future internet

MDPI

Article

Fuzzy-Based MEC-Assisted Video Adaptation Framework for HTTP Adaptive Streaming

Waqas ur Rahman

School for Architecture, Built Environment, Computing and Engineering, Birmingham City University, Birmingham B4 7AP, UK; waqas.rahman@bcu.ac.uk

Abstract

As the demand for high-quality video streaming applications continues to rise, multi-access edge computing (MEC)-assisted streaming schemes have emerged as a viable solution within the context of HTTP adaptive streaming (HAS). These schemes aim to enhance both quality of experience (QoE) and utilization of network resources. HAS faces a significant challenge when applied to mobile cellular networks. Designing a HAS scheme that fairly allocates bitrates to users ensures a high QoE and optimizes bandwidth utilization remains a challenge. To this end, we designed an MEC- and client-assisted adaptation framework for HAS, facilitating collaboration between the edge and client to enhance users' quality of experience. The proposed framework employs fuzzy logic at the user end to determine the upper limit for the video streaming rate. On the MEC side, we developed an integer nonlinear programming (INLP) optimization model that collectively enhances the QoE of video clients by considering the upper limit set by the client. Due to the NP-hardness of the problem, we utilized a greedy algorithm to efficiently solve the quality adaptation optimization problem. The results demonstrate that the proposed framework, on average, (i) improves users' QoE by 30%, (ii) achieves a fair allocation of bitrates by 22.6%, and (iii) enhances network utilization by 4.2% compared to state-of-the-art approaches. In addition, the proposed approach prevents playback interruptions regardless of the client's buffer size and video segment duration.

Keywords: video quality adaptation; DASH; adaptive bitrate streaming; QOE; fuzzy logic

Academic Editors: Paolo Bellavista, Zhihui Lu and Qiang Duan

Received: 10 July 2025
Revised: 21 August 2025
Accepted: 2 September 2025
Published: 8 September 2025

Citation: Rahman, W.u. Fuzzy-Based MEC-Assisted Video Adaptation Framework for HTTP Adaptive Streaming. *Future Internet* **2025**, *17*, 410. https://doi.org/10.3390/fi17090410

Correction Statement: This article has been republished with a minor change. The change does not affect the scientific content of the article and further details are available within the backmatter of the website version of this article.

1. Introduction

HTTP adaptive streaming (HAS) is the leading video delivery system that delivers high-quality video content to users. It allows HTTP clients to adjust the quality according to the fluctuating network conditions [1].

In HAS, videos are encoded at multiple bitrates/resolutions and stored on the server. The HTTP client initiates the video session by requesting the manifest file from the server. In response, the server sends the manifest file to the client, which includes descriptions of the segments, such as video quality, segment bitrate, and other relevant information. The adaptive bitrate (ABR) algorithm runs at the client-chosen video quality according to the client's context. The algorithm aims at selecting the highest feasible bitrate while ensuring uninterrupted and seamless streaming.

Traditionally, HAS algorithms utilize a client-based adaptive approach. The most significant advantage of a client-based approach is scalability. Each client makes its own decision based on the information available at its end. However, the clients are unaware of

the decisions made by other clients, which can lead to unfairness and higher frequency of video quality switches [2–4].

Lately, edge computing has emerged to offer enhanced performance compared to cloud computing through server deployment within the radio access network (RAN) [5–7]. It brings computational power and storage capability to the edge of the network. Unlike the client-based HAS approach, the information about radio link conditions and each client's request are available at the edge. Furthermore, the clients can share client-side information, such as buffer level and observed throughput, with the edge cloud. This knowledge can be leveraged to assist ABR algorithms running at the edge in making better bitrate selection decisions, enhancing the user experience and equitable allocation of resources among all clients. Also, MEC-assisted algorithms can equitably assign resources among HAS clients.

In MEC-assisted adaptive streaming, the MEC server assumes a critical role in centralized network resource management, leveraging real-time insights from the RAN and client data. By facilitating client coordination and assisting in the determination of suitable video bitrates based on channel conditions, the MEC server ensures that the adaptive streaming system collectively optimizes clients QoE, bandwidth utilization, and fairness.

Leveraging MEC-assisted adaptive streaming presents significant challenges. The MEC-assisted streaming framework needs to be interoperable with the DASH standard. This compatibility should be achieved without necessitating any modifications to either the servers or clients. Another key challenge is the design of an MEC-assisted HAS system to present an optimized HAS solution to enhance the QoE and fairness between streaming clients while efficiently utilizing the bandwidth. The existing MEC-assisted adaptation frameworks select the bitrates for upcoming segments at the edge cloud without requiring the client's involvement in the bitrate selection process [8–10]. The clients share client-side information, including buffer level, observed throughput, and media presentation description (MPD), but they are not directly involved in the decision-making process. The adaptation algorithm running at the edge aims to jointly optimize the clients' user experience. The drawback of this approach is that the adaptation algorithm might overlook the needs of some clients in the process of maximizing the QoE for all clients. While a client is facing the risk of playback interruptions, the adaptation scheme may not necessarily choose the most cautious bitrate to avoid them. Rather, it might favor objectives such as enhancing video quality, ensuring fairness, and optimizing bandwidth usage. For example, a video client near the base station observes a higher bandwidth than a client near the edge of the cell. Based on the bandwidth, the client near the base station could download the video at a higher resolution compared to the client near the edge of the cell. Downloading a low-resolution video for the HAS client near the base station improves fairness but degrades video quality and bandwidth efficiency, whereas selecting a high-resolution video for the client at the edge of cell enhances fairness and video quality, but it results in buffer underflow. Therefore, the proposed framework facilitates collaboration between the edge cloud and clients during the quality selection process. Based on the client-side network conditions, the client sends a naive suggestion to the edge cloud that is treated as the upper limit for the video streaming rate. The proposed method utilizes a fuzzy-based adaptive streaming method, which monitors throughput and buffer level to suggest an appropriate bitrate to the edge cloud. The aim of this client-side adaptation approach is to guarantee that the recommended bitrate effectively reduces the likelihood of playback interruptions. Consequently, the edge cloud can focus on optimizing video quality and ensuring fair bitrate distribution without worrying about rebuffering during decision making.

This work offers the following contributions:

- Architecture: We leverage edge computing to design a hybrid MEC and client adaptation architecture for HAS, facilitating collaboration between the edge and client and improving users' QoE and network utilization;
- Client-side adaptation: We design a fuzzy-based ABR algorithm that recommends the upper limit for the video streaming rate to the edge cloud solely based on client-side information;
- Joint optimization problem formulation: Following the client's recommendation, we formulate an integer nonlinear programming (INLP) optimization model to jointly optimize QoE, network utilization, and the equitable distribution of bitrates among all competing clients;
- Heuristic methods: Due to the NP-hardness of the problem, we design a greedy algorithm that produces a sub-optimal solution. We present a heuristic approach that functions in polynomial time;
- Analysis: We analyze the efficiency of the proposed framework by conducting comprehensive experiments and compare the results with other state-of-the-art schemes. The results illustrate that the proposed approach provides significant enhancements in terms of QoE, network utilization, and fairness, with an average improvement of 30%, 22.6%, and 4.2%, respectively.

2. Related Work

2.1. Client-Based Methods

The aim of HAS algorithms is to enhance the QoE of the users. Initially, client-side HAS schemes have been proposed that picked the bitrate according to the instantaneous throughput and buffer level [11–16]. The algorithms rely on fixed control laws that lead to inconsistent performance under varying client and service settings [10]. Also, client-based algorithms prioritize different video quality objectives. The throughput-based algorithms perform well under stable network links. However, they experience a high frequency of playback interruptions and fail to mitigate the video bitrate switching ratio. The authors in [10] showed that, similar to throughput-based methods, buffer-based algorithms also experience high number of playback stalls. In [16], the authors employed fuzzy logic to pick the video quality of the segments. The algorithm observes buffer level and changes in buffer level to avoid rebuffering. This approach has a limitation in that it does not account for segment duration and throughput fluctuations. For example, when the available buffer is 8 s, and the duration of the segment is 2 s, it means that there are 4 segments stored in the buffer. In contrast, when the duration of the segment is 8 s, it implies that just one video segment is buffered. In the latter scenario, even a minor throughput variation could lead to buffer underflow. Hence, it is crucial to take segment duration and throughput variation into account as input variables for FLC. Akshan et al. [17] also employed bitrate adaptation using fuzzy logic controller (FLC), which observes throughput and buffer level to the select video quality and download scheduling to mitigate the negative effect of ON–OFF switching. Mowaf et al. [18] extended the work presented in [17] and incorporated the available power in the adaptation process to enhance the longevity of the client's battery. However, this extension in battery life comes at the cost of video quality. In [19], the authors introduced a video adaptation controller that considers segment duration and playback buffer size in addition to buffer level and estimated throughput to select bitrates. The objective of their work was to guarantee enhanced QoE across different HTTP client settings and video characteristics. However, this work proposed a client-based adaptation algorithm where clients are oblivious of each other's conditions.

When multiple HTTP clients compete for bandwidth, the rate adaptation algorithms unfairly assign variable and degraded video to the users [3]. Li et al. [2] proposed the Panda (Probe AND Adapt) algorithm, aimed at addressing issues related to fairness and instability in video quality in a multi-client setting. Although the algorithm works efficiently in a stable wired network, its performance degrades under dynamic cellular links.

2.2. Edge-Assisted Methods

Yang et al. [20] developed a proof-of-concept for an edge-assisted streaming service. Mehrabi et al. [21] introduced an MEC-assisted adaptation solution designed to improve QoE and fairness simultaneously. The authors in [10] showed that the solution does not provide guaranteed user experience under a variety of client and server settings.

