applied sciences

MDPI

*Article*

# Fuzzy-Based Quality Adaptation Algorithm for Improving QoE from MPEG-DASH Video

**Waqas ur Rahman, Md Delowar Hossain and Eui-Nam Huh ***

Department of Computer Science and Engineering, Kyung Hee University, Yongin-si 17104, Korea; waqasrahman@khu.ac.kr (W.u.R.); delowar@khu.ac.kr (M.D.H.)
* Correspondence: johnhuh@khu.ac.kr; Tel.: +82-31-201-3778

**Abstract:** Video clients employ HTTP-based adaptive bitrate (ABR) algorithms to optimize users' quality of experience (QoE). ABR algorithms adopt video quality based on the network conditions during playback. The existing state-of-the-art ABR algorithms ignore the fact that video streaming services deploy segment durations differently in different services, and HTTP clients offer distinct buffer sizes. The existing ABR algorithms use fixed control laws and are designed with predefined client/server settings. As a result, adaptation algorithms fail to achieve optimal performance across a variety of video client settings and QoE objectives. We propose a buffer- and segment-aware fuzzy-based ABR algorithm that selects video rates for future video segments based on segment duration and the client's buffer size in addition to throughput and playback buffer level. We demonstrate that the proposed algorithm guarantees high QoE across various video player settings and video content characteristics. The proposed algorithm efficiently utilizes bandwidth in order to download high-quality video segments and to guarantee high QoE. The results from our experiments reveal that the proposed adaptation algorithm outperforms state-of-the-art algorithms, providing improvements in average video rate, QoE, and bandwidth utilization, respectively, of 5% to 18%, about 13% to 30%, and up to 45%.

**Keywords:** fuzzy logic; adaptive bitrate; DASH; video streaming; QoE



**Citation:** 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. https://doi.org/10.3390/app11115270

Academic Editor: Ludmila Dymova

Received: 30 April 2021
Accepted: 3 June 2021
Published: 6 June 2021



## 1. Introduction

Multimedia content accounts for the majority of internet traffic. According to the Cisco Visual Networking Index, 82% of global mobile data traffic will be video by 2022 [1]. To handle traffic demand related to multimedia, HTTP adaptive streaming (HAS) solutions are often used.

Figure 1 shows a simple MPEG-DASH streaming scenario between a video server and a DASH client. In MPEG-DASH, the video content stored in the server is encoded at multiple video rates and fragmented into small segments of fixed duration, $\tau$. The video client starts the streaming session by asking for the Media Presentation Description (MPD) file from the server by using HTTP GET requests. The video client downloads the segments at the most suitable video rates available from the server. The adaptive bitrate (ABR) algorithm runs on the HTTP client and dynamically adjusts the video quality based on the network conditions. The objective of the ABR algorithm is to optimize the QoE of video clients. A comprehensive survey on QoE determined the factors that affect user experience [2]. These QoE objectives include selecting the highest set of video bitrates, avoiding unnecessary video bitrate changes, and avoiding playback interruptions. Playback interruptions and selection of video bitrates affect the user experience the most [3]. However, these video quality objectives are contradictory. On one hand, selecting the highest available video rate increases the risk of playback interruption; on the other hand, mitigating the risk of interruptions during playback means selecting lower-quality segments. Therefore, the selection of the most feasible video rate is a challenging task.

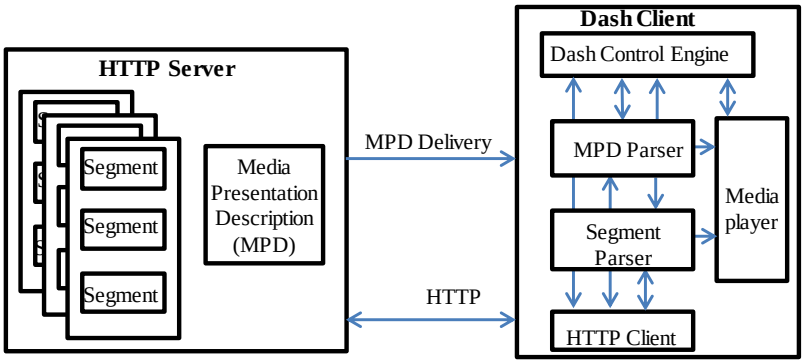



**Figure 1.** A simple MPEG-DASH streaming scenario.

The majority of existing algorithms employ fixed control rules to pick the video rate based on available throughput [4,5], the amount of video in the playback buffer [6], or a combination of the two [7–11]. These algorithms require significant tuning, and performance varies from one client or group of network settings to the other.

Video streaming services deploy segment durations ($\tau$) differently. Microsoft's Smooth Streaming and Adobe's HTTP Dynamic Streaming offer segment durations of two seconds and four seconds, respectively [12,13]. The client can buffer a larger number of smaller segments, compared to larger segments, as shown in Figure 2. More segments reduce the risk of playback interruption. Let us assume the current buffer level is four seconds, and the segment duration is two seconds. This means the total number of segments the buffer can hold is two. If the segment duration is four seconds, then only one segment 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. This shows that instead of observing only buffer level, clients should observe the ratio of the buffer level to the segment duration in order to adapt video quality. Moreover, using the smaller segment duration, the client has more opportunities to adapt the video rate, compared to using the larger duration. In a highly unstable network, the client could adjust the video rate more quickly by downloading smaller segments.

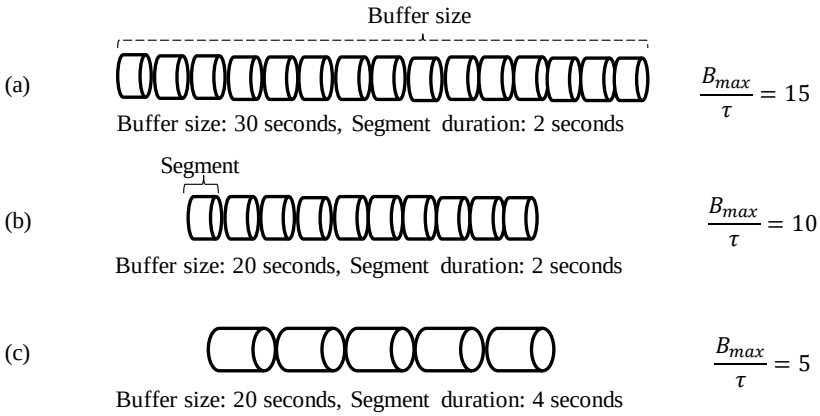



**Figure 2.** Comparing (**a**,**b**) shows that the larger buffer size allows the client to buffer more segments. Comparing (**b**,**c**) shows that longer segment durations decrease the number of segments that can be buffered.

Similarly, different video players offer different playback buffer sizes, $B_{max}$. The buffer-based algorithms adapt the video rates aggressively or conservatively based on the playback buffer level. As the buffer level increases, the algorithms select the video rate aggressively. A smaller playback buffer would fill up quickly, compared to a larger buffer. This would allow the adaptation algorithms to aggressively increase the video rate. However, a larger buffer reduces the risk of playback interruption in case of a mismatch

between the selected video rate and the available bandwidth. The ABR algorithms should be able to guarantee QoE under different settings. However, the existing algorithms ignore important parameters, such as segment duration and the client's playback buffer size, when selecting video quality. This results in inconsistent performance from algorithms under different settings.

In this paper, we present a buffer- and segment-aware fuzzy-based ABR algorithm for a single cell with multiple clients in order to optimize the QoE of competing clients. The objective of the proposed algorithm is to achieve optimal performance across a variety of video client settings and QoE objectives. To this end, the proposed algorithm considers segment duration, client playback buffer size, estimated throughput, and the amount of video in the playback buffer to select rates for the next video segments. We conducted extensive experiments to evaluate the performance of the proposed algorithm with multiple segment durations, playback buffer sizes, and video sequences. The results from our extensive experiments reveal that the proposed long video adaptation algorithm outperforms state-of-the-art algorithms, with average improvements in video rate, QoE, and bandwidth utilization, respectively, ranging from 5% to 18%, by about 13% to 30%, and by up to 45%. Moreover, the proposed algorithm manages the clients' playback buffers to avoid unnecessary rebuffering.

The rest of this paper is organized as follows. Section 2 reviews the existing video streaming algorithms. Section 3 describes the QoE maximization problem. Section 4 describes the proposed fuzzy-based logic controller, while Section 5 details the quality adaptation algorithm, and Section 6 presents the simulation results. Finally, Section 7 concludes the paper.

