] J. Kua, G. Armitage, and P. Branch, "A survey of rate adaptation techniques for dynamic adaptive streaming over HTTP," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1842-1866, 2017.

[27] D. Stohr, A. Frommgen, A. Rizk, "Where are the sweet spots? A systemic approach to reproducible dash playe rcomparisons," in Proc. ACM Conf. Multimedia, 2017, pp. 1113-1121.

[28] M. Mehrabi, D. You, V. Latzko, H. Salah, M. Reisslein, F. H. Fitzek," Device-Enhanced MEC: Multi-Access Edge Computing (MEC) Aided by End Device Computation and Caching: A Survey," *IEEE Access*, vol. 12, no. 7, pp. 166079-108, 2019.

[29] S. R. Yang, Y. J. Tseng, C. C. Huang, W. C. Lin, "Multi-access edge computing enhanced video streaming: Proof-of-concept implementation and prediction/QoE models, " *IEEE Trans. Veh. Technol.*, vol. 68, no. 2, pp. 1888-902, 2018.

[30] A. Martin, R. Viola, M. Zorrilla, J. Flórez, P. Angueira, J. Montalbán, "MEC for Fair, Reliable and Efficient Media Streaming in Mobile Networks," *IEEE Trans. Broadcast*, 2019.

[31] K. Bilal, A. Erbad, "Edge computing for interactive media and video streaming," *in Proc. IEEE Conf. Fog Mobile Edge Comput*., 2017, pp. 68-73.

[32] Ma G, Wang Z, Zhang M, Ye J, Chen M, Zhu W. Understanding performance of edge content caching for mobile video streaming. IEEE Journal on Selected Areas in Communications. 2017 Mar 9;35(5):1076-89.

[33] W. U. Rahman, C. S. Hong and E. Huh, "Performance Evaluation of Edge Computing Assisted Adaptive Streaming Algorithms," *in Proc. IEEE Conf. Info. Netwk.*, 2020, pp. 226-231.

[34] Progressive Download and Dynamic Adaptive Streaming Over HTTP, document 3GPP TS 26.247 V12.1.0, 2013. [Online]. Available: http://goo.gl/4EJbvd.

[35] A. Zambelli. Microsoft Corporation. IIS smooth streaming technical overview. [Online]. Available: https://www.bogotobogo.com/VideoStreaming/Files/iis8/IIS_Smooth_Streaming_Technical_Overview.pdf

[36] Adobe. Configure HTTP Dynamic Streaming and HTTP Live Streaming. [Online]. Available: https://helpx.adobe.com/adobe.../configure-dynamic-streaming-live-streaming.html

[37] R. Pantos: HTTP Live Streaming. [Online]. https://tools.ietf.org/html/rfc8216. Accessed 12 May 2020

[38] F. Dobrian, A. Awan, D. Joseph, A. Ganjam, J. Zhan, V. Sekar, I. Stoica, and H. Zhang, "Understanding the impact of video quality on user engagement," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. 362-373, 2015.

[39] P. Ni, R. Eg, A. Eichhorn, C. Griwodz, and P. Halvorsen, "Flicker effects in adaptive video streaming to handheld devices," *in Proc. ACM Conf. Multimedia*, 2011, pp. 463–47.

[40] Y. Liu, S. Dey, D. Gillies, F. Ulupinar, and M. Luby, "User experience modeling for DASH video," *in Proc. IEEE Packet Video Workshop*, 2013 pp. 1–8.

[41] Y. Shen, L. Yitong, H. Yang, and D. Yang, "Quality of Experience study on dynamic adaptive streaming based on HTTP," *IEICE Trans. Commun.*, vol. 98, no. 1, pp. 62-70, 2015.

[42] Y. Qi and M. Dai, "The effect of frame freezing and frame skipping on video quality," *in Proc. IEE Conf. Intell. Inf. Hiding Multimedia Signal Process*., 2006, pp. 423–426.

[43] N. Staelens, J. De Meulenaere, M. Claeys, G. Van Wallendael, W. Van den Broeck, J. De Cock, R. Van de Walle, P. Demeester, and F. De Turck, "Subjective quality assessment of longer

duration video sequences delivered over HTTP adaptive streaming to tablet devices," *IEEE Trans. Broadcast*., vol. 60, no. 4, pp. 707–714, 2014.

[44] Q. Huynh-Thu and M. Ghanbari, "Temporal aspect of perceived quality in mobile video broadcasting," *IEEE Trans. Broadcast.*, vol. 54, no. 3, pp. 641-651, 2008.

[45] A. K. Moorthy, L. K. Choi, A. C. Bovik, and G. De Veciana, "Video quality assessment on mobile devices: Subjective, behavioral and objective studies," *IEEE J. Sel. Topics Signal Process*, vol. 6, no. 6, pp. 652-671, 2012.

[46] T. Hoßfeld, M. Seufert, C. Sieber, and T. Zinner, "Assessing effect sizes of influence factors towards a QoE model for HTTP adaptive streaming," in *Proc. of IEEE Conf. Multimedia Experience*, 2013, pp. 111–116.

[47] X. Yin, A. Jindal, V. Sekar, B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over HTTP", *ACM SIGCOMM Comput. Commun. Rev*., vol. 45, no. 4, pp. 325-338, 2015.

[48] J. Jiang, V. Sekar, and H. Zhang, "Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive," *in Proc. ACM Conf. Emerging Netw. Experiments Technol*., 2012, pp. 97–108.

[49] R. Jain, D. Chiu, and W. Hawe, "A quantitative measure of fairness and discrimination for resource allocation in shared computer system," Technical Report, December 1984.

[50] "ns-3 LTE module". [Online]. Available: https://www2.nsnam.org/tutorials/tutorials/consortium13/lte-tutorial.pdf

[51] MPEG-DASH / Media Source demo., http://dash-mse-test.appspot.com/

[52] H. T. Le, N. P. Ngoc, C. T. Truong," Bitrate adaptation for seamless on-demand video streaming over mobile networks," *Signal Process.: Image Commun*., vol. 65, pp.154-64, 2018.

[53] T. C. Thang, Q. D. Ho, J. K. Kang, A. T. Pham, "Adaptive streaming of audiovisual content using MPEG DASH," *IEEE Trans. Consum. Electron*., vol. 58, no. 1, pp. 78-85, 2011.

[54] "Average YouTube video length as of December 2018, by category". [Online]. Available: https://www.statista.com/statistics/1026923/youtube-video-category-average-length/

[55] W. U. Rahman, K. Chung, "SABA: Segment and buffer aware rate adaptation algorithm for streaming over HTTP," *Multimedia Systems*, vol. 24, no. 5, pp. 509-29, 2018.