Kim et al. [22] proposed an edge-assisted scheme that considers resource allocation along with QoE and fairness. All these edge-assisted solutions use optimization models to address the problem of joint adaptation at the edge [21–23]. In [24], authors proposed a Reinforcement Learning (RL)-based framework that incorporates edge collaboration to enhance QoE in multi-client environments. By reallocating clients to optimal edge servers and training RL models specific to network conditions, the approach improves both overall QoE and fairness across users. These solutions are purely edge-based solutions where the clients do not have any input in the selection of the video quality. All the decision-making rests with a single point in the architecture.

Yan et al. [23] employed a QoE continuum model emphasizing the significance of the most recently downloaded segments for QoE evaluation in contrast to earlier segments. The study did not address the equitable and efficient utilization of bandwidth by HAS clients.

Our work presents a hybrid MEC- and client-based framework for HAS. First, the HTTP clients utilize a fuzzy controller to choose bitrates for the upcoming segments and share them with the edge server. The server locks these values as the highest achievable bitrates for the clients. The optimization algorithm running at the edge jointly optimizes QoE, bandwidth utilization, and fairness for streaming in MEC environments.

2.3. User Experience

In [25], authors identified the factors influencing QoE. These factors include video bitrate of the downloaded segments, bitrate switches, and playback rebuffering events. It has been demonstrated that QoE is influenced by playback interruptions and bitrate [26–33]. These factors present a contradiction because increasing the video bitrate may lead to more frequent rebuffering. Additionally, studies have demonstrated that frequent changes in video bitrate negatively affect user engagement [33]. The goal of HAS algorithms is to enhance all video quality factors to enhance the user experience. The average bitrate of the streamed segments during the video session is calculated according to

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

where R_{ij}^{k} represents the ith bitrate selected by the jth client for the kth segment, and S represents the number of segments downloaded during the streaming session.

Frequent changes in video quality also degrade the QoE. Average change in quality from one segment to another is given by

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

Playback interruptions affect the QoE the most [34]. The HTTP client experiences rebuffering if the segment download time ($\tau \times R_{ij}^k / T^k$) is higher than the buffer level. The rebuffering time during the streaming session, IR, is $\sum_{k=1}^{S} \left(\tau \times R_i^k / T^k - B^k \right)_+$.

To assess the proposed method, we utilize a QoE model used by Model Predictive Control (MPC): Linear QoE [35].

$$QoE_j = \sum_{k=1}^{S} q\left(R_{ij}^k\right) - \mu IR_j - \gamma \sum_{k=1}^{S} \left| q\left(R_{ij}^k\right) - q\left(R_{ij}^{k-1}\right) \right| \quad (3)$$

where $q\left(R_{ij}^k\right)$ represents the quality for the kth segment encoded at the ith bitrate, and γ and μ are non-negative weights assigned to quality changes and interruption time, respectively.

When multiple clients share a bottleneck link, the client inequitably and inefficiently utilizes the bandwidth [2,3,28]. To maximize bandwidth utilization, we aim to select for the clients the most feasible bitrate such that it minimizes the difference between their combined bitrates and the bandwidth at the base station. A value close to 0 ensures that the clients in aggregate are using the highest feasible bitrate to enhance QoE. The bandwidth inefficiency is determined as follows:

$$IE_j = |R_{ij}^{(t)} + \sum_{v \neq j} R_{iv}^{(t)} - BW^{(t)}| \quad (4)$$

where $BW^{(t)}$ is the available bandwidth at time t, $R_{ij}^{(t)}$ represents the bitrate picked by the jth client at time t, and $\sum_{v \neq j} R_{iv}^{(t)}$ is the sum of the bitrates of the video clients at time t.

To ensure a fair allocation of bitrates among clients, we select for each client the most appropriate bitrate that has the smallest deviation from the average video rates assigned to other competing clients. The fairness index at time t is computed as follows:

$$F_j = |R_{ij}^{(t)} - R_{avg}| \quad (5)$$

where $R_{avg} = \left(\frac{1}{N-1}\right) \sum_{v \neq j} R_{iv}^t$ is the average of the bitrates streamed other video clients.

Low inefficiency values signify that the clients downloaded the highest feasible bitrates, while a low value for fairness indicates that clients shared the bandwidth fairly.

3. Proposed Design

3.1. Architecture Overview

Figure 1 shows the hybrid MEC-assisted adaptive streaming model. The video is encoded into a set of bitrates R = {R_1, R_2, R_3, …, R_m} and fragment into segments of duration, τ. A set of N clients requests kth segment from the server, and each client is indexed by j, where j = 1, 2, …, N. The system parameters and their descriptions are given in Table 1.

To initiate the streaming session, the client requests metadata about the video content available at the server. In response, the server shares the metadata, so the edge and client have the information on the data available at the server. In the proposed design, a fuzzy-based HAS adaptation method running at the client determines the upper limit for the streaming rate, R_s, according to client-side information and shares it with the edge cloud. Under the traditional HAS framework, the network relays the bitrate selected by the client to the server. As shown in Figure 1, in hybrid MEC- and client-assisted architecture, the edge cloud acknowledges R_s as the maximum achievable bitrate for the HAS client. The client's suggestion for the next segment is based on the client- and server-side information, including available quality levels, observed throughput, buffer level, buffer size, segment duration, etc. [10]. In addition, the clients can also share client-side information, including

buffer level, observed throughput, initial playout delay, playlist, and MPD information. The 3rd Generation Partnership Project (3GPP) has standardized QoE reporting for clients using an HTTP POST request that carries metadata in XML format [36]. However, as the number of clients increases, frequent sharing of client-side application-level information with the edge cloud increases unnecessary overhead. Hence, the proposed scheme incorporates a fuzzy logic controller at the client, utilizing client-side data to provide only a preliminary recommendation to the edge cloud. Since the adaptation at the edge cloud is cell-wide, it might overlook the needs of an individual client. For example, a client may be in danger of experiencing playback interruption, but the adaptation block at the edge cloud prioritizes optimizing cell wide QoE and fairness. To this end, the adaptation block at the edge treats the bitrate suggestion by the client as the upper bound, which reflects the maximum rate supported by the client.

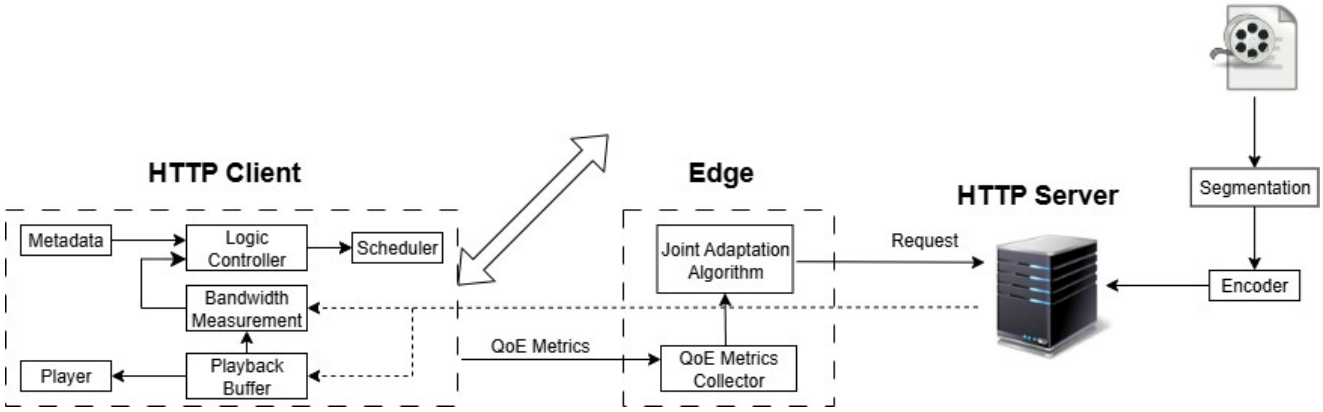

Figure 1. Block diagram of the hybrid MEC-assisted HAS framework.

At the edge, the proposed integer nonlinear programming (INLP) optimization model (discussed in Section 3.3) jointly optimizes the QoE, fairness, and bandwidth utilization of HAS clients. The optimization model treats the recommended bitrate, R_s, from the client as the upper limit for the video streaming rate. The presence of integer decision variables makes it computationally impractical to tackle the problem through a brute force method. The reason is that the complexity increases exponentially as the number of clients grows, rendering it unfeasible for large-scale HAS scheduling. To this end, we propose an online heuristic algorithm (discussed in Section 3.4) that operates at the edge with an overall complexity that remains polynomial. Based on the video quality optimization of the clients, the adaptation block's outcomes are subsequently transmitted to the server for downloading the upcoming segments.

Table 1. Description of parameters.