## 2. Related Work

Over the last decade, multiple ABR schemes have been proposed to improve QoE. The ABR algorithms can be divided into three methods: (1) throughput-based, (2) buffer-based, and (3) hybrids. Throughput-based algorithms pick the video rate of the segments considering the available throughput, $T(i)$, observed during the download of the $i$th segment [4,14–17]. In an unstable network, the available throughput may fluctuate quickly. Because the video rate is selected based on throughput, an unstable network would lead to more video rate switches, which degrades the user experience. In order to minimize video rate–switching frequency, the smoothed throughput measure, $T^s(i)$, is employed:

$$T^s\ (i+1) = (1-\delta) \times T^s(i) + \delta \times T(i) \tag{1}$$

where the value of weighing factor $\delta$ is within the range [0, 1]. The authors in [18] provided pseudocode for the adaptation algorithm of Google's MPEG-DASH Media Source demo. The algorithm selects video rates only based on the throughput observed during download of the segments. The algorithm uses a moving average of two different bandwidth estimation coefficients to reflect small- and large-scale bandwidth fluctuations. It then selects the smaller 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.

Because adaptation algorithms ignore buffer level, selecting video rates based on smoothed throughput increases the risk of playback interruption. Li et al. [19] showed that the throughput-based schemes cannot accurately estimate bandwidth when multiple clients compete for access through a network bottleneck. This leads to degradation in the selected video rates, unfair and inefficient utilization of bandwidth, and a higher frequency of video rate switches.

Some ABR algorithms observe only the playback buffer to select the video quality [20–23]. Huang et al. [6] proposed a buffer-based algorithm (BBA) that divides the playback buffer into multiple predefined thresholds ($B_1$, $B_2$, $B_3$, and $B_{max}$) such that $B_1 < B_2 < B_3 < B_{max}$. The algorithm respectively increases or decreases the video rate aggressively or conservatively based on whether the playback buffer level increases above the next higher buffer threshold or decreases below the next lower buffer threshold. In [21],

the authors formulated video rate adaptation as a utility maximization problem. They devised an online algorithm called BOLA that uses Lyapunov optimization to improve the user experience.

Multiple researchers have used a combination of playback buffer and throughput to select video quality [7,8,11,23–28]. Such algorithms combine the approaches used by throughput-based and buffer-based algorithms to select the quality of the next segment. Vergados et al. [25] proposed the Fuzzy-based DASH (FDASH) adaptation algorithm, which considers buffer level and variations in its level to mitigate playback interruption. The authors in [26] extended the work in [25] and also considered the available power in the selection of the segments in order to extend the lifetime of mobile video clients. Kim et al. [26] modified FDASH by employing history-based throughput estimation, a segment bit-rate filtering module (SBFM), and a *Start Mechanism* to improve the user experience. In [28], the authors used segment size in addition to throughput and the playback buffer size for video rate adaptation. Akshan et al. [29] proposed a fuzzy-based adaptation controller for DASH that takes observed throughput and playback buffer as input to pick the video rates. Multiple studies have shown that when multiple DASH clients compete for bandwidth in a wired network, it results in unfair allocation of bandwidth due to ON-OFF generated traffic [4,6]. The mechanism proposed in [29] also minimizes ON-OFF traffic by selecting video rates higher than the observed throughput to allow the clients to download segments continuously. However, this approach work would lead to poor performance in a cellular network. The reason is that unlike the wired network, the throughput in a cellular network depends on multiple factors, including propagation distance, fading, interference, and user mobility. The authors in [30] use a fuzzy-based adaptation mechanism that uses the moving average of the observed throughput and playback buffer level variations to minimize the video rate switches. Then, the fuzzy logic system selects the video rate based on the difference between the moving average and real values. Hou et al. [31] proposed a three-input fuzzy controller including throughput, buffer level, and buffer level variations to enhance the user experience.

The existing algorithms employ fixed control rules to pick the video rate based on the observed throughput and playback buffer level. Furthermore, the algorithms ignore important video client and content characteristics including segment duration and buffer size. This leads to the inconsistent performance of the algorithms under different video content and client settings. Different from existing works, the proposed fuzzy-based algorithm adapts video rates by exploiting not only estimated throughput and playback buffer level but also video player settings and video content characteristics. The objective of the proposed approach is to achieve optimal performance across a variety of video client settings and QoE objectives.

## 3. QoE Maximization

The HTTP server stores video content encoded into a set of $m$ video rates, $R = \{R_1, R_2, R_3, \ldots, R_m\}$. Each representation of a video is split into fixed-duration segments $\tau$. The segments are downloaded into the playback buffer, which holds the unviewed video. Let $B^k \in [0, B^{max}]$ be the playback buffer level upon download of the $k$th segment. The client downloads the $k$th segment encoded at the $i$th video rate, ${R_i}^k$. The size of the segment is ${R_i}^k \times \tau$. The download time of the $k$th segment will be $({R_i}^k \times \tau / T^k)$ where $T^k$ is the throughput observed when downloading the $k$th segment.

The objective of ABR algorithms is to improve the QoE of video clients to achieve long-term user engagement [32]. As mentioned in Section 1, the key QoE factors that affect user experience include video rate, video rate variations, and playback interruptions.

The average video bitrate over downloaded segments is given by:

$$Q = \frac{\sum_{k=1}^{S} \left( R_i^k \right)}{S} \tag{2}$$

where $R_i^k$ is the $i$th video rate ($k$ is the segment index), and $S$ is the total number of segments downloaded by the client.

Magnitude for the changes in the video rate from one segment to another is given by:

$$QS = \frac{\sum_{k=2}^{S} R_{ij}^k - R_{ij}^{k-1}}{Number\ of\ Switches} \tag{3}$$

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

In this study, we used the same QoE metric used by the authors in [17], which is defined as follows:

$$\mathrm{QoE}_j = \sum_{k=1}^{S} q\left(R_i^k\right) - \mu IR_j - \sum_{k=1}^{S} \left| q\left(R_i^k\right) - q\left(R_i^{k-1}\right) \right| \tag{4}$$

For a video fragmented into $S$ segments, $q(R_i^k)$ maps the video rate to the quality perceived by the viewer. The authors in [17] used $q(R_i^k) = R_i^k$ and $\mu = 3000$, signifying that a playback interruption of 1 s receives the same penalty as reducing the bitrate of a segment by 3000 kbps. We used the same values in our evaluation. In this study, we calculated the average QoE per segment (that is, the total QoE metric divided by the number of segments). In Section 6, we evaluate the QoE of the algorithms by using Equation (4).

In addition, we also calculate mean opinion score (MOS) using the calculations proposed in [10] to evaluate the performance of the algorithms. These MOS calculations were also used by the authors of [8] in their study to evaluate the algorithms.

$$MOS_j = \max(0.81 \times \mu - 0.96 \times \sigma - 4.95 \times \varphi + 0.17, 0) \tag{5}$$

$$\mu = \sum_{k=1}^{S} \frac{QL_k}{S} \tag{6}$$

$$\sigma = \frac{\sqrt{\sum_{k=1}^{S} (QL_k - \mu)^2}}{S} \tag{7}$$

$$\varphi = \frac{7}{8} \times \max\left(\frac{\ln(F_{req}}{6} + 1,\ 0\right) + \frac{1}{8} \times \left(\frac{\min(FT_{avg}, 15}{15}\right) \tag{8}$$

where $F_{freq}$ and $FT_{avg}$ represent the playback interruption frequency and the average interruption time, respectively. $QL_k$, $\mu$, and $\sigma$ represent the quality level for the $k$th segment, average quality level and the standard deviation of the quality level, respectively.

**The QoE maximization problem:** The problem of video rate adaptation for QoE maximization is given by:

$$\max_{R_1, \ldots R_K} \mathrm{QoE}_1^K \tag{9}$$

Subject to:

$$B^{k+1} = \left(B^k + \tau - \left(\tau \times \frac{R_i^k}{T^k}\right)\right)_+ \tag{10}$$

$$0 \leq B^k \leq B_{max} \tag{11}$$

$$R_i^k \in R,\ \forall\, 1 \leq i \leq m \tag{12}$$

Given the throughput history observed from downloading previous segments, $\{T^t,\ t \in [t^1,\ t^{k+1}]\}$, the optimization problem provides output for the video rate decision: $(R_1, \ldots, R_k)$. The notation $(x)_+ = \max(x, 0)$ ensures that the term is always positive. Constraint (7) guarantees that the clients do not experience any playback interruptions for the entire streaming duration. Constraint (8) ensures that the buffer level stays between 0 and

$B_{max}$. Finally, constraint (9) specifies that the discrete video rate downloaded by the client from the server belongs to the set of available video rates.

## 4. The Fuzzy Logic Controller

The goal of the proposed fuzzy-based scheme is to select a video rate from $R = \{R_1, R_2, R_3, \ldots, R_m\}$ to improve the user experience. Video rates and playback interruptions are important factors when improving QoE [2,3,32]. In addition, a lot of video rate switches were found to annoy viewers [33]. We consider the following parameters to improve QoE:

(1) Maximize the video rates;
(2) Minimize playback interruptions;
(3) Minimize the frequency of video rate switches.