N	Number of clients
S	Total segments streamed by the clients
R	Set of bitrates stored at the server
R_i^k	*kth* segment encoded at the *ith* bitrate
R_s	Bitrate recommended by the client to edge server
R_{avg}	Average of the bitrates currently streamed by the competing clients

Table 1. *Cont.*

R_{max}, R_{min}	Maximum and minimum bitrate in set R
B^k	Buffer level at the download of *kth* segment
B_{max}	Buffer size
T^k	Throughput during the download of *kth* segment
m	Available bitrates at the server
k	Current segment index
τ	Duration of the segment
ρ	The degree of variation in throughput during the download of the last two segment
BW	Bandwidth at base station
W_j	Bandwidth allocated to the *jth* client
Q_j	Average video bitrate achieved by the *jth* client
QS_j	Average bitrate changes experience by the *jth* client
F_j	Fairness contribution by allocating bitrate to client j during the streaming session
IE_j	Bandwidth inefficiency by allocating bitrates to client j during the streaming session
IR	Total rebuffering time
X	Output of FLC
x_{ij}	Decision variable that defines number of clients streaming the ith bitrate stored in the server
α, β, φ, θ	Vide rate, bitrate switching, fairness, and bandwidth inefficiency weighting parameters
δ_s	Switching level threshold
δ_{IE}	Inefficiency threshold
δ_F	Fairness threshold

3.2. Fuzzy-Logic Controller

The aim of the fuzzy-based adaptation method is to suggest the most feasible bitrate, R_s, to the edge-cloud based on only client-side information. The edge cloud treats this suggestion as the maximum bitrate supported by the client. The fuzzy-based quality adaptation algorithm considers the following goals to select the bitrate:

(1) Maximize quality;
(2) Avoid rebuffering;
(3) Minimize bitrate switches.

Here, we discuss the details of fuzzy-based quality adaptation algorithm and the integration of FLC in the HAS client to pick a suitable bitrate. At the client side, we utilize an FLC based on the Mamdani model [37]. The structure of FLC comprises four main modules: (1) fuzzification module, (2) fuzzy inference engine, (3) defuzzification module, and (4) fuzzy rule base [38].

The FLC functions through a repeating cycle of four stages:

Stage 1: Recording the crisp measurement (input) of all variables that represent the controller process's conditions.

Stage 2: Mapping these measurements into corresponding fuzzy sets to represent measurement uncertainties. This process is referred to as fuzzification.

Stage 3: The fuzzified measurements are processed by the inference engine, which assesses the control rules stored in the fuzzy rule base. The result of the inference engine is a fuzzy set.

Stage 4: The final step, known as defuzzification, involves converting the fuzzy set into a single crisp value.

The proposed FLC applies three input variables: (1) B/τ; (2) the changes in the throughput during the streaming of the last two segments, $\rho = |log(T^k/T^{k-1})|$; and (3) $\Delta B/\tau$, where $\Delta B = B^k - B^{k-1}$ represents the variation in the buffer level between the download of kth and $kth-1$ segment.

Most of the HAS algorithms adapt bitrates according to changes in the observed throughput and buffer level. However, buffer level does not provide complete knowledge about the amount of data downloaded in the client's buffer. Video streaming services provide different segment durations. For instance, Adobe's HTTP Dynamic Streaming (HDS) and Apple HTTP Live Streaming (HLS) provide segment durations of four and ten seconds, respectively [16,17]. As shown in Figure 2, if buffer size is thirty seconds, and segment duration deployed by the streaming service is two seconds, only fifteen segments can fill the buffer. If the segment duration offered by the server is four seconds, the number of segments that can fit into the buffer decreases. Therefore, we use the B/τ as input for the FLC.

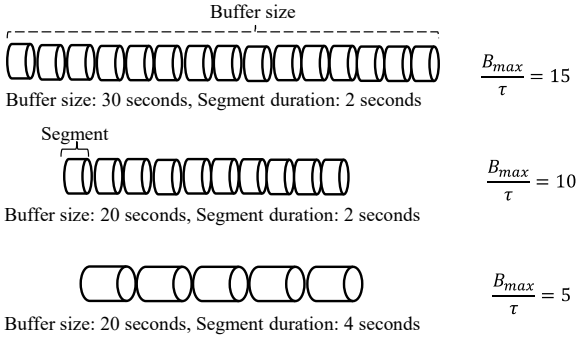

Figure 2. Larger buffer size enables the client to store a greater number of segments. A longer segment duration leads to a reduction in the number of segments that can be stored in the buffer.

The algorithm adapts the bitrate as the observed throughput varies. Therefore, it is important to distinguish between minor and major variations in the throughput. If there is a sudden significant decrease in throughput, the adaptation algorithm would quickly switch the bitrate to avoid playback interruption. To this end, we defined a variable, $\rho = |log(T^k/T^{k-1})|$, that differentiates between minor and substantial variations in the throughput. Figure 3 illustrates the changes in ρ as throughput fluctuates during the streaming of video segments. Figure 3 shows that ρ remains close to zero when throughput variations are small but increases as throughput changes become more significant. Although ρ can distinguish between large and small variations in throughput, it does not offer insight into whether the throughput is increasing or decreasing. To this end, the fluctuations in the buffer (ΔB) helps the client to determine whether the observed throughput is increasing or decreasing compared to the chosen bitrate. However, similar to the buffer level itself, ΔB does not entirely reflect the changes in the buffer level. For instance, when the buffer increases by four seconds during the download of a two-second segment, it indicates that the buffer level becomes filled to a capacity twice that of the segment duration. However, if the duration of the segment is four seconds, and the buffer level also increases by four seconds, it means that the buffer level becomes filled to a capacity matching that of the segment duration. Hence, the third variable we consider as an input for the controller is $\Delta B/\tau$.

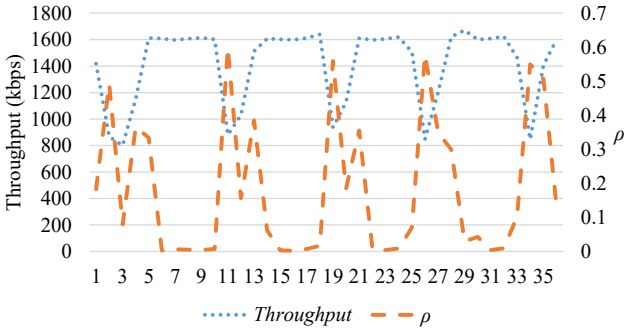

Figure 3. Variation in ρ as throughput fluctuates.

In Figure 4a, the membership functions of B/τ are depicted, encompassing three linguistic variables: dangerous, low, and safe. If the number of segments in buffer are less than (B_d/τ), the buffer level is in dangerous zone. Here, B_d is given by

$$B_d = \min(\tau, 20\% \times B_{max}) \tag{6}$$

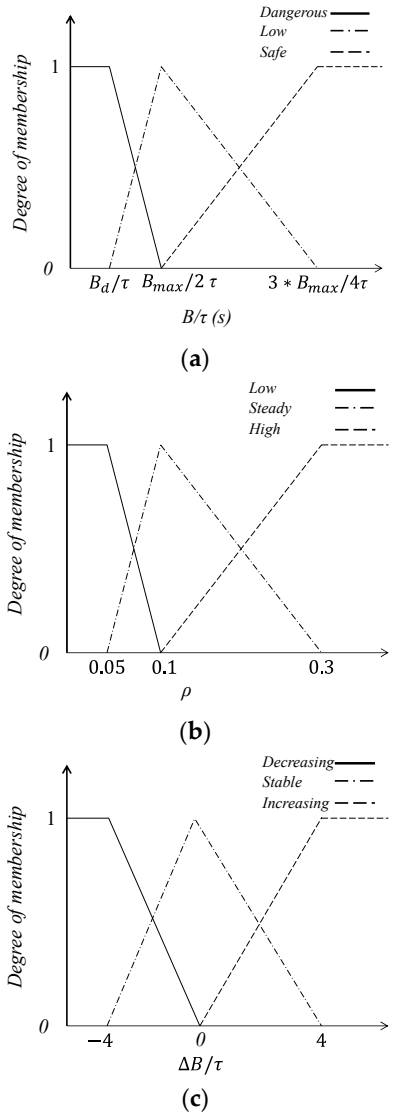

Figure 4. Membership functions for (**a**) B/τ, (**b**) $\rho = log(T_k/T_{k-1})$, and (**c**) $\Delta B/\tau$.

As the segment duration increases, it takes more time to download the segment. In an unstable environment where the selected bitrate exceeds the throughput, the risk of

playback interruption increases. Therefore, it becomes important to take segment duration into consideration in the selection of B_d. However, when dealing with a small buffer size, it is not feasible to select B_d based solely on segment duration. For instance, if the buffer size is 20 s, and the segment duration is 10 s, the rate adaptation algorithm will consistently select a conservative bitrate. This cautious approach is necessary because even a slight mismatch between the bitrate and throughput could lead to rebuffering events. Therefore, buffer size also becomes an important parameter in the selection of B_d. To this end, B_d is set equal to the minimum of segment duration and 20% of buffer size.

When the risk of playback interruptions is high, video quality is reduced to minimize rebuffering. As the client's buffer increases and stays close to $B_{max}/2$, it is in the low zone, and there is less likelihood of rebuffering. When the buffer rises above $B_{max}/2$, the chances of rebuffering reduces further, and the buffer level is in the safe zone. The adaptation algorithm can increase the video rate aggressively. Figure 4b shows the membership functions of ρ, in which three linguistic variables are considered: small, medium, and large. In the event of negligible variations in throughput, the value of ρ fluctuates around 0.05. The current bitrate can be maintained without concern for a drop in buffer level. As the value of ρ increases above 0.1, the bitrate is adjusted gradually. When ρ exceeds 0.3, the bitrate is adapted aggressively to minimize the risk of playback interruption. Figure 4c shows the membership functions of $\Delta B/\tau$. The following variables are considered: decreasing, stable, and increasing. The x-axis in Figure 4c represents changes in buffer level in terms of the number of segments. A decrease in the buffer level indicates that the downloaded bitrate is greater than the throughput. It forces the adaptation algorithm to drop the bitrate for the upcoming segments. When the buffer fills up, it signifies that the bandwidth is not fully utilized. This offers an opportunity to enhance the bitrate.

The controller's output, represented as O, indicates an adjustment factor for the bitrate of the next segment. Figure 5 illustrates the linguistic variables used for the output of the Fuzzy Logic Controller (FLC), which include large decrease (LD), small decrease (SD), no change (NC), small increase (SI), and large increase (LI). The output of the FLC signifies whether to increase, decrease, or stay at the current bitrate. A small output value corresponds to a decrease in bitrate, while a large value corresponds to an increase. Output values less than 0.5 are categorized as a large decrease, reflecting a sudden drop in the buffer level due to a sudden decrease in throughput. Output values between 0.25 and 1 are categorized as a small decrease. An output value around 1 indicates no change in bitrate. Output values above 1 suggest an opportunity to increase the bitrate. For values close to 1.5, the output is categorized as a small increase, and for values above 1.5, it is categorized as a large increase.

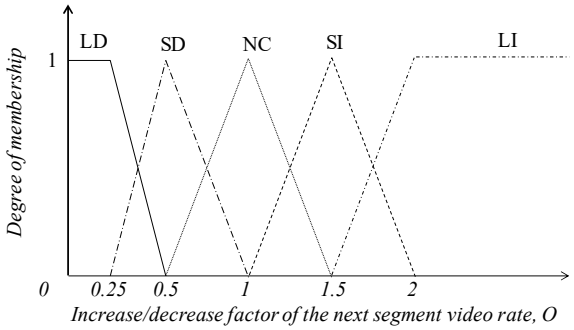

Figure 5. Output of the FLC.

The fuzzy if–then rules for FLC are given in Table 2. Table 2 illustrates that when B/τ falls into the dangerous zone, the bitrate is decreased to reduce the chances of rebuffering.

The FLC considers both ρ and $\Delta B/\tau$ to decide whether the bitrate should be decreased gradually or abruptly (*Rule 1* to *Rule 9*).

Table 2. Description of parameters.

Rule (R_i)	B/τ	$\Delta B/\tau$	ρ	O
Rule 1	Dangerous	Decreasing	Small	LD
Rule 2	Dangerous	Decreasing	Medium	LD
Rule 3	Dangerous	Decreasing	Large	LD
Rule 4	Dangerous	Stable	Small	LD
Rule 5	Dangerous	Stable	Medium	SD
Rule 6	Dangerous	Stable	Large	SD
Rule 7	Dangerous	Increasing	Small	LD
Rule 8	Dangerous	Increasing	Medium	SD
Rule 9	Dangerous	Increasing	Large	SD
Rule 10	Low	Decreasing	Small	NC
Rule 11	Low	Decreasing	Medium	SD
Rule 12	Low	Decreasing	Large	SD
Rule 13	Low	Stable	Small	NC
Rule 14	Low	Stable	Medium	NC
Rule 15	Low	Stable	Large	NC
Rule 16	Low	Increasing	Small	NC
Rule 17	Low	Increasing	Medium	NC
Rule 18	Low	Increasing	Large	SI
Rule 19	Safe	Decreasing	Small	SI
Rule 20	Safe	Decreasing	Medium	SI
Rule 21	Safe	Decreasing	Large	NC
Rule 22	Safe	Stable	Small	SI
Rule 23	Safe	Stable	Medium	SI
Rule 24	Safe	Stable	Large	LI
Rule 25	Safe	Increasing	Small	LI
Rule 26	Safe	Increasing	Medium	LI
Rule 27	Safe	Increasing	Large	LI

LD = large decrease; SD = small decrease; NC = no change; SI = small increase; LI = large increase.

When B/τ falls into low zone, the FLC mainly prioritizes maintaining the current bitrate or making gradual adjustments. First, the controller examines whether the buffer level is decreasing or remaining stable. If the buffer level decreases, and there is a danger of buffer level falling back into the danger zone (*Rule 11*, *Rule 12*), the bitrate is decreased. Otherwise, the bitrate remains unchanged (*Rule 10*). If the buffer level increases, and the rate of change of buffer is large, the controller increases the bitrate (*Rule 18*). Otherwise, the bitrate is maintained to minimize the frequency of bitrate switching (*Rule 13* to *Rule 17*).

When buffer level enters the safe zone, the risk of playback interruption is minimal. If there is a large drop in buffer level, the bitrate is maintained to minimize frequent changes in the bitrate (R_{21}); otherwise, the bitrate is gradually increased at the expense of buffer level (*Rule 19*, *Rule 20*). If $\Delta B/\tau$ is stable, the bitrate is increased gradually (*Rule 22*, *Rule 23*),

whereas if $\Delta B/\tau$ is increasing, the controller focuses on quickly increasing the bitrate (*Rule 25 to Rule 27*).

The center of sum method is used to de-fuzzify the output, O, to a crisp value according to

$$O = \frac{\sum_j A_j \times \varphi_j}{\sum_j A_j} \tag{7}$$

where A_j represents the areas of the respective output membership functions, φ_j represents the center of the area, and j denotes the number of output membership functions.

As explained earlier, the crisp output of FLC, O, denotes the bitrate of the upcoming segment in relation to the throughput. The algorithm's control decisions are limited because finite bitrates are available at the server. Therefore, the algorithm picks a bitrate according to the crisp output as follows:

$$R_s = \max\{r \in R\} < O \times T_S \tag{8}$$

where T_s is the weighted throughput:

$$T_S^{k+1} = (1 - \delta) \times T_S^k + \delta \times T^k \tag{9}$$

where value of the weighting factor δ lies within the range of 0 to 1.

Every client communicates the recommended bitrate, R_s, to the edge cloud to jointly allocate the bitrates to the HTTP clients.

3.3. Joint Optimization Problem

The objective of the HAS algorithm is to optimize the user experience to achieve long-term engagement [28,39]. The problem of maximizing utility (joint QoE optimization) is formulated as an integer non-linear programming (INLP) optimization model.

$$\text{Maximize } U_j = \alpha \times Q_j - \beta \times QS_j - \varphi \times F - \theta \times IE \tag{10}$$

Subject to

$$T^k = \frac{1}{t^{k+1} - t^k + \Delta t^k} \int_{t^k - \Delta t^k}^{t^{k+1}} T_t dt \tag{11}$$

$$\sum_{j=1}^{N} x_{ij}^{(t)} \geq 0, \ \forall\, 1 \leq i \leq m \tag{12}$$

$$\sum_{j=1}^{N} x_{ij}^{(t)} \times W_j^{(t)} \leq BW, \ \forall\, 1 \leq i \leq N \tag{13}$$

$$0 \leq B_j^{(t)} \leq B_j^{max} \tag{14}$$

$$R_{ij}^{(t)} \leq R_s, \ \forall\, 1 \leq i \leq m, 1 \leq j \leq N \tag{15}$$

$$R_{ij}^{(t)} \in R, \ \forall\, 1 \leq i \leq m, 1 \leq j \leq N \tag{16}$$

We establish four weighting parameters, namely α, β, φ, and θ, each within the range of 0–1 (subject to the constraint $\alpha + \beta + \varphi + \theta = 1$). These parameters are used to regulate the respective bitrates, bitrate changes, fair bitrate allocation, and equitable distribution of bandwidth among video clients. The values of the weighting parameters are set to $\alpha = 0.4$, $\beta = 0.4$, $\varphi = 0.1$, and $\theta = 0.1$. More weightage is given to bitrate and bitrate switches since they affect the QoE. The decision variable $x_i^{(t)}$ defines the number of video clients downloading video bitrate at time t. The only decision variables here are integer variables $x_i^{(t)}$ and R_{ij}^k. The variable B_j^k is dependent on the decision variables. The values of the remaining variable are predetermined.

Objective function (10) aims to jointly optimize QoE, fairness, and the bandwidth utilization given the available throughput during the streaming session $\{T^k, t \in [t^1, t^{k+1}]\}$. Constraint (12) specifies that more than one client can concurrently download the video content. Constraint (13) guarantees that the cumulative bandwidth allocated to the clients does not surpass the total bandwidth available at the base station. Constraint (14) ensures uninterrupted playback for the video clients throughout the streaming session. Constraint (15) guarantees that the bitrates assigned to the clients by the MEC do not surpass the recommended bitrate, R_s, from the client. And finally, constraint (16) indicates that the discrete bitrates allocated to the HAS clients by MEC belong to the set $R = \{R_1, R_2, R_3, \ldots, R_m\}$, available at the server.

3.4. Online Optimization Algorithm

This section presents the online optimization algorithm for solving the joint adaptation problem at the edge. The presence of integer decision variables, as discussed in Section 3.3, makes it computationally impractical to tackle the problem through a brute force method. The reason is that the complexity increases exponentially as the number of clients increases, rendering it unfeasible for large-scale HAS scheduling.

The pseudo-code for the proposed algorithm is provided in Algorithm 1. The algorithm picks the *ith* bitrate from the set R for the upcoming segment, represented as R_{next}. The bitrate picked for the previous segment is labeled as R_{prev}. For the first segment, where clients lack information about the available resources, and the buffer is empty, the system opts for the lowest attainable bitrate. In the subsequent segments, bitrates are determined using the MEC-assisted adaptation algorithm. As explained in Section 3.2, the client shares with edge cloud the highest bitrate it can stream, according to the throughput and buffer level. The online adaptation algorithm (Subroutine 1) takes into account three predetermined threshold values, δ_s, δ_F, and δ_{IE}, for the switching level, fairness, and the bandwidth inefficiency index, respectively. δ_s and δ_{IE} are computed as follows:

$$\delta_s = |\, max\,\{r \in R\} < T^k - max\,\{r \in R\} < T^{k-1}\,|$$

$$\delta_{IE} = |max\,\{r \in R\} < \; T^k - T^k\,|$$

where $max\,\{r \in R\} < T_k$ represents the highest bitrate in set R that is less than the throughput observed during the download of the *kth* segment. R_{max} and R_{min} are the highest and the lowest available bitrates in the set R, respectively. The switching index linked to the chosen bitrate is calculated as $|r - R_{prev}|$, and the bandwidth inefficiency index is calculated as $|r - T^k|$. The fairness threshold, δ_F, is set to 0.6, and the fairness index is calculated as $1 - (r - R_{avg})/(R_{max} - R_{min})$, yielding a value between 0 and 1.