In this section, we discuss the details of the proposed buffer- and segment-aware fuzzy-based quality adaptation algorithm. We also discuss how a fuzzy logic controller (FLC) is incorporated in the client in order to select the most feasible video rate. In this work, we used an FLC based on the Mamdani model [34]. The structure of the fuzzy logic controller is shown in Figure 3. It consists of four components: a fuzzification module, a fuzzy interference engine, a fuzzy rule base, and a defuzzification module.

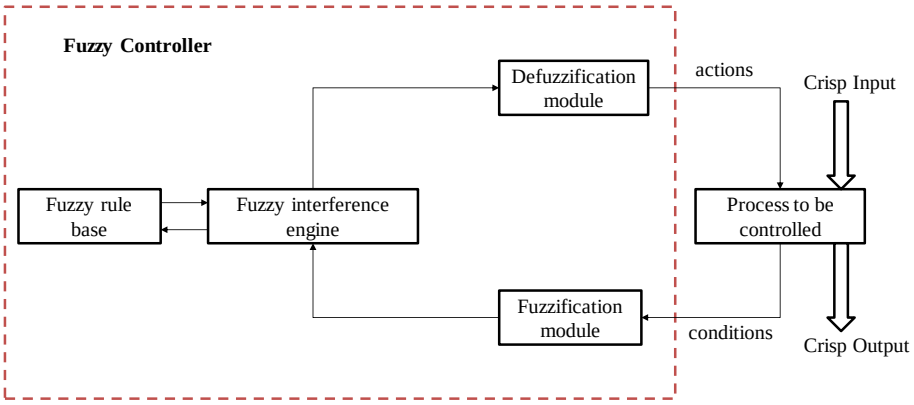



**Figure 3.** Structure of the fuzzy logic controller.

The controller operates by repeating a cycle of following steps:

- Step 1: Crisp measurements (inputs) are taken of all variables that represent the conditions of the controller process.
- Step 2: These measurements are mapped into corresponding fuzzy sets to express measurement uncertainties. This step is called fuzzification.
- Step 3: The fuzzified measurements are then used by the interference engine. The interference engine evaluates the control rules stored in the fuzzy rule base. The output of this evaluation is a fuzzy set.
- Step 4: The defuzzification module converts the fuzzy set into a single crisp value. This is the final step, called defuzzification.

In this work, the FLC applies three variables as input: (1) the ratio of the client's playback buffer level to the segment duration ($B/\tau$), (2) the ratio of $\Delta B$ to the segment duration, ($\Delta B/\tau$) where $\Delta B = B^k - B^{k-1}$ is the difference in the buffer level upon download of the last two segments, and (3) the ratio of throughput to the selected video rate ($T/R_i^k$).

The existing algorithms select video rates based on the available buffer level. As explained in Section 1, buffer level alone does not give us complete information. It is important to note here that segment durations change from one video content provider to the next. If the available playback buffer level is eight seconds, and the segment duration is two seconds, four segments can fit in the buffer. However, if the segment duration is four seconds, it means only two segments can fit into the buffer. Therefore, in this work, we consider the ratio of buffer level to segment duration.

The change in the buffer level ($\Delta B$) is the result of a mismatch between available throughput and the video rate. $\Delta B$ helps the algorithm determine whether to increase or decrease the video rate. However, $\Delta B$ alone does not accurately capture the variations in the buffer level during the download of a segment. Let us understand this by using an example. If the segment duration is two seconds, and the change in the buffer level during the download of a segment is four seconds, it means that during the download of the segment, a portion of the buffer equal to twice the segment duration gets filled up. However, if the segment duration is four seconds and the change in the buffer level is also four seconds, the portion of the buffer level that gets filled up is equal to one segment. Therefore, we consider the ratio of $\Delta B$ to segment duration as input for the FLC.

In addition to $B/\tau$ and $\Delta B/\tau$, the ratio of throughput to video rate helps to determine if the video rate should be switched abruptly or gradually. Therefore, the FLC applies the ratio of throughput to video rate as input.

Figure 4a shows the membership functions of the ratio of buffer level to segment duration, in which three linguistic variables are considered: Dangerous, Low, and Safe. When the buffer level is close to empty, video quality is compromised in order to mitigate the risk of playback interruption. As the playback buffer fills up, video quality is increased to improve the user experience.

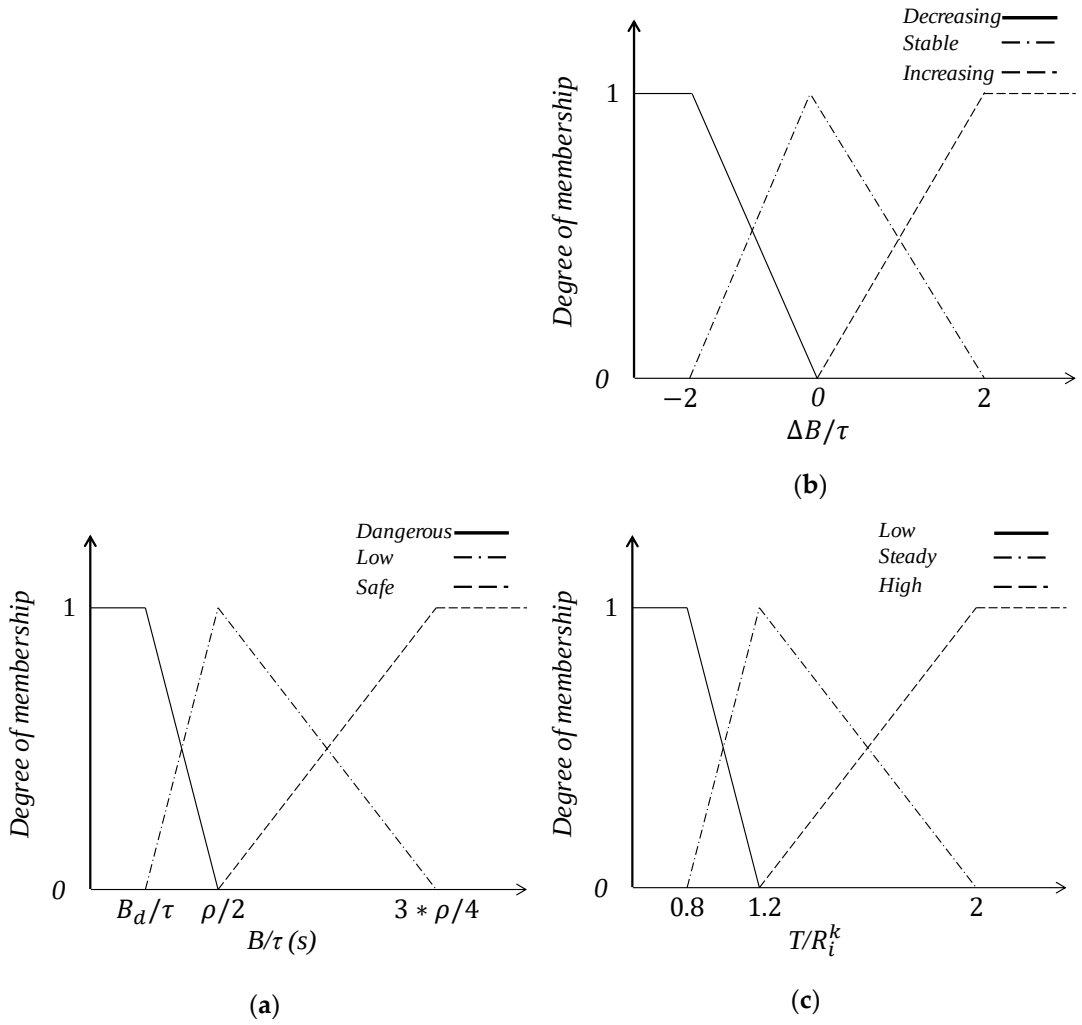



**Figure 4.** Membership functions for (**a**) the ratio of buffer level to segment duration, $B/\tau$; (**b**) the ratio of $\Delta B$ to segment duration, $\Delta B/\tau$; and (**c**) the ratio of throughput to the selected video rate, $T/R_i^k$.

Figure 4b shows the membership functions of the ratio of $\Delta B$ to segment duration in which three more linguistic variables are considered: Decreasing, Stable, and Increasing.

These linguistic variables describe the behavior from the rate of change in the buffer between subsequent segment downloads. Ideally, the buffer level should remain stable. A decrease in the buffer level forces the adaptation algorithm to decrease the video rate, whereas an increase in the buffer level suggests that the available throughput is not being utilized completely. That provides an opportunity to increase the video rate and improve video quality.

Figure 4c shows the membership functions of the ratio of throughput to selected video rate, in which three more linguistic variables are considered: Low, Steady, and High. These linguistic variables describe the extent of the mismatch between throughput and video rate. The adaptation algorithm adapts the video quality as the throughput fluctuates over time. Ideally, the adaptation algorithm selects the highest available video rate that is less than the throughput in order to efficiently utilize bandwidth.

The output of the fuzzy logic controller, $F$, represents an increase or decrease in the video rate of the next segment. As shown in Figure 5, the linguistic variables for the output are Large Decrease (LD), Small Decrease (SD), No Change (NC), Small Increase (SI), and Large Increase (LI).

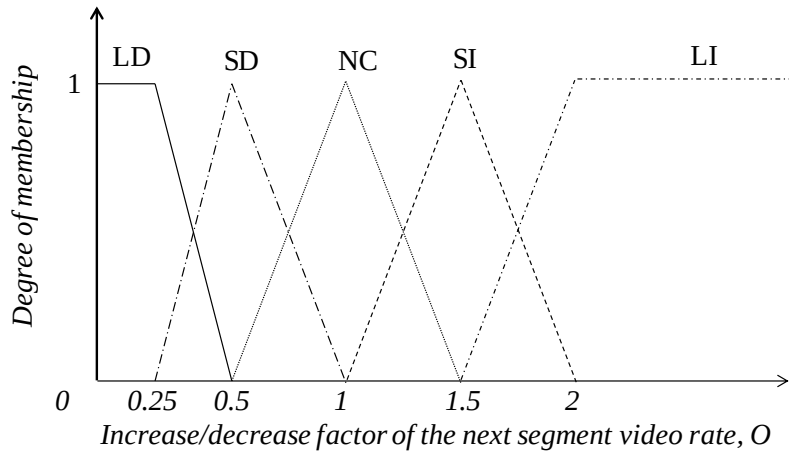



**Figure 5.** Output of the fuzzy logic controller.

The fuzzy rules provided in Table 1 are applied to map fuzzy output to fuzzy input. Each rule represents a fuzzy if–then statement. The output of the rules is aggregated into a fuzzy set. As shown in Table 1, when $B/\tau$ is in the dangerous zone, FLC decreases the video rate to minimize the risk of buffer underflow. The FLC observes $\Delta B/\tau$ and $T/R_i^k$ to determine whether to decrease the video rate abruptly or gradually. The FLC decreases the video rate gradually only when both $\Delta B/\tau$ and $T/R_i^k$ are not in the decreasing zone or the low zone, respectively.

**Table 1.** FDASH fuzzy rules.

| Rule ($R_i$) | $B/\tau$ | $\Delta B/\tau$ | $T/R_i^k$ | $O$ |
|---|---|---|---|---|
| $R_1$ | Dangerous | Decreasing | Low | *LD* |
| $R_2$ | Dangerous | Decreasing | Steady | *LD* |
| $R_3$ | Dangerous | Decreasing | High | *LD* |
| $R_4$ | Dangerous | Stable | Low | *LD* |
| $R_5$ | Dangerous | Stable | Steady | *SD* |
| $R_6$ | Dangerous | Stable | High | *SD* |
| $R_7$ | Dangerous | Increasing | Low | *LD* |
| $R_8$ | Dangerous | Increasing | Steady | *SD* |
| $R_9$ | Dangerous | Increasing | High | *SD* |
| $R_{10}$ | Low | Decreasing | Low | *SD* |
| $R_{11}$ | Low | Decreasing | Steady | *NC* |
| $R_{12}$ | Low | Decreasing | High | *NC* |
| $R_{13}$ | Low | Stable | Low | *NC* |
| $R_{14}$ | Low | Stable | Steady | *NC* |
| $R_{15}$ | Low | Stable | High | *NC* |
| $R_{16}$ | Low | Increasing | Low | *NC* |
| $R_{17}$ | Low | Increasing | Steady | *NC* |
| $R_{18}$ | Low | Increasing | High | *SI* |
| $R_{19}$ | Safe | Decreasing | Low | *SI* |
| $R_{20}$ | Safe | Decreasing | Steady | *SI* |
| $R_{21}$ | Safe | Decreasing | High | *LI* |
| $R_{22}$ | Safe | Stable | Low | *SI* |
| $R_{23}$ | Safe | Stable | Steady | *SI* |
| $R_{24}$ | Safe | Stable | High | *LI* |
| $R_{25}$ | Safe | Increasing | High | *LI* |
| $R_{26}$ | Safe | Increasing | High | *LI* |

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

As $B/\tau$ enters the low zone, the FLC focuses on staying at the current video rate to minimize the frequency of video rate changes. However, when $\Delta B/\tau$ and $T/R_i^k$ are in the decreasing zone and low zone, respectively, the FLC decides to slowly decrease the video rate. When $\Delta B/\tau$ and $T/R_i^k$ are in the increasing zone and high zone, respectively, the FLC decides to slowly increase the video rate.

Finally, when $B/\tau$ enters the safe zone, there are enough segments available in the buffer to increase video quality. However, if $\Delta B/\tau$ and $T/R_i^k$ are in the decreasing zone and low zone, respectively, the FLC decides to keep the current quality. That mitigates the risk of the buffer level dropping to the low zone or dangerous zone, and it minimizes video rate switches.

The output, $F$, is de-fuzzified to a crisp value using the center of sum method and is calculated as:

$$F = \frac{\sum_j A_j \times x_j}{\sum_j A_j} \tag{13}$$

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

## 5. Proposed Fuzzy-Based Adaptation Scheme

The DASH client downloads a video stream stored in multiple discrete video rates in the HTTP server. The set of video rates available for the video stream is denoted by $R$. The client adaptively selects a video rate from the available video rates in set $R$. This is achieved through the fuzzy logic controller. Based on the crisp output, $F$, the proposed scheme first selects $R_s$ as follows:

$$R_s = \max\{r \in R\} < F \times T^S \tag{14}$$

where $T^s$ is the smooth throughput calculated using Equation (1), in which the value of weighting factor $\delta$ is set to 0.8. One of the challenges faced by ABR algorithms is that control decisions available to algorithms are coarse-grained because limited video rates are available for a given video. As seen in Equation (14), for the next video rate the proposed scheme selects the highest available video rate that is less than ($F \times T^S$).

Next, to minimize video rate switches, Algorithm 1 is invoked after the selection of $R_s$. This algorithm first determines if $R_s$ is greater or smaller than the previously selected video rate, $R_{prev}$. If $R_s$ is greater than $R_{prev}$, the algorithm checks the buffer level. If the buffer level is within the danger zone, ($B^k < B_{dang}$) (line 5), then the video rate remains unchanged (line 6). Since the buffer level is already low, increasing the video rate increases the risk of playback interruption. Otherwise, the algorithm selects $R_s$ as the video rate for the next segment (line 8). The buffer threshold, $B_{dang}$, is equal to the minimum segment duration and part of the buffer size, as follows:

$$B_{dang} = \min \text{ (Segment duration, 20\% of buffer size)} \quad (15)$$

**Algorithm 1:** Video Rate Switch Minimization

1 **Input:** $R_s$: Video rate selected based on crisp output of FLC; $R_{prev}$: video rate of the last downloaded segment; $B^k$: current buffer level; $T^s$: smoothed throughput; $D$: segment duration; $B_{max}$: buffer size; $B_{pred}$: predicted buffer level.
2 **Output:** $R_{next}$**:** Video rate for next segment
3 ***if*** $R_s > R_{prev}$
4 $B_{pred} = B^k + (T^S / R_s - 1) * D$
5 **if** ($B_{prev} <= B_{dang}$)
6 $R_{next} = R_{prev}$
7 ***else***
8 $R_{next} = R_s$
9 ***else if*** $R_s < R_{prev}$
10 $B_{pred} = B^k + (T^S / R_s - 1) * D$
11 ***if*** ($B_{pred} >= 0.5 \times B_{max}$)
12 $R_{next} = R_{prev}$
13 ***else***
14 $R_{next} = R_s$

As explained in Section 1, video streaming services offer segments of different durations. As the segment duration increases in an unstable environment, the risk of buffer underflow increases. Therefore, segment duration should be considered in the selection of $B_{dang}$. However, with long segment durations and a small buffer size, it is not feasible to select $B_{dang}$ based only on segment duration. For example, if the available segment is 10 s, and the buffer size is 20 s, setting $B_{dang}$ equal to the segment duration means the client selects the video rate cautiously most of the time. Therefore, buffer size should also be considered in the selection of $B_{dang}$. To this end, we set $B_{dang}$ equal to the minimum segment duration plus 20% of the buffer size.