Subroutine 1 initially evaluates all the bitrates less than R_s that meet the criteria for bitrate switching, fairness, and bandwidth inefficiency thresholds (lines 3–5). From the pool of bitrates that meet these criteria, the algorithm chooses the most suitable bitrate with the highest achievable utility objective value for the current segment (lines 6–9). If there is no bitrate available to meet the required conditions, first, the fairness condition is bitrate that optimizes the utility objective function is assigned to the clients for the upcoming segment (line 14). If no such bitrate is available, bandwidth inefficiency is compromised next. The algorithm checks all the bitrates that only satisfy the switching condition (lines 16–18). The bitrate that maximizes the utility value is selected for streaming for the upcoming segment (line 19). Finally, if none of the bitrates $\leq R_s$ meet the switching condition, the bitrate with the highest objective value is downloaded for the upcoming segment (lines 21–22).

Algorithm 1: MEC-assisted Adaptation

Input: N: Number of DASH clients, R: set of available discrete bitrates, t_{dur}: video duration

Output: Binary allocations, x_{ij}, and integer bitrate allocation, r_{ij}, for each client, $1 \leq j \leq N$

For each client $1 \leq j \leq N$ ***do***

$maxUtility = -\infty$

$R_{next} = 0$

If *First Segment = = True*

$R_{next} = R_{min}$

Else If *First Segment = = False*

Run Online Adaptation Algorithm

If t_{dur} *= = End of Streaming Session* ***then***

$Utility = Utility + maxUtility$

Subroutine 1: Online Adaptation Algorithm

1: Update $B_i(t)$

2: Compute $\delta_S, \delta_{IE}, \delta_F, r_{avg}$

3: ***For*** each bitrate $r \in R$ in decreasing order

4: ***If*** allocation of r satisfies (12) *and* $r \leq R_s$

5: ***If*** $|r - R_{prev}| \leq \delta_s$ *and* $1 - \frac{r - r_{avg}}{R_{max} - R_{min}} > \delta_F$ *and* $|r - T_i| \leq \delta_{IE}$

6: $U = \delta \times r - \beta \times |r - R_{prev}| - \varphi \times |r - r_{avg}| - \mu \times |r - T_i|$

7: **If** $U > maxUtility$

8: $maxUtility = U$

9: $R_{next} = r$

End For

10: ***If*** $R_{next} = 0$

11: ***For***each bitrate $r \in R$ in decreasing order

12: ***If*** allocation of r satisfies (12) *and* $r \leq R_s$

13: ***If*** $|r - R_{prev}| \leq \delta_S$ *and* $|r - T_i| \leq \delta_{IE}$

14: Perform operations in lines 6–9

End For

15: ***If*** $R_{next} = 0$

16: ***For*** each bitrate $r \in R$ in decreasing order

17: ***If*** allocation of r satisfies (12) *and* $r \leq R_s$

18: ***If*** $|r - T_i| \leq \delta_{IE}$

19: Perform operations in lines 6–9

End For

20: ***If*** $R_{next} = 0$

21: ***For*** each bitrate $r \in R \leq R_s$ in decreasing order

22: Perform operations in lines 6–9

End For

23: Update weighting parameters $\rho, \beta, \varphi,$ *and* θ

24: Compute Video Quality, Switching, QoE, Fairness, and Utility Function according to (1), (2), (3), (4), (5), (9)

25: Update Buffer Level

26: Return U_i

3.5. Computational Complexity

The calculation of observed throughput during the streaming of each segment requires $O(\tau)$ time. The computation of switching and bandwidth inefficiency thresholds (δ_s and δ_{IE}) takes $O(\mid R \mid)$ units of time. Similarly, the execution of the MEC-assisted adaptation algorithm leads to the computational complexity of $O(\tau + \mid R \mid)$. Combining all the aforementioned elements gives N video clients an overall time complexity of $O(N\ (\tau + \mid R \mid))$. Therefore, the overall complexity for the proposed heuristic algorithm remains in polynomial time.

4. Performance Evaluation

We implemented the MEC scenario as depicted in Figure 1 by using the simulation software ns-3. Table 3 shows the two video sequences used for evaluation: Tears of Steel (Dataset 1) and Big Buck Bunny (Dataset 2). Both videos are animations, with *Big Buck Bunny* having a duration of 9 min 56 s and *Tears of Steel* approximately 10 min. The encoding ladder spans multiple resolutions: bitrates below 150 kbps are encoded at 320 × 240, those between 150 kbps and 500 kbps at 480 × 360, between 500 kbps and 700 kbps at 854 × 480, between 700 kbps and 2 Mbps at 1280 × 720, and above 2 Mbps at 1920 × 1080. We used the trace file from [40,41] to create the video segments. We adopted the algorithms proposed in ECAAS [23], FDASH [16], ECAA [21], and FQAA [18] as benchmarks to show the effectiveness of the proposed scheme. Table 4 provides an overview of the characteristics of the HAS algorithms. ECAA and ECAAS are MEC-assisted algorithms that assign bitrates based on information available at edge cloud and do not run any adaptation algorithm at the client end. In the case of the proposed hybrid scheme, the MEC and clients collaborate to determine the video streaming rates. The proposed scheme employs a fuzzy logic controller at the client end to determine the upper limit for the video streaming rate. Based on the input from the clients, the proposed MEC-assisted online adaptation algorithm runs an optimization model to improve the QoE of video clients while also selecting bitrates fairly and efficiently utilizing network resources. FDASH and FQAA are algorithms based on fuzzy logic that run only on the video client's end. Table 5 presents the experiments used to analyze algorithms. For each configuration, the experiment was conducted 10 times, and the average results are reported. Fairness is quantified by Jain's fairness index [42], and bandwidth inefficiency is computed using $\frac{|\sum_j R_{ij}^{(t)} - W|}{W}$.

Table 3. Datasets.

Dataset	Name	Genre	Codec	Video Length	Bitrates
1	Tears of Steel	Animation	AVC	10:00	184, 380, 459, 693, 1270, 1545, 2000, 2530, 3750, 5379, 7861, and 11,321 kbps
2	Big Buck Bunny	Animation	AVC	9:56	45, 88, 128, 177, 217, 255, 323, 378, 509, 577, 782, 887, 1008, 1207, 1473, 2087, 2409, 2944, 3340, 3613, and 3936 kbps

Table 4. Properties of HAS Algorithms.

Algorithm	Adaptation Setting	Parameters Observed
Proposed	Hybrid MEC and client-based adaptation	B/τ, *rho*, $\Delta B/\tau$
ECAA	MEC-assisted	Buffer level, Throughput

Table 4. *Cont.*

Algorithm	Adaptation Setting	Parameters Observed
ECAAS	MEC-assisted	Buffer level, Throughput
FDASH	Client-based	ΔB, B
FQAA	Client-based	ΔB, B

Table 5. Experiment settings used to evaluate the algorithms.

Dataset	Buffer Size	Segment Duration	Mobility	No. of Clients
1	15 s	2 s	75 km/h	10
1	30 s	4 s	75 km/h	10
1	60 s	4 s	75 km/h	10
2	15 s	2 s	75 km/h	10
2	60 s	4 s	75 km/h	10

4.1. Dataset 1

First, we analyze the algorithms when downloading segments from dataset 1. As demonstrated in Table 5, ten clients, all traveling at vehicular speed, are concurrently streaming the video. Most algorithms employ fixed control rules for bitrate selection, resulting in inconsistent performance across varying settings [10]. Therefore, we analyze how the algorithms perform across different client and server configurations, as given in Table 5. In the first experiment, buffer size and segment duration are configured to be 15 and 2 s, respectively. In the second experiment, the buffer size and segment duration are adjusted to 30 and 4 s, respectively. In the third experiment, the buffer size increased to 60 s.

Table 6 illustrates the performance of the algorithms when using dataset 1. Table 6 shows that the proposed ECAAS and ECAA algorithms maintained high bitrate irrespective of the segment duration or buffer size. However, the ECAAS and ECAA algorithms attained a high bitrate at the cost of experiencing a high frequency of bitrate changes. The proposed algorithm aims to balance between achieving high QoE, maintaining fairness among the clients, and high utilization of bandwidth. This resulted in a slightly greater number of bitrate switches when compared to client-side fuzzy-based algorithms (FDASH and FQAA), which solely aim to achieve high QoE. The FDASH and FQAA had a low number of bitrate switches, but they achieved a low average bitrate. Figure 6a,b illustrate the response of the proposed and FDASH algorithms to throughput changes, respectively. Figure 6 depicts that the proposed algorithm maintains consistent video rates during minor variations in the throughput but abruptly adjusts the bitrate when faced with significant fluctuations. On the other hand, the FDASH algorithm maintains consistent bitrate even under large fluctuations. The reason is that the FDASH algorithm considers the average throughput of the last 60 s to adapt the bitrate. Although the algorithm effectively minimizes bitrateS fluctuations, it increases the risk of playback interruptions as well. Moreover, it also exhibits delays in improving video quality as the throughput increases.