If $R_s$ is less than $R_{prev}$ (line 9) and the predicted video rate is greater than 0.5 times the buffer size (line 11), the algorithm stays at the video rate selected for the previous segment. The reason is that there is enough buffer available to take the risk of staying at the current video rate. Otherwise, the algorithm reduces the video rate to $R_s$.

## 6. Simulation Results

In this section, we implement the proposed buffer- and segment-aware fuzzy-based adaptive video streaming to evaluate its performance. We implemented the experiments by utilizing the ns-3 simulation software. We modified code available from [35] to perform our experiments. The efficiency of the proposed DASH scheme was evaluated against other adaptation algorithms, namely, FDASH [25], DBT [11], QLSA [16], and DASH-Google [18]. The topology implemented in this paper is shown in Figure 6. Clients

downloaded segments over an LTE network. Configuration details for the underlying LTE cellular network are in Table 2. The clients moved at vehicular speed within the cell. A grid-based road topology was used to simulate mobility. The clients' arrival times were uniformly distributed within the first 30 s.

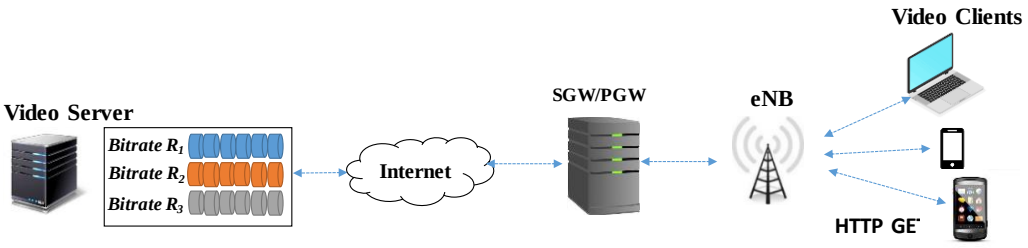



**Figure 6.** Network topology.

**Table 2.** Cellular network configuration.

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

The HAS server stores two test sequences: Tears of Steel (Dataset 1) and Big Buck Bunny (Dataset 2). The available video segments were produced using trace files of several DASH streams from [36]. Table 3 presents the client/server settings used to evaluate the algorithms. The experiment was repeated 10 times for each setting, and the average of the results is presented in this section. The videos were streamed for 10 min to evaluate the algorithms. As the experiments detailed in Table 3 were repeated, the performance of each algorithm was analyzed for 500 min. In the following results, we calculated QoE using Equation (4), and bandwidth inefficiency at time $t$ was calculated with $\frac{\left|\sum_j R_i^{(t)} - W\right|}{W}$ [7], where $W$ is the total bandwidth available at the base station.

**Table 3.** Experiment settings used to analyze the performance of the algorithms.

| Experiment | Dataset | Buffer Size | Segment Duration | Number of Clients |
|---|---|---|---|---|
| 1 | 1 | 15 s | 2 s | 10 |
| 2 | 1 | 30 s | 4 s | 10 |
| 3 | 1 | 60 s | 4 s | 10 |
| 4 | 2 | 15 s | 2 s | 10 |
| 5 | 2 | 60 s | 4 s | 10 |

### *6.1. Dataset 1*

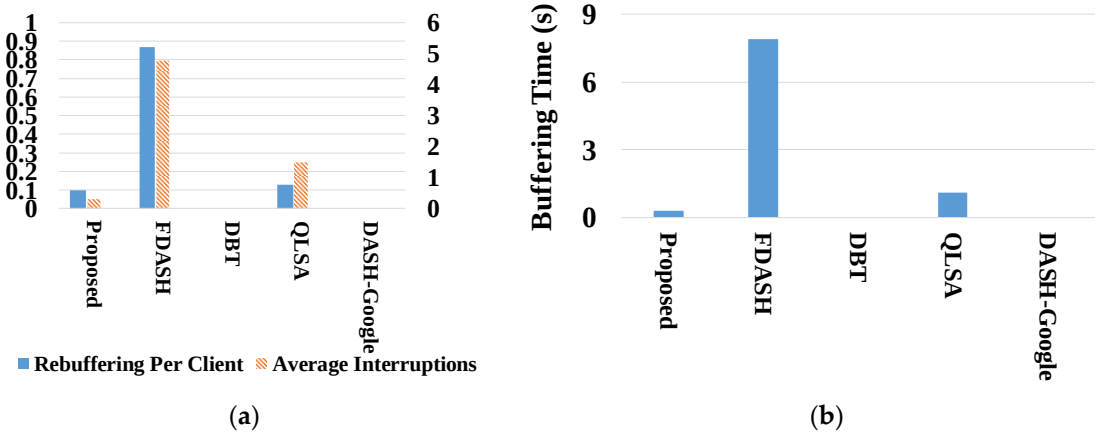
Here, we analyze the performance of the proposed algorithm when streaming video segments from Dataset 1. As shown in Table 3, 10 clients simultaneously competed for the bandwidth. In the first experiment, the buffer size and segment duration were set to 15 s and 2 s, respectively. In the next segment, buffer size and segment duration were increased to 30 s and 4 s, respectively. In the final experiment, buffer size was increased to 60 s.

Table 4 displays the performance of the algorithms when the buffer size was set to 15 s and the segment duration was 2 s. Because the buffer size was small, the risk of playback interruption was high. In case of large fluctuations in the throughput, aggressively selecting the video rate could lead to playback interruption. Table 4 shows that the proposed

algorithm selected the highest video rate from among the competing clients by efficiently utilizing the video rate. However, the proposed algorithm experienced a slightly high number of video rate switches. The reason behind the greater number of video rate switches is that when the buffer level increases, the proposed algorithm quickly increases the video rate by efficiently utilizing the bandwidth. Similarly, when the buffer level drops, the proposed algorithm reacts quickly to decrease the video rate to mitigate playback interruptions. Because the buffer level was small, and the throughput fluctuated as a mobile user moved at vehicular speed, this led to the algorithm downloading high-quality segments at the expense of slightly more video rate switches. Figure 7 shows that the proposed algorithm avoided unnecessary video playback interruptions while downloading high-quality segments. The rebufferings-per-client metric represents the ratio of clients that experienced a playback interruption to the total number of clients, while average interruptions is the number of times a client experienced a playback interruption. Table 4 shows that the DBT algorithm achieved a lower video rate than the proposed algorithm; however, the DBT algorithm had slightly higher QoE and MOS values, compared to the proposed algorithm, because it experienced fewer video rate switches. The DBT algorithm avoided switching video rates unless the buffer level increased above or decreased below predefined thresholds, irrespective of fluctuations in bandwidth. This approach helps mitigate unnecessary video rate switches. FDASH is a fuzzy-based algorithm that employs fuzzy logic to select the video rate and control the playback buffer level. In order to minimize video rate switches, the FDASH algorithm switches the video rate only if the predicted buffer level for the next 60 s increases above or decreases below a predefined threshold. If a video client has a small buffer, this approach minimizes video rate switches; however, it leads to more playback interruptions, which degrades QoE. Table 4 and Figure 7 indicate that the FDASH experienced the lowest number of video rate switches, but also experienced the greatest number of playback interruptions. The QLSA and DASH-Google algorithms achieved low video rates while experiencing more video rate switches, which degraded QoE.

**Table 4.** Performance of the algorithms with a buffer size of 15 s and segment duration of 2 s.

| Adaptation Algorithms | Proposed | FDASH | DBT | QLSA | DASH-Google |
|---|---|---|---|---|---|
| Average Video Rate (kbps) | 1307.55 | 1196.86 | 1223.38 | 1205.04 | 1216.87 |
| Switching Ratio | 0.28 | 0.05 | 0.10 | 0.32 | 0.45 |
| Inefficiency | 0.16 | 0.41 | 0.28 | 0.23 | 0.19 |
| Average of Switches (kbps) | 615.56 | 788.79 | 596.77 | 607.66 | 521.73 |
| QoE | 1145.33 | 834.32 | 1166.78 | 992.39 | 930.74 |
| MOS | 3.20 | 1.81 | 3.31 | 2.63 | 2.42 |



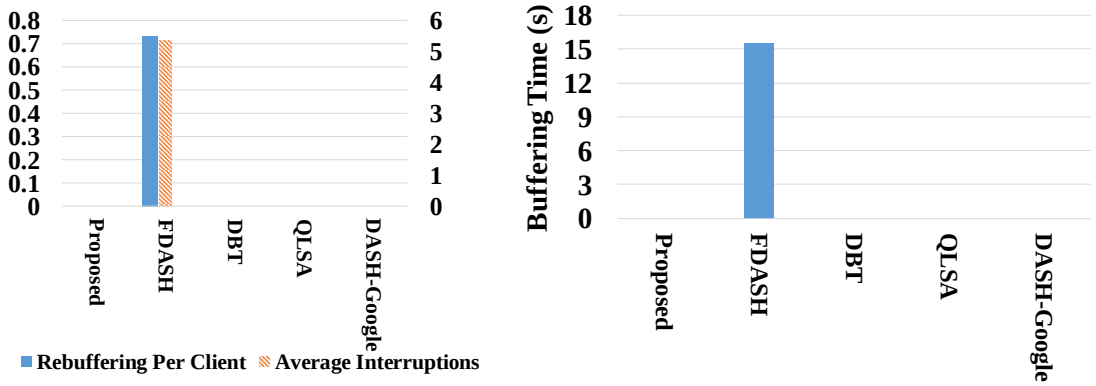

(**a**) (**b**)

**Figure 7.** Comparison of (**a**) rebufferings per client, and average number of interruptions, and (**b**) average buffering time when the clients downloaded Dataset 1. The buffer size and segment duration were set to 15 s and 2 s, respectively.

Next, we set the buffer size and segment duration to 30 s and 4 s, respectively. Although the buffer size increased, the ratio of buffer size to segment duration was the same as in the previous experiment. Table 5 shows that the proposed algorithm achieved a high video rate by efficiently utilizing the available bandwidth. Moreover, the proposed algorithm avoided playback interruptions. Table 5 shows that the proposed algorithm achieved the highest QoE and MOS values. The DBT algorithm achieved the highest QoE and MOS values in the previous experiment. However, because the buffer size increased, the performance of the algorithm degraded. The algorithm mitigated unnecessary video rate switches and avoided playback interruptions at the expense of a low video rate. As explained earlier, the DBT algorithm divides the buffer level into predefined buffer thresholds. As the buffer fills up, the algorithm selects video rates more aggressively. The DBT algorithm does not adapt the playback buffer thresholds as the buffer size changes. As the buffer size increases, the distance between the predefined thresholds also increases. This reduces the switching ratio but at the expense of a lower video rate. Table 5 and Figure 8 show that the FDASH algorithm achieved the highest video rate and the lowest switching ratio, but at the expense of a large number of rebuffering events. As a result, the QoE of the FDASH algorithm degraded. The QLSA and DASH-Google algorithms achieved high video rates but at the expense of a large number of video rate switches.

**Table 5.** Performance of algorithms at a buffer size of 30 s and a segment duration of 4 s.

| Adaptation Algorithms | Proposed | FDASH | DBT | QLSA | DASH-Google |
|---|---|---|---|---|---|
| Average Video Rate (kbps) | 1328.39 | 1397.88 | 1033.22 | 1269.27 | 1285.14 |
| Switching Ratio | 0.29 | 0.09 | 0.13 | 0.38 | 0.44 |
| Inefficiency | 0.16 | 0.29 | 0.33 | 0.23 | 0.19 |
| Average of Switches (kbps) | 765.51 | 285.99 | 668.30 | 551.19 | 534.21 |
| QoE | 1106.78 | 511.92 | 887.51 | 1030.42 | 1021.27 |
| MOS | 3.45 | 1.61 | 2.68 | 3.3 | 3.28 |




(**a**) (**b**)

**Figure 8.** Comparison of (**a**) rebufferings per client and average number of interruptions, and (**b**) average buffering time when the clients downloaded Dataset 1. The buffer size and segment duration were set to 30 s and 4 s, respectively.

Next, we increased the buffer size to 60 s. Table 6 shows that the proposed algorithm achieved the highest video rate, efficiently utilizing the bandwidth while avoiding playback interruptions. The proposed algorithm achieved the highest QoE and MOS values. The FDASH algorithm achieved a high video rate while mitigating the frequency of video rate switches. Figure 9 shows that FDASH was the only algorithm to experience playback interruptions. As a result, the FDASH algorithm had a lower video rate than the proposed algorithm. Table 6 shows that the FDASH algorithm achieved the second highest QoE value; however, its MOS value is the second lowest. The reason is that the playback interruption events heavily impact the MOS values. The DBT algorithm had the fewest

video rate switches. The DBT algorithm was able to minimize the switching ratio, but it also had the lowest video rate among the competing clients. The DASH-Google algorithm achieved a high video rate, but also experienced the most video rate switches.

**Table 6.** Performance of algorithms at a buffer size of 60 s and a segment duration of 4 s.

| Adaptation Algorithms | Proposed | FDASH | DBT | QLSA | DASH-Google |
|---|---|---|---|---|---|
| Average Video Rate (kbps) | 1274.64 | 1236.42 | 1055.77 | 1080.90 | 1225.20 |
| Switching Ratio | 0.26 | 0.1 | 0.05 | 0.32 | 0.47 |
| Inefficiency | 0.20 | 0.251 | 0.32 | 0.32 | 0.15 |
| Average of Switches (kbps) | 664.76 | 5986.98 | 834.45 | 572.24 | 639.08 |
| QoE | 1092.89 | 986.27 | 981.41 | 849.02 | 949.86 |
| MOS | 3.39 | 3.02 | 3.14 | 2.99 | 3.16 |

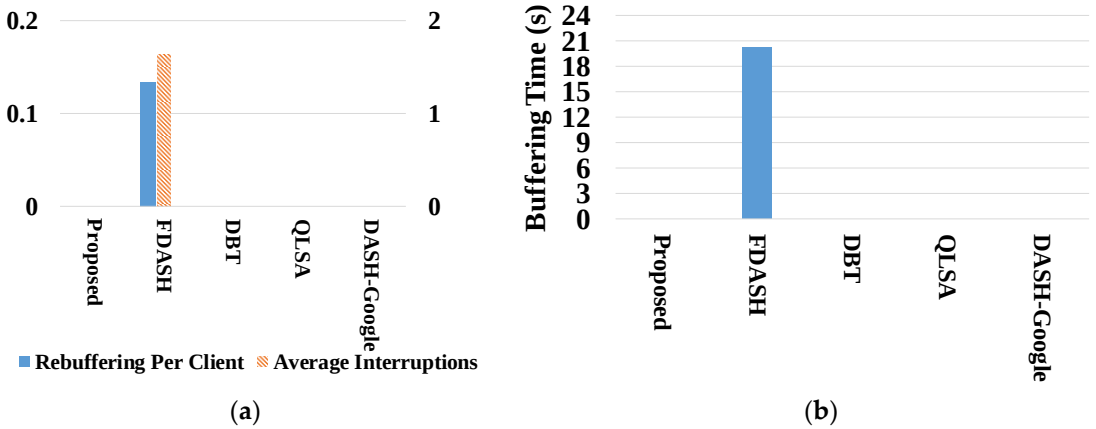



**Figure 9.** Comparison of (**a**) rebufferings per client, and average number of interruptions, and (**b**) average buffering time when the clients downloaded Dataset 1. The buffer size and segment duration were set to 60 s and 4 s, respectively.

### *6.2. Dataset 2*

Here, we analyze the performance of the algorithms when streaming the Big Buck Bunny video (Dataset 2). In the first experiment, the buffer size and segment duration were set to 15 s and 2 s, respectively. In the next segment, buffer size and segment duration were increased to 60 s and 4 s, respectively.

Table 7 shows the performance of the algorithms when the buffer size and segment duration were set to 15 s and 2 s, respectively. Like the previous experiments, the proposed algorithm achieved the highest video rate and had the lowest bandwidth inefficiency. The proposed algorithm was able to avoid rebuffering events and achieved the best user experience. The QLSA and DASH-Google algorithms achieved a high video rate but also experienced a high rate-switching ratio. The DBT algorithm experienced few video rate switches and avoided buffer underflow. However, it had the lowest video rate among the competing algorithms. Similar to the experiments conducted with Dataset 1, the FDASH algorithm experienced the fewest video rate switches but at the expense of more playback interruptions. Figure 10 shows that only the FDASH algorithm experienced playback interruptions.