Table 6 also provides a comparison of the average number of interruptions per client and buffering time, respectively. The average interruptions quantify the number of times a client experienced rebuffering during the streaming session. Table 6 illustrates that the FDASH algorithm experiences a high number of interruptions when the buffer size is small. As the buffer size increases, the rebuffering ratio decreases. As explained earlier, the FDASH and FQAA algorithms take the average throughput of the last 60 s to predict the throughput for the upcoming segment, and this leads to the algorithms reacting late

to the changes in the throughput. This increases the probability of playback stalling. In contrast, the proposed algorithm does not experience any interruption regardless of the buffer size. In the proposed scheme, the fuzzy-based bitrate suggestion to the edge cloud ensures that the likelihood of rebuffering is minimized. The FLC observes the buffer level, the variations in the buffer level, and the degree of changes in the throughput to ensure that the selected bitrate does not lead to buffer underflow. The MEC-assisted ECAA and ECAAS algorithms also experience playback interruptions. Both the ECAA and ECAAS algorithms select the maximum suitable bitrate for the upcoming segment according to the following condition: max ($R^{thr\ max}$ and $R^{buff\ max}$), where $R^{thr\ max}$ is the maximum bitrates less than the segment throughput ($R^{thr\ max} = max\ (R_i{}^k < T^k)$, and $R^{buff\ max}$ is the maximum bitrate that does not lead to rebuffering given the current buffer level and throughput. $R^{buff\ max}$ is calculated using the following equation:

$$B^{k+1} = B^k + \tau - \left(\tau \times \frac{R_i^k}{T^k}\right) \tag{17}$$

where B^k is the buffer level at the end of the download of the kth segment, and τ is the segment duration. If the segment duration is 4 s, the available buffer at the start of the download of the $(k+1)^{th}$ segment is also 4 s, and the current segment throughput is 1000 kbps. Downloading dataset 1, according to Equation (17), $R^{buff\ max}$ will be 1545 kbps, i.e., the highest bitrate that does not result in rebuffering. However, if the throughput during the download of the $(k+1)^{th}$ segment drops to 500 kbps, the bitrate of 1545 kbps will lead to rebuffering. In an unstable environment where the throughput fluctuates during the streaming session, this approach is unable to prevent rebuffering. As the segment duration increases, the likelihood of playback interruptions due to sudden drops in throughput also increases. Figures 7 and 8 show the adaptive behavior of the proposed and ECAA algorithm to the variations in the throughput, respectively. In Figure 8, the ECAA algorithm experiences playback interruption during the download of the 13th segment. As explained above, the ECAA and ECAAS algorithms first calculate max ($R^{thr\ max}$ and $R^{buff\ max}$) to pick the highest bitrate that does not lead to rebuffering given the current throughput and buffer level. For the 13th segment, $R^{thr\ max}$ and $R^{buff\ max}$ are 1540 and 2000 kbps, respectively. The joint adaptation algorithm at the edge cloud analyzes all bitrates less than or equal to 2000 kbps and selects 1540 kbps as the bitrate that offers the highest quality, fairness, and bandwidth efficiency. However, as the throughput dropped further during the download of the segment, the client experienced rebuffering. Since the joint adaptation algorithm does not prioritize minimizing rebuffering, this highlights the necessity for a more effective mechanism to minimize rebuffering and foster collaboration between the client and edge cloud. To this end, the proposed algorithm utilizes a fuzzy logic controller to aid the joint adaptation algorithm in recommending the highest feasible bitrate that reduces the likelihood of playback interruption.

Table 6 shows the QoE of the algorithms using the quality objective model given in Equation (3). Table 6 illustrates that the proposed algorithm achieves high user experience in all three experiments. Video bitrate and playback interruptions have the greatest impact on QoE [28]. The proposed algorithm maintains a high bitrate while preventing the buffer level from depleting. The ECAAS algorithm also achieves a high bitrate, but its QoE drops due to the high number of bitrate switches and rebuffering. The ECAA algorithm achieves the highest QoE when buffer size and segment duration are configured to 15 s and 2 s, respectively. However, its QoE decreases when buffer size and segment duration are increased, as this leads to more rebuffering events and a reduction in video bitrate. The

FQAA and FDASH algorithms result in poor QoE, as the clients experience a high number of rebuffering events and download low-quality segments.

Table 6. Performance of the algorithms when the client downloaded dataset 1.

		Proposed	ECAAS	ECAA	FDASH	FQAA
Average Video Rate	τ = 2s, Bmax = 15s	1332.00	1340.23	1462.57	1196.86	1100.00
	τ = 4s, Bmax = 30s	1350.00	1382.83	1246.00	1328.39	1200.00
	τ = 4s, Bmax = 60s	1280.00	1386.93	1357.00	1236.42	1125.00
Switching Ratio	τ = 2s, Bmax = 15s	0.31	0.57	0.34	0.05	0.05
	τ = 4s, Bmax = 30s	0.39	0.72	0.49	0.09	0.09
	τ = 4s, Bmax = 60s	0.30	0.77	0.34	0.10	0.10
Fairness	τ = 2s, Bmax = 15s	0.88	0.89	0.88	0.85	0.84
	τ = 4s, Bmax = 30s	0.89	0.86	0.86	0.84	0.85
	τ = 4s, Bmax = 60s	0.89	0.88	0.88	0.84	0.84
Inefficiency	τ = 2s, Bmax = 15s	0.15	0.15	0.10	0.16	0.17
	τ = 4s, Bmax = 30s	0.15	0.17	0.34	0.16	0.15
	τ = 4s, Bmax = 60s	0.15	0.19	0.11	0.25	0.22
QoE	τ = 2s, Bmax = 15s	1150.00	962.84	1208.49	834.32	850.00
	τ = 4s, Bmax = 30s	1116.00	648.16	617.00	511.92	725.00
	τ = 4s, Bmax = 60s	1088.00	382.66	1058.00	986.89	1000.00
Average Interruption	τ = 2s, Bmax = 15s	0.00	0.00	0.20	4.70	2.10
	τ = 4s, Bmax = 30s	0.00	3.50	2.80	5.30	1.12
	τ = 4s, Bmax = 60s	0.00	5.00	1.80	1.63	0.00
Buffering Time	τ = 2s, Bmax = 15s	0.00	0.00	0.50	7.90	3.69
	τ = 4s, Bmax = 30s	0.00	5.80	7.80	15.60	5.23
	τ = 4s, Bmax = 60s	0.00	8.60	2.20	20.30	0.00

The proposed scheme also attains high fairness and efficiently utilizes network resources across all experiments, followed by ECAA and ECAAS. The proposed online algorithm ensures that the selected bitrates maximize the objective function that aims to jointly enhance fairness and bandwidth utilization. The ECAA and ECAAS algorithms also consider fairness and bandwidth utilization during video selection process. Due to rebuffering events, the algorithms are compelled to select bitrates that result in a slight degradation of fairness and efficiency values. When the buffer level approaches zero, the algorithms prioritize protecting the buffer over fairness and bandwidth efficiency, based on the condition *max* ($R^{thr\ max}$ and $R^{buff\ max}$). If we analyze dataset 1, we can observe that the bitrates are more widely spaced, particularly at the higher bitrates. Given the distribution of bitrates in the dataset, a forced drop in bitrate due to the increasing risk of playback interruption impacts fairness. The client-based algorithms achieve low fairness and bandwidth efficiency value since each client is selecting bitrate oblivious to other clients. Neither FDASH nor FQAA consider fairness and network efficiency during their adaptation process. Each greedy client is aiming to achieve the highest feasible bitrate according to the network conditions irrespective of the decision made by other clients.

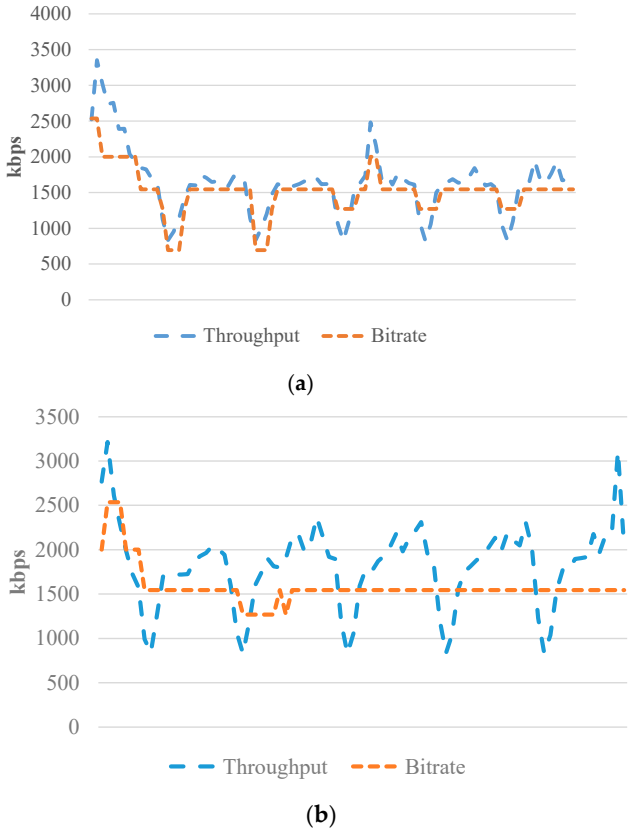

Figure 6. Response of the algorithms to throughput fluctuations. (**a**) Proposed; (**b**) FDASH.

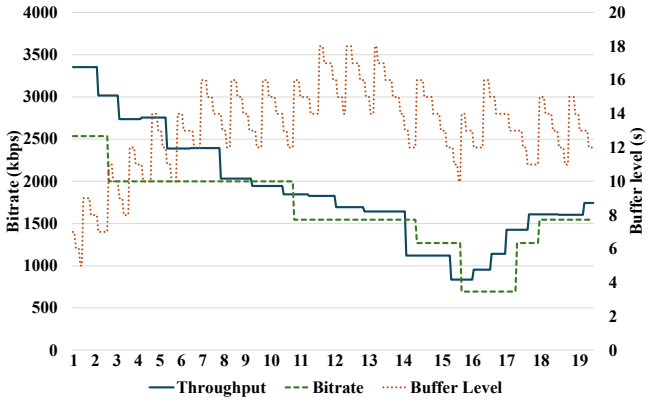

Figure 7. Response of the proposed method to fluctuations in the throughput.

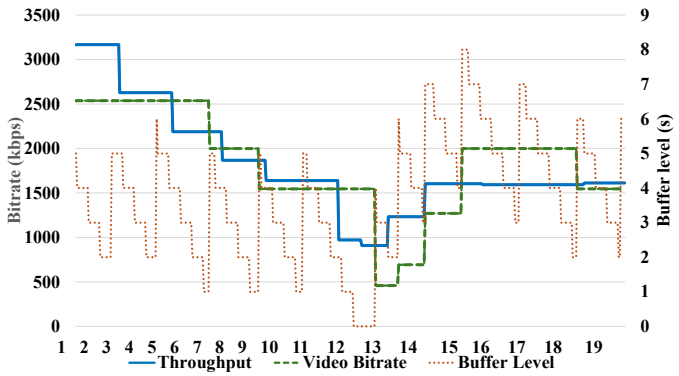

Figure 8. Response of the ECAA algorithm to variations in the throughput.

4.2. Dataset 2

Next, we analyze the algorithms when downloading segments from dataset 2 given in Table 3. In the fourth experiment, buffer size and segment duration are configured to 15 and 2 s, respectively. In the final experiment, the buffer size and segment duration are set to 60 and 4 s, respectively.

Table 7 reveals that the MEC-assisted algorithms download high-quality segments compared to the fuzzy-based algorithms. The proposed algorithm achieves an average bitrate comparable to that of the ECAAS and ECAA algorithms while experiencing fewer bitrate switches. Table 7 illustrates that only the proposed algorithm avoided experiencing any playback rebuffering. The ECAAS algorithm achieved a high bitrate at the cost of encountering playback rebuffering. Similarly, the client-based algorithms also experienced playback stalling. Table 7 demonstrates that when the segment duration increased to 4 s, despite an increase in buffer size, the algorithms encountered a higher frequency of rebuffering. The reason is that a longer segment duration increases the probability of rebuffering when there is a discrepancy between the chosen bitrate and the available throughput.

Table 7. Performance of the algorithms when the client downloaded dataset 2.

		Proposed	ECAAS	ECAA	FDASH	FQAA
Average Video Rate	τ = 2s, Bmax = 15s	1346.00	1331.00	1423.00	1274.00	1099.00
	τ = 4s, Bmax = 60s	1238.00	1472.00	1256.00	1138.00	1055.00
Switching Ratio	τ = 2s, Bmax = 15s	0.40	0.70	0.32	0.07	0.11
	τ = 4s, Bmax = 60s	0.40	0.67	0.59	0.14	0.18
Fairness	τ = 2s, Bmax = 15s	0.91	0.91	0.87	0.87	0.85
	τ = 4s, Bmax = 60s	0.91	0.92	0.88	0.85	0.84
Inefficiency	τ = 2s, Bmax = 15s	0.15	0.11	0.15	0.30	0.33
	τ = 4s, Bmax = 60s	0.16	0.10	0.26	0.28	0.28
QoE	τ = 2s, Bmax = 15s	1274.00	987.00	1304.00	1112.00	990.00
	τ = 4s, Bmax = 60s	1100.00	816.00	701.00	963.00	975.00
Average Interruption	τ = 2s, Bmax = 15s	0.00	1.50	1.00	2.30	1.20
	τ = 4s, Bmax = 60s	0.00	3.55	1.22	4.00	2.00
Buffering Time	τ = 2s, Bmax = 15s	0.00	1.60	1.75	3.70	3.50
	τ = 4s, Bmax = 60s	0.00	4.17	3.20	1.52	1.23

The proposed scheme consistently attains the highest QoE irrespective of the segment duration and the buffer size. The proposed algorithm was able to download high-quality segments without experiencing any playback interruption. The ECAAS algorithm streamed high-bitrate segments but also encountered frequent bitrate switches and playback rebuffering. The ECAA algorithm attained high QoE when buffer size and segment duration were set to 15 s and 2 s, respectively. However, with the increase in segment duration, the QoE deteriorated due to the high number of bitrate switches and rebuffering. The client-based algorithms were effective in reducing bitrate switches, but this came at the cost of video quality and rebuffering.

Table 7 also reveals that the proposed and ECAAS algorithms achieve both high fairness and efficiently utilize network resources. The bitrates in dataset 2 are more closely spaced than those in dataset 1. As a result, we can observe that algorithms achieve greater fairness and bandwidth utilization when streaming dataset 2 compared to dataset 1. In pursuit of maximizing fairness and optimizing bandwidth utilization, the ECAAS algorithm

makes a trade-off by compromising on video quality. The ECAAS algorithm experiences a high number of bitrate switches and playback interruptions, degrading the QoE. The proposed algorithm effectively utilizes available bandwidth, allocates video quality fairly among clients, and ensures a high QoE. The client-based algorithms solely prioritize improving the user experience of individual clients. Consequently, their utilization of bandwidth is suboptimal, and due to a lack of awareness regarding other clients' decisions, they unfairly select the bitrates for each client.

4.3. Summary

Table 8 provides an overview of the adaptation algorithms' average performance throughout all experiments, highlighting that the proposed scheme attained the highest QoE and fairness while also ensuring the efficient utilization of available bandwidth. Additionally, the proposed method successfully prevented playback interruption in all experiments.

Table 8. Performance of Algorithms Averaged Over All Experiments.

	Proposed	ECAAS	ECAA	FDASH	FQAA
QoE	1145.6	759.3	977.6	881.6	908
Fairness	0.9	0.89	0.87	0.85	0.84
Inefficiency	0.15	0.15	0.19	0.23	0.23

In the summary below, consecutive numbers correspond to the results compared to ECAA, FDASH, FQAA, and ECAAS, respectively. The proposed algorithm did the following:

(1) Increased the QoE by 17%, 30%, 26%, and 51%;
(2) Increased bandwidth efficiency by 2.9%, 6.25%, 6.8%, and 1.1%;
(3) Outperformed other algorithms in fairness by 21%, 34.7%, and 34.7% while achieving similar efficiency to the ECAAS algorithm.

5. Conclusions

This work introduces a hybrid MEC- and client-assisted framework for adaptive streaming to enhance users' engagement, network utilization, and the equitable allocation of bitrates among clients through collaboration between the edge and client. The proposed framework employs fuzzy logic at the client to determine the streaming rate upper limit for the clients. We also formulated the problem of jointly allocating bitrates to the clients as an integer nonlinear programming (INLP) optimization model subject to the network's bandwidth constraints. In addition, we proposed a heuristic approach to address the time complexity of solving the INLP model. The results showed that the proposed approach improves performance by an average of 30% in QoE, 4.2% in bandwidth efficiency, and 22.6% in fairness compared to state-of-the-art algorithms. Additionally, the findings indicate that the proposed approach effectively prevented any playback interruptions under different client and server settings.

In the future, a further expansion of this framework could involve coordination between multiple edge nodes for load balancing. At present, the system focuses on client–MEC interaction within a single node, but edge servers often operate in clusters with varying workloads. Extending the framework to allow cooperative resource sharing and handover mechanisms across edge nodes would improve system scalability and enhance robustness under highly dynamic traffic conditions.

Funding: This research received no external funding.

Data Availability Statement: Dataset available on request from the author.