**Table 7.** Performance of algorithms at a buffer size of 15 s and a segment duration of 2 s.

| Adaptation Algorithms | Proposed | FDASH | DBT | QLSA | DASH-Google |
|---|---|---|---|---|---|
| Average Video Rate (kbps) | 1418.92 | 1274.77 | 1229.03 | 1380.94 | 1394.02 |
| Switching Ratio | 0.28 | 0.07 | 0.13 | 0.42 | 0.48 |
| Inefficiency | 0.11 | 0.30 | 0.25 | 0.19 | 0.14 |
| Average of Switches (kbps) | 504.56 | 583.10 | 427.44 | 494.58 | 383.51 |
| QoE | 1286.39 | 1112.9 | 1093.30 | 1205.33 | 1135.70 |
| MOS | 3.67 | 2.97 | 3.07 | 3.43 | 3.18 |

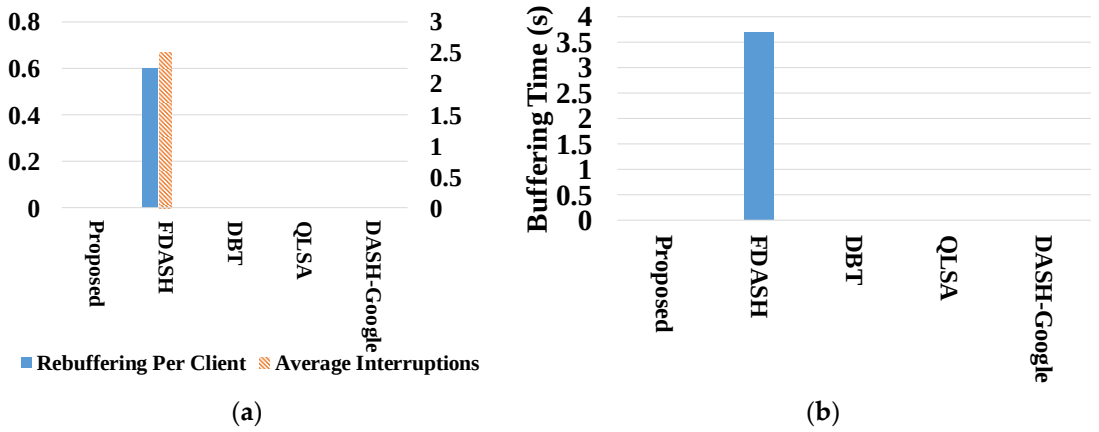


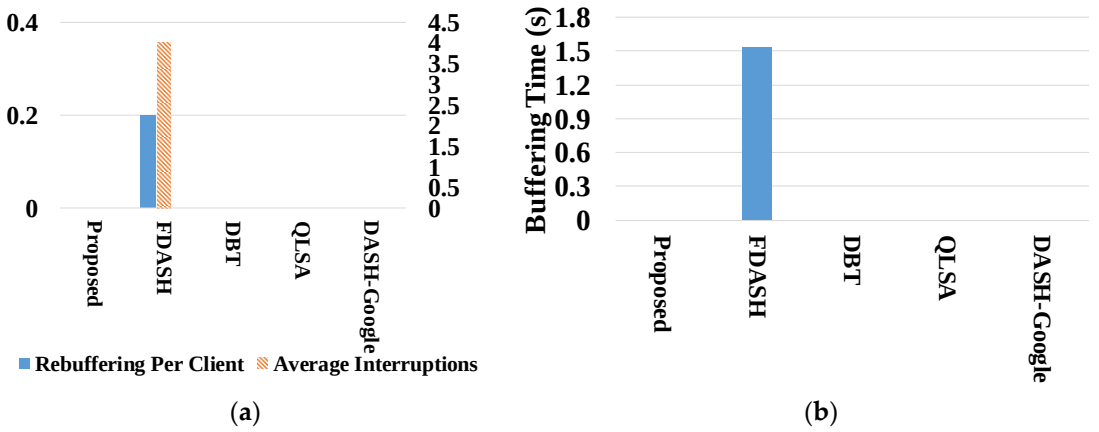

**Figure 10.** Comparison of (**a**) rebufferings per client, and average number of interruptions, and (**b**) average buffering time when the clients downloaded Dataset 2. The buffer size and segment duration were set to 15 s and 2 s, respectively.

When the buffer size and segment duration were set to 15 s and 2 s, respectively, a comparison of Tables 4 and 7 indicates that the DBT algorithm achieved the highest QoE and MOS values when downloading Dataset 1, but it achieved the lowest QoE value and the second lowest MOS value when streaming Dataset 2. Similarly, Table 4 shows that the QLSA algorithm achieved the second lowest QoE and MOS values among the competing clients while downloading Dataset 1, whereas it achieved the second highest QoE and MOS values when streaming Dataset 2. As explained in Section 1, the existing algorithms employ fixed control rules to pick the video rate. This results in inconsistent performance from the algorithms as the client settings and video content characteristics change. On the other hand, the proposed algorithm achieved a high QoE, irrespective of client settings or dataset characteristics.

Next, we increased the buffer size and segment duration to 60 s and 4 s. Table 8 shows that the proposed and QLSA algorithms achieved high video rates. Both algorithms were able to avoid playback interruptions. The proposed algorithm achieved a slightly lower switching ratio, compared to QLSA, which resulted in slightly better user experience. The DBT algorithm achieved the fewest video rate switches at the expense of low video quality. FDASH and DASH-Google achieved similar video rates. However, a high frequency of video rate switches led to low user experience from DASH-Google. Figure 10 shows that only the FDASH algorithm experienced playback interruptions. A comparison of Figures 10 and 11 shows that the number of buffering events and the buffering time decreased with an increase in buffer size.

**Table 8.** Performance of the algorithms at a buffer size of 60 s and a segment duration of 4 s.

| Adaptation Algorithms | Proposed | FDASH | DBT | QLSA | DASH-Google |
|---|---|---|---|---|---|
| Average Video Rate (kbps) | 1272.61 | 1138.06 | 1043.21 | 1272.98 | 1152.71 |
| Switching Ratio | 0.38 | 0.14 | 0.10 | 0.45 | 0.68 |
| Inefficiency | 0.20 | 0.28 | 0.31 | 0.24 | 0.14 |
| Average of Switches (kbps) | 450.32 | 436.45 | 644.35 | 525.30 | 412.12 |
| QoE | 1030.34 | 963.60 | 953.00 | 1006.99 | 905.28 |
| MOS | 3.23 | 2.88 | 3.04 | 3.09 | 2.95 |




(**a**) (**b**)

**Figure 11.** Comparison of (**a**) rebufferings per client and average number of interruptions, and (**b**) average buffering time when the clients downloaded Dataset 1. The buffer size and segment duration were set to 60 s and 4 s, respectively.

### *6.3. Summary*

Table 9 shows the average performance of the adaptation algorithms from all the experiments, and demonstrates that the proposed algorithm achieved the highest video rate and the lowest bandwidth inefficiency; this guaranteed the highest user experience among the state-of-the-art algorithms. Furthermore, the proposed algorithm avoided unnecessary playback interruptions during all the experiments. However, it experienced slightly more video rate switches. The reason is that that the proposed algorithm simultaneously focuses on efficiently utilizing bandwidth while mitigating playback interruptions. In order to achieve this, the proposed algorithm quickly reacts to changes in throughput. That results in slightly more video rate switches, but this approach helps the algorithm in downloading high-quality video segments while avoid playback interruptions. DBT and FDASH waited for the buffer volume to increase above (or decrease below) predefined thresholds. That minimized the switching ratio, but compromised video quality. Moreover, the FDASH algorithm estimates the buffer level for the next 60 s. In an environment where the bandwidth is unstable and the buffer size is small, this approach leads to a high frequency of rebuffering events. The throughput-based algorithms such as QLSA and DASH-Google do not have information on the playback buffers; therefore, they did not risk conservative reactions to changes in bandwidth. This resulted in more video rate switches.

**Table 9.** Average performance of the algorithms over all the experiments.

| Adaptation Algorithms | Proposed | FDASH | DBT | QLSA | DASH-Google |
|---|---|---|---|---|---|
| Average Video Rate (kbps) | 1274.64 | 1236.42 | 1055.77 | 1080.90 | 1225.20 |
| Switching Ratio | 0.26 | 0.1 | 0.05 | 0.32 | 0.47 |
| Inefficiency | 0.16 | 0.30 | 0.29 | 0.242 | 0.16 |
| Average of Switches (kbps) | 600.14 | 1616.26 | 634.26 | 550.19 | 498.13 |
| QoE | 1146.74 | 881.80 | 1016.4 | 1016.83 | 988.57 |

The results also indicate that the proposed algorithm guaranteed high QoE irrespective of buffer size, segment duration, and video sequence. However, the performance of the other state-of-the-art algorithms varied from one setting to the other. The reason is that these algorithms employ fixed control strategies for all settings.

In the following summary, consecutive numbers represent the results compared to FDASH, DBT, QLSA and Google-Dash, in that order.

The proposed algorithm

(1) increased the average video rate by 5.7%, 18.2%, 6.3%, and 5.2%;
(2) increased bandwidth efficiency by 30%, 45%, and 31%, whereas it achieved the same efficiency as the Google-Dash algorithm;
(3) outperformed other schemes in terms of QoE by 30%, 12.8%, 12.7%, and 16%.

## 7. Conclusions

In this paper, we presented a fuzzy-based quality adaptation algorithm that exploits video content characteristics and client-side settings to jointly optimize the user experience of HAS clients in a cellular environment. The objective of the proposed algorithm is to simultaneously meet conflicting video-quality objectives in order to optimize QoE. Simulation results revealed that the proposed fuzzy-based quality adaptation algorithm outperformed other state-of-the-art algorithms. The results demonstrate that the proposed algorithm guaranteed an improved user experience, irrespective of client playback buffer size, segment duration, and video sequences. The proposed algorithm, on average, improved video quality by over 10%, increased bandwidth efficiency by over 35%, and improved QoE by over 18%.

In the future, we plan to extend this research to an edge cloud–based fuzzy-based quality adaptation algorithm. Furthermore, we also plan to evaluate the proposed algorithm by varying the number of competing clients, the client movement speed, and the client arrival times.

**Author Contributions:** This paper represents the results of collaborative teamwork. Methodology, W.u.R.; project administration, E.-N.H.; software, W.u.R. and M.D.H.; supervision, E.-N.H.; writing—original draft, W.u.R.; writing—review and editing, W.u.R., M.D.H. and E.-N.H. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Data Availability Statement:** The dataset used to carry out the experiments can be accessed at: //ftp.itec.aau.at/datasets/DASHDataset2014 (accessed on 1 June 2021).

**Acknowledgments:** This work was supported by Institute for Information & communications Technology Planning&Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2021-0-00818, Machine Learning Based Low Bandwidth Image Communication Edge Computing System for Proactive Anomaly Detection on Smart Plant Environment).

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

## References

1. Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2017–2022 White Paper. 2019. Available online: https://s3.amazonaws.com/media.mediapost.com/uploads/CiscoForecast.pdf (accessed on 20 December 2020).
2. Seufert, M.; Egger, S.; Slanina, M.; Zinner, T.; Hoßfeld, T.; Tran-Gia, P. A survey on quality of experience of HTTP adaptive streaming. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 469–492. [CrossRef]
3. Huynh-Thu, Q.; Ghanbari, M. Temporal aspect of perceived quality in mobile video broadcasting. *IEEE Trans. Broadcast.* **2008**, *54*, 641–651. [CrossRef]
4. Jiang, J.; Sekar, V.; Zhang, H. Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive. *IEEE/ACM Trans. Netw.* **2014**, *22*, 326–340. [CrossRef]
5. Sun, Y.; Yin, X.; Jiang, J.; Sekar, V.; Lin, F.; Wang, N.; Liu, T.; Sinopoli, B. CS2P: Improving video bitrate selection and adaptation with data-driven throughput prediction. In Proceedings of the 2016 ACM SIGCOMM Conference, Florianópolis, Brazil, 22–26 August 2016; pp. 272–285. [CrossRef]

6. Huang, T.Y.; Johari, R.; McKeown, N.; Trunnell, M.; Watson, M. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. In Proceedings of the 2014 ACM Conference on SIGCOMM, New York, NY, USA, 17–22 August 2014; pp. 187–198. [CrossRef]
7. Rahman, W.; Chung, K. A novel adaptive logic for dynamic adaptive streaming over HTTP. *J. Vis. Commun. Image Represent.* **2017**, *49*, 433–446. [CrossRef]
8. Rahman, W.; Chung, K. SABA: Segment and buffer aware rate adaptation algorithm for streaming over HTTP. *Multimed. Syst.* **2018**, *24*, 509–529. [CrossRef]
9. Rahman, W.; Hong, C.S.; Huh, E.N. Edge computing assisted joint quality adaptation for mobile video streaming. *IEEE Access* **2019**, *7*, 129082–129094. [CrossRef]
10. Claeys, M.; Lleatré, S.; Famaey, J.; Wu, T.; Van Leekwijck, W.; De Turck, F. Design and optimisation of a (FA)*Q*-learning-based HTTP adaptive streaming client. *Connect. Sci.* **2014**, *26*, 25–43. [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. Zambelli, A.; Microsoft Corporation. IIS Smooth Streaming Technical Overview. Available online: http://www.microsoft.com/enus/download/details.aspx?id=17678 (accessed on 1 June 2021).
13. Adobe. Configure HTTP Dynamic Streaming and HTTP Live Streaming. Available online: https://helpx.adobe.com/adobe.../configure-dynamic-streaming-live-streaming.html (accessed on 1 June 2021).
14. Liu, C.; Bouazizi, I.; Gabbouj, M. Rate adaptation for adaptive HTTP streaming. In Proceedings of the 2011 IEEE International Conference on Multimedia and Expo, Barcelona, Spain, 11–15 July 2011; pp. 169–174. [CrossRef]
15. MPEG-DASH/Media Source Demo. Available online: http://dash-mse-test.appspot.com/ (accessed on 1 June 2021).
16. Azumi, M.; Kurosaka, T.; Bandai, M. A QoE-aware quality-level switching algorithm for adaptive video streaming. In Proceedings of the 2015 IEEE Global Communications Conference (GLOBECOM), San Diego, CA, USA, 6–10 December 2015; pp. 1–5. [CrossRef]
17. Yin, X.; Jindal, A.; Sekar, V.; Sinopoli, B. A control-theoretic approach for dynamic adaptive video streaming over HTTP. In Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication, London, UK, 17–21 August 2015; pp. 325–338. [CrossRef]
18. Ayad, I.; Im, Y.; Ketller, E.; Ha, S. A practical evaluation of rate adaptation algorithms in http-based adaptive streaming. *Comput. Netw.* **2018**, *133*, 90–103. [CrossRef]
19. 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]
20. Huang, T.Y.; Johari, R.; McKeown, N. Downton abbey without the hiccups: Buffer-based rate adaptation for HTTP video streaming. In Proceedings of the 2013 ACM SIGCOMM Workshop On Future Human-Centric Multimedia Networking, Hong Kong, China, 16 August 2013; pp. 187–198. [CrossRef]
21. Spiteri, K.; Urgaonkar, R.; Sitaraman, R.K. BOLA: Near-Optimal Bitrate Adaptation for Online Videos. *IEEE/ACM Trans. Netw.* **2018**, *28*, 1–9. [CrossRef]
22. Rahman, W.; Chung, K. Chunk size aware buffer-based algorithm to improve viewing experience in dynamic HTTP streaming. *IEICE Trans. Commun.* **2016**, *99*, 767–775. [CrossRef]
23. 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]
24. 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. *Signal Image Video Process.* **2019**, *13*, 1367–1375. [CrossRef]
25. Vergados, J.D.; Michalas, A.; Sgora, A.; Vergados, D.D.; Chatzimisios, P. FDASH: A fuzzy-based MPEG/DASH adaptation algorithm. *IEEE Syst. J.* **2015**, *10*, 859–868. [CrossRef]
26. Mowaf, M.; Taqieddin, E.; Al-Dahoud, H. Energy efficient fuzzy-based DASH adaptation algorithm. *Digit. Commun. Netw.* **2021**, *7*, 113–119. [CrossRef]
27. Jun, K.H.; Seul, S.Y.; Tae, K.J. A modification of the fuzzy logic based dash adaptation scheme for performance improvement. *Wirel. Commun. Mob. Comput.* **2018**. [CrossRef]
28. Rahman, W.; Chung, K. Buffer-based adaptive bitrate algorithm for streaming over HTTP. *KSII Trans. Internet Inf. Syst.* **2015**, *9*, 4585–4622. [CrossRef]
29. 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, Oregon, 18–20 March 2015; pp. 31–36. [CrossRef]
30. Ma, L.V.; Park, J.; Nam, J.; Ryu, H.; Kim, J. A fuzzy-based adaptive streaming algorithm for reducing entropy rate of dash bitrate fluctuation to improve mobile quality of service. *Entropy* **2017**, *19*, 477. [CrossRef]
31. Hou, Y.; Xue, L.; Li, S.; Xing, J. User-experience-oriented fuzzy logic controller for adaptive streaming. *Comput. J.* **2018**, *61*, 1064–1074. [CrossRef]
32. Dobrian, F.; Sekar, V.; Awan, A.; Stoica, I.; Joseph, D.A.; 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]

33. Ni, P.; Eg, R.; Eichhorn, A.; Griwodz, C.; Halvorsen, P. Flicker effects in adaptive video streaming to handheld devices. In Proceedings of the 19th ACM International Conference on Multimedia, Scottsdale Arizona, AZ, USA, 28 November–1 December 2011; pp. 463–472. [CrossRef]
34. Mamdani, E.H.; Assilian, S. An experiment in linguistic synthesis with a fuzzy logic controller. *Int. J. Man-Machine Stud.* **1975**, *7*, 1–13. [CrossRef]
35. Available online: https://github.com/djvergad/dasht/ (accessed on 1 June 2021).
36. Available online: http://ftp.itec.aau.at/datasets/DASHDataset2014/ (accessed on 1 June 2021).