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

References

1. *ISO/IEC 23009-1:2012*; Information Technology—Dynamic Adaptive Streaming Over HTTP (DASH)—Part 1: Media Presentation Description and Segment Formats. International Organization for Standardization: Geneva, Switzerland, 2012.
2. Li, Z.; Zhu, X.; Gahm, J.; Pan, R.; Hu, H.; Begen, A.C.; Oran, D. Probe and adapt: Rate adaptation for HTTP video streaming at scale. *IEEE J. Sel. Areas Commun.* **2014**, *32*, 719–733. [CrossRef]
3. Huang, T.Y.; Nikhil, H.; Brandon, H.; Nick, M.; Ramesh, J. Confused, timid, and unstable: Picking a video streaming rate is hard. In Proceedings of the 2012 Internet Measurement Conference, Boston, MA, USA, 14–16 November 2012; pp. 225–238. [CrossRef]
4. Akhshabi, S.; Anantakrishnan, L.; Begen, A.C.; Dovrolis, C. What happens when HTTP adaptive streaming players compete for bandwidth? In Proceedings of the 22nd International Workshop on Network and Operating System Support for Digital Audio and Video, Toronto, ON, Canada, 7–8 June 2012; pp. 9–14. [CrossRef]
5. Tran, T.X.; Hajisami, A.; Pandey, P.; Pompili, D. Collaborative mobile edge computing in 5G networks: New paradigms, scenarios, and challenges. *IEEE Commun. Mag.* **2017**, *55*, 54–61. [CrossRef]
6. Yi, S.; Hao, Z.; Zhang, Q.; Zhang, Q.; Shi, W.; Li, Q. LAVEA: Latencyaware video analytics on edge computing platform. In Proceedings of the Second ACM/IEEE Symposium on Edge Computing, San Jose, CA, USA, 12–14 October 2017; pp. 2573–2574.
7. Wang, D.; Peng, Y.; Ma, X.; Ding, W.; Jiang, H.; Chen, F.; Liu, J. Adaptive wireless video streaming based on edge computing: Opportunities and approaches. *IEEE Trans. Serv. Comput.* **2019**, *12*, 685–697. [CrossRef]
8. Li, Y.; Frangoudis, P.A.; Hadjadj-Aoul, Y.; Bertin, P. A mobile edge computing-assisted video delivery architecture for wireless heterogeneous networks. In Proceedings of the IEEE Symposium on Computers and Communications, Heraklion, Crete, Greece, 3–6 July 2017; pp. 534–539.
9. Rahman, W.U.; Hong, C.S.; Huh, E.-N. Edge Computing Assisted Joint Quality Adaptation for Mobile Video Streaming. *IEEE Access* **2019**, *7*, 129082–129094. [CrossRef]
10. Rahman, W.U.; Amin, M.B.; Hossain, M.D.; Seon Hong, C.; Huh, E.-N. QoE optimization for HTTP adaptive streaming: Performance evaluation of MEC-assisted and client-based methods. *J. Vis. Commun. Image R.* **2022**, *82*, 103415. [CrossRef]
11. Hung, L.T.; Ngoc, N.P.; Truong, C.T. Bitrate adaptation for seamless on-demand video streaming over mobile networks. *Signal Process. Image Commun.* **2018**, *65*, 154–164. [CrossRef]
12. Park, J.; Kim, M.; Chung, K. Buffer-based rate adaptation scheme for HTTP video streaming with consistent quality. *Comput. Sci. Inform. Syst.* **2021**, *18*, 1139–1157. [CrossRef]
13. Rahman, W.U.; Huh, E.N. Content-aware QoE optimization in MEC-assisted Mobile video streaming. *Multimedia Tools and Applications.* **2023**, *82*, 42053–42085. [CrossRef]
14. Huang, W.; Zhou, Y.; Xie, X.; Wu, D.; Chen, M.; Ngai, E. Buffer State is Enough: Simplifying the Design of QoE-Aware HTTP Adaptive Video Streaming. *IEEE Trans. Broadcast.* **2018**, *64*, 590–601. [CrossRef]
15. Karn, N.K.; Zhang, H.; Jiang, F.; Yadav, R.; Laghari, A.A. Measuring bandwidth and buffer occupancy to improve the QoE of HTTP adaptive streaming. *IEEE Signal Process. Image Commun.* **2019**, *13*, 1367–1375. [CrossRef]
16. Vergados, D.J.; Michalas, A.; Sgora, A.; Vergados, D.D.; Chatzimisios, P. FDASH: A Fuzzy-Based MPEG/DASH Adaptation Algorithm. *IEEE Syst. J.* **2016**, *10*, 859–868. [CrossRef]
17. Sobhani, A.; Yassine, A.; Shirmohammadi, S. A fuzzy-based rate adaptation controller for DASH. In Proceedings of the 25th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video, Portland, OR, USA, 18–20 March 2015.
18. Mowafi, M.; Taqieddin, E.; Al-Dahoud, H. Energy efficient fuzzy-based DASH adaptation algorithm. *Digit. Commun. Netw.* **2021**, *7*, 113–119. [CrossRef]
19. Rahman, W.U.; Hossain, M.D.; Huh, E.-N. Fuzzy-Based Quality Adaptation Algorithm for Improving QoE from MPEG-DASH Video. *Appl. Sci.* **2021**, *11*, 5270. [CrossRef]
20. Yang, S.-R.; Tseng, Y.-J.; Huang, C.-C.; Lin, W.-C. Multi-Access Edge Computing Enhanced Video Streaming: Proof-of-Concept Implementation and Prediction/QoE Models. *IEEE Trans. Veh. Technol.* **2019**, *68*, 1888–1902. [CrossRef]
21. Mehrabi, A.; Siekkinen, M.; Ylä-Jääski, A. Edge computing assisted adaptive mobile video streaming. *IEEE Trans. Mob. Comput.* **2018**, *18*, 787–800. [CrossRef]
22. Kim, M.; Chung, K. Edge Computing Assisted Adaptive Streaming Scheme for Mobile Networks. *IEEE Access* **2021**, *9*, 2142–2152. [CrossRef]
23. Yan, Z.; Xue, J.; Chen, C.W. Prius: Hybrid Edge Cloud and Client Adaptation for HTTP Adaptive Streaming in Cellular Networks. *IEEE Trans. Circuits Syst. Video Technol.* **2017**, *27*, 209–222. [CrossRef]

24. Kang, J.; Chung, K. RL-based HTTP adaptive streaming with edge collaboration in multi-client environment. *J. Netw. Comput. Appl.* **2024**, *223*, 103833. [CrossRef]
25. Seufert, M.; Egger, S.; Slanina, M.; Zinner, T.; Hobfeld, T.; Tran-Gia, P. A Survey on Quality of Experience of HTTP Adaptive Streaming. *IEEE Commun. Surv. Tutor. Tutor.* **2015**, *17*, 469–492. [CrossRef]
26. Dobrian, F.; Sekar, V.; Awan, A.; Stoica, I.; Joseph, D.; Ganjam, A.; Zhan, J.; Zhang, H. Understanding the impact of video quality on user engagement. *ACM SIGCOMM Comput. Commun. Rev.* **2011**, *41*, 362–373. [CrossRef]
27. Georgopoulos, P.; Elkhatib, Y.; Broadbent, M.; Mu, M.; Race, N. Towards network-wide QoE fairness using openflow-assisted adaptive video streaming. In Proceedings of the SIGCOMM Workshop on Future Human-Centric Multimedia Networking (FhMN), Hong Kong, China, 16 August 2013; pp. 15–20.
28. Alberti, C.; Renzi, D.; Timmerer, C.; Mueller, C.; Lederer, S.; Battista, S.; Mattavelli, M. Automated QoE evaluation of dynamic adaptive streaming over HTTP. In Proceedings of the 5th International Workshop on Quality of Multimedia Experience (QoMEX), Klagenfurt am Wörthersee, Austria, 3–5 July 2013; pp. 58–63.
29. Yamagishi, K.; Hayashi, T. Parametric quality-estimation model for adaptive-bitrate-streaming services. *IEEE Trans. Multimed.* **2017**, *19*, 1545–1557. [CrossRef]
30. Garcia, M.N.; Robitza, W.; Raake, A. On the accuracy of short-term quality models for long-term quality prediction. In Proceedings of the 7th International Workshop on Quality of Multimedia Experience (QoMEX), Costa Navarino, Greece, 26–29 May 2015; pp. 1–6.
31. Rodríguez, D.Z.; Rosa, R.L.; Alfaia, E.C.; Abrahão, J.I.; Bressan, G. Video quality metric for streaming service using DASH standard. *IEEE Trans. Broadcast.* **2016**, *62*, 628–639. [CrossRef]
32. Duanmu, Z.; Zeng, K.; Ma, K.; Rehman, A.; Wang, Z. A quality-of-experience index for streaming video. *IEEE J. Sel. Top. Signal Process* **2017**, *11*, 154–166. [CrossRef]
33. Rodriguez, D.Z.; Abrahao, J.; Begazo, D.C.; Rosa, R.L.; Bressan, G. Quality metric to assess video streaming service over TCP considering temporal location of pauses. *IEEE Trans. Consumer Electron.* **2012**, *58*, 985–992. [CrossRef]
34. Huynh-Thu, Q.; Ghanbari, M. Temporal Aspect of Perceived Quality in Mobile Video Broadcasting. *IEEE Trans. Broadcast.* **2008**, *54*, 641–651. [CrossRef]
35. Yin, X.; Jindal, A.; Sekar, V.; Sinopoli, B. A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP. *ACM SIGCOMM Comput. Commun. Rev.* **2015**, *45*, 325–338. [CrossRef]
36. Progressive Download and Dynamic Adaptive Streaming Over HTTP, Document 3GPP TS 26.247 V12.1.0. 2013. Available online: https://www.etsi.org/deliver/etsi_ts/126200_126299/126247/18.00.00_60/ts_126247v180000p.pdf (accessed on 5 March 2025).
37. Mamdani, E.H.; Assilian, S. An experiment in linguistic synthesis with a fuzzy logic controller. *Int. J. Man-Mach. Stud.* **1975**, *7*, 1–13. [CrossRef]
38. Petrangeli, S.; Hooft, J.V.D.; Wauters, T.; Turck, F.D. Quality of experience-centric management of adaptive video streaming services: Status and challenges. *ACM Trans. Multimed. Comput. Commun. Appl.* **2018**, *14*, 1–29. [CrossRef]
39. Lee, C.C. Fuzzy logic in control systems: Fuzzy logic controller. I. *IEEE Trans. Syst. Man Cybern.* **1990**, *20*, 404–418. [CrossRef]
40. DASHDataset2014. Available online: http://ftp.itec.aau.at/datasets/DASHDataset2014/ (accessed on 1 November 2023).
41. Lederer, S.; Müller, C.; Timmerer, C. Dynamic adaptive streaming over HTTP dataset. In Proceedings of the 3rd Multimedia Systems Conference, Chapel Hill, NC, USA, 22–24 February 2012.
42. Rajendra, J.; Chiu, D.; Hawe, W. *A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Computer System*; Technical Report; Digital Equipment Corporation: Maynard, MA, USA, 1